MathChat: Benchmarking Mathematical Reasoning and Instruction Following in Multi-Turn Interactions

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated impressive capabilities in mathematical problem-solving, particularly in single-turn question-answering formats. However, real-world scenarios often involve mathematical question-answering that requires multi-turn or interactive information exchanges, and the performance of LLMs on these tasks is still underexplored. This paper introduces MathChat, a comprehensive benchmark specifically designed to evaluate LLMs across a broader spectrum of mathematical tasks. These tasks are structured to assess the models' abilities in multi-turn interactions and open-ended generation. We evaluate the performance of various state-of-the-art LLMs on the MathChat benchmark, and we observe that while these models excel in single-turn question answering, they significantly underperform in more complex scenarios that require sustained reasoning and dialogue understanding. To address the above limitations of existing LLMs when faced with multi-turn and open-ended tasks, we develop $\text{MathChat}_{\text{sync}}$, a synthetic dialogue-based math dataset for LLM fine-tuning, focusing on improving models' interaction and instruction-following capabilities in conversations. Experimental results emphasize the need for training LLMs with diverse, conversational instruction tuning datasets like $\text{MathChat}_{\text{sync}}$. We believe this work outlines one promising direction for improving the multi-turn mathematical reasoning abilities of LLMs, thus pushing forward the development of LLMs that are more adept at interactive mathematical problem-solving and real-world applications.

1 Introduction

Mathematical reasoning has been an essential task for computers for decades (Boblow, 1968). With the explosion in Large Language Model (LLM) development (Brown et al., 2020; Achiam et al., 2023; Touvron et al., 2023a,b; Jiang et al., 2023; Team et al., 2024), mathematical reasoning has been widely recognized as a key ability for assessing these models. Most math reasoning benchmarks such as GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), MAWPS (Koncel-Kedziorski et al., 2016), ASDiv (Miao et al., 2020) and MathVista (Lu et al., 2024) feature the format of single-turn question answering (QA), where the input is a single question and the output is the solution. Recent studies (Yu et al., 2024; Yue et al., 2024; Gou et al., 2024; Luo et al., 2023; Tang et al., 2024) have scaled up such QA data by distilling synthetic data from stronger LLMs like GPT-4 (Achiam et al., 2023) or utilizing human-annotated datasets of rationales in diverse formats (Yue et al., 2024; Liang et al., 2023), continually pushing the limits of math QA accuracy. For example, on one of the most widely recognized benchmarks, GSM8K, accuracy has increased from 10.4% with a 175B-parameter model (Brown et al., 2020) to 88.2% achieved by a 7B-parameter model (Shao et al., 2024) in the past few years.

While math-specialized LLMs have shown promising progress on single-round QA benchmarks, their mathematical capabilities have not been verified in more complex scenarios. For instance, in real-world applications, such as interactive chatbots (Lee and Yeo, 2022; Jančařík et al., 2022) and problem-solving assistants (Nguyen et al., 2019), math tasks extend beyond single-round QA and require much more advanced reasoning and instruction following abilities such as dialogue understanding, diagnostic reasoning, educational feedback, etc. *Can the established math-specialized LLMs perform as well on multi-round math reasoning as they do on single-round tasks?* This question has not been comprehensively studied, although many recent studies have identified critical weaknesses of state-of-the-art LLM reasoners that could happen in multi-round interactions, such as long-context reasoning (Chen et al., 2024),

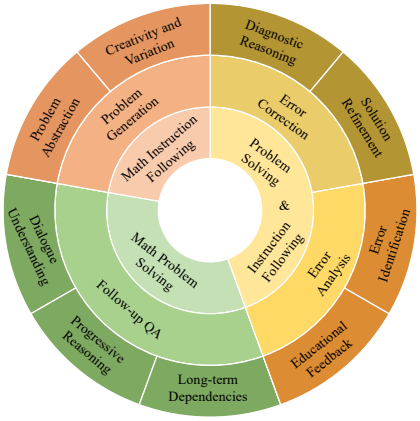

Figure 1: Taxonomy of MathChat. The inner ring represents the task categories involved in MathChat. The intermediate ring lists the evaluation tasks in MathChat. The outer ring shows the tested capabilities in our tasks beyond simple math problem solving. See detailed descriptions in Section 2.

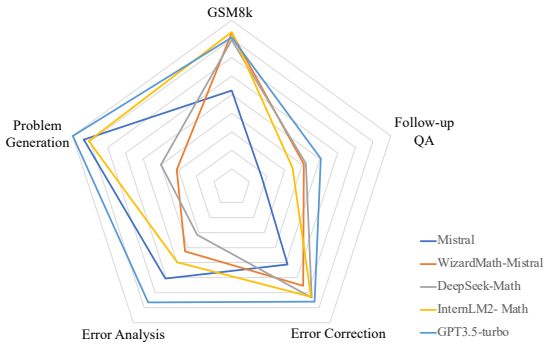

Figure 2: The performance comparison among various LLMs. Math LLMs (e.g., Deepseek-Math) have great performance on single-round QA dataset GSM8K, achieving similar performance to GPT-3.5. However, they significantly underperform GPT3.5 on MathChat, which requires more advanced reasoning abilities. We average the evaluation metrics in each task and scale all values into 0-1 for better visibility.

self-reflection ability (Huang et al., 2023), error identification (Authors, 2024), and educational content generation (Kasneci et al., 2023).

Therefore, in this paper, we advance the exploration of LLMs' mathematical reasoning abilities by introducing a new benchmark, MathChat. Figure 1 shows the hierarchical ability taxonomy derived from the tasks in MathChat (e.g., those in Figure 3), which are more advanced than the capabilities tested by single-round QA and addresses the above limitations noted in state-of-the-art LLMs.

Based on our MathChat benchmark, we find that current state-of-the-art math-specialized LLMs that are fine-tuned on extensive mathematical QA data struggle to reliably answer multi-turn questions and understand instructions that extend beyond single-round QA. Specifically, on open-ended tasks like ERROR ANALYSIS and PROBLEM GENERATION in Figure 3, the fine-tuned LLMs fail catastrophically since they can hardly understand the provided instructions. These shortcomings are perhaps unsurprising for models like MetaMath (Yu et al., 2024), which was trained exclusively on augmented question-answer pairs from single-turn math datasets GSM8K and MATH. The tasks in MathChat obviously represent a shift in distribution that challenges such models. However, even models like WizardMath (Luo et al., 2023) that were trained on more diverse data including open-ended dialogues and evolving instructions fail to achieve satisfactory performance on MathChat. We have also tried to reform our multi-turn math reasoning problem into a one-round math QA task by including all dialogue history in the question part, no significant performance improvement is observed. These results indicate potential over-tuning and data saturation towards the single-turn QA data inside current math LLMs, and also highlight a crucial open problem for the field of LLM development:

How can we empower math-focused LLMs to engage in multi-turn dialogues and follow diverse instructions without significantly compromising their problem-solving abilities?

To address the identified research challenge, we conduct an exploratory study to investigate various training data mixture strategies by leveraging extensive public math QA data, general-domain instruction tuning data, general-domain dialogue data, and our constructed synthetic dialogue-based math data ($\text{MathChat}_{\text{sync}}$). The results indicate that the model trained with $\text{MathChat}_{\text{sync}}$ significantly outperforms the baselines fine-tuned on other mixture datasets on open-ended tasks and surpasses the base LLMs on problem-solving tasks (see Section 4 for more details).

In summary, this paper makes two main contributions. First, we introduce and release a benchmark MathChat dedicated to multi-turn math reasoning and conversation, aimed at advancing the development of a more generalized reasoner and assistant in mathematical contexts—a capability that existing math-specific LLMs currently lack. Second, we demonstrate that integrating synthetic math-dialogue dataset $\text{MathChat}_{\text{sync}}$ with supervised fine-tuning (SFT) markedly enhances performance on open-ended tasks within MathChat,

without compromising much accuracy on direct problem-solving tasks. The resulting fine-tuned LLMs surpass their counterparts trained on various combinations of existing datasets. We believe this paper offers a new perspective on the evaluation of math-specific LLMs and advances the goal of developing a general math reasoning assistant.

2 MathChat

We introduce MathChat, designed to provide a deeper and more comprehensive examination of LLMs' abilities in multi-turn mathematical reasoning and instruction-following. MathChat contains four novel tasks (FOLLOW-UP QA, ERROR CORRECTION, ERROR ANALYSIS, and PROBLEM GENERATION) inspired by previous studies in the education domain that reveal the importance of following a sequence of Initiate-Response-Follow-up (Lim et al., 2020), learning from self-made errors (Heemsoth and Heinze, 2016), and posing new problems with solutions (Silver, 1994). The first two tasks focus on multi-turn mathematical problem-solving and reasoning, whereas the final two tasks evaluate the models' ability to follow mathematical instructions and respond to open-ended questions. All tasks within MathChat are sourced from the testing set of GSM8K, which we expanded using GPT-4 (we use gpt-4-0125-preview version in this paper.) to suit our specific requirements. While our benchmark is based on GPT-4, we have implemented robust quality assurance measures. We use the human-annotated GSM8k dataset as a seed for generating new tasks, ensuring that the foundation of our benchmark is rooted in high-quality data. Additionally, additional verification are involved in verifying the correctness of reference responses, especially for the first two tasks with deterministic answers. As a result, each task category contains the same number of samples as the GSM8K testing set—1,319. Table 1 shows some basic statistics of our benchmark and Figure 3 shows some examples. All prompts used to generate the task data can be found in the Appendix A.8.

Follow-up QA In this task, we form a three-round dialogue between a human user and an AI assistant. The initial round consists of a question from the original GSM8K testing dataset, with its ground truth answer. We then use GPT-4 to generate two additional questions that require a deeper understanding of the original question, and they are added to the existing dialogue to initiate new rounds of conversations. The correct answers are produced by GPT-4 and subsequently verified and revised by two different LLMs (i.e., GPT-4 and Claude). After several rounds of verification and revision by the LLMs themselves, we successfully annotate 92.4% of problems. For the remaining problems that do not meet the verification criteria, we manually create the follow-up questions and solutions to control the problem quality. After that, we randomly sample 50 problems from this task and find that all of them are correct and high-quality. During the evaluation phase, we present only the questions to the models, and the answers generated by the evaluated LLM are used in the subsequent round as shown in Figure 3, following methodologies similar to general-domain benchmarks like MT-bench (Zheng et al., 2023).

Follow-up QA Question (First Round)	46.25
Follow-up QA Question (Second Round)	34.43
Follow-up QA Question (Third Round)	41.60
Follow-up QA Answer (First Round)	52.78
Follow-up QA Answer (Second Round)	87.16
Follow-up QA Answer (Third Round)	93.84
Error Correction Wrong Attempt	54.82
Error Correction Mistake Correction	75.27
Error Analysis Wrong Attempt	66.17
Error Analysis Mistake Analysis	94.69
Problem Generation New Problem	55.37
Problem Generation New Answer	105.13

Table 1: Average lengths in MathChat benchmark. The first-round QA is essentially GSM8k testing set. We can find that our MathChat has more informative answers than GSM8k.

Error Correction In this task, we present a question to GPT-4 and instruct it to intentionally generate an incorrect answer. The incorrectness of this response is easily verified by comparing it to the correct answer in the GSM8K dataset. We use this incorrect QA pair as the dialogue history and clearly state that the solution is incorrect, and then we prompt the AI assistant to correct the wrong answer. For this task, verifying the accuracy of the ground truth answer is unnecessary because the correct final response should match the original answer provided in the source dataset GSM8k. Thus, during evaluation, we simply check whether the LLM's corrected answer aligns with the original answer. To ensure our benchmark contains a diverse set of error types, we conduct an analysis in Appendix A.6.

Error Analysis LLMs have been proven to have weak error analysis abilities (Huang et al., 2023;

<table>
<tr><th colspan="2">MathChat Examples</th></tr>
<tr><th>Follow-up QA</th><th>Error Correction</th></tr>
<tr><td><User>: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
<Assistant>: It takes 2/2=<<2/2=1>>1 …[14 words omitted] …The answer is 3.
<User> : If making the same robe requires twice as much white fiber as initially stated, how many bolts of fabric are needed now for one robe?
<Assistant>: Initially, it was stated …[61 words omitted] … The answer is 6.
<User>: If one robe requires 3 bolts of fabric and the production of each subsequent robe requires one additional bolt of fabric, how many bolts are needed for 3 robes?
<Assistant>: The first robe requires … [56 words omitted] … The answer is 12.</td><td><User>: John buys 2 pairs of shoes for each of his 3 children. They cost $60 each. How much did he pay?
<Assistant>: Let's see. If John buys 2 pairs of shoes for $60, then for 3 children, that's 3 times $60. So, he would have spent $180.
<User>: That's not quite right. Could you check your calculation again?
<Assistant>: Oh, I see the mistake now. … [38 words omitted] … The answer should be $360.</td></tr>
<tr><th>Problem Generation</th><th>Error Analysis</th></tr>
<tr><td><User>: Your task is to create a similar math problem based on a given seed problem. The generated problem should be accompanied by a detailed solution.
<Assistant>: Understood, please give me the seed problem.
<User> : Seed problem: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Seed Solution: It takes 2/2=<<2/2=1>>1 …[14 words omitted] … Answer is 3.
<Assistant>: New Problem: A bakery uses 4 pounds of flour to make a batch of 24 cupcakes and twice that amount of sugar in ounces. How many ounces of sugar does the bakery use?
Solution: First, we convert the… [42 words omitted] …The answer is 128.</td><td><User>: Please give me a math problem and I will answer that. You need to analyze my solution and correct it if I make errors.
<Assistant>: John buys 2 pairs of shoes for each of his 3 children. They cost $60 each. How much did he pay?
<User>: If he bought 2 pairs for each of his 3 children, that's 2 + 3 = 5 pairs of shoes. And if each pair costs $60, then he must have spent 5 * $60 = $300 in total.
<Assistant>: You're on the right track, but there seems to be a mistake … [76 words omitted] … The answer should be $360.</td></tr>
</table>

Figure 3: The examples of the four tasks in our MathChat benchmark. We place all tasks under a scenario of a dialogue between the user and the assistant. The part with a *italics* font is the input to the LLMs and the **highlighted** parts are generated by LLMs and to be evaluated.

Authors, 2024). The initial QA pair for the ERROR ANALYSIS task is similar to that used in the ERROR CORRECTION task, where the evaluated LLM is presented with an incorrect solution to a problem. However, the tasks diverge from the second round: while ERROR CORRECTION focuses on rectifying the answer, ERROR ANALYSIS further requires the model to first recognize that an error exists, then analyze the error and correct it. Although the two tasks share similarities in targeting errors, they pose distinctly different challenges for LLMs, especially those specialized in mathematics. These models are trained to solve problems directly, aligning well with the goal of ERROR CORRECTION. In contrast, ERROR ANALYSIS demands that the model not only understand the instructions but also identify and articulate the cause of errors before correcting them. To enhance data diversity in our benchmark, we generate a different batch of incorrect attempts for the ERROR ANALYSIS task, separate from those used in ERROR CORRECTION.

Problem Generation The final task in MathChat, Problem Generation, has been a direction of interest in both computer science and education for many years (Polozov et al., 2015; Wang et al., 2021; Zhou et al., 2023b). In this task, we provide the LLM with an original question-solution pair from the source dataset as part of the dialogue history. We then ask the LLM to create a new problem-solution pair that either delves deeper into the same topic or applies the same mathematical principles in a different context. This task is notably different from the typical mathematical QA, as it requires a model to generate questions rather than solve them. It challenges models to exhibit both creative and reasoning capabilities.

3 Evaluation of Existing LLMs on MathChat

We assess a variety of baseline LLMs using the MathChat benchmark. Detailed experimental settings such as the descriptions of baseline models are located in Appendix A.5.1.

3.1 Evaluation Metrics

For the problem-solving tasks (FOLLOW-UP QA and ERROR CORRECTION), we extract the last numerical value that appeared in the model's response and compare it to the ground truth number. This approach aligns with the evaluation metrics used in most prior studies on math word problem solving. For the instruction-following tasks (ERROR ANALYSIS and PROBLEM GENERATION), we utilize GPT-4 to assign scores from 1 to 5 (higher is better) based on three carefully designed multi-dimensional criteria, similar to (Zheng et al., 2023; Kim et al., 2024). The ERROR ANALYSIS task evaluates instruction following (IF), error diagnosis (ED), and solution accuracy (SA). The PROBLEM GENERATION task assesses IF, SA, and problem quality (PQ). A detailed description of these evalua-

tion rubrics is available in Appendix A.9. All these metrics are measured on a scale of 1 to 5. Empirically, for instruction following tasks, a score of 1 to 2 indicates the failure to understand the instructions. A score of 2 to 3 signifies a basic understanding of the instructions, but the generated responses are often wrong. A score of 3 to 4 means the model has a good understanding of the instructions and can generate corresponding answers, though mistakes may still occur sometimes. A score higher than 4 indicates a very good response, which is usually fluent and relevant, with mistakes being rare.

3.2 Prompting Template

For math-specific LLMs like MetaMath and WizardMath, which are typically trained on specific QA templates, our MathChat involves multi-turn dialogues that do not strictly adhere to the formats of their training data. To fully exploit their potential in evaluation, we test these models in two settings: (i) using the chat template[1] of their base models, and (ii) adapting their specific QA templates to include our dialogue history in the question part, i.e., reforming our multi-turn math reasoning problem to a one-round math QA task. For each task, we report results from the better-performing setting. Empirically, we find that for tasks requiring problem-solving skills, such as FOLLOW-UP QA and ERROR CORRECTION, the second setting significantly outperforms the first. However, performance is nearly identical across both settings for the instruction following tasks. These experimental evaluations reveal that solving the challenging tasks in our benchmark requires models to possess deeper understanding and comprehension abilities. For models that cannot perform well on our tasks, it is not merely due to their unfamiliarity with chat-template data.

3.3 Result Analysis and Observations

Overall, while most math-specific LLMs (except for MAmmoTH) outperform GPT-3.5-turbo only in the Round1 of Follow-up QA (see the first column in Table 2), they fall short in all other tasks (other columns in Table 2). These outcomes suggest that current math-specific models are overly tuned to single-round QA data, and the significant performance drop in multi-round and complex tasks further validates the rigor of our benchmark, challenging the models' diverse capabilities in mathematical reasoning, as illustrated in Figure 1. We further investigate the models' performance in each task:

Follow-up QA. In Rounds 2 and 3 of the FOLLOW-UP QA tasks, models face significant challenges in multi-round math reasoning, with accuracy reductions ranging from 20% to 50%. This decline indicates that while math-specific LLMs initially outperform general-purpose LLMs and even GPT-3.5-turbo in Round 1, their performance deteriorates more significantly in subsequent rounds. Theoretically, if a model maintains consistent accuracy across all dialogue rounds, with a first-round accuracy of x_1, the expected second-round accuracy would be x_1^2 due to error propagation. Interestingly, when comparing the square of the first-round accuracy (x_1^2) with the actual second-round accuracy (x_2), we observe a contrasting pattern: $x_1^2 > x_2$ for all math-specific LLMs, indicating a decline, whereas $x_1^2 < x_2$ for all other general-purpose models. This finding demonstrates that while math-specific LLMs excel at solving math problems in a single round, they show weaker progressive reasoning capabilities within dialogues.

Error Correction. In the ERROR CORRECTION task, a clear distinction also exists between math-specific LLMs and general-purpose LLMs. Notably, general LLMs exhibit higher accuracy in correcting errors than in directly solving problems (i.e., the first-round follow-up QA), whereas the reverse is true for math-specific LLMs. This adaptive behavior is evident in general-purpose LLMs but is noticeably lacking in math-specific LLMs, suggesting their weak ability to learn and reason from errors due to the over-tuning on single-round QA tasks. The difficulty of this task in our benchmark further emphasizes the need for models to go beyond single-round accuracy and develop robust error-correction abilities.

Error Analysis. The ERROR ANALYSIS task requires that models first identify errors in a given solution before proceeding to analyze and correct them. In practice, we find that math-specialized LLMs often misinterpret the task's instruction about analyzing the solution and instead simply repeat the previous answer, or just validate the incorrect solution as correct. Conversely, only GPT-3.5-turbo relatively performs well in verifying the solution and pinpointing errors. This task presents a significant challenge for open-source mathemat-

[1] https://huggingface.co/docs/transformers/main/en/chat_templating.

	Follow-up QA			Error Correction	Error Analysis			Problem Generation		
	R1*	R2	R3							
	Acc.			Acc.	IF	ED	SA	IF	PQ	SA
General-Purpose 7B LLMs:										
LLaMA2-chat	15.09	11.67	8.12	38.82	2.64	1.83	1.87	4.02	3.83	3.33
Mistral-Instruct	32.06	20.40	13.70	51.20	**3.50**	**2.82**	2.77	**4.44**	4.30	**3.80**
Gemma-it	37.60	17.65	10.57	46.15	3.07	2.05	**3.11**	3.09	3.75	2.48
Math-specialized 7B LLMs:										
MAmmoTH	66.85	32.16	19.31	54.15	2.55	1.75	1.79	2.03	1.95	2.42
MetaMath	77.18	43.98	32.16	56.30	2.51	1.26	1.34	2.28	2.32	2.35
WizardMath	83.20	44.81	**36.86**	68.22	2.62	1.81	1.95	1.53	1.54	1.60
DeepSeek-Math	79.40	**48.19**	35.70	**74.34**	1.87	1.38	1.47	1.95	1.96	2.08
InternLM2-Math	**83.80**	40.20	28.64	72.70	2.88	2.24	2.35	4.31	**4.31**	3.50
GPT-3.5-turbo	74.68	55.26	45.59	75.90	4.12	3.64	3.71	4.62	4.62	4.23
GPT-4-turbo	94.62	76.36	73.41	81.11	4.60	4.35	4.45	4.94	4.94	4.87
GPT-4o	95.68	77.67	73.03	83.09	4.84	4.60	4.68	4.91	4.94	4.82

* The first round performance is essentially the performance on the original GSM8K dataset.

Table 2: The performance of three open-sourced general-purpose LLMs, five math-specialized LLMs, and GPT-3.5-turbo on MathChat. All open-sourced models are in the size of 7B. R1, R2, and R3 denote different rounds in Follow-up QA. Evaluation metrics: Acc. (%), and others from 1 (lowest) to 5 (highest), such as IF = Instruction Following, ED = Error Diagnosis, SA = Solution Accuracy and PQ = Problem Quality. We **bold** the best performance achieved by open-sourced models.

ical LLMs, indicating a common limitation: their ability to identify and analyze errors. The high failure rate in this task also shows the challenging nature of our benchmark.

Problem Generation. The PROBLEM GENERATION task, similar to ERROR ANALYSIS, requires models to understand instructions that go beyond answering the given question. This task assesses several abilities: a model must accurately comprehend the given instruction, understand the provided problem-solution pair, and generate a new and relevant problem-solution pair. We observe that all general-purpose LLMs and only one math-specific model InternLM2-Math perform well. Other math LLMs, which are heavily optimized for problem-solving, struggle with this task. Empirically, we find that those models still consistently attempt to solve problems even when clearly instructed to create new problems. The difficulty of adapting to problem generation highlights the rigidity of current math-specific models, suggesting that these models are overly tuned to solve problems and, as a result, find it challenging to adapt to other tasks.

4 Enhancement via Supervised Fine-Tuning

Given the above challenges highlighted by our benchmark, it is natural to seek solutions to address these issues. In this section, we explore the performance improvements of general-purpose models enhanced by various supervised fine-tuning (SFT) strategies. See Appendix A.7 for case studies.

4.1 Baselines

We first build a series of Mistral 7B baseline models by applying supervised fine-tuning with existing datasets. First, **Mistral-Math** is developed to specialize Mistral-Instruct in math reasoning. This is achieved via fine-tuning the model by Arithmo (akjindal53244, 2023) compilation, which includes three existing datasets: MetaMath (Yu et al., 2024), MathInstruct (Yue et al., 2024), and Lila-OOD (Mishra et al., 2022). These dataset totally comprises about 540,000 entries. Second, **Mistral-Math-IT** is then built for enhancing the instruction following ability of Mistral-Math. We utilized the Alpaca-GPT4 dataset (Peng et al., 2023), which includes 52,000 instruction-following instances generated by GPT-4. We also use LIMA (Zhou et al., 2023a), which contains 1,000 high-quality prompts and responses from human interactions. Last, **Mistral-Math-IT-Chat** gains the ability to engage in conversation by tuning with a dialogue dataset. We subsample the Ultra-chat200k (Ding et al., 2023) to 50,000 dialogues to minimize the training workload. Empirically, we find that this subsampling does not significantly affect performance on MathChat compared to using the entire Ultrachat-200k dataset. Similarly, a series of Gemma 7B models are developed using the same SFT setting, and named following the same format.

4.2 Dialogue Dataset MathChat$_{\text{sync}}$

While the Ultra-chat200k dataset includes dialogues spanning a variety of topics, math-related

conversations should be specifically highlighted and incorporated into the SFT process. We thus introduce and release a new dataset $\text{MathChat}_{\text{sync}}$, which is created by sampling QA pairs from Arithmo as seed examples. We then tasked GPT models to engage in real-world dialogues based on these seeds, enriching the dataset with diverse and contextually relevant mathematical discussions. The details of the generation prompts are provided in the Appendix A.9. Due to budget constraints, we generated 16,132 dialogues using GPT-4 and 131,346 dialogues using GPT-3.5-turbo, resulting in a total of 147,478 dialogues in the $\text{MathChat}_{\text{sync}}$ dataset. This dataset can serve as an augmented resource during the SFT stage for future math LLMs, enabling them to engage in dialogues without compromising their ability to reason in single-round QA. Since $\text{MathChat}_{\text{sync}}$ already includes samples in forms of instruction and dialogue, Mistral and Gemma are tuned using both Arithmo and $\text{MathChat}_{\text{sync}}$, resulting in **Mistral-MathChat** and **Gemma-MathChat** models, respectively.

4.3 Result Analysis and Observations

Table 3 presents the results of two series of LLMs that have been fine-tuned from Mistral and Gemma models. The evaluation follows the same settings on MathChat as presented in Table 2. Generally, the results suggest that our method of augmenting the training corpus enhances performance across all tasks. Notably, incorporating general-purpose instruction tuning data from sources such as Alpaca and UltraChat can improve performance on mathematical tasks. This improvement may stem partly from the inclusion of mathematical content within these datasets. The addition of high-quality instruction data predominantly may also boost the LLMs' natural language comprehension, thereby enhancing their ability to solve math problems. Moreover, the model fine-tuned with our $\text{MathChat}_{\text{sync}}$ dataset demonstrates markedly superior overall performance. Appendix A.1 shows how we scale and calculate the overall score and Table 4 contains a more comprehensive comparison in terms of the overall performance. Since $\text{MathChat}_{\text{sync}}$ is created in a very simple and straightforward way, we believe that scaling up the quality and amount of such math dialogue data can bring more performance improvement, which we leave as our future work. Detailed analysis on each task follows.

Follow-up QA. When performing SFT with existing datasets, adding instruction-following, dialogue or our $\text{MathChat}_{\text{sync}}$ datasets generally enhances the performance on follow-up QA tasks. Notably, we observe that performance improvements in the second and third rounds are significantly greater compared to the initial round of the original GSM8K QA. A likely explanation is that these datasets contain longer-context QA pairs, which enable the model to reason based on the dialogue history rather than focusing predominantly on more immediate contexts.

Error Correction. Fine-tuned models exhibit better accuracy than base LLMs in error correction, yet integrating additional datasets has not markedly boosted performance. This limited improvement suggests that essential skills such as DIAGNOSTIC REASONING and SOLUTION REFINEMENT, indicated in Figure 1, are not effectively learned from the used datasets. Additionally, we observed that our $\text{MathChat}_{\text{sync}}$ data negatively affects this task. Upon examining the error cases, we discovered that models trained with $\text{MathChat}_{\text{sync}}$ indeed have a better understanding of "correcting the error", where they try to make improvements over previous incorrect attempts rather than simply making new attempts. This contrasts with models trained purely on problem-solving datasets, which tend to give completely new solutions. The lower performance of the model trained with $\text{MathChat}_{\text{sync}}$ may be attributed to the dataset's lack of manual filtering of incorrect cases. We leave the quality control problem and analysis in our future work.

Error Analysis. Similar to Error Correction, learning the ability to perform error analysis is challenging when using SFT with math QA and general instruction tuning datasets. Although the performance on this task is not exceptionally high, the inclusion of math-dialogue data in SFT has proven to be a viable method for enhancing LLMs' capabilities in error analysis. Our analysis in Figure 5 also reveals that the models fine-tuned with existing datasets (i.e., three baselines) typically affirm the correctness of previous answers and terminate their responses prematurely. In contrast, our $\text{MathChat}_{\text{sync}}$ dataset aids LLMs in understanding how to conduct error analysis.

Problem Generation. On problem generation task, we observe that the base models already have reasonable performance and SFT without

	Follow-up QA			Error Correction	Error Analysis			Problem Generation			(Scaled) Overall
	R1*	R2	R3		IF	ED	SA	IF	PQ	SA	
	Acc.			Acc.							Average
Mistral 7B Series:											
Mistral-Instruct	32.06	20.40	13.70	51.20	**3.50**	2.82	**2.77**	4.44	4.30	3.80	0.550
Mistral-Math	70.20	32.31	24.60	**70.22**	2.18	1.60	1.71	3.54	3.28	3.75	0.519
Mistral-Math-IT	70.73	40.59	27.74	69.54	2.34	1.65	1.76	4.08	3.81	4.16	0.565
Mistral-Math-IT-Chat	**71.79**	39.22	27.36	69.15	2.31	1.50	1.63	4.39	4.20	4.28	0.574
Mistral-MathChat (Ours)	71.02	**41.02**	**27.97**	67.96	3.40	**2.89**	2.67	**4.70**	**4.58**	**4.43**	**0.661**
Gemma 7B Series:											
Gemma-it	37.60	17.65	10.57	46.15	3.07	2.05	**3.11**	3.09	**3.75**	2.48	0.463
Gemma-Math	70.73	29.70	19.92	62.68	1.69	1.29	1.32	3.24	3.09	3.44	0.464
Gemma-Math-IT	72.02	43.36	32.57	62.60	1.76	1.40	1.46	3.34	3.32	3.61	0.508
Gemma-Math-IT-Chat	**74.68**	46.35	**33.64**	**63.85**	2.05	1.64	1.70	3.64	3.48	**3.99**	0.549
Gemma-MathChat (Ours)	72.14	**47.10**	32.64	61.86	**3.43**	**2.90**	2.90	**3.77**	3.72	3.74	**0.623**

Table 3: Performance of LLMs that are fine-tuned with different datasets. The best performance is **bold** and the second best is underlined for each series.

MathChat$_{sync}$ generally hurts the performance. However, a notable finding is the increase in Solution Accuracy (SA) scores following SFT, which suggests that fine-tuning on mathematical data helps the model recognize the importance of solution correctness and extend this awareness to generation tasks. Furthermore, our MathChat-enhanced SFT model records the best performance on this task, demonstrating the versatile utility of dialogue-enhanced training in mathematical contexts.

5 Related Work

Mathematical Reasoning. Recently, LLMs have demonstrated success in solving math word problems through techniques like Chain of Thought (CoT) (Wei et al., 2022; Kojima et al., 2022), Program of Thought (PoT) (Chen et al., 2023), and sampling methods (Wang et al., 2022). These studies primarily focus on improving performance via better prompting design or inference strategies. Some researchers also attempted extensive pre-training on math-related corpora to obtain foundational mathematical LLMs (Lewkowycz et al., 2022; Taylor et al., 2022; Azerbayev et al., 2024). As for the evaluation of mathematical reasoning, popular benchmarks include GSM8K, MAWPS (Koncel-Kedziorski et al., 2016), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), MathVista (Lu et al., 2024), MathVerse (Zhang et al., 2024), etc., and all of them are in single-round QA format. State-of-the-art (SOTA) models such as MetaMath (Yu et al., 2024), WizardMath, MathInstruct (Yue et al., 2024), ToRA (Gou et al., 2024), OpenMathInstruct (Toshniwal et al., 2024) augment extensive amount of math QA pairs from LLMs or humans as the additional training set to boost the performance.

Multi-Turn Dialogues in Reasoning. The advancement of dialogue capabilities in LLMs, particularly their proficiency in multi-turn interactions, has been a key focus in LLM research (Ding et al., 2023; Tunstall et al., 2023; Zheng et al., 2023). These are many studies on the intersection of math reasoning and interaction. Frieder et al. (2024) explores error types within ChatGPT-generated math solutions, including reasoning errors and miscalculations. An et al. (2023) proposes using error analysis to improve the accuracy of final solutions. CheckMate (Collins et al., 2024) a prototype platform for human-LLM interaction focused on qualitative evaluation. Our work distinguishes itself by examining an under-explored direction of open-ended multi-turn dialogues: the benchmarking and analysis of combined mathematical reasoning and instruction-following on LLMs.

6 Conclusion

This paper introduces the MathChat benchmark as a new evaluative framework for assessing the capabilities of large language models (LLMs) in mathematical problem-solving and open-ended QA within multi-turn dialogue contexts. We demonstrate that while existing math-specialized LLMs excel at single-turn question-answering tasks, they significantly struggle with more complex, open-ended tasks that require understanding and following multi-turn instructions. We also collect and release a fine-tuning dataset MathChat$_{sync}$ with math-centered dialogue interactions. LLMs trained with MathChat$_{sync}$ show marked improvements in handling complex tasks in MathChat that require higher levels of comprehension and adaptability.

Limitation

A notable limitation of our work is that the MathChat dataset is fully synthetic, generated based on GPT-4. The quality of synthetic datasets is a significant concern, and we acknowledge that this can affect the overall performance and generalization of models trained using such data. While synthetic datasets inherently lack the richness of real-world data, we have implemented two key measures to enhance quality and mitigate these concerns. First, we did not rely solely on GPT-4 for dataset generation. Instead, we augmented the high-quality, human-annotated GSM8K dataset to generate new tasks. This augmentation approach has been shown to be effective in generating new training data and benchmarks. We believe this improves the quality of the MathChat benchmark compared to datasets that are entirely synthetic. Second, for the two problem-solving tasks with deterministic answers, we employed human annotators alongside model verification processes to ensure the correctness of the model-generated responses. These measures collectively address concerns about dataset quality and contribute to a more reliable evaluation of mathematical reasoning in LLMs.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

akjindal53244. 2023. Arithmo. https://github.com/akjindal53244/Arithmo.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*.

Anonymous Authors. 2024. Evaluating mathematical reasoning of large language models: A focus on error identification and correction. In *ARR Feb Submission*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*.

DG Boblow. 1968. Natural language input for a computer problem-solving system. *Semantic Information Processing*, pages 146–226.

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

Changyu Chen, Xiting Wang, Ting-En Lin, Ang Lv, Yuchuan Wu, Xin Gao, Ji-Rong Wen, Rui Yan, and Yongbin Li. 2024. Masked thought: Simply masking partial reasoning steps can improve mathematical reasoning learning of language models. *arXiv preprint arXiv:2403.02178*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Katherine M Collins, Albert Q Jiang, Simon Frieder, Lionel Wong, Miri Zilka, Umang Bhatt, Thomas Lukasiewicz, Yuhuai Wu, Joshua B Tenenbaum, William Hart, et al. 2024. Evaluating language models for mathematics through interactions. *Proceedings of the National Academy of Sciences*, 121(24):e2318124121.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2024. Mathematical capabilities of chatgpt. *Advances in neural information processing systems*, 36.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *The Twelfth International Conference on Learning Representations*.

Tim Heemsoth and Aiso Heinze. 2016. Secondary school students learning from reflections on the rationale behind self-made errors: A field experiment. *The Journal of Experimental Education*, 84(1):98–118.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Antonín Jančařík, Jarmila Novotná, and Jakub Michal. 2022. Artificial intelligence assistant for mathematics education. In *Proceedings of the 21st European Conference on e-Learning-ECEL*, pages 143–148.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *NeurIPS*, 35:22199–22213.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *NAACL*, pages 1152–1157.

Dabae Lee and Sheunghyun Yeo. 2022. Developing an ai-based chatbot for practicing responsive teaching in mathematics. *Computers & Education*, 191:104646.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Zhenwen Liang, Dian Yu, Xiaoman Pan, Wenlin Yao, Qingkai Zeng, Xiangliang Zhang, and Dong Yu. 2023. Mint: Boosting generalization in mathematical reasoning via multi-view fine-tuning. *arXiv preprint arXiv:2307.07951*.

Woong Lim, Ji-Eun Lee, Kersti Tyson, Hee-Jeong Kim, and Jihye Kim. 2020. An integral part of facilitating mathematical discussions: Follow-up questioning. *International Journal of Science and Mathematics Education*, 18:377–398.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*.

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

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

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, et al. 2022. Lila: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5807–5832.

Hien D Nguyen, Vuong T Pham, Dung A Tran, and Trung T Le. 2019. Intelligent tutoring chatbot for solving mathematical problems in high-school. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–6. IEEE.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Oleksandr Polozov, Eleanor O'Rourke, Adam M Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. 2015. Personalized mathematical word problem generation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Edward A Silver. 1994. On mathematical problem posing. *For the learning of mathematics*, 14(1):19–28.

Zhengyang Tang, Xingxing Zhang, Benyou Wan, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Zichao Wang, Andrew Lan, and Richard Baraniuk. 2021. Math word problem generation with mathematical consistency and problem context constraints. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5986–5999.

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

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. 2024. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*.

Longhui Yu, Weisen Jiang, Han Shi, YU Jincheng, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. Mammoth: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, et al. 2024. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? *arXiv preprint arXiv:2403.14624*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, et al. 2023a. Lima: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zihao Zhou, Maizhen Ning, Qiufeng Wang, Jie Yao, Wei Wang, Xiaowei Huang, and Kaizhu Huang. 2023b. Learning by analogy: Diverse questions generation in math word problem. *arXiv preprint arXiv:2306.09064*.

A Appendix

A.1 Overall Results

To facilitate a more thorough and direct comparison across different models on our MathChat benchmark, we have formulated three comprehensive metrics based on two key aspects: problem-solving accuracy and open-ended task quality. Initially, we normalize all sub-metrics to a 0-1 scale. For problem-solving tasks, including follow-up QA and error correction, accuracies are normalized by dividing each by 100. For open-ended tasks, which

Model	Overall Average	Task Average	Category Average
LLaMA2-chat	0.424	0.418	0.384
Mistral-Instruct	0.550	0.544	0.507
Gemma-it	0.463	0.463	0.432
MAmmoTH	0.422	0.442	0.424
MetaMath	0.451	0.470	0.463
WizardMath	0.454	0.492	0.476
DeepSeek-Math	0.452	0.500	0.476
InternLM2-Math	0.617	<u>0.635</u>	<u>0.608</u>
Gemma-Math	0.464	0.491	0.463
Gemma-Math-IT	0.508	0.528	0.511
Gemma-Math-IT-Chat	0.549	0.564	0.548
Mistral-Math	0.519	0.549	0.514
Mistral-Math-IT	0.565	0.586	0.557
Mistral-Math-IT-Chat	0.574	0.593	0.565
Gemma-MathChat (Ours)	<u>0.623</u>	0.622	<u>0.608</u>
Mistral-MathChat (Ours)	**0.661**	**0.664**	**0.638**

Table 4: Overall results of 7B LLMs. The best models are **bold** and the second best is underlined.

are graded on a 1 to 5 scale, we normalize by dividing the scores by 5. We then define three metrics: 1) Overall Average: the average score of all ten sub-metrics listed in Tables 2 and 3; 2) Task Average: the average score across the four tasks; 3) Category Average: the average score of the two categories, i.e., problem-solving and open-ended QA.

The results in Table 4, based on the metrics defined above, indicate that the model with a Mistral backbone, fine-tuned with our $\text{MathChat}_{\text{sync}}$ dataset, achieves the best performance across all three metrics. This proves the effectiveness of our SFT dataset and suggests that there is still potential for improvement in math-specific LLMs.

A.2 Analysis of Answer Qualities

To evaluate the answer qualities of various models on our MathChat benchmark, we analyzed 500 outputs each from Mistral, InternLM2-Math (i.e., the best math-specialized LLM in Table 2), Mistral-Math, and our Mistral-$\text{MathChat}_{\text{sync}}$ model across tasks such as Follow-up QA, Error Analysis, and Problem Generation. We employed GPT-4 to categorize these outputs according to a predefined set of output categories. Our analysis revealed that the Mistral-$\text{MathChat}_{\text{sync}}$ models excel in tasks requiring open-ended responses, like error analysis and problem generation, while performing comparably in problem-solving tasks. The following sections detail these results:

LLMs + $\text{MathChat}_{\text{sync}}$ SFT achieves state-of-the-art accuracy in follow-up QA. As shown in Figure 4, all three math-specific models significantly outperform the original Mistral model, with our $\text{MathChat}_{\text{sync}}$ model slightly surpassing the other two, showing the strong mathematical problem solving ability is still maintained after $\text{MathChat}_{\text{sync}}$ fine-tuning.

LLMs + $\text{MathChat}_{\text{sync}}$ SFT exhibits strong error identification and correction abilities. Figure 5 shows that although the Mistral model identifies errors in mathematical problems, it falls short in offering corrections. InternLM2-Math and Math-SFT show reduced error detection capabilities due to their intensive training on straightforward math QA. In contrast, our $\text{MathChat}_{\text{sync}}$ model demonstrates a robust capacity for both identifying and correcting errors.

LLMs + $\text{MathChat}_{\text{sync}}$ SFT demonstrates superior performance in problem generation. As shown in 6, our $\text{MathChat}_{\text{sync}}$ model excels in problem generation tasks, while the other two math-specific models (InternLM2-Math and Math-SFT) struggle with instruction following and basic comprehension, highlighting the effectiveness of our $\text{MathChat}_{\text{sync}}$ fine-tuning approach.

A.3 Connection Between Few-Shot Prompting and MathChat

We conducted an experiment to compare the performance of models using few-shot prompting and

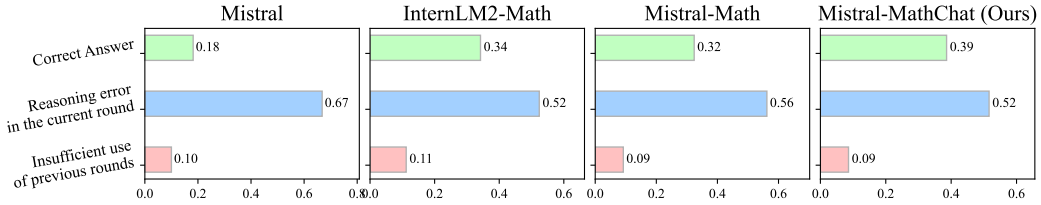

Figure 4: The Round3 answer quality in follow-up QA task.

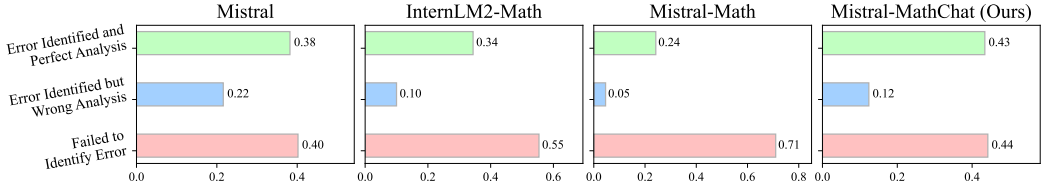

Figure 5: The answer quality in Error Analysis task.

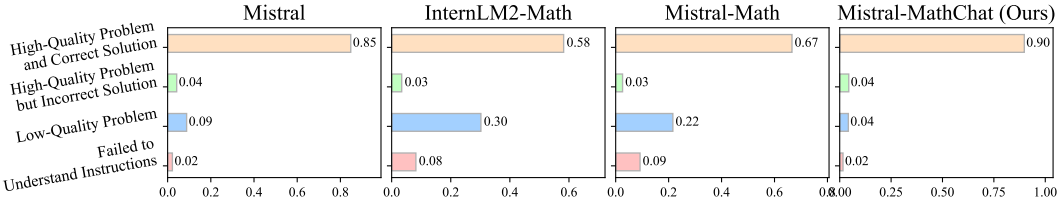

Figure 6: The answer quality in Problem Generation task.

Model	**GSM8K Zero Shot (R1)**	**GSM8K Few Shot**	**Performance Drop (R1 to R2)**	**Performance Change**
Mistral-Instruct	32.06	40.56	36.38%	+26.49%
Gemma-it	37.60	36.54	53.04%	-2.82%
MAmmoTH	66.85	59.36	51.88%	-11.20%
MetaMath	77.18	72.25	43.04%	-6.39%
WizardMath	83.20	78.99	46.15%	-5.06%
DeepSeek-Math	79.40	80.97	39.29%	+1.98%
InternLM2-Math	83.80	76.88	52.02%	-8.26%

Table 5: Comparison of GSM8K Zero Shot and Few Shot Performance, Performance Drop (R1 to R2), and Performance Change (Zero Shot to Few Shot).

MathChat on the GSM8K dataset. Our goal was to understand the differences in performance when using these two methods. In particular, we hypothesized that the additional context provided by few-shot prompting would positively impact the models' performance, similar to how MathChat requires information from the extra conversational context. The results in Table 5 show a general trend that few-shot prompting improves performance over zero-shot, but not uniformly. For instance, models like Mistral-Instruct and DeepSeek-Math experienced notable gains in performance from zero-shot to few-shot prompting, with increases of 26.49% and 1.98%, respectively. However, other models, such as MetaMath and MAmmoTH, saw a performance drop in the transition from zero-shot to few-shot, indicating that not all models leverage few-shot prompting effectively. Interestingly, models that benefit from few-shot prompting tend to have a smaller performance drop from R1 to R2. This indirectly supports your guess that there is a correlation between multi-turn reasoning and few-shot learning. We believe this is because both tasks require models to have strong long-context comprehension and reasoning abilities.

A.4 Impact of Removing Prior Conversation Context

In this experiment, we evaluate the performance of various models on the second and third rounds (R2 and R3) of questions, both with and without the context provided by the previous rounds. Specifically, we report the performance when running on R2 and R3 questions individually, without the context from R1 and R1/2 in Table 6, respectively. The results indicate that removing the prior conversation context negatively impacts all models' performance. This confirms that when a model can engage with the full conversation context in MathChat, it significantly enhances subsequent rounds of problem-solving. These findings highlight the importance of conversational context in evaluating a model's reasoning ability, further validating the effectiveness of our benchmark.

A.5 Experiment Details

A.5.1 Existing LLM Baselines

We test three general-purpose, open-source models: LLaMA2-7B-chat (Touvron et al., 2023b), Mistral-7B-Instruct (Jiang et al., 2023) and Gemma-7B-it (Team et al., 2024). Additionally, we examine five math-specific LLMs: MAmmoTH (Yue et al., 2024) create and release MathInstruct, a math problem-solving dataset including CoT-style and PoT-style annotations and perform Supervised Fine-Tuning (SFT) on various base LLMs. In this paper, we use their released MAmmoTH-Mistral-7B variant. MetaMath-Mistral-7B (Yu et al., 2024) is trained on augmented math data based on GSM8K and MATH. WizardMath-7B-v1.1 (Luo et al., 2023) utilizes both SFT and reinforcement learning from evol-instruct Feedback on math instructions. InternLM2-7B-Math (Ying et al., 2024) and DeepSeek-7B-Math (Shao et al., 2024) incorporate pre-training, SFT, and preference alignment focused on a mathematical corpus. We also present the performance of GPT-3.5-turbo, GPT-4-turbo and the latest GPT-4o.

A.5.2 Supervised Fine-tuning Implementation

We utilize Mistral-7B and Gemma-7B as our backbone models and conduct fine-tuning using Low-Rank Adaptation (LoRA) (Hu et al., 2021), with the rank set to 8 and alpha to 16. In our training process, we do not employ any specific templates or prefixes for the QA pairs but utilize the default chat template of the base models for transforming dialogues. The implementation is based on Pytorch along with the DeepSpeed (Rasley et al., 2020) Library, and the models are trained on 8 NVIDIA V100 GPUs, each with 32GB of memory. We opt for float-16 (FP16) precision to decrease memory demands and computational requirements. The fine-tuning is carried out over three epochs, with a batch size of 32 and a learning rate of 3e-5. The cumulative training time for integrating all three types of datasets amounts to approximately 72 hours, and the training time for SFT with Math + $\text{MathChat}_{\text{sync}}$ is around 30 hours.

A.6 Error Type Analysis

To ensure our benchmark contains a diverse array of error types, we randomly sampled 500 errors from our error correction task and used GPT-4 to determine their error types. The distribution of errors are shown in Figure 7: Calculation Errors were most frequent, accounting for 41.8% of the total. Reasoning Errors constituted 32.6%, indicating challenges in logical thinking and strategizing the steps required to solve problems. Conceptual Errors, making up 9.6%, pointed to difficulties in understanding underlying mathematical concepts. Ambiguity in solutions was noted in 13.8% of cases, where the provided solution is ambiguous

Model	R2 (Original)	R3 (Original)	R2 (Without R1)	R3 (Without R1/R2)
Mistral-Instruct	20.40	13.70	13.50	10.00
Gemma-it	17.65	10.57	15.16	6.60
MAmmoTH	32.16	19.31	21.75	9.25
MetaMath	43.98	32.16	30.47	17.82
WizardMath	44.81	36.86	41.70	29.80
DeepSeek-Math	48.19	35.70	48.14	35.18
InternLM2-Math	40.20	28.64	38.13	24.34

Table 6: Performance comparison of R2 and R3 with and without prior context from R1 and R1/2.

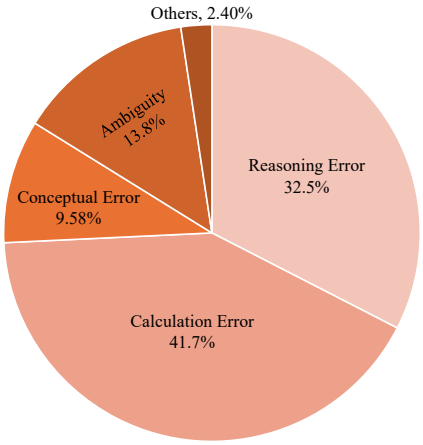

Figure 7: The distribution of error types on error correction task.

or unclear. This range of error types highlights the broad spectrum of challenges that MathChat contains, making our benchmark a robust tool for diagnosing and improving error correction and analysis ability across a variety of categories.

A.7 Case Study

Follow-up QA Figure 8 displays the responses from four LLMs on the follow-up QA task, specifically focusing on the third round of each model's response. The Mistral-instruct and Mistral-Math models, despite performing well in the first two rounds, exhibit reasoning errors in their third-round outputs. The InternLM2-Math model demonstrates a correct reasoning chain but makes a calculation error, resulting in an incorrect answer. These results indicate that the three models struggle with long-context reasoning, leading to increased errors as the number of dialogue turns rises. In contrast, our model, trained with $\text{MathChat}_{\text{sync}}$, consistently performs well and successfully solves the third-round problem.

Error Analysis Figure 9 shows the responses from four LLMs on an error analysis task. This base model - Gemma-it 7B correctly identified the calculation error regarding the sheep in Toulouse and corrected the user's response. However, the feedback was verbose and included unnecessary details, potentially leading to confusion. The models fine-tuned with existing datasets failed to recognize the error in the user's solution, incorrectly affirming the erroneous calculation. This indicates a lack of training focus on error identification and correction capabilities in these models. Trained with our MathChat dataset, the model successfully identified and corrected the calculation error in a concise and clear manner. The analysis shows the importance of targeted, dialogue-rich training datasets like $\text{MathChat}_{\text{sync}}$ in developing LLMs that are capable of effective educational interaction. The superior performance of the MathChat-trained model demonstrates its potential as a valuable tool in educational settings, offering precise and understandable corrections that can aid learning.

Problem Generation Figure 10 presents the responses from four LLMs on the problem generation task. The Mistral-Instruct and Mistral-Math models provide incorrect solutions when generating a new problem, while the InternLM2-Math model fails to follow the instructions for generating solutions to the new problem. Only our model, trained with $\text{MathChat}_{\text{sync}}$, successfully understands the instructions and delivers a high-quality solution.

A.8 Prompt for Generating Task Data

The prompts used to generate our MathChat benchmark is provided in Figure 11, 12, 13 and 14.

A.9 Prompt for Evaluating Open-ended Tasks

We provide the prompts used for evaluating the results of Error Analysis and Problem Generation in Figure 15 and 16.

Follow-up QA Case Study

Dialogue History of Mistral-instruct: (Reasoning Error)

<User>: *The expenditure of Joseph in May was $500. In June, his expenditure was $60 less. How much was his total expenditure for those two months?*

<Assistant>: **To find the total expenditure for May and June, we first need to determine Joseph's expenditure in June. We know that his expenditure in May was $500 and it was $60 less in June.\n\nJoseph's expenditure in June = May's expenditure - $60\n = $500 - $60\n = $440\n\nNow that we have the expenditure for both months, we can find the total by adding them together.\n\nTotal expenditure = May's expenditure + June's expenditure\n = $500 + $440\n = $940\n\nSo, the answer is $940.**

<User>: *Considering Joseph's expenditures, if in July he decided to increase his spending by 20% compared to June, what was his expenditure in July?*

<Assistant>: ***To find Joseph's expenditure in July, we first need to determine how much more he spent compared to June, which was a 20% increase.\n\nPercentage increase = Total increase / Original amount * 100%\n = $X / $440 * 100%\n\nWe don't know the exact value of $X (the total increase), but we can find it by setting up an equation:\n\nTotal increase = Percentage increase * Original amount\n = 20% * $440\n = 0.2 * $440\n = $88\n\nNow that we know the total increase, we can find Joseph's expenditure in July:\n\nJoseph's expenditure in July = June's expenditure + Total increase\n = $440 + $88\n = $528\n\nSo, the answer is $528.***

<User>: *Now, if Joseph decides to save 15% of his total expenditure in July for an emergency fund and the rest he plans to split equally to support three causes he believes in, how much money will each cause receive?*

<Assistant>: **First, let's calculate the amount Joseph saved for his emergency fund in July:\n\nAmount saved = Percentage saved * Total expenditure\n = 15% * $528\n = 0.15 * $528\n = $80.20 (rounded to the nearest cent)\n\nNow, we'll find the amount left for the three causes:\n\nAmount for causes = Total expenditure - Amount saved\n = $528 - $80.20\n = $447.80\n\nSince Joseph plans to split the money equally among the three causes, we can find the amount each cause will receive:\n\nAmount for one cause = Amount for causes / Number of causes\n = $447.80 / 3\n =$149.27.**

Dialogue History of InternLM2-Math: (Calculation Error)

<User>: *The expenditure of Joseph in May was $500. In June, his expenditure was $60 less. How much was his total expenditure for those two months?*

<Assistant>: **To find the total expenditure for those two months, we need to add the expenditure in May and June. \nIn June, the expenditure was $60 less than in May, so we subtract $60 from $500 to get the June expenditure. \n$500 - $60 = $440\nNow, we add the expenditure in May and June: $500 + $440 = $940\nTherefore, the total expenditure for those two months was $940.\nThe answer is 940.**

<User>: *Considering Joseph's expenditures, if in July he decided to increase his spending by 20% compared to June, what was his expenditure in July?*

<Assistant>: **If Joseph increased his spending by 20% in July compared to June, we need to find 20% of the June expenditure and add it to the June expenditure. \n20% of $440 is (20/100) * $440 = $88\nSo, the expenditure in July was $440 + $88 = $528\nTherefore, Joseph's expenditure in July was $528.\nThe answer is 528.**

<User>: *Now, if Joseph decides to save 15% of his total expenditure in July for an emergency fund and the rest he plans to split equally to support three causes he believes in, how much money will each cause receive?*

<Assistant>: **First, we need to find 15% of the July expenditure to determine how much Joseph will save for his emergency fund. \n15% of $528 is (15/100) * $528 = $79.2\nSo, Joseph will save $79.2 for his emergency fund.\nThe remaining amount that Joseph will split equally among three causes is $528 - $79.2 = $448.8\nTo find out how much each cause will receive, we divide the remaining amount by 3: $448.8 / 3 = $149 \nTherefore, each cause will receive $149.6\nThe answer is 149. (Calculation Error)**

Dialogue History of Mistral-Math: (Reasoning Error)

<User>: *The expenditure of Joseph in May was $500. In June, his expenditure was $60 less. How much was his total expenditure for those two months?*

<Assistant>: **Joseph's expenditure in May was $500.\nIn June, his expenditure was $60 less, so it was $500 - $60 = $440.\nTo find his total expenditure for those two months, we add his expenditure in May and June: $500 + $440 = $940.\nTherefore, Joseph's total expenditure for those two months was $940.**

<User>: *Considering Joseph's expenditures, if in July he decided to increase his spending by 20% compared to June, what was his expenditure in July?*

<Assistant>:**If Joseph's expenditure in June was $440, and he decided to increase his spending by 20% in July, we can calculate his expenditure in July by adding 20% of $440 to $440.\n20% of $440 is (20/100) * $440 = $88.\nSo, Joseph's expenditure in July would be $440 + $88 = $528.\nThe answer is: 528**

<User>: *Now, if Joseph decides to save 15% of his total expenditure in July for an emergency fund and the rest he plans to split equally to support three causes he believes in, how much money will each cause receive?*

<Assistant>: **If Joseph's expenditure in July was $528, and he decides to save 15% of it for his emergency fund, we can calculate the amount he saves by multiplying $528 by 15/100.\n15/100 * $528 = $84.\nSo, Joseph saves $84 for his emergency fund.\nThe remaining amount he plans to split equally among three causes is $528 - $84 = $444.\nTo find out how much each cause will receive, we divide $444 by 3.\n$444 / 3 = $148.\nTherefore, each cause will receive $148.\nThe answer is: 148**

Dialogue History of Mistral-MathChat (Ours): (Correct Solution!)

<User>: *The expenditure of Joseph in May was $500. In June, his expenditure was $60 less. How much was his total expenditure for those two months?*

<Assistant>: **Joseph's expenditure in May was $500.\nIn June, his expenditure was $60 less, so it was $500 - $60 = $440.\nTo find the total expenditure for those two months, we add the expenditures together: $500 + $440 = $940.\nThe answer is: 940**

<User>: *Considering Joseph's expenditures, if in July he decided to increase his spending by 20% compared to June, what was his expenditure in July?*

<Assistant>: **To find the expenditure in July, we first calculate the 20% increase of $440, which is $440 * 0.20 = $88. Then, we add this increase to the expenditure in June: $440 + $88 = $528. So, Joseph's expenditure in July was $528.**

<User>: *Now, if Joseph decides to save 15% of his total expenditure in July for an emergency fund and the rest he plans to split equally to support three causes he believes in, how much money will each cause receive?*

<Assistant>: **First, we calculate 15% of $528, which is $528 * 0.15 = $79.20. This amount is saved for the emergency fund. The remaining amount for the three causes is $528 - $79.20 = $448.80. Since Joseph plans to split this equally among three causes, each cause will receive $448.80 / 3 = $149.60.**

Figure 8: This figure illustrates the responses of four different LLMs in a follow-up QA task.

Error Analysis Case Study

Dialogue History:

<User>: Please give me a math problem and I will answer that. You need to analyze my solution and correct it if I make errors.

<Assistant>: Let's solve a math problem. Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. Together, how many sheep do Toulouse, Charleston, and Seattle have if Seattle has 20 sheep?

<User>: Okay, let me try to solve it. If Seattle has 20 sheep, then Charleston has 4 times 20, which is 80 sheep. Then, if Toulouse has twice as many sheep as Charleston, that would be 2 times 20, giving Toulouse 40 sheep. So, added together, they should have 20 sheep + 80 sheep + 40 sheep = 140 sheep. Is that correct? Please check my work.

Gemma-Math Response: (Fail to identify errors)

Yes, you are correct. Your solution is correct. Seattle has 20 sheep, Charleston has 80 sheep, and Toulouse has 40 sheep. Added together, they have 20 + 80 + 40 = 140 sheep. You solved the problem correctly.

Gemma-Math-IT Response: (Fail to identify errors)

Yes, you are correct! Toulouse, Charleston, and Seattle have a total of 140 sheep.

Gemma-it Response: (Too verbose)

You correctly calculated the number of sheep for Charleston, but you doubled the number of sheep for Toulouse instead of multiplying it by 2. And you did not add the number of sheep for Seattle, Charleston, and Toulouse correctly. Corrected solution: Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. Together, how many sheep do Toulouse, Charleston, and Seattle have if Seattle has 20 sheep? Okay, let me try to solve it. If Seattle has 20 sheep, then Charleston has 4 times 20, which is 80 sheep. Then, if Toulouse has twice as many sheep as Charleston, that would be 2 times 80, giving Toulouse 160 sheep. So, added together, they should have 20 sheep + 80 sheep + 160 sheep = 260 sheep

Gemma-Math-IT-Chat Response: (Fail to identify errors)

Yes, that is correct. You have correctly calculated the number of sheep each city has and then added them together to find the total number of sheep they have. So, Toulouse, Charleston, and Seattle have a total of 140 sheep.

Gemma-MathChat Response (Ours): (Correct and fluent)

You made a mistake in your calculation. Let's go through it again. If Seattle has 20 sheep, then Charleston has 4 times 20, which is 80 sheep. If Toulouse has twice as many sheep as Charleston, that would be 2 times 80, giving Toulouse 160 sheep. So, added together, they should have 20 sheep + 80 sheep + 160 sheep = 260 sheep. Please check your calculation again.

Figure 9: This figure illustrates the responses of four different LLMs in an error analysis task. The base Gemma-it model identifies and corrects the error but with verbosity, models fine-tuned with existing datasets fail to detect the error, and the $\text{MathChat}_{\text{sync}}$ model successfully identifies and corrects the error succinctly and accurately.

Problem Generation Case Study

Dialogue History

<User>: *Your task is to create a similar math problem based on a given seed problem. The generated problem should be accompanied by a detailed solution.*

<Assistant>: *Understood, please give me the seed problem.*

<User> : *Seed problem: Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 7 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?*
*Solution: He makes $10.5 selling lemons each year because 7 x 1.5 = <<7*1.5=10.5>>10.5\n He earns $7.5 each year from the lemon tree because 10.5 - 3 = <<10.5-3=7.5>>7.5\n It will take 12 years to earn enough to pay off the tree because 90 / 7.5 = <<90/7.5=12>>12\n He will make money in year 13 because 12 + 1 = <<12+1=13>>13. Answer is 13.*

Response of Mistral-instruct: (Low-quality Solution)

<Assistant>: ***Problem: Sarah is raising a rabbit. The rabbit hutch costs $120 to build. Each year, the rabbit produces 12 baby rabbits, which she can sell for $5 each. It costs $5 a year to feed and care for the rabbit. How many years will it take before she starts earning money on the rabbit farm?\n\n***

Solution: Sarah makes $11 selling baby rabbits each year because 12 x 5 = <<12*5=60>>60, but she has to subtract the cost of caring for the mother rabbit, so she makes $11 - $5 = <<11-5=6>>6\nShe earns $6 each year from the rabbit farm because 11 - 5 = 6\nIt will take 20 years to earn enough to pay off the rabbit hutch because 120 / 6 = <<120/6=20>>20\nShe will make money in year 21 because 20 + 1 = <<20+1=21>>21.

Response of InternLM2-Math: (No Solution Provided)

<Assistant>: **Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 7 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. If Carlos sells all the lemons he grows and uses the money to pay off the initial planting cost, how many years will it take for him to earn $100 in total from the lemon tree.**

Response of Mistral-Math: (Low-quality Solution)

<Assistant>: **Carlos is planting an apple tree. The tree will cost $100 to plant. Each year it will grow 8 apples, which he can sell for $2 each. It costs $4 a year to water and feed the tree.\nHow many years will it take before he starts earning money on the apple tree?**
Solution:\nHe makes $16 selling apples each year because 8 x 2 = <<8*2=16>>16\nHe earns $12 each year from the apple tree because 16 - 4 = <<16-4=12>>12\nIt will take 9 years to earn enough to pay off the apple tree because 100 / 12 = <<100/12=8.33>>8.33\nHe will make money in year 10 because 9 + 1 = <<9+1=10>>10\nThe answer is: 10

Response of Mistral-MathChat: (Good Response!)

<Assistant>: **Sara is planting a cherry tree. The tree will cost $120 to plant. Each year it will grow 10 cherries, which she can sell for $2 each. It costs $5 a year to water and feed the tree. How many years will it take before she starts earning money on the cherry tree?**
Solution: She makes $20 selling cherries each year because 10 x 2 = <<10*2=20>>20\nShe earns $15 each year from the cherry tree because 20 - 5 = <<20-5=15>>15\nIt will take 8 years to earn enough to pay off the tree because 120 / 15 = <<120/15=8>>8\nShe will make money in year 9 because 8 + 1 = <<8+1=9>>9

Figure 10: This figure illustrates the responses of four different LLMs in a problem generation task.

Prompt for Generating MathChat$_{\text{sync}}$

The prompt for generating MathChat$_{\text{sync}}$ is shown in Figure 17.

System Prompt for Follow-up QA

Objective: To create a dialogue math problem-solving session involving two characters A and B that contains two follow-up question-answerings, where A acts as the questioner and B as the responder.

Initial Round:

A (Questioner): Begins the session by asking a seed math problem.
B (Responder): Responds with the correct answer to the seed problem.

There should be exact four follow-up rounds in the response in the format of A:...\n B:...\n A:...\n B....\n.

A: Asks a follow-up question that is more challenging than the original problem, but logically connected to it. The answer should be a single value.
B: Provides a correct and detailed solution to the first follow-up question. End the response with 'The answer is \ANSWER{THE_FINAL_ANSWER}'.
Second Follow-Up Round:

A: Poses another follow-up question, further increasing in difficulty from the first follow-up, and maintaining a logical connection to the previous questions. The answer should be a single value.
B: Responds with a correct and comprehensive solution to the second follow-up question. End the response with 'The answer is \ANSWER{THE_FINAL_ANSWER}'.

Guidelines:

Complexity: Ensure that each follow-up question is more challenging than the preceding one, introducing new complexities or requiring deeper understanding.
Accuracy: B must provide accurate and mathematically sound answers.
Explanation: B should include clear explanations for each solution, demonstrating the thought process and mathematical principles used.
Clarity: Both A and B should use clear, concise language appropriate for the intended educational level of the math problems.
Creativity: A is encouraged to be creative in formulating follow-up questions that are engaging and thought-provoking.

Figure 11: The system prompt for generating FOLLOW-UP QA task data.

System Prompt for Error Correction

Objective: To create a dialogue-based interaction centered around a math problem between two characters A and B, where A presents the original problem and B attempts to solve it, initially providing an incorrect solution, and then revising it to align with the correct answer.

There should be exact four rounds in the response in the format of A:...\n B:...\n A:...\n B....\n. The dialogue should follow the structure below:

1. A starts the dialogue by presenting a math problem. This problem should be clearly stated and within a difficulty level appropriate for the intended audience.
First Attempt at Solution by B.

2. B responds to the problem with an attempt to solve it. Importantly, this first attempt must give an incorrect answer value, demonstrating a common misunderstanding or error that could be made in solving such a problem.
Request for Revision by A:

3. After B's response, A points out that the solution is incorrect and prompt B to reconsider its approach and give a new answer. No need to explain the mistake at this point. Just ask B to revise the solution.

4. Taking into account the feedback from A, B revises its solution. This time, the answer should be correct and align with the seed answer provided initially. B should also explain the reasoning behind the revised solution, highlighting the correction of the initial mistake. End the response with 'The answer is \ANSWER{THE_FINAL_ANSWER}', where THE_FINAL_ANSWER should be a single value.

Notes:
Use the seed problem and answer provided to guide the dialogue. The final answer should be the same as the seed answer.
Ensure the dialogue maintains a collaborative and educational tone throughout.
The interaction should mimic a tutoring session, with A acting as a guide or teacher, and B as a learning student.
The math problem, incorrect solution, and subsequent dialogue should be tailored to the target audience's understanding level and learning objectives.
Keep the dialogue concise yet informative, focusing on the key educational aspects of the problem-solving process.

Figure 12: The system prompt for generating ERROR CORRECTION task data.

System Prompt for Error Analysis

Objective: To create a dialogue-based interaction centered around a math problem between two characters A and B, where A presents the original problem and B attempts to solve it, initially providing an incorrect solution, and then A pointing out the error and revising it to align with the correct answer.

There should be exact three rounds in the response in the format of A:...\n B:...\n A:...\n. The dialogue should follow the structure below:

1. A starts the dialogue by presenting a math problem. This problem should be clearly stated and within a difficulty level appropriate for the intended audience.
First Attempt at Solution by B.

2. B responds to the problem with an attempt to solve it. Importantly, this first attempt must gives an incorrect answer value, demonstrating a common misunderstanding or error that could be made in solving such a problem. And request for an evaluation and analysis by A:

3. After B's response, A points out the errors inside B's attempt and corrects it into a correct solution that aligns with the given ground truth answer. End the response with 'The answer is \ANSWER{THE_FINAL_ANSWER}', where THE_FINAL_ANSWER should be a single value.

Notes:
Use the seed problem and answer provided to guide the dialogue. The final answer should be the same as the seed answer.
Ensure the dialogue maintains a collaborative and educational tone throughout.
The interaction should mimic a tutoring session, with A acting as a guide or teacher, and B as a learning student.
Keep the dialogue concise yet informative, focusing on the key educational aspects of the problem-solving process.

Figure 13: The system prompt for generating ERROR ANALYSIS task data.

System Prompt for Problem Generation

Objective: Creating new math problems based on a given seed problem. The generated problems should either explore the same topic in greater depth or apply the same mathematical principles in a different context. Each problem should be accompanied by a detailed solution that demonstrates the correct application of the mathematical principles involved.

Instructions:

1. Analyze the Seed Problem: Carefully read and understand the seed math problem provided. Identify the key mathematical concepts and principles it involves.

2. Determine the Focus: Choose whether to delve deeper into the same topic as the seed problem or to explore a different topic. In either case, ensure the new problem applies the same fundamental mathematical principles.

3. Create a New Problem: Craft a new math problem. If delving deeper into the same topic, make the problem more complex or nuanced. If exploring a different topic, find a creative way to apply the same principles. Ensure the problem is clear, concise, and mathematically sound.

4. Provide a Solution: Along with the problem, provide a step-by-step solution. The solution should be detailed enough to demonstrate the correct application of the mathematical principles involved. The final solution must be a single value instead of multiple values.

5. Ensure Variety and Creativity: When generating multiple problems, aim for a variety of contexts and applications. Avoid repetitive or overly similar problems to ensure a rich and diverse set of data.

6. Check for Accuracy and Clarity: Before finalizing, review the problem and solution for mathematical accuracy and clarity in expression. The problem should be challenging yet solvable, and the solution should be logical and well-explained.

Return the generated problem and solution in the following format without any additional information:

New Problem: [New Problem]
Solution: [Solution]

Figure 14: The system prompt for generating PROBLEM GENERATION task data.

Evaluation Prompt for Error Analysis

Evaluate the large language model's ability to identify and correct errors in an attempted solution to a math word problem. The evaluation focuses on the model's comprehension, analytical reasoning, and problem-solving capabilities within the context of mathematical problem-solving. Use the following criteria for scoring:

1. Understanding and Instruction Adherence: Assess how well the AI model understands the given task and follows the instructions. Consider whether the AI model accurately grasps the context and objectives of the task.
2. Identification of the Wrong Attempt: Evaluate the AI model's capability to identify and generate a reasonable and correct analysis of the wrong attempt. Assess the depth and accuracy of the analysis.
3. Correction of the Wrong Solution: Measure the effectiveness of the AI model in correcting the previously wrong solution into a correct one. This not only involves providing the correct answer but also explaining the correct approach to solving the problem, ensuring the explanation is mathematically sound and logically structured.

Scoring Guidelines (1-5 points):
1 point: The model shows very poor understanding and adherence to instructions, provides incorrect or irrelevant analysis of the wrong attempt, and fails to correct the solution or makes it worse.
2 points: The model demonstrates limited understanding and partial adherence to instructions, offers an inaccurate or shallow analysis of the wrong attempt, and corrects the solution with significant errors or misunderstandings.
3 points: The model shows fair understanding and adherence to instructions, provides a moderately accurate analysis of the wrong attempt with some correct elements, and corrects the solution with noticeable errors or logical flaws.
4 points: The model demonstrates good understanding and adherence to instructions, offers a well-reasoned and mostly accurate analysis of the wrong attempt, and corrects the solution effectively with minor mistakes or areas for improvement.
5 points: The model exhibits excellent understanding and strict adherence to instructions, provides a detailed and accurate analysis of the wrong attempt, and corrects the solution perfectly with a clear, logical, and mathematically sound explanation.

For each of the three aspects, provide a score along with a concise rationale for each score. Explain how the AI model's performance aligns with the evaluation criteria and contributes to effectively identifying, analyzing, and correcting the mathematical error. End the response for each score with "Score 1: {SCORE}", "Score 2: {SCORE}", and "Score 3: {SCORE}". The SCORE must be a number from 1-5.

Figure 15: The system prompt for evaluating ERROR ANALYSIS results using GPT-4.

Evaluation Prompt for Problem Generation

Evaluate the large language model's ability to generate a problem and solution based on a provided seed problem. The task assesses the model's understanding, creativity in problem generation, and accuracy in solution. Use the following criteria for scoring:

1. Understanding and Instruction Adherence: Assess whether the AI model fully grasps the task and adheres to the instructions given. Consider how well the generated problem aligns with the seed problem's topic or mathematical principles.
2. Problem Relevance and Quality: Evaluate the relevance and quality of the generated problem. Determine if it explores the same topic more deeply or applies the same mathematical principles in a different context, while also assessing the problem's complexity and ingenuity.
3. Solution Accuracy: Check the correctness of the solution provided for the generated problem. Ensure the solution is logically sound, mathematically accurate, and effectively solves the problem.

Scoring Guidelines (1-5):

1 point: The model does not understand the task, generates an unrelated problem, and provides an incorrect or irrelevant solution.
2 points: The model shows limited understanding of the task, creates a problem somewhat related to the seed problem, but the solution has significant errors or is partially irrelevant.
3 points: The model demonstrates a moderate understanding, generates a problem that is relevant and has quality, and provides a solution that is mostly correct with some errors or inconsistencies.
4 points: The model exhibits a good understanding, creates a relevant and well-constructed problem, and provides a solution that is largely correct with minor mistakes.
5 points: The model shows an excellent understanding of the task, generates a highly relevant and challenging problem, and provides a perfectly accurate and comprehensive solution.

When scoring, consider the overall effectiveness of the AI model in generating a coherent and related problem-solution pair. Provide a score for each criterion, and a rationale for each score, detailing how the AI model's performance aligns with the evaluation criteria and contributes to the quality of the generated content. End the response for each score with "Score 1: {SCORE}", "Score 2: {SCORE}", and "Score 3: {SCORE}". The SCORE must be a number from 1-5.

Figure 16: The system prompt for evaluating PROBLEM GENERATION results using GPT-4.

Prompt for MathChat$_{sync}$ Generation

You are given a seed math mathematical problem and its answer, both of which are human-annotated and 100% correct. The objective is to create a simulated multi-round conversation between a human user (<User>) and an AI assistant (<Assistant>) based on the given math problem. The conversation should explore various aspects of the problem, including but not limited to direct solutions, rephrasings, follow-up queries, solution evaluations, and requests for similar problems. The dialogue must adhere to the following guidelines:

Conversation Participants:
<User>: The human user, who will initiate queries, seek clarifications, always ask questions.
<Assistant>: The AI assistant, tasked with providing clear, accurate, and educational responses to the user's inquiries.
Dialogue Structure:
The conversation must be limited to a maximum of five rounds.
Each round consists of a question from the <User> followed by an answer from the <Assistant>.
Content Guidelines:
Make sure all the conversations are related to the math problem itself, do not include any irrelevant chat like thank you and bye-bye, etc.
The Content may involve but not limited to rephrasing the problem, seeking further explanations, deliberately giving wrong answers and asking for correction, or asking for additional, similar problems that could appear in real life.

Input Format:
Seed Problem: <problem>
Seed Answer: <answer>

Desired output format:
<User> ...
<Assistant> ...
up to five rounds of conversation
<User> ...
<Assistant> ...

Figure 17: The system prompt for generating the MathChat$_{sync}$ dataset for supervised fine-tuning.