

# S<sup>3</sup>C-MATH: Spontaneous Step-level Self-correction Makes Large Language Models Better Mathematical Reasoners

Anonymous submission

## Abstract

Self-correction is a novel method that can stimulate the potential reasoning abilities of large language models (LLMs). It involves detecting and correcting errors during the inference process when LLMs solve reasoning problems. However, recent works do not regard self-correction as a spontaneous and intrinsic capability of LLMs. Instead, such correction is achieved through post-hoc generation, external knowledge introduction, multi-model collaboration, and similar techniques. In this paper, we propose a series of mathematical LLMs called S<sup>3</sup>C-MATH, which are able to perform **Spontaneous Step-level Self-correction for Mathematical reasoning**. This capability helps LLMs to recognize whether their ongoing inference tends to contain errors and simultaneously correct these errors to produce a more reliable response. We proposed a method, which employs a step-level sampling approach to construct step-wise self-correction data for achieving such ability. Additionally, we implement a training strategy that uses above constructed data to equip LLMs with spontaneous step-level self-correction capacities. Our data and methods have been demonstrated to be effective across various foundation LLMs, consistently showing significant progress in evaluations on GSM8K, MATH, and other mathematical benchmarks. To the best of our knowledge, we are the first to introduce the spontaneous step-level self-correction ability of LLMs in mathematical reasoning.

## Introduction

Reasoning is one of the essential foundational abilities of large language models (LLMs), showing the capacity of LLMs to tackle complex real-world problems. Nowadays, researchers are increasingly focusing on the performance of LLMs in specific reasoning tasks like mathematics, code, logic, common-sense, etc. (Sun et al. 2024). Mathematics is one of the significant branches of reasoning ability of LLMs, and solving a mathematical problem can demonstrate an LLM’s ability to decompose, reason, and summarize complex problems. In recent work, the chain-of-thought (CoT) inference (Wei et al. 2023) has been proven to be a method that can significantly enhance an LLM’s ability to solve reasoning problems (Suzgun et al. 2022; Wang and Zhou 2024). CoT induces the model to explicitly output the reasoning process, gradually deriving a series of intermediate processes or sub-goals, thereby enabling the model to correctly answer reasoning questions. However, during the

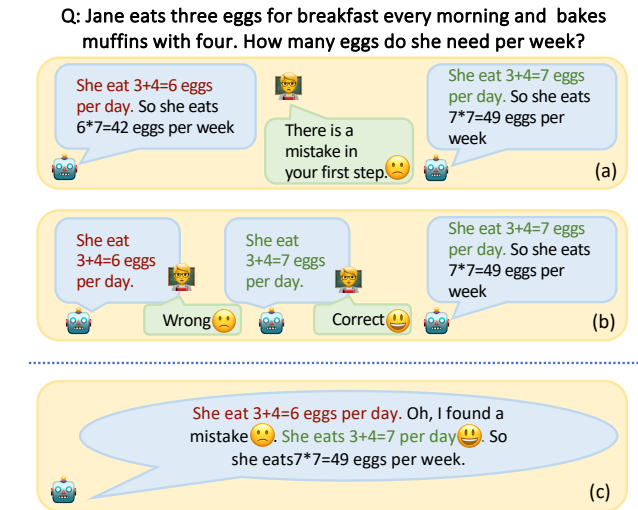


Figure 1: Different ways to implement self-correction. (a) Using more capable LLMs like GPT4 or a trained model to generate the feedback; (b) Apply such process at the step-level; (c) Our work, the LLMs spontaneously identify errors in their output and correct them immediately. Red text denotes the wrong step, and green text denotes the content after correction.

step-by-step reasoning process, an LLM may still generate errors (Chen et al. 2024; Wang et al. 2024). These errors can be propagated to subsequent reasoning processes, leading to incorrect outputs from the LLMs.

To alleviate potential errors that may occur during model reasoning, self-correction methods have been proposed. These methods can check whether the model makes mistakes during the reasoning process, pinpoint the errors, and enable the model to generate a better answer based on the provided feedback (Kamoi et al. 2024; Pan et al. 2024). Existing research has already leveraged such error correction processes to make the model’s response more reliable (Puerto et al. 2024; Paul et al. 2024; Xu et al. 2024a). The crux of self-correction lies in **how to generate feedback** for the model’s output, and **how to generate a better response** based on this feedback (Tyen et al. 2024).

In terms of generating feedback, for some specific tasks, external tools or knowledge can be used as feedback. For instance, in the task of code generation, the code generated by the model can be handed over to the compiler. If the code cannot compile, the model can be informed that an error exists, and the error stack can be provided to the LLMs (Zheng et al. 2024; Li et al. 2024a). For tasks that are difficult to judge with external tools, some research provides the model’s original answer to more capable models such as GPT4 or a trained critic model specifically for error detection, thereby generating corresponding feedback (Zhang et al. 2024b). Such self-correction is usually applied at the instance-level, i.e., after the model’s complete output, feedback generation and error correction are completed. Given that most reasoning tasks are solved in a CoT manner, these self-correction methods have begun to be applied at the step-level, providing the model with more granular signals. We illustrate these self-correction methods in panels (a) and (b) of Figure 1.

However, we believe that the aforementioned methods still seem unnatural in LLMs’ inference. Utilizing multiple models to address a single problem does not provide a fair assessment of the LLM’s reasoning ability, and such an approach cannot be considered an end-to-end capability of the model. Therefore, some researchers have begun to propose implementing all steps of self-correction within the same model, using multi-task learning to enable a single model to master both problem-solving and error correction tasks (Madaan et al. 2023; Liu et al. 2024). Yet, we do not think this is enough. Multi-stage processing requires LLMs to generate several times, thus increasing inference time and computational costs. Moreover, this behavior is not spontaneous from the model itself, but a preset fixed process in multi-stage processing, which cannot demonstrate the model’s capacity for error recognition. Based on these considerations, we propose **spontaneous step-level self-correction** capability for LLMs, which allows the LLM to spontaneously identify errors in its on-going output and correct them immediately, ultimately yielding more reliable responses. This process is illustrated in panel (c) of Figure 1.

In this paper, we propose an ingenious method for constructing self-correction data to achieve spontaneous step-level self-correction. This method utilizes existing step-by-step reasoning instruction data, generating potentially erroneous steps through sampled generation, and validating whether there are indeed errors in the steps through the *pass@k* validation. We insert the erroneous steps and the markers used to trigger self-correction into the correct steps in the existing data and construct S<sup>3</sup>C-MATHQA, which includes 532K self-correction data, based on MetaMathQA (Yu et al. 2024). In order to make self-correction more accurate and generalize to more mathematical problems, we annotate each correction case in a fine-grained manner, adding the reflection and improvements. During the training stage, we mix S<sup>3</sup>C-MATHQA and the original 395K MetaMathQA to create 927K SFT data, and use loss-masks to ignore the loss of the erroneous steps. This method ensures that the LLMs are introduced with a new self-correction capability while maintaining their original

effectiveness. Our experiments show stable and consistent improvements across multiple mathematical datasets such as GSM8K (Cobbe et al. 2021) and MATH (Hendrycks et al. 2021) on both generalist LLMs like Meta-Llama-3-8B (Meta AI 2024) and math-specialized LLMs like DeepSeek-Math (Shao et al. 2024). Our main contributions are as follows:

- We introduce spontaneous step-level self-correction capability of LLMs, which can automatically correct potential errors and generate a more reliable response in the reasoning process. To the best of our knowledge, we are the first to propose this capability in mathematical reasoning of LLMs.
- We propose a method for constructing spontaneous step-level self-correction data, which could be used to quickly and effectively build self-correction data from existing step-by-step SFT data. With the proposed method, we have constructed S<sup>3</sup>C-MATHQA, a 532K SFT dataset to produce the ability of spontaneous step-level self-correction.
- We employ a fine-tuning strategy incorporating loss-mask on the S<sup>3</sup>C-MATHQA, resulting in the training of S<sup>3</sup>C-MATH, a series of models with spontaneous step-level self-correction capabilities in mathematical reasoning. This strategy enables the LLMs to identify potential errors during the output process and autonomously correct them. The trained S<sup>3</sup>C-MATH, with this capability, achieves performance improvements on multiple mathematical benchmarks.

## Related Works

### Mathematical Reasoning of LLMs

Mathematical reasoning, one of the foundational abilities of LLMs, demonstrates the model’s capability to solve complex real-world mathematical problems, which can be enhanced during various stages of LLMs’ training. In the pre-training stage, continuing pre-training on large amounts of mathematical corpora can improve the internal mathematical reasoning ability of LLMs from a knowledge perspective (Ying et al. 2024; Shao et al. 2024; Zhou et al. 2024). During supervised instruction tuning, mathematical abilities could be further enhanced by training on problem-answer pairs, leading to the same answer formatting and better reasoners (Yu et al. 2024; Li et al. 2024b). For the preference alignment stage, preference optimization (Chen et al. 2024) and reinforcement learning (Lightman et al. 2023; Wang et al. 2024) are used to improve LLMs at a fine-grained level.

### Self-Correction of LLMs

Self-correction is an advanced technique to enhance an LLM’s output by refining its initial response during the inference process, which can be implemented through various methodologies (Kamoi et al. 2024; Pan et al. 2024). Strategies of self-correction center around the concept of feedback generation. Essentially, the LLM must evaluate whether errors exist in its initial output, identify these errors, and pinpoint the steps needed for correction. The timing of self-correction can be categorized into two main types: post-hoc

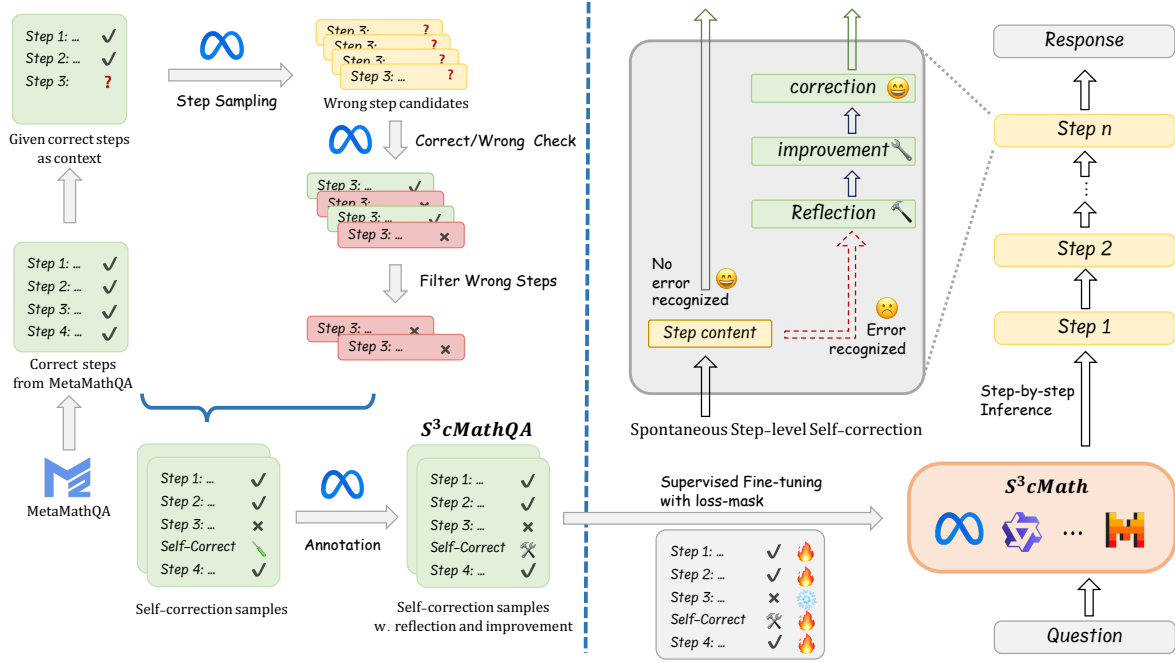


Figure 2: An overview of our works. The left part of this figure illustrates the process of constructing S<sup>3</sup>C-MATHQA, and the right part depicts the training of S<sup>3</sup>C-MATH and the procedure during its reasoning.

and real-time. In the post-hoc approach, feedback is generated after the initial inference, based on which the LLM is prompted to re-generate the response (Paul et al. 2024; Jiang et al. 2023). In contrast, the real-time approaches integrate feedback into the generation context during the inference process, enabling the LLM to produce a completion with the necessary corrections on the fly. Additionally, feedback of the LLMs’ contents could be generated in two types: external knowledge and tools to such as code compilers (Zheng et al. 2024; Li et al. 2024a) or LLMs themselves. Feedback generation by LLMs can be bifurcated into cross-model and same-model approaches. In the cross-model approach, a more advanced LLM or a dedicatedly trained critic model is employed to produce the feedback (Li, Patel, and Du 2024; Cohen et al. 2023; Du et al. 2023). On the other hand, the same-model approach utilizes the same LLM for both the initial generation and the feedback, albeit with different prompts to guide the correction process (Thorne and Vlachos 2021; Tyen et al. 2024; Paul et al. 2024).

### Synthesised Mathematical Data

Synthesized data generated from existing LLMs has been proven effective for training another LLM (Yu et al. 2024; Liu and Yao 2024; Li et al. 2024b; Zhu et al. 2024; Xu et al. 2024b; Zhou et al. 2024), which can be used in both pre-training and supervised fine-tuning. For pre-training, data can be synthesized to expand the scope of mathematical knowledge that LLMs can learn from. For instance, continuing pre-training on the synthesized Jiuzhang dataset (Zhou et al. 2024) significantly improves LLMs’ performance on multiple mathematical benchmarks. For supervised fine-

tuning, there are also various methods to synthesize supervised instruction pairs. For example, MetaMathQA (Yu et al. 2024) augments questions from the original training set and generates answers for those bootstrapped questions. More recently, KPMath (Zhu et al. 2024) extracts key points from existing data to analyze and synthesize more complex mathematical word problems.

### Approach

The main works of this paper can be divided into two parts. The first part is to construct self-correction SFT data S<sup>3</sup>C-MATHQA based on existing step-by-step mathematical instruction pairs. The second is to use S<sup>3</sup>C-MATHQA to carry out instruction fine-tuning, ultimately enabling the model to acquire spontaneous step-level self-correction capabilities.

### Data Construction

We divide our proposed self-correction data construction process into two steps. The first step involves generating erroneous steps from existing correct CoT steps, which will be used to create step-level self-correction instances. The second step concerns reflections and improvements for the above step-level self-correction instances, which serve as detailed guidance for LLMs to better acquire the ability to do spontaneous step-level self-correction. In this paper, we use the MetaMathQA (Yu et al. 2024) dataset, which includes 395K mathematical reasoning step-by-step samples, as the seed dataset.

**Wrong Step Sampling** Before we start, we carry out instruction tuning on the Meta-Llama-3-8B (Meta AI 2024)

using the MetaMathQA data, resulting in our model  $M$  used for sampling erroneous steps. Since we need to sample on the training set, in order to mitigate the reduction of diversity caused by model over-fitting, we adopt the idea of cross-validation and conduct a 5-fold cross-training. That is, we evenly divide the MetaMathQA data into five parts, use four parts of the data to train the model, and the remaining one part of the data for sampling erroneous steps. This process is repeated five times on different parts of the data. Hereafter, we denote the data used for model training as  $s$ , and the remaining data used for sampling erroneous steps as  $\bar{s}$ . The model that undergoes instruction tuning on  $s$  is denoted as  $M_s$ .

First, we use the trained model  $M_s$  to perform step-wise sampling on the split-out data  $\bar{s}$ , obtaining a set of steps that may contain errors. The specific method is that for each step  $x_i$  in an SFT sample response  $x$ , we concatenate all the steps from the first to the  $i^{th}$  step into a context and give it to  $M_s$  for continuation, until the model outputs the next step, obtaining  $candidates(\hat{x}_{i+1})$ . We use a high temperature and sample multiple steps at the same time to ensure the diversity of steps. The candidates of the  $(i+1)^{th}$  steps of case  $x$  could be represented as:

$$candidates(\hat{x}_{i+1}) = M_s^k(x_1 \oplus \dots \oplus x_i), x \in \bar{s}$$

where  $\oplus$  indicates concatenation of existing correct steps, and  $M_s^k(x_1 \oplus \dots \oplus x_i)$  represents the set of responses generated by  $M_s$  for  $x_1 \oplus \dots \oplus x_i$   $k$  times.

Next, we will use the trained model to evaluate the correctness of the generated candidates ( $\hat{x}_{i+1}$ ). Specifically, for each step  $\hat{x}_{i+1}$  in the candidate set, we concatenate the correct preceding steps  $x_1, \dots, x_i$  as context before the step  $\hat{x}_{i+1}$ , and ask the model  $M_s$  to continue generating until the model actively ends the generation (i.e., encounters the `<end-of-sequence>` token). We will determine whether the step is correct based on whether the answer generated by model  $M_s$  matches the standard answer. In order to make more accurate judgments and reduce the cases where correct steps are mistakenly considered as wrong, we use  $pass@k$  to evaluate a step. Specifically, we sample multiple model outputs with a high temperature of 1.0 for 16 times, and only when none of the contents generated by the model match the correct answer do we consider the step to be wrong. Whether  $\hat{x}_{i+1}$  is correct can be represented as:

$$\hat{x}_{i+1} \begin{cases} \text{Correct} & \text{if } \exists o \in M_s^k(x_1 \oplus \dots \oplus x_i \oplus \hat{x}_{i+1}), A(o) = A(gt) \\ \text{Wrong} & \text{if } \forall o \in M_s^k(x_1 \oplus \dots \oplus x_i \oplus \hat{x}_{i+1}), A(o) \neq A(gt) \end{cases}$$

where  $A(o)$  represents the answer extraction for the output  $o$ , while  $A(gt)$  stands for the answer extraction for the golden truth.

**Reflection and Improvement Generation** Based on the correct steps in the existing data and the erroneous steps we sampled in the previous section, we inserted the erroneous steps and the flags used to trigger self-correction into the correct steps, thus forming the direct self-correction data. However, another significant challenge of self-correction is how

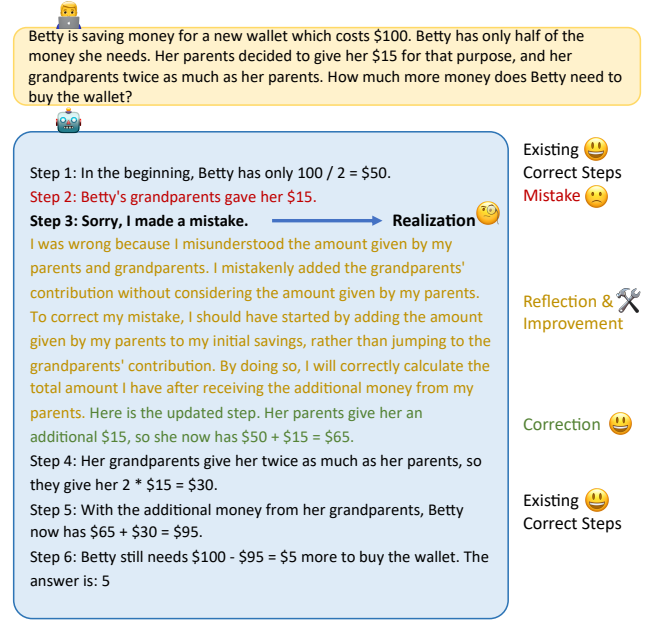


Figure 3: An example from  $S^3C$ -MATHQA. Steps 1, 4, 5, and 6 are copied from the correct steps already exist in MetaMathQA. Step 2 is a wrong step sampled from the method we proposed. Step 3 is the indicator of LLM’s error realization, reflection, improvement, and the new corrected step. During training, we do not compute the loss for the red texts.

the LLMs can produce more accurate content when they know that errors already exist. To enable the LLMs to better learn to correct the identified mistakes, we annotated the acquired step-level self-correction samples in two ways, reflection and improvement, using Meta-Llama-3-70B (Meta AI 2024). Reflection involves the model analyzing where errors occurred in the existing answers, while improvement entails generating ways to improve based on the existing output and reflection. When generating these two annotations, we provide the model with a prompt, as well as a question, previous step, wrong step, and correct step, and ask the model to make annotations based on this information. We present a data example and its components in Figure 3.

## Tuning and Evaluation

After constructing the aforementioned data, we implement SFT on it. We test our trained model on multiple mathematical evaluation benchmarks to validate the effectiveness of our data and methods.

**Instruction Tuning** We conduct SFT on the constructed dataset. Since our SFT data is self-correction data, it contains at least one erroneous step. Learning from these erroneous steps might harm the reasoning performance of the model itself. To avoid being affected by this situation, we use a loss mask to control the learning objectives of the model. Specifically, we applied a mask operation to the erroneous step, preventing the model from learning this error, thereby

Base Model	SFT Data	GSM8K		MATH		SVAMP		Mathematics	
		<i>P@1</i>	<i>M@32</i>	<i>P@1</i>	<i>M@32</i>	<i>P@1</i>	<i>M@32</i>	<i>P@1</i>	<i>M@32</i>
Generalist Models									
Meta-Llama-3-8B (Meta AI 2024)	MetaMathQA	81.12	86.20	30.58	39.92	81.40	<b>86.60</b>	18.27	<b>28.21</b>
	+ S <sup>3</sup> C-MATHQA- w/o. R&I	81.65	<b>88.10</b>	<u>32.32</u>	41.00	80.50	84.60	17.56	25.48
	+ S <sup>3</sup> C-MATHQA- w/o. I	<u>82.34</u>	86.96	31.98	<u>41.44</u>	80.40	85.60	<b>19.32</b>	26.52
	+ S <sup>3</sup> C-MATHQA	<b>82.94</b>	<u>87.34</u>	<b>33.14</b>	<b>41.60</b>	<b>81.80</b>	<u>86.40</u>	<u>19.08</u>	<u>27.53</u>
Mistral-7B-v0.3 (Mistral AI team 2023)	MetaMathQA	73.46	82.34	24.04	31.34	77.20	83.30	<b>17.35</b>	<u>27.98</u>
	+ S <sup>3</sup> C-MATHQA- w/o. R&I	75.36	81.80	<b>27.30</b>	32.28	75.60	83.40	13.48	25.89
	+ S <sup>3</sup> C-MATHQA- w/o. I	<b>75.89</b>	<b>82.87</b>	25.18	<u>32.96</u>	<b>78.60</b>	<u>83.80</u>	14.67	25.89
	+ S <sup>3</sup> C-MATHQA	<u>75.51</u>	<u>82.64</u>	<u>25.48</u>	<b>33.74</b>	<u>78.40</u>	<b>84.90</b>	<u>16.07</u>	<b>28.10</b>
Meta-Llama-3-70B (Meta AI 2024)	MetaMathQA	88.55	91.74	<u>45.70</u>	<u>55.40</u>	87.40	<u>91.00</u>	29.08	40.30
	+ S <sup>3</sup> C-MATHQA- w/o. R&I	89.16	<u>92.72</u>	44.16	53.46	87.60	90.70	28.48	39.02
	+ S <sup>3</sup> C-MATHQA- w/o. I	<u>89.39</u>	92.57	45.66	54.50	<b>89.10</b>	<b>91.70</b>	<u>32.62</u>	<u>42.02</u>
	+ S <sup>3</sup> C-MATHQA	<b>91.66</b>	<b>93.33</b>	<b>46.22</b>	<b>55.80</b>	<u>87.70</u>	<u>91.00</u>	<b>34.17</b>	<b>42.65</b>
Math-specialized Models									
DeepSeek-Math-Base-7B (Shao et al. 2024)	MetaMathQA	79.30	85.90	38.22	46.86	80.60	86.40	25.98	42.95
	+ S <sup>3</sup> C-MATHQA- w/o. R&I	82.18	87.72	40.08	50.92	81.80	86.40	23.75	43.93
	+ S <sup>3</sup> C-MATHQA- w/o. I	<b>83.02</b>	<u>87.87</u>	<u>40.48</u>	<u>51.60</u>	<b>82.50</b>	<u>86.70</u>	<u>26.25</u>	<u>44.20</u>
	+ S <sup>3</sup> C-MATHQA	<u>82.49</u>	<b>88.17</b>	<b>41.40</b>	<b>52.14</b>	<u>82.20</u>	<b>87.60</b>	<b>28.27</b>	<b>45.68</b>
Qwen2-Math-7B (Qwen Team 2024)	MetaMathQA	84.08	89.08	51.32	60.64	85.60	<b>90.40</b>	<b>42.44</b>	52.02
	+ S <sup>3</sup> C-MATHQA- w/o. R&I	84.38	<u>89.39</u>	<u>51.94</u>	<u>62.80</u>	86.70	89.50	39.73	51.61
	+ S <sup>3</sup> C-MATHQA- w/o. I	<b>84.91</b>	<b>89.61</b>	<b>52.24</b>	<b>63.70</b>	<u>87.30</u>	<u>90.30</u>	<u>40.83</u>	<b>52.23</b>
	+ S <sup>3</sup> C-MATHQA	<u>84.76</u>	<b>89.61</b>	51.76	62.40	<b>87.40</b>	89.40	39.35	<u>52.20</u>

Table 1: Our main experimental results (%) on four mathematical reasoning tasks (GSM8K, MATH, SVAMP and Mathematics) under *Pass@1* and *Majority@32* settings. The term "w/o. R&I" represents that the self-correction part of the SFT samples do not include reflection and improvements, while "w/o. I" indicates that the self-correction part of the SFT samples do not include improvements. We abbreviate *Pass@1* as *P@1* and *Majority@32* as *M@32*. The highest accuracy is indicated in **bold**, while the second highest is underlined.

maintaining the performance of the existing SFT data. As shown in the red section of Figure 3, we will not calculate the loss for it.

**Evaluation Settings** We test the performance of our models on the evaluation sets of GSM8K (Cobbe et al. 2021), MATH (Hendrycks et al. 2021), SVAMP (Patel, Bhatamishra, and Goyal 2021), and Mathematics (Saxton et al. 2019). For each sample in the test set, we only perform inference once. Each test sample in these datasets contains a question and a golden answer. We extract the final answer from the model’s output and matched it with the golden answer to ultimately determine whether the sample was answered correctly. The evaluation uses the *vllm* framework (Kwon et al. 2023) for inference, and we evaluate the model’s single-inference accuracy, i.e., the *pass@1* metric, using greedy decoding. Self-consistency (Wang et al. 2023) is considered a stable way to evaluate the model’s sampling method. To more accurately assess the improvements brought about by our strategy, we also test the accuracy under the *maj@32* setting, setting the temperature to 0.7 during sampling decoding.

## Experiments

### Baseline

Since our data is built on MetaMathQA (Yu et al. 2024) dataset, we use the model fine-tuned on MetaMathQA as the baseline for our experiments. We compare the performance difference between our trained model and the baseline model on several mathematical benchmarks. All our SFT experiments use the same configuration. We use Megatron-LM as the training framework and set the initial learning rate to 1e-5, but except 1e-6 for Mistral-7B-v0.3 (Mistral AI team 2023), with a 0.03 warm-up ratio, and decay to 0 using a cosine schedule. The model’s max length is 8192, the global batch size is 128, and the number of training epochs is 3. We pack all SFT data to accelerate the training process. We conducted our SFT experiments on 32 Nvidia A100 GPUs.

### Main results

Based on MetaMathQA 395K, We applied our proposed method to synthesize S<sup>3</sup>C-MATHQA, 532K self-correction data. For all instances in S<sup>3</sup>C-MATHQA, we generated reflection and improvement using our proposed methods. The main motivation of our paper is to seamlessly introduce

the end-to-end intrinsic self-correction capability into the model. Therefore, we merged the original MetaMathQA dataset with our synthesized data S<sup>3</sup>C-MATHQA, yielding a total of 927K data for our SFT data corpus.

To demonstrate the generalization effect of our data, we conducted SFT experiments on different foundation LLMs using the MetaMathQA and S<sup>3</sup>C-MATHQA. We adopted both generalist LLMs and math-specialized LLMs. For the generalist LLMs, we utilized three different size base models: Meta-Llama-3-8B(Meta AI 2024), Mistral-7B-v0.3(Mistral AI team 2023), and Meta-Llama-3-70B(Meta AI 2024). For the math-specialized models, we used the DeepSeek-Math(Shao et al. 2024) and Qwen2-Math-7B models(Qwen Team 2024).

We evaluated the trained model on GSMK, MATH, SVAMP, and Mathematics, with the results shown in Table 1. Our experimental results demonstrate that our model can make stable improvements across multiple mathematical evaluation benchmarks. On Meta-Llama-3-8B, we improved the accuracy of GSM8K from 81.1% to 82.9%, and the accuracy of MATH from 30.6% to 33.1%. On Deepseek-Math-Base, we increased the accuracy of GSM8K from 79.3% to 82.5%, and the accuracy of MATH from 38.2% to 41.4%. Our method achieved consistent improvements on both the generalist LLMs and the math-specialized LLMs, thereby validating the effectiveness of the method we proposed.

In our experimental results, we compared three methods: direct correction, correction after reflection, and correction after reflection and generation of improvement. The results showed that on multiple datasets, training the model with fine-grained analysis information can enhance the LLM’s performance in mathematical abilities.

## Analysis

### Ablation Study

**Data Distribution Change vs. Self-correction Introduction** Since we introduced new data into the original MetaMathQA dataset, even though our data’s queries all come from MetaMathQA, there is still a possibility of changing the query distribution of the SFT data. Our error path sampling method may generate more self-correction data on more challenging problems, thus more difficult queries may appear more frequently in our SFT data. To ablate the impact of these situations on our results and to prove that our improvement comes from self-correction itself rather than the distribution changes caused by sampling, we conducted the following ablation experiment. The specific method of this experiment is that, based on the queries of the 532K self-correction data we generated, we over-sampled the original data from MetaMathQA, resulting in a total of 927K data. The distribution of queries in the over-sampled data is the same as that in our proposed 927K data.

As shown in Table 2, the experiment demonstrates that altering the distribution of the query does not enhance the model’s performance, further validating the effectiveness of our method.

**Step-Level correction vs. Instance-level correction** The method we propose is a step-level self-correction, the ad-

SFT Data	GSM8K		MATH	
	P@1	M@32	P@1	M@32
MetaMathQA	81.12	86.20	30.58	39.92
+ S <sup>3</sup> C-MATHQA- w/o. R&I	<b>81.65</b>	<b>88.10</b>	<b>32.32</b>	<b>41.00</b>
MetaMathQA dist-aligned	78.54	85.22	30.92	38.62

Table 2: Data Distribution Change vs. Self-correction Introduction. "MetaMathQA dist-aligned" represents data that has the same distribution as the query in "MetaMathQA + S<sup>3</sup>C-MATHQA", but uses the response from MetaMathQA.

vantage of which is that the model can directly correct the errors just occurred in the output process in a timely manner, thereby avoiding unnecessary error propagation. We can also generalize self-correction case-wise, that is, allowing the model to output all the content at once, and then let the model probe the generated content for errors. If errors are found, the model will automatically do self-correction, thereby producing a more accurate answer. To prove that the step-level self-correction in our work is more effective, we carried out this ablation experiment. Specifically, we saved the complete model output used to verify the error step at the error step sampling stage. According to our method, the answer paths sampled here cannot ultimately derive the correct answer. We take the entire error path as the context to generate instance-level correction data. Consistent with our proposed method, we also do not calculate loss on the erroneous reasoning path.

SFT Data	GSM8K		MATH	
	P@1	M@32	P@1	M@32
MetaMathQA	81.12	86.20	30.58	39.92
+ instance-level correction	79.98	87.26	32.22	40.60
+ S <sup>3</sup> C-MATHQA- w/o. R&I	<b>81.65</b>	<b>88.10</b>	<b>32.32</b>	<b>41.00</b>

Table 3: Step-Level correction vs. Instance-level correction. "instance-level correction" utilized the same sample size for training as S<sup>3</sup>C-MATHQA.

As shown in Table 3, experimental results indicate that while instance-level self-correction can yield some improvement, the extent of this enhancement is not as significant as that achieved with the step-level approach we employed.

### Generate from existing paths vs. Generate new paths

One of the key features of our method is to directly construct self-correction data from the existing step-by-step mathematical data. Therefore, in our correction data, apart from the erroneous step and the corresponding correction behavior, the remaining correct steps are completely consistent with the original SFT data. This allows us to seamlessly introduce the intrinsic self-correction ability into the LLMs. Currently, in exploring the mathematical capabilities of the LLM, scholars use Monte Carlo Tree Search (MCTS) for step sampling (Wang et al. 2024; Zhang et al. 2024a), which allows for more diverse steps and decouples from the ex-



<b>Question</b>	<i>Cecilia just bought a new puppy. According to her veterinarian, she has to feed the puppy 1 cup of dog food every day for the first 180 days. Then she has to feed the puppy 2 cups of dog food every day for the rest of its life. If one bag of dog food contains 110 cups, how many bags of dog food will Cecilia use in the first year?</i>
w/o. self-correction	<p>Step 1: In the first 180 days, Cecilia will feed the puppy 1 cup of dog food every day, so she will use 180 cups of dog food.</p> <p>Step 2: In the rest of the puppy’s life, she will feed the puppy 2 cups of dog food every day, so she will use 2 cups x 365 days = 730 cups of dog food.</p> <p>Step 3: In total, Cecilia will use 180 cups + 730 cups = 910 cups of dog food in the first year.</p> <p>Step 4: Since one bag of dog food contains 110 cups, Cecilia will need 910 cups / 110 cups per bag = 8.27 bags of dog food.</p> <p>Step 5: Since she cannot buy a fraction of a bag, she will need to round up to the nearest whole number.</p> <p>Step 6: Therefore, Cecilia will use 9 bags of dog food in the first year. The answer is: 9</p>
w. self-correction	<p>Step 1: In the first 180 days, Cecilia will need to feed the puppy 1 cup of dog food every day, so she will need 1 cup x 180 days = 180 cups of dog food.</p> <p>Step 2: In the rest of the puppy’s life, she will need to feed the puppy 2 cups of dog food every day, so she will need 2 cups x 365 days = 730 cups of dog food.</p> <p>Step 3: Sorry, I made a mistake. I incorrectly assumed that the rest of puppy’s life would be 365 days, which is the number of days in a year, and used that as the number of days to calculate the total cups of dog food needed for the rest of the puppy’s life. However, the problem states that the rest of its life is not just 365 days.</p> <p>To correct this, I should have used a variable to represent the number of days the puppy will live after the first 180 days, and then multiplied that variable by 2 cups to get the total cups of dog food needed for the rest of the puppy’s life. This will allow me to consider the possibility that the puppy may live for a longer or shorter period of time. Here is the updated step. In the rest of the puppy’s life, she will need to feed the puppy 2 cups of dog food every day, so she will need 2 cups x (365 days - 180 days) = 370 cups of dog food.</p> <p>Step 4: In total, Cecilia will need 180 cups + 370 cups = 550 cups of dog food in the first year.</p> <p>Step 5: Since one bag of dog food contains 110 cups, Cecilia will need 550 cups / 110 cups per bag = 5 bags of dog food in the first year. The answer is: 5.</p>

Table 4: A case study from the GSM8K test dataset. Red text denotes the process of spontaneous step-level self-correction.

isting SFT data. To demonstrate that our direct sampling of erroneous steps from existing SFT data and construction of self-correction data is more effective, we designed this ablation experiment. We implemented an MCTS to generate correct and incorrect steps for all queries in MetaMathQA, and used these steps to construct self-correction data. For a fair comparison, we also included the original MetaMathQA data and constructed MCTS self-correction data of the same scale as the data volume as S<sup>3</sup>C-MATHQA.

SFT Data	GSM8K		MATH	
	P@1	M@32	P@1	M@32
MetaMathQA	81.12	86.20	30.58	39.92
+ MCTS corrections	79.98	86.96	30.96	40.12
+ S <sup>3</sup> C-MATHQA- w/o. R&I	<b>81.65</b>	<b>88.10</b>	<b>32.32</b>	<b>41.00</b>

Table 5: Generate from existing paths vs. Generate new paths. ”MCTS corrections” utilized the same sample size for training as S<sup>3</sup>C-MATHQA.

As shown in Table 5, the experiments have demonstrated that the step-level self-correction data generated by MCTS cannot achieve the effectiveness of our method. The data produced by our method is more closely aligned with the original SFT data, enabling the introduction of spontaneous step-level self-correction capabilities within the model while preserving the original SFT data’s effectiveness.

### Case study

To more clearly demonstrate the spontaneous step-level self-correction ability generated by our method in LLMs, we provide an evaluation output of a model trained on the

DeepSeek-Math base. We present the results of greedy decoding produced by training on MetaMathQA and S<sup>3</sup>C-MATHQA respectively in Table 5. We can see that after training on S<sup>3</sup>C-MATHQA, the model spontaneously realizes its mistake in the second step where it incorrectly assumed ‘the rest of its life’ to represent 365 days, and proceeds to recalculate, arriving at the final result. This case can also show that our method does not affect the original data to the greatest extent, as the models trained on these two sets of data produce almost identical greedy outputs on the same case in the first two steps.

### Conclusion

In this paper, we introduce a novel capability for LLMs, the spontaneous step-level self-correction ability, which allows LLMs to recognize errors in their outputs in real-time and correct them simultaneously, thereby generating a more reliable answer. We propose a method for constructing training data for this capability based on wrong step sampling and build the S<sup>3</sup>C-MATHQA based on MetaMathQA. We applied SFT on both the generalist and math-specialized LLMs using the proposed S<sup>3</sup>C-MATHQA, achieving consistent and stable improvements on multiple mathematical benchmarks. Our work demonstrates that LLMs can possess a spontaneous step-level self-correction ability. There are still areas for improvement in our work, such as generating diverse steps when sampling error steps, examining the quality of reflection and improvement, etc. In the future, we will continue on this work to increase the accuracy during the self-correction process and extend this method to more reasoning tasks, truly generalizing this ability and building a better LLM reasoner.

## References

- Chen, G.; Liao, M.; Li, C.; and Fan, K. 2024. Step-Level Value Preference Optimization for Mathematical Reasoning. Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems.
- Cohen, R.; Hamri, M.; Geva, M.; and Globerson, A. 2023. LM vs LM: Detecting Factual Errors via Cross Examination. <https://arxiv.org/abs/2305.13281v1>.
- Du, Y.; Li, S.; Torralba, A.; Tenenbaum, J. B.; and Mordatch, I. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate.
- Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset.
- Jiang, Z.; Xu, F.; Gao, L.; Sun, Z.; Liu, Q.; Dwivedi-Yu, J.; Yang, Y.; Callan, J.; and Neubig, G. 2023. Active Retrieval Augmented Generation. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 7969–7992. Singapore: Association for Computational Linguistics.
- Kamoi, R.; Zhang, Y.; Zhang, N.; Han, J.; and Zhang, R. 2024. When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs.
- Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention.
- Li, C.; Dong, G.; Xue, M.; Peng, R.; Wang, X.; and Liu, D. 2024a. DotaMath: Decomposition of Thought with Code Assistance and Self-Correction for Mathematical Reasoning.
- Li, C.; Wang, W.; Hu, J.; Wei, Y.; Zheng, N.; Hu, H.; Zhang, Z.; and Peng, H. 2024b. Common 7b language models already possess strong math capabilities. *arXiv preprint arXiv:2403.04706*.
- Li, R.; Patel, T.; and Du, X. 2024. PRD: Peer Rank and Discussion Improve Large Language Model Based Evaluations.
- Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let’s Verify Step by Step.
- Liu, D.; Nassereldine, A.; Yang, Z.; Xu, C.; Hu, Y.; Li, J.; Kumar, U.; Lee, C.; and Xiong, J. 2024. Large Language Models Have Intrinsic Self-Correction Ability.
- Liu, H.; and Yao, A. C.-C. 2024. Augmenting math word problems via iterative question composing. *arXiv preprint arXiv:2401.09003*.
- Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegrefe, S.; Alon, U.; Dziri, N.; Prabhunoye, S.; Yang, Y.; Gupta, S.; Majumder, B. P.; Hermann, K.; Welleck, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback.
- Meta AI. 2024. Introducing Meta Llama 3: The Most Capable Openly Available LLM to Date. <https://ai.meta.com/blog/meta-llama-3/>.
- Mistral AI team. 2023. Mistral 7B. <https://mistral.ai/news/announcing-mistral-7b/>.
- Pan, L.; Saxon, M.; Xu, W.; Nathani, D.; Wang, X.; and Wang, W. Y. 2024. Automatically Correcting Large Language Models: *Surveying the Landscape of Diverse Automated Correction Strategies*. *Transactions of the Association for Computational Linguistics*, 12: 484–506.
- Patel, A.; Bhattamishra, S.; and Goyal, N. 2021. Are NLP Models Really Able to Solve Simple Math Word Problems?
- Paul, D.; Ismayilzada, M.; Peyrard, M.; Borges, B.; Bosse-lut, A.; West, R.; and Faltings, B. 2024. REFINER: Reasoning Feedback on Intermediate Representations. In Graham, Y.; and Purver, M., eds., *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1100–1126. St. Julian’s, Malta: Association for Computational Linguistics.
- Puerto, H.; Chubakov, T.; Zhu, X.; Madabushi, H. T.; and Gurevych, I. 2024. Fine-Tuning with Divergent Chains of Thought Boosts Reasoning Through Self-Correction in Language Models.
- Qwen Team. 2024. Introducing Qwen2-Math | Qwen. <https://qwenlm.github.io/blog/qwen2-math/>.
- Saxton, D.; Grefenstette, E.; Hill, F.; and Kohli, P. 2019. Analysing Mathematical Reasoning Abilities of Neural Models.
- Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models.
- Sun, J.; Zheng, C.; Xie, E.; Liu, Z.; Chu, R.; Qiu, J.; Xu, J.; Ding, M.; Li, H.; Geng, M.; Wu, Y.; Wang, W.; Chen, J.; Yin, Z.; Ren, X.; Fu, J.; He, J.; Yuan, W.; Liu, Q.; Liu, X.; Li, Y.; Dong, H.; Cheng, Y.; Zhang, M.; Heng, P. A.; Dai, J.; Luo, P.; Wang, J.; Wen, J.-R.; Qiu, X.; Guo, Y.; Xiong, H.; Liu, Q.; and Li, Z. 2024. A Survey of Reasoning with Foundation Models.
- Suzgun, M.; Scales, N.; Schärli, N.; Gehrmann, S.; Tay, Y.; Chung, H. W.; Chowdhery, A.; Le, Q. V.; Chi, E. H.; Zhou, D.; and Wei, J. 2022. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them.
- Thorne, J.; and Vlachos, A. 2021. Evidence-Based Factual Error Correction. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 3298–3309. Online: Association for Computational Linguistics.
- Tyen, G.; Mansoor, H.; Carbune, V.; Chen, P.; and Mak, T. 2024. LLMs Cannot Find Reasoning Errors, but Can Correct Them given the Error Location. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics ACL 2024*, 13894–13908. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics.
- Wang, P.; Li, L.; Shao, Z.; Xu, R. X.; Dai, D.; Li, Y.; Chen, D.; Wu, Y.; and Sui, Z. 2024. Math-Shepherd: Verify and Reinforce LLMs Step-by-Step without Human Annotations.



Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models.

Wang, X.; and Zhou, D. 2024. Chain-of-Thought Reasoning Without Prompting.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Xu, X.; Diao, S.; Yang, C.; and Wang, Y. 2024a. Can We Verify Step by Step for Incorrect Answer Detection? *arXiv preprint arXiv:2402.10528*.

Xu, X.; Xiao, T.; Chao, Z.; Huang, Z.; Yang, C.; and Wang, Y. 2024b. Can LLMs Solve longer Math Word Problems Better? *arXiv preprint arXiv:2405.14804*.

Ying, H.; Zhang, S.; Li, L.; Zhou, Z.; Shao, Y.; Fei, Z.; Ma, Y.; Hong, J.; Liu, K.; Wang, Z.; Wang, Y.; Wu, Z.; Li, S.; Zhou, F.; Liu, H.; Zhang, S.; Zhang, W.; Yan, H.; Qiu, X.; Wang, J.; Chen, K.; and Lin, D. 2024. InternLM-Math: Open Math Large Language Models Toward Verifiable Reasoning.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2024. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models.

Zhang, D.; Zhoubian, S.; Yue, Y.; Dong, Y.; and Tang, J. 2024a. ReST-MCTS\*: LLM Self-Training via Process Reward Guided Tree Search.

Zhang, Y.; Khalifa, M.; Logeswaran, L.; Kim, J.; Lee, M.; Lee, H.; and Wang, L. 2024b. Small Language Models Need Strong Verifiers to Self-Correct Reasoning.

Zheng, T.; Zhang, G.; Shen, T.; Liu, X.; Lin, B. Y.; Fu, J.; Chen, W.; and Yue, X. 2024. OpenCodeInterpreter: Integrating Code Generation with Execution and Refinement.

Zhou, K.; Zhang, B.; Wang, J.; Chen, Z.; Zhao, W. X.; Sha, J.; Sheng, Z.; Wang, S.; and Wen, J.-R. 2024. JiuZhang3.0: Efficiently Improving Mathematical Reasoning by Training Small Data Synthesis Models. *arXiv preprint arXiv:2405.14365*.

Zhu, X.; Li, J.; Liu, Y.; Ma, C.; and Wang, W. 2024. Key-Point-Driven Mathematical Reasoning Distillation of Large Language Model. *arXiv preprint arXiv:2407.10167*.

## Reproducibility Checklist

1. Does this paper make theoretical contributions? (yes)
  - (a) All assumptions and restrictions are stated clearly and formally. (yes)
  - (b) All novel claims are stated formally (e.g., in theorem statements). (yes)
  - (c) Proofs of all novel claims are included. (yes)
  - (d) Proof sketches or intuitions are given for complex and/or novel results. (yes)
  - (e) Appropriate citations to theoretical tools used are given. (yes)
  - (f) All theoretical claims are demonstrated empirically to hold. (yes)
  - (g) All experimental code used to eliminate or disprove claims is included. (no)
2. Does this paper rely on one or more datasets? (yes)
  - (a) A motivation is given for why the experiments are conducted on the selected datasets (yes)
  - (b) All novel datasets introduced in this paper are included in a data appendix. (no)
  - (c) All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes)
  - (d) All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes)
  - (e) All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes)
  - (f) All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisfying. (NA)
3. Does this paper include computational experiments? (yes)
  - (a) Any code required for pre-processing data is included in the appendix. (no).
  - (b) All source code required for conducting and analyzing the experiments is included in a code appendix. (no)
  - (c) All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes)
  - (d) All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes)
  - (e) If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes)
  - (f) This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes)
  - (g) This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes)
  - (h) This paper states the number of algorithm runs used to compute each reported result. (yes)
  - (i) Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (no)
  - (j) The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (no)
  - (k) This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes)
  - (l) This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (no)