MathFimer: Enhancing Mathematical Reasoning by Expanding Reasoning Steps through Fill-in-the-Middle Task

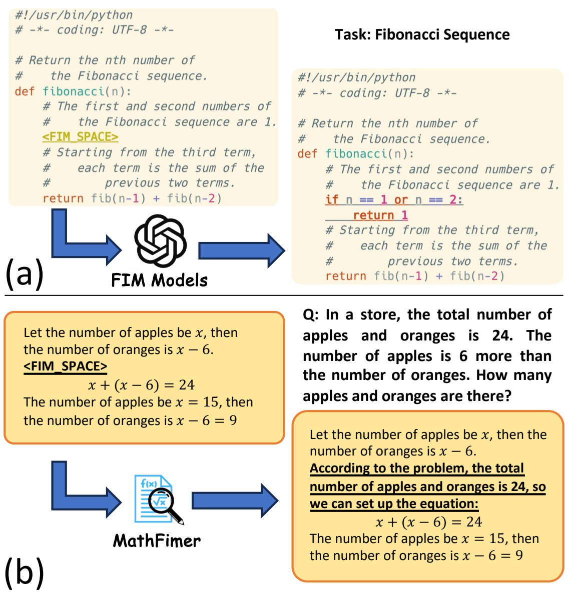
Anonymous ACL submission

Abstract

Mathematical reasoning represents a critical frontier in advancing large language models (LLMs). While step-by-step approaches have emerged as the dominant paradigm for mathematical problem-solving in LLMs, the quality of reasoning steps in training data fundamentally constrains the performance of the models. Recent studies has demonstrated that more detailed intermediate steps can enhance model performance, yet existing methods for step expansion either require more powerful external models or incur substantial computational costs. In this paper, we introduce MathFimer, a novel framework for mathematical reasoning step expansion inspired by the "Fill-in-the-middle" task from code completion. By decomposing solution chains into prefix-suffix pairs and training models to reconstruct missing intermediate steps, we develop a specialized model, MathFimer-7B, on our carefully curated NuminaMath-FIM dataset. We then apply these models to enhance existing mathematical reasoning datasets by inserting detailed intermediate steps into their solution chains, creating MathFimer-expanded versions. Through comprehensive experiments on multiple mathematical reasoning datasets, including MathInstruct, MetaMathQA and etc., we demonstrate that models trained on MathFimer-expanded data consistently outperform their counterparts trained on original data across various benchmarks such as GSM8K and MATH. Our approach offers a practical, scalable solution for enhancing mathematical reasoning capabilities in LLMs without relying on powerful external models or expensive inference procedures.

1 Introduction

Recent advances in large language models (LLMs) (OpenAI, 2023; DeepSeek-AI, 2025) have demonstrated remarkable capabilities across various reasoning tasks (Gao et al., 2024; Xu et al., 2025a), from logical deduction to complex problem-solving (Phan et al., 2025). Among these, mathematical reasoning stands as a particularly challenging frontier (Sun et al., 2024; Xu et al., 2025b), serving as a critical benchmark for evaluating an LLM's ability to perform structured, multi-step reasoning processes.



```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# Return the nth number of
#    the Fibonacci sequence.
def fibonacci(n):
    # The first and second numbers of
    #    the Fibonacci sequence are 1.
    <FIM_SPACE>
    # Starting from the third term,
    #    each term is the sum of the
    #        previous two terms.
    return fib(n-1) + fib(n-2)
```

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# Return the nth number of
#    the Fibonacci sequence.
def fibonacci(n):
    # The first and second numbers of
    #    the Fibonacci sequence are 1.
    if n == 1 or n == 2:
        return 1
    # Starting from the third term,
    #    each term is the sum of the
    #        previous two terms.
    return fib(n-1) + fib(n-2)
```

Q: In a store, the total number of apples and oranges is 24. The number of apples is 6 more than the number of oranges. How many apples and oranges are there?

Let the number of apples be x, then the number of oranges is $x-6$.
<FIM_SPACE>
$$x+(x-6)=24$$
The number of apples be $x=15$, then the number of oranges is $x-6=9$

Let the number of apples be x, then the number of oranges is $x-6$.
According to the problem, the total number of apples and oranges is 24, so we can set up the equation:
$$x+(x-6)=24$$
The number of apples be $x=15$, then the number of oranges is $x-6=9$

Figure 1: We developed MathFimer inspired by the fill-in-the-middle task in code reasoning of LLMs. Panel **(a)** demonstrates an example where the FIM model completes a given code context, while Panel **(b)** shows how MathFimer, as proposed in this paper, extends the steps of an existing step-by-step answer.

A key breakthrough in improving LLMs' mathematical reasoning capabilities has been the introduction of chain-of-thought (CoT) prompting (Wei et al., 2023), where models explicitly articulate intermediate steps in their problem-solving process. This approach has not only enhanced solution accuracy but has also provided valuable insights into the models' reasoning mechanisms. However, the effectiveness of CoT prompting raises a fundamental question: ***What characteristics of training data are crucial for developing LLMs that can gener-***

ate high-quality reasoning chains and arrive at correct mathematical solutions?

Prior research has revealed that the granularity and completeness of reasoning steps in training data significantly impact a model's reasoning capabilities (Jin et al., 2024). Models trained on more detailed step-by-step solutions tend to exhibit superior performance in mathematical reasoning tasks. This observation has led to various approaches for expanding reasoning steps in training data, including the use of stronger external models and sophisticated search algorithms like Monte Carlo Tree Search (MCTS) (Zhou et al., 2023; Wu et al., 2024; Liu et al., 2024). However, these current approaches to improving reasoning steps face three main challenges. First, they rely on using even larger models to create better steps, which creates a cycle where we constantly need bigger models to make improvements (Guan et al., 2025; Toshniwal et al., 2024). Second, these methods require substantial computing resources, particularly when using advanced techniques like MCTS to explore different reasoning paths. Third, instead of building upon existing human-verified steps, these methods often generate entirely new reasoning chains, which can introduce unexpected errors and reduce the reliability of solutions.

These limitations motivate our central research question: ***Can we develop a more efficient and reliable method for expanding reasoning steps while preserving the validity of existing human-generated solutions?*** Drawing inspiration from the "Fill-in-the-middle" task in code reasoning, where LLMs successfully complete missing code segments based on surrounding context, we propose a novel approach to this problem. Rather than generating entirely new reasoning chains, we explore whether the FIM paradigm can be adapted to supplement missing steps in existing reasoning processes or insert more detailed explanations into steps that are already sufficient.

Building on this insight, we propose MathFimer, a framework for enhancing mathematical reasoning through step expansion. We first construct NuminaMath-FIM by decomposing NuminaMath-CoT (Li et al., 2024) solutions into prefix-suffix pairs with missing intermediate steps. Using this dataset, we train a step-expansion model MathFimer-7B on math-specialized base model Qwen2.5-Math-7B (Yang et al., 2024). This model learns to supplement intermediate reasoning steps while preserving the original solution structure.

We apply MathFimer-7B to expand the reasoning steps in several existing mathematical reasoning datasets and evaluate their impact through comprehensive experiments. Our results demonstrate that training on MathFimer-expanded data consistently improves model performance across various mathematical reasoning benchmarks, including GSM8K and MATH. This improvement is observed across both general-purpose and math-specialized foundation models, with expanded datasets leading to more detailed reasoning steps and higher solution accuracy compared to the original training data.

Our main contributions are threefold:

- We propose a novel step expansion framework inspired by code completion techniques, introducing MathFimer to enhance mathematical reasoning through targeted insertion of intermediate steps in existing solutions.

- We develop and release a specialized training dataset (NuminaMath-FIM) along with a step-expansion model MathFimer-7B, providing a practical and scalable solution for improving mathematical reasoning datasets.

- Through extensive experiments across multiple benchmarks and model architectures, we demonstrate that our approach consistently improves mathematical reasoning performance, offering new insights into the relationship between step granularity and reasoning quality in LLMs.

2 Approach

In this paper, we propose a reasoning step expansion method that enhances the quality of existing data by filling in possible missing steps at the step level. This is achieved through the fill-in-the-middle (FIM) task, which supplements existing CoT data. Specifically, the work presented in this paper can be divided into two parts: the first part involves training the aforementioned FIM models, and the second part applies the trained FIM models to extend steps in existing data. Figure 2 shows an overview of our work.

2.1 FIM Model Training

The goal of this section is to train a Fill-in-the-Middle (FIM) model for mathematical reasoning tasks, which can generate the missing intermediate

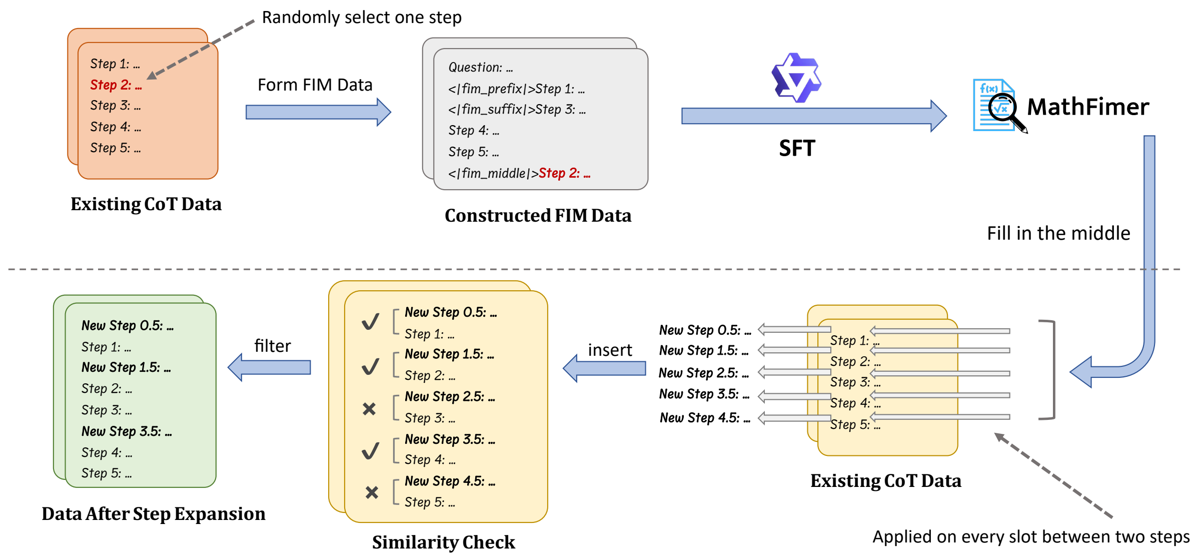


Figure 2: An overview of our work. The upper part illustrates how we construct FIM training data from existing CoT data and train FIM models, MathFimer, which works on chain-of-thought. The lower part demonstrates the process where MathFimer is used to expand the steps of existing CoT data for more detailed reasoning.

steps between a mathematical problem, its preceding steps, and its succeeding steps. This can be expressed as:

$$\mathbf{FIM}(\mathrm{Q}, \mathrm{P}, \mathrm{S}) \Rightarrow \mathrm{M}$$

where **FIM** refers to the model we are training, Q (question) represents the mathematical problem, P (prefix) refers to the preceding steps, S (suffix) refers to the succeeding steps, and M (middle) denotes the intermediate steps between P and S.

We construct the data for training the FIM model using the existing high-quality mathematical reasoning dataset, NuminaMath-CoT. NuminaMath-CoT includes mathematical reasoning data of varying difficulty levels, containing 853K mathematical question-and-answer pairs, providing us with more generalizable data.

Specifically, we first performed a step-by-step decomposition of the NuminaMath-CoT data, transforming the standard answers into individual steps. We provide the details of step decomposition in Appendix B. Then, for each case, we randomly select one step and treat all the preceding steps as the prefix and all the succeeding steps as the suffix. This can be represented as:

$$(P, S, M) = (y_{1 \ldots i-1}, y_{i+1 \ldots n}, y_i), y_i \in Y$$

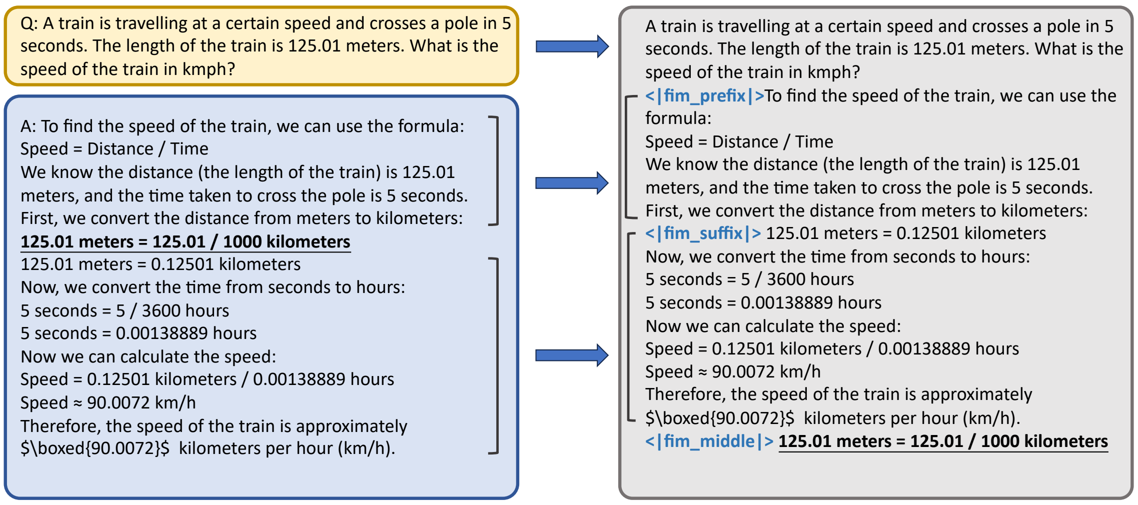
where y_i is a step randomly selected from the answer Y, which contains n steps.

For the organization format of the FIM training data, we refer to the work of Bavarian et al. (2022) and adopt the PSM(Prefix-Suffix-Middle) sequence order. We use three special tokens: `<|fim_prefix|>`, `<|fim_suffix|>`, and `<|fim_middle|>`, to construct the format for the FIM training data. An example of the FIM data construction is provided in Figure 3.

For each case in NuminaMath-CoT, we performed three rounds of random sampling as described above. As a result, for each mathematical problem, we constructed three FIM data entries, which together formed our FIM training set, NuminaMath-FIM, consisting of 2.5M training samples for FIM task. Next, we conducted SFT on a math-specialized base model, Qwen2.5-Math-7B(Yang et al., 2024). Specifically, we only computed the loss for the tokens after `<|fim_middle|>`, ultimately obtaining the FIM model MathFimer-7B for step expansion.

2.2 Expansion of Reasoning Steps

After training MathFimer-7B, we can use it to expand the reasoning steps in existing mathematical solutions. Specifically, for each pair of consecutive steps in the original solution, we perform an inference using the FIM model to generate potentially missing intermediate steps or provide more detailed reasoning between them. This can be for-



Q: A train is travelling at a certain speed and crosses a pole in 5 seconds. The length of the train is 125.01 meters. What is the speed of the train in kmph?

A: To find the speed of the train, we can use the formula:
Speed = Distance / Time
We know the distance (the length of the train) is 125.01 meters, and the time taken to cross the pole is 5 seconds.
First, we convert the distance from meters to kilometers:
125.01 meters = 125.01 / 1000 kilometers
125.01 meters = 0.12501 kilometers
Now, we convert the time from seconds to hours:
5 seconds = 5 / 3600 hours
5 seconds = 0.00138889 hours
Now we can calculate the speed:
Speed = 0.12501 kilometers / 0.00138889 hours
Speed ≈ 90.0072 km/h
Therefore, the speed of the train is approximately $\boxed{90.0072}$ kilometers per hour (km/h).

A train is travelling at a certain speed and crosses a pole in 5 seconds. The length of the train is 125.01 meters. What is the speed of the train in kmph?
<|fim_prefix|>To find the speed of the train, we can use the formula:
Speed = Distance / Time
We know the distance (the length of the train) is 125.01 meters, and the time taken to cross the pole is 5 seconds.
First, we convert the distance from meters to kilometers:
<|fim_suffix|> 125.01 meters = 0.12501 kilometers
Now, we convert the time from seconds to hours:
5 seconds = 5 / 3600 hours
5 seconds = 0.00138889 hours
Now we can calculate the speed:
Speed = 0.12501 kilometers / 0.00138889 hours
Speed ≈ 90.0072 km/h
Therefore, the speed of the train is approximately $\boxed{90.0072}$ kilometers per hour (km/h).
<|fim_middle|> **125.01 meters = 125.01 / 1000 kilometers**

Figure 3: An example of NuminaMath-FIM. The left side represents a mathematical problem and its corresponding solution from NuminaMath-CoT, while the right side shows the FIM data constructed from it. The underlined portion represents a randomly selected step from all the steps, with the blue tokens `<|fim_prefix|>`, `<|fim_suffix|>`, and `<|fim_middle|>` being three special tokens. During supervised fine-tuning, we only compute the loss for the underlined portion.

mally expressed as follows:

$$\hat{y_i} = \mathbf{FIM}(Q, y_1 \dots y_{i-1}, y_i \dots y_n)$$

where i represents each position in the original answer, n is the total number of steps in the original answer, y_i is the i-th step in the original answer, **FIM** is the trained MathFimer model, Q is the question for the sample, and $\hat{y_i}$ is the missing part generated by the FIM model between the i-th step and the subsequent steps.

In our experiments, we observed that when the original steps are already sufficiently detailed, the model tends to generate content that is very similar to the subsequent step y_i . Therefore, after the FIM model generates the supplementary step $\hat{y_i}$, we added a similarity calculation step. Specifically, we compute the sequence similarity between $\hat{y_i}$ and y_i . We set a threshold η and mark those generated steps with a similarity greater than η as `invalid`. In this paper, we set $\eta = 0.8$.

Next, we insert the steps generated by the FIM model into the original steps. Specifically, if the similarity score in the previous step is not labeled as invalid, we will insert it into the original sequence. More precisely, for any step $\hat{y_i}$ where the similarity score is less than η, we insert it between step y_{i-1} and step y_i. This insertion operation is carried out between each pair of original steps, ultimately constructing a more detailed answer with additional steps.

To evaluate the effectiveness and generalization of MathFimer-7B in expanding reasoning steps, we used it to extend the reasoning steps on several existing step-by-step reasoning datasets, including a mixture of GSM8K(Cobbe et al., 2021) and MATH(Hendrycks et al., 2021), MathInstruct-CoT(Yue et al., 2023), MetaMathQA(Yu et al., 2024), NuminaMath-CoT(Li et al., 2024), and ScaleQuestMath(Ding et al., 2024). For all datasets, we only used the training set. We conducted instruction fine-tuning experiments on multiple foundation LLMs. For general-purpose LLMs, we selected Meta-Llama-3.1-8B and Meta-Llama-3.1-70B(AI, 2024), and for math-specialized LLMs, we chose Qwen2.5-Math-7B and Qwen2.5-Math-72B(Yang et al., 2024). After instruction fine-tuning, we evaluated performance on multiple mathematical reasoning benchmarks, including GSM8K(Cobbe et al., 2021), MATH(Hendrycks et al., 2021), Math Odyssey(Fang et al., 2024), and OlympiadBench-EN(He et al., 2024).

3 Experiments

3.1 Settings

We conducted supervised instruction fine-tuning experiments on both general-purpose and math-

Dataset	FIM Model	Elementary Math		Competition Math		AVERAGE
		GSM8K	MATH	Odyssey	OB-EN	
Base Model: Meta-Llama3.1-8B						
GSM8K+MATH	/	67.55	18.32	21.59	1.78	27.31
	MathFimer-7B	$73.16_{+5.61}$	$21.84_{+3.52}$	$21.34_{-0.25}$	$2.52_{+0.74}$	$29.72_{+2.41}$
MathInstruct-CoT	/	67.78	18.74	22.11	2.37	27.75
	MathFimer-7B	$75.21_{+7.43}$	$22.90_{+4.16}$	$24.42_{+2.31}$	$3.56_{+1.19}$	$31.52_{+3.77}$
MetaMathQA	/	84.15	34.66	29.05	6.37	38.56
	MathFimer-7B	$84.69_{+0.54}$	$35.12_{+0.46}$	$28.79_{-0.26}$	$6.81_{+0.44}$	$38.85_{+0.29}$
Base Model: Meta-Llama3.1-70B						
GSM8K+MATH	/	89.23	40.22	38.30	8.74	44.12
	MathFimer-7B	$92.72_{+3.49}$	$44.36_{+4.14}$	$37.79_{-0.51}$	$12.15_{+3.41}$	$46.76_{+2.63}$
MathInstruct-CoT	/	89.31	41.96	36.50	9.19	44.24
	MathFimer-7B	$90.98_{+1.67}$	$44.72_{+2.76}$	$39.33_{+2.83}$	$12.15_{+2.96}$	$46.8_{+2.56}$
MetaMathQA	/	90.52	49.06	40.36	13.48	48.36
	MathFimer-7B	$92.57_{+2.05}$	$51.34_{+2.28}$	$38.30_{-2.06}$	$14.81_{+1.33}$	$49.26_{+0.9}$
Base Model: Qwen2.5-Math-7B						
GSM8K+MATH	/	82.71	50.90	36.25	15.41	46.32
	MathFimer-7B	$85.37_{+2.66}$	$51.92_{+1.02}$	$34.7_{-1.55}$	$14.37_{-1.04}$	$46.59_{+0.27}$
MathInstruct-CoT	/	86.28	59.80	44.22	20.59	52.72
	MathFimer-7B	$90.30_{+4.02}$	$58.86_{-0.94}$	$43.44_{-0.78}$	$20.00_{-0.59}$	$53.15_{+0.43}$
MetaMathQA	/	93.18	70.22	49.10	34.81	61.83
	MathFimer-7B	$93.10_{-0.08}$	$79.08_{+8.86}$	$52.70_{+3.6}$	$41.04_{+6.23}$	$66.48_{+4.65}$
Base Model: Qwen2.5-Math-72B						
GSM8K+MATH	/	93.25	70.74	50.13	30.37	61.12
	MathFimer-7B	$94.24_{+0.99}$	$75.16_{+4.42}$	$52.70_{+2.57}$	$36.30_{+5.93}$	$64.6_{+3.48}$
MathInstruct-CoT	/	91.36	69.26	46.27	26.67	58.39
	MathFimer-7B	$92.49_{+1.13}$	$71.70_{+2.44}$	$46.02_{-0.25}$	$29.63_{+2.96}$	$59.96_{+1.57}$
MetaMathQA	/	90.22	57.68	42.93	20.00	52.71
	MathFimer-7B	$92.95_{+2.73}$	$63.40_{+5.72}$	$47.30_{+4.37}$	$24.89_{+4.89}$	$57.14_{+4.43}$

Table 1: Our main experimental results (%) on four mathematical reasoning tasks (GSM8K, MATH, Math Odyssey and OlympiadBench-EN). The evaluation results are obtained by sampling the model 16 times with a temperature of 0.7 and calculating the average accuracy.

specialized foundation LLMs. To demonstrate the generalizability of the proposed method, we carried out experiments with different model sizes. We selected the original data before applying MathFimer-7B as the baseline for each experimental group and compared the performance improvements achieved after applying our proposed method for step expansion. In all experiments, we maintained identical training settings, only varying the data used for training. Specifically, we used Megatron-LM as the framework for SFT, with a model max_length set to 8192 and a global batch size of 128 (GSM8K+MATH datasets were set to 32 due to their smaller sample sizes, as a large batch size would result in an insufficient number of optimization steps). The learning rate for training was set to 1e-5. We packed all training samples for faster training. All SFT experiments were conducted on 64 Ascend H910B-64G.

For evaluation, we employ vLLM(Kwon et al., 2023) as the inference framework. To reduce evaluation variance, each question is sampled 16 times with a temperature setting of 0.7, and the average accuracy is calculated. To determine whether the model-generated answers are correct, we utilize LLM-as-a-judge, thereby mitigating evaluation errors caused by answer extraction and rule-based comparison. All model inferences in this study are conducted on NVIDIA A100-80G GPUs, with 1-card inference for 7B/8B models and 4-cards inference for 70B/72B models.

3.2 Main Results

We conducted our experiments on base models of different sizes, including both general-purpose and math-specialized models. Specifically, we evaluated Meta-Llama-3.1-8B, Meta-Llama-3.1-70B, Qwen2.5-Math-7B, and Qwen2.5-Math-72B. We employed the MathFimer-7B model, which was trained based on Qwen2.5-Math-7B, to perform a single round of step expansion. For comparative analysis, we selected five datasets: GSM8K+MATH, MathInstruct-CoT, MetaMathQA, NuminaMath-CoT, and ScaleQuest-Math, to examine whether step expansion via MathFimer-7B leads to improved performance on relevant mathematical reasoning benchmarks. For evaluation, we used the GSM8K, MATH, Math Odyssey, and OlympiadBench-EN datasets. Among them, GSM8K and MATH primarily assess elementary-level mathematical problems, while Math Odyssey and OlympiadBench-EN consist of competition-level mathematics questions.

We present all our main results in Table 1, and our full experimental results in Appendix D. As shown in the results, our method achieves consistent improvements across different base models and datasets. Specifically, for Meta-Llama3.1-8B, MathInstruct-CoT, when expanded using MathFimer, increases the average accuracy from 27.75% to 32.52%, yielding a 3.77 percentage point improvement. Similarly, for Qwen2.5-Math-72B, MetaMathQA, after step expansion via MathFimer, raises the average accuracy from 52.71% to 57.14%, achieving a gain of 4.43%.

Due to computational resource constraints, we perform only a single round of step expansion in our main experiment to observe the general applicability of our proposed MathFimer. However, MathFimer is capable of iterative step expansion, meaning that previously expanded steps can be further refined. We explore the scalability of step expansion in more detail in the Analysis section 4.2.

4 Analysis

4.1 Disentangling Model Effects

To disentangle the impact of our FIM methodology from model distillation effects, we conducted a systematic ablation study addressing a critical question: *To what extent do our performance gains stem from the FIM-based step expansion versus knowledge transfer from the base model?*

We designed a controlled experiment using Qwen2.5-Math-7B as the base model. We first generated distillation datasets by fine-tuning the base model on NuminaMath-CoT and using it to generate solutions for GSM8K+MATH, MathInstruct, and MetaMathQA. We then applied MathFimer-7B's step expansion to these distilled datasets to isolate the contribution of our FIM approach.

Dataset	FIM Model	GSM8K	MATH	Odyssey	OB-EN
G+M	/	67.55	18.32	21.59	1.78
	7B	73.16+5.61	21.84+3.52	21.34-0.25	2.52+0.74
G+M (distill)	/	81.58	29.32	27.76	4.44
	7B	82.41+0.83	32.6+3.28	28.19+0.43	6.59+2.15
MI-CoT	/	67.78	18.74	22.11	2.37
	7B	75.21+7.43	22.9+4.16	24.42+2.31	3.56+1.19
MI-CoT (distill)	/	83.32	35.90	32.90	6.22
	7B	86.2+2.88	37.88+1.98	32.85-0.05	8.63+2.41
MMQA	/	84.15	34.66	29.05	6.37
	7B	84.69+0.54	35.12+0.46	28.79-0.26	6.81+0.44
MMQA (distill)	/	84.23	35.18	24.42	6.81
	7B	87.57+3.34	36.98+1.8	26.16+1.74	8.11+1.3

Table 2: Performance decomposition experimental results. For the abbreviations in the table, G+M refers to GSM8K+MATH, MI-CoT refers to MathInstruct-CoT, MMQA refers to MetaMathQA, Odyssey refers to Math Odyssey, and OB-EN refers to OlympiadBench-EN. Experiments are conduct on Meta-Llama-3.1-8B.

The results in Table 2 reveal several key insights. First, while distillation alone yields substantial improvements (e.g., MATH accuracy increases from 18.32% to 29.32% for G+M), MathFimer's step expansion provides additional gains even on distilled data (+3.28%). This pattern is consistent across datasets, with MI-CoT showing similar additive benefits (+2.88% on GSM8K). The smaller magnitude of improvements on distilled data compared to original data (e.g., +3.52 % vs +3.28 % for G+M on MATH) suggests that while knowledge transfer from the base model contributes significantly to overall performance, our FIM-based step expansion provides complementary benefits through structural enhancement of reasoning chains.

4.2 Analysis of Iteration Effects

Our iterative step expansion experiments demonstrate the robust scalability of MathFimer. As shown in Table 3, each iteration of step expansion consistently improves reasoning performance across most benchmarks. Notably, on the GSM8K benchmark, MI-CoT achieves substantial gains of +7.43%, +12.43%, and +15.54% percentage points over three iterations, reaching 83.32% accuracy. Similar patterns emerge on MATH, with consistent improvements culminating in a +9.42% percentage point gain.

Dataset	Iter	GSM8K	MATH	Odyssey	OB-EN
G+M	0	67.55	18.32	21.59	1.78
	1	73.16$_{+5.61}$	21.84$_{+3.52}$	21.34$_{-0.25}$	2.52$_{+0.74}$
	2	77.03$_{+9.48}$	23.5$_{+5.18}$	21.08$_{-0.51}$	6.07$_{+4.29}$
	3	78.7$_{+11.15}$	25.54$_{+7.22}$	22.37$_{+0.78}$	6.67$_{+4.89}$
MI-CoT	0	67.78	18.74	22.11	2.37
	1	75.21$_{+7.43}$	22.9$_{+4.16}$	24.42$_{+2.31}$	3.56$_{+1.19}$
	2	80.21$_{+12.43}$	26.68$_{+7.94}$	27.76$_{+5.65}$	4.44$_{+2.07}$
	3	83.32$_{+15.54}$	28.16$_{+9.42}$	26.48$_{+4.37}$	6.67$_{+4.3}$

Table 3: Iteration effect analysis results. Model used for step expansion is MathFimer-7B.

This iterative enhancement suggests that MathFimer effectively constructs increasingly sophisticated reasoning chains, where each expansion cycle introduces valuable intermediate steps that contribute to improved reasoning capabilities. The consistent performance gains across different datasets and iteration counts validate the scalability of our approach and its ability to leverage extended reasoning chains for enhanced reasoning.

4.3 Impact of Model Scale

To investigate the relationship between model capacity and step expansion capability, we conducted a systematic comparison between MehtFimer-1.5B, MathFimer-7B and MathFimer-72B. We trained MathFimer-1.5B on Qwen2.5-Math-1.5B, MathFimer-72B on Qwen2.5-Math-72B using identical training data and hyperparameters as MathFimer-7B to ensure fair comparison.

Our experimental results, as presented in Table 4, reveal an interesting finding: the performance gap between MathFimer-7B and MathFimer-72B is notably small across all benchmarks. For instance, on GSM8K+MATH, performance is nearly identical across all three model sizes (73.09%, 73.16%, and 73.09% on GSM8K). This pattern of comparable performance persists across different datasets and evaluation metrics, suggesting that step expansion quality may not be significantly bottlenecked by model capacity. These results indicate that the step

Dataset	FIM Model	GSM8K	MATH	Odyssey	OB-EN
G+M	/	67.55	18.32	21.59	1.78
	MathFimer-1.5B	73.09$_{+5.54}$	22.76$_{+4.44}$	21.59	1.78
	MathFimer-7B	73.16$_{+5.61}$	21.84$_{+3.52}$	21.34$_{-0.25}$	2.52$_{+0.74}$
	MathFimer-72B	73.09$_{+5.54}$	21.84$_{+3.52}$	23.39$_{+1.8}$	2.07$_{+0.29}$
MI-CoT	/	67.78	18.74	22.11	2.37
	MathFimer-1.5B	73.01$_{+5.23}$	21.84$_{+3.1}$	22.62$_{+0.51}$	3.26$_{+0.89}$
	MathFimer-7B	75.21$_{+7.43}$	22.9$_{+4.16}$	24.42$_{+2.31}$	3.56$_{+1.19}$
	MathFimer-72B	73.92$_{+6.14}$	23.06$_{+4.32}$	24.68$_{+2.57}$	2.67$_{+0.3}$

Table 4: Different model size of MathFimer.

expansion task might be effectively addressed with relatively modest model sizes, potentially due to the structured nature of mathematical reasoning steps and the explicit decomposition in our approach.

4.4 Compare with Prompt-based Fill

We try to compare our method with prompt-based step expansion using general-purpose models, although we found it challenging to ensure a fair comparison. Prompt-based methods typically rely on external LLMs and repeated inference, which introduces additional computational costs and tuning complexity. In contrast, our approach leverages existing data and directly trains a FIM model, making step expansion more efficient and scalable without requiring external resources.

Dataset	FIM Model	GSM8K	MATH	Odyssey	OB-EN
G+M	/	67.55	18.32	21.59	1.78
	MathFimer-1.5B	**73.09**	**22.76**	21.59	1.78
	Llama-3.2-3B-Instruct	68.76	18.88	**22.39**	**2.52**
MI-CoT	/	67.78	18.74	22.11	2.37
	MathFimer-1.5B	**73.01**	**21.84**	**22.62**	**3.26**
	Llama-3.2-3B-Instruct	71.30	20.34	22.16	3.04

Table 5: Comparison with prompt-based fill method. For the abbreviations in the table, G+M refers to GSM8K+MATH, MI-CoT refers to MathInstruct-CoT. Experiments are conduct on Meta-Llama-3.1-8B.

To provide a more concrete comparison, we conducted an ablation experiment leveraging prompt-based fill approach. Specifically, we prompted Llama-3.2-3B-Instruct in a zero-shot manner to expand intermediate steps in original answers, and perform SFT on this expanded dataset with the same settings of former experiments. As shown in Table 5, our approach consistently outperforms prompt-based expansion(on G+M, 73.09% vs. 68.76%), both in accuracy and quality of generated intermediate reasoning. This highlights the effectiveness of training specialized FIM models for automatic step enrichment.

Type	Method	Base Model	GSM8K	MATH
Fill-in-the-middle	MathFimer(ours)	Meta-Llama3.1-8B	84.15 -> 86.58 (+2.43)	34.66 -> 37.04 (+2.38)
Preference-based	DPO (Lai et al., 2024)	Qwen2-7B	unknown	54.80 -> 55.00 (+0.20)
	Step-DPO (Lai et al., 2024)	Qwen2-7B	88.20 -> 88.50 (+0.30)	54.80 -> 55.80 (+1.00)
Rejection Sampling	RFT (Wang et al., 2024)	Mistral-7B	77.90 -> 79.00 (+1.10)	28.60 -> 29.90 (+1.30)
RL-based	PPO (Wang et al., 2024)	Mistral-7B	77.90 -> 81.80 (+3.90)	28.60 -> 31.30 (+2.70)

Table 6: Comparison with other methods for reasoning enhancement.

4.5 Evaluation of Filled Steps

To assess the correctness of the generated steps, we conducted a quantitative evaluation using Qwen2.5-Math-PRM-7B as a process reward model. Specifically, we scored the reasoning steps both before and after expansion, and the results, reported as the proportion of steps achieving a PRM scores, are summarized in Table 7. After expansion, approximately 90% of the steps achieved a score exceeding this threshold, indicating that the expanded steps exhibit a high degree of correctness.

PRM score	Proportion	0-0.2	0.2-0.4	0.4-0.6	0.6-0.8	0.8-1.0
GSM8k+MATH	raw	2.66%	2.14%	2.69%	5.50%	87.01%
	expanded	2.29%	1.87%	2.16%	4.14%	89.55%
MathInstruct-CoT	raw	2.19%	2.04%	2.48%	5.38%	87.91%
	expanded	2.21%	2.44%	3.51%	6.16%	85.67%
MetaMathQA	raw	1.37%	0.90%	1.21%	2.10%	94.42%
	expanded	1.57%	1.32%	1.65%	3.06%	92.40%

Table 7: PRM score distribution before and after step insertion with MathFimer-7B. The PRM scores range from 0 to 1.

Moreover, when comparing the PRM scores of expanded steps to those of the original reasoning chains, we observe that the correctness is largely preserved. In some cases, the expanded steps even outperform the original ones. These findings suggest that our step-expansion method does not degrade answer quality and, in fact, can improve the plausibility and completeness of intermediate reasoning. This supports the viability of our approach as a reliable step enhancement strategy for improving LLM-generated reasoning.

4.6 Compare with Other Methods

To better showcase how MathFimer compares with other methods for enhancing reasoning in LLMs, we collected several representative approaches applied to the MetaMathQA benchmark, including Direct Preference Optimization(DPO), Step-DPO, Rejection Sampling Fine-tuning(RFT) and Proximal Policy Optimization(PPO). As shown in Table 6, our method achieves comparable performance under the same experimental settings. We summarize the comparative results of MathFimer and other methods as follows.

Different optimization targets: Methods like reinforcement learning and preference optimization typically focus on optimizing for final answer correctness, while our approach specifically targets the quality and granularity of intermediate steps. These approaches are actually complementary rather than competitive - our expanded data could potentially serve as better starting points for RL.

Orthogonality and compatibility: Our FIM-based approach can actually be used alongside RL and preference optimization. The expanded steps we generate could serve as higher-quality starting points for these optimization methods, potentially leading to even better results when combined.

Computational efficiency: Our method is significantly more efficient than RL-based approaches, which require substantial computational resources for reward modeling and policy optimization. MathFimer can be applied using smaller models (even 1.5B parameters) with minimal overhead.

5 Conclusion

In this paper, we introduce the Fill-in-the-middle (FIM) paradigm into mathematical reasoning chains. We construct NuminaMath-FIM by decomposing solutions into prefix-suffix pairs, where intermediate steps are held out for reconstruction. Through training on these prefix-middle-suffix triplets, we develop MathFimer models that can effectively expand reasoning steps while preserving solution coherence. Our comprehensive experiments across multiple mathematical reasoning datasets demonstrate that MathFimer-enhanced data consistently improves model performance with relative improvements of 7.43% on GSM8K and 8.86% on MATH.

Limitations

While our MathFimer framework demonstrates promising results in enhancing mathematical reasoning through step expansion, we identify several important limitations that warrant careful consideration and future investigation.

Domain Generalization While our approach demonstrates effectiveness in mathematical reasoning, its applicability to other reasoning domains remains uncertain. The current implementation and evaluation focus exclusively on mathematical problem-solving, leaving open questions about the framework's generalizability to domains such as code reasoning, logical deduction, and commonsense reasoning, where solution structures and validation requirements may differ significantly.

Generation Reliability Our step expansion process inherently relies on model generation, introducing potential risks of error propagation. Despite overall improvements in reasoning quality, we currently lack robust mechanisms for verifying the logical consistency and mathematical correctness of inserted steps. This limitation becomes particularly critical when applying multiple iterations of step expansion, where errors could potentially accumulate.

Methodological Limitations The framework's effectiveness inherently depends on the quality of initial training data and may inherit biases from base models. Additionally, the current approach primarily focuses on expanding existing solution patterns rather than generating novel solution approaches, potentially limiting its applicability to extremely complex or unconventional problems.

References

Meta AI. 2024. The llama 3 herd of models. *arXiv preprint*.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle. *arXiv preprint*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: Process supervision without process. *arXiv preprint*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yuyang Ding, Xinyu Shi, Xiaobo Liang, Juntao Li, Qiaoming Zhu, and Min Zhang. 2024. Unleashing reasoning capability of llms via scalable question synthesis from scratch. *arXiv preprint*.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. 2024. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *arXiv preprint*.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint*.

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

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *arXiv preprint*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *arXiv preprint*.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint*.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint*.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2024. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. In *First Conference on Language Modeling*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint*.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, et al. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint*.

Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, Yue Wu, Wenhai Wang, Junsong Chen, Zhangyue Yin, Xiaozhe Ren, Jie Fu, Junxian He, Wu Yuan, Qi Liu, Xihui Liu, Yu Li, Hao Dong, Yu Cheng, Ming Zhang, Pheng Ann Heng, Jifeng Dai, Ping Luo, Jingdong Wang, Ji-Rong Wen, Xipeng Qiu, Yike Guo, Hui Xiong, Qun Liu, and Zhenguo Li. 2024. A survey of reasoning with foundation models. *arXiv preprint*.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

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

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*.

Xin Xu, Shizhe Diao, Can Yang, and Yang Wang. 2024a. Can we verify step by step for incorrect answer detection? *arXiv preprint arXiv:2402.10528*.

Xin Xu, Tong Xiao, Zitong Chao, Zhenya Huang, Can Yang, and Yang Wang. 2024b. Can llms solve longer math word problems better? *arXiv preprint arXiv:2405.14804*.

Xin Xu, Qiyun Xu, Tong Xiao, Tianhao Chen, Yuchen Yan, Jiaxin Zhang, Shizhe Diao, Can Yang, and Yang Wang. 2025a. Ugphysics: A comprehensive benchmark for undergraduate physics reasoning with large language models. *arXiv preprint arXiv:2502.00334*.

Xin Xu, Jiaxin Zhang, Tianhao Chen, Zitong Chao, Jishan Hu, and Can Yang. 2025b. Ugmathbench: A diverse and dynamic benchmark for undergraduate-level mathematical reasoning with large language models. *arXiv preprint arXiv:2501.13766*.

Yuchen Yan, Jin Jiang, Yang Liu, Yixin Cao, Xin Xu, Mengdi zhang, Xunliang Cai, and Jian Shao. 2024. S^{3}c-math: Spontaneous step-level self-correction makes large language models better mathematical reasoners. *arXiv preprint*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint*.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. 2024. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint*.

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

Yifan Zhang, Yifan Luo, Yang Yuan, and Andrew Chi-Chih Yao. 2024. Autonomous data selection with language models for mathematical texts. *arXiv preprint*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. 2024. Jiuzhang3.0: Efficiently improving mathematical reasoning by training small data synthesis models. *arXiv preprint*.

A Related Works

A.1 Mathematical Reasoning of LLMs

Mathematical reasoning is one of the advanced capabilities of large language models (LLMs). By transforming real-world mathematical problems into a sequence of sub-problems and engaging in step-by-step thinking, the model's ability to solve related mathematical tasks is enhanced (Wei et al., 2023). Currently, the mathematical reasoning ability of models can be strengthened at various stages of LLM's training. During the pre-training phase, reasoning-related knowledge texts, such as mathematical forum discussions, textbooks, and so on, are typically used for enhancement (Paster et al., 2023; Zhang et al., 2024). Additionally, a large number of synthetic step-by-step reasoning question-answer pairs are used to train the model, allowing it to learn various reasoning patterns. In the instruction fine-tuning (SFT) phase, high-quality question-answer pairs are usually employed to help the model master the pattern of step-by-step thinking, thereby enabling it to solve reasoning problems (Ding et al., 2024; Zhou et al., 2024; Xu et al., 2024b). After SFT, researchers also use techniques such as outcome supervision and process supervision to reinforce the model's mathematical reasoning process, ensuring that the model generates more accurate reasoning steps during inference (Lightman et al., 2023; Xu et al., 2024a; Wang et al., 2024; Zhang et al., 2025).

A.2 Expansion of Reasoning Steps

Just as the scaling law in model training applies, there is also a scaling law for LLMs during test-time. The former improves the model's reasoning ability by providing more training data (Hoffmann et al., 2022), while the latter increases the model's computational load during inference to enhance calculation accuracy, thereby improving performance (Brown et al., 2024; Snell et al., 2024). Expanding reasoning steps is one way to enhance the test-time computation of LLMs. By generating more detailed reasoning steps during inference, the model's reasoning performance can be improved.

There are several ways to expand reasoning steps. For example, in a training-free approach, prompts like Chain-of-Thought (Wei et al., 2023) can guide the model to perform more detailed reasoning. Using self-consistency (Wang et al., 2023b) to perform multiple reasoning paths and vote on the most consistent answers is another option. Additionally, methods like tree-search combined with a verifier can be used to select the optimal reasoning path (Chen et al., 2024; Feng et al., 2024; Guan et al., 2025). On the other hand, training-based approaches involve transforming training data into more detailed steps (Jin et al., 2024; Ying et al., 2024) or incorporating behaviors like planning (Wang et al., 2023a) and self-correction (Yan et al., 2024), which can increase the model's computation during test-time, thus improving reasoning performance.

B Details of Step Decomposition

Following previous research (Lightman et al., 2023; Wang et al., 2024), in this paper, we constructed a detailed set of rules for step segmentation. These rules primarily divide steps based on natural language sentences, while additionally handling common mathematical elements such as formulas, making the steps more reasonable.

Our decomposition approach for NuminaMath-CoT solutions into individual steps employed a combination of rule-based parsing and mathematical structure recognition:

- **Step Identification:** We primarily used explicit step markers as boundaries (e.g., "Step 1:", "First,", "Next,", "Finally,"). When these weren't present, we identified natural breakpoints in the reasoning through sentence boundaries that introduce new mathematical operations.

- **Mathematical Structure Parsing:** We parsed solutions to identify self-contained mathematical units, such as individual equation formations, algebraic manipulations, numerical computations, and logical deductions.

- **Granularity Control:** We ensured each step contained a single conceptual operation or transformation, avoiding steps that combined multiple reasoning actions.

Since our method involves expansion between steps, a more fine-grained segmentation approach allows for more reasonable expansion.

C Prompts

We provide the prompt we use to do zeroshot prompt-based fill with general-purpose LLMs in 4.

Prompt for zero-shot prompt-based Fill

I will give you a question and its answer, please make the answer more detailed. You can only insert new steps into the existing steps, DO NOT modify the existing steps. Please directly output the new answer.

Question: {question}

Answer: {answer}

Figure 4: Prompt for zero-shot prompt-based step fill.

D Full Experiments result

We provide our full experimental results on NuminaMath-CoT and ScaleQuest-Math in Table 8.

Dataset	FIM Model	Elementary Math		Competition Math		AVERAGE
		GSM8K	MATH	Odyssey	OB-EN	
Base Model: Meta-Llama3.1-8B						
GSM8K+MATH	/	67.55	18.32	21.59	1.78	27.31
	MathFimer-7B	73.16$_{+5.61}$	21.84$_{+3.52}$	21.34$_{-0.25}$	2.52$_{+0.74}$	29.72$_{+2.41}$
MathInstruct-CoT	/	67.78	18.74	22.11	2.37	27.75
	MathFimer-7B	75.21$_{+7.43}$	22.9$_{+4.16}$	24.42$_{+2.31}$	3.56$_{+1.19}$	31.52$_{+3.77}$
MetaMathQA	/	84.15	34.66	29.05	6.37	38.56
	MathFimer-7B	84.69$_{+0.54}$	35.12$_{+0.46}$	28.79$_{-0.26}$	6.81$_{+0.44}$	38.85$_{+0.29}$
NuminaMath-CoT	/	89.08	48.10	36.76	13.04	46.75
	MathFimer-7B	91.21$_{+2.13}$	50.5$_{+2.4}$	38.3$_{+1.54}$	14.52$_{+1.48}$	48.63$_{+1.89}$
ScaleQuest-Math	/	91.21	59.52	38.82	20.74	52.57
	MathFimer-7B	91.05$_{-0.16}$	59.56$_{+0.04}$	40.36$_{+1.54}$	21.63$_{+0.89}$	53.15$_{+0.58}$
Base Model: Meta-Llama3.1-70B						
GSM8K+MATH	/	89.23	40.22	38.30	8.74	44.12
	MathFimer-7B	92.72$_{+3.49}$	44.36$_{+4.14}$	37.79$_{-0.51}$	12.15$_{+3.41}$	46.76$_{+2.63}$
MathInstruct-CoT	/	89.31	41.96	36.50	9.19	44.24
	MathFimer-7B	90.98$_{+1.67}$	44.72$_{+2.76}$	39.33$_{+2.83}$	12.15$_{+2.96}$	46.8$_{+2.56}$
MetaMathQA	/	90.52	49.06	40.36	13.48	48.36
	MathFimer-7B	92.57$_{+2.05}$	51.34$_{+2.28}$	38.3$_{-2.06}$	14.81$_{+1.33}$	49.26$_{+0.9}$
NuminaMath-CoT	/	96.44	66.36	47.30	31.70	60.45
	MathFimer-7B	96.36$_{-0.08}$	67.82$_{+1.46}$	46.79$_{-0.51}$	33.33$_{+1.63}$	61.08$_{+0.63}$
ScaleQuest-Math	/	94.24	74.02	52.44	35.70	64.10
	MathFimer-7B	95$_{+0.76}$	74.42$_{+0.4}$	49.36$_{-3.08}$	36.89$_{+1.19}$	63.92$_{-0.18}$
Base Model: Qwen2.5-Math-7B						
GSM8K+MATH	/	82.71	50.90	36.25	15.41	46.32
	MathFimer-7B	85.37$_{+2.66}$	51.92$_{+1.02}$	34.7$_{-1.55}$	14.37$_{-1.04}$	46.59$_{+0.27}$
MathInstruct-CoT	/	86.28	59.80	44.22	20.59	52.72
	MathFimer-7B	90.3$_{+4.02}$	58.86$_{-0.94}$	43.44$_{-0.78}$	20$_{-0.59}$	53.15$_{+0.43}$
MetaMathQA	/	93.18	70.22	49.10	34.81	61.83
	MathFimer-7B	93.1$_{-0.08}$	79.08$_{+8.86}$	52.7$_{+3.6}$	41.04$_{+6.23}$	66.48$_{+4.65}$
NuminaMath-CoT	/	85.37	55.16	43.19	17.33	50.26
	MathFimer-7B	87.72$_{+2.35}$	53$_{-2.16}$	42.16$_{-1.03}$	16.74$_{-0.59}$	49.91$_{-0.36}$
ScaleQuest-Math	/	93.78	70.52	50.13	34.81	62.31
	MathFimer-7B	93.86$_{+0.08}$	79.38$_{+8.86}$	54.24$_{+4.11}$	40.44$_{+5.63}$	66.98$_{+4.67}$
Base Model: Qwen2.5-Math-72B						
GSM8K+MATH	/	93.25	70.74	50.13	30.37	61.12
	MathFimer-7B	94.24$_{+0.99}$	75.16$_{+4.42}$	52.7$_{+2.57}$	36.3$_{+5.93}$	64.6$_{+3.48}$
MathInstruct-CoT	/	91.36	69.26	46.27	26.67	58.39
	MathFimer-7B	92.49$_{+1.13}$	71.7$_{+2.44}$	46.02$_{-0.25}$	29.63$_{+2.96}$	59.96$_{+1.57}$
MetaMathQA	/	90.22	57.68	42.93	20.00	52.71
	MathFimer-7B	92.95$_{+2.73}$	63.4$_{+5.72}$	47.3$_{+4.37}$	24.89$_{+4.89}$	57.14$_{+4.43}$
NuminaMath-CoT	/	96.29	77.54	55.27	43.26	68.09
	MathFimer-7B	96.13$_{-0.16}$	77.4$_{-0.14}$	55.01$_{-0.26}$	44.15$_{+0.89}$	68.17$_{+0.08}$
ScaleQuest-Math	/	94.09	80.22	54.24	44.30	68.21
	MathFimer-7B	94.47$_{+0.38}$	80.82$_{+0.6}$	55.27$_{+1.03}$	43.7$_{-0.6}$	68.57$_{+0.35}$

Table 8: Our main experimental results (%) on four mathematical reasoning tasks (GSM8K, MATH, Math Odyssey and OlympiadBench-EN). The evaluation results are obtained by sampling the model 16 times with a temperature of 0.7 and calculating the average accuracy.