Benchmarking Large Language Models via Random Variables

Anonymous ACL submission

Abstract

Recent studies have raised concerns about the reliability of current mathematical benchmarks, highlighting issues such as simplistic design and potential data contamination. Therefore, creating a reliable benchmark that effectively evaluates the genuine capabilities of large language models (LLMs) in mathematical reasoning remains a significant challenge. To address this, we propose **RV-Bench**, a framework for **Bench**marking LLMs via **R**andom **V**ariables in mathematical reasoning. Specifically, the background content of a random variable question (RV question) mirrors the original problem in existing benchmarks, but the variable combinations are randomized, making it "unseen" by the LLMs. Models must completely understand the question pattern of the original problem to correctly answer RV questions with various variable values. As a result, the LLM's genuine capability in mathematical reasoning is reflected by its accuracy and robustness on RV-Bench. We conducted extensive experiments on over 30 representative LLMs across more than 1000 RV questions. Our findings suggest that LLMs exhibit an imbalance in proficiency between encountered and "unseen" data domains. Proficiency generalization across similar mathematical reasoning tasks is verified to be limited by accuracy and robustness, but it can still be enhanced through test-time scaling.

1 Introduction

The emergence of LLMs has led to impressive results across a wide range of applications, including machine translation (Zhang et al., 2023; Zhu et al., 2024), text summarization (Liu et al., 2024d), and question answering (Kamalloo et al., 2023). With advancements in LLMs' reasoning capabilities (Huang and Chang, 2023), their performance on complex tasks such as code generation (Chen et al., 2021; Hong et al., 2024b), planning (Huang et al., 2024a), and, particularly, mathematical reasoning and computation (Romera-Paredes et al., 2024), has become a central focus within the LLM research community (Zhao et al., 2023). As this area continues to be a prominent focus of LLM research, numerous promising methods (Luo et al., 2023; Xu et al., 2024b) and benchmarks (Fang et al., 2024) have been developed to enhance LLMs' performance on mathematical tasks.

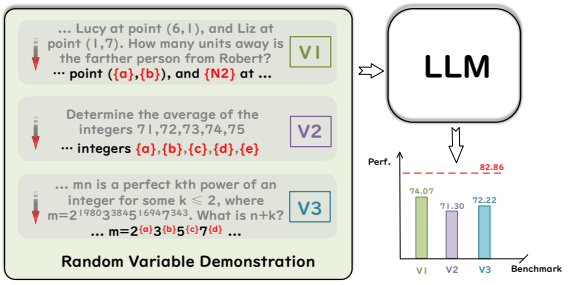

Figure 1: When mathematical problems are presented with identical content but various variable combinations, LLMs experience a significant drop in accuracy. This discrepancy poses challenges in evaluating the genuine capabilities of LLMs in mathematical reasoning.

However, **are existing benchmarks of LLMs in mathematical reasoning truly reliable?** Figure 1 illustrates a discrepancy within the well-known MATH (Hendrycks et al., 2021b) dataset. In our pilot experiments, powerful LLMs like GPT-4o (Achiam et al., 2023) perform well on MATH problems but still experience a significant drop in accuracy when answering questions with the same content but various variable combinations (Mirzadeh et al., 2024), as detailed in Section 4.2. This discrepancy raises two potential concerns about the existing evaluation framework: **1) The existing benchmarks' design may be overly simplistic for contemporary LLMs,** as they typically only evaluate performance on fixed-variable problems. The LLMs may not genuinely understand the problem but instead "guess" the correct answer (Dong et al., 2024); **2) The problems in widely-used benchmarks might be encountered**

by LLMs through data contamination during training, allowing the models to achieve high accuracy solely on the original problems (Ni et al., 2024) but not completely understand the inherent question pattern. These concerns present a significant challenge in evaluating the genuine capabilities of LLMs (Deng et al., 2024).

The advanced study presents an in-depth analysis about the probabilistic modeling of LLMs during reasoning process obscures the fact that they are not genuinely capable of formal reasoning (Shi et al., 2023; Jiang et al., 2024). Additionally, potential issues such as data leakage and overfitting during LLM training are also being studied (Xu et al., 2024a). Given that mathematics is a foundational topic applicable across a wide range of semantic scenarios, the increasing popularity and prevalence of math datasets like GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) raise the risk of potential data contamination. Although recent studies on contamination detection (Chern et al., 2024; Ni et al., 2024) can signal unreliable results, they fail to reflect the genuine performance of LLMs, as data contamination occurs during the training phase and remains non-intervenable (Kapoor and Narayanan, 2023).

The phenomenon mentioned above raises a critical issue: current benchmarks may not truly reflect the performance of LLMs (Balloccu et al., 2024; Mirzadeh et al., 2024). In this context, **effectively benchmarking LLMs for genuine mathematical reasoning capabilities remains a significant challenge.** To address this, we propose **RV-Bench** in this paper as a solution for benchmarking LLMs in mathematical reasoning using **random variable questions (RV questions)**, which provide a variety of "unseen" questions with specific variable combinations. This novel study introduces a genuine and effective benchmark, RV-Bench, that addresses concerns 1) and 2) mentioned above.

Specifically, we construct question functions based on the original problems from two selected mathematical data sources: MATH (Hendrycks et al., 2021b) and LeetCode-Math[1]. These functions generate instantiated questions with random variables and their corresponding answers. The RV questions are then collected to evaluate LLMs. Unlike existing math benchmarks (Cobbe et al., 2021; Hendrycks et al., 2021b), RV-Bench includes questions with a wide range of variable combinations, rather than fixed ones. Furthermore, RV-Bench provides "unseen" questions, allowing LLMs to demonstrate their genuine performance even if the model has been exposed to certain benchmarks (Mirzadeh et al., 2024). To achieve high accuracy in RV-Bench, an LLM must completely understand the inherent question pattern to correctly answer the RV questions, effectively reflecting its genuine capabilities in mathematical reasoning.

Our contributions are listed following:

- We construct the **RV-Bench** leaderboard, providing a comprehensive evaluation of the genuine mathematical reasoning capabilities of LLMs. A macroscopic analysis of RV-Bench quantifies the degree of question pattern understanding in existing LLMs.
- By comparing the LLMs' accuracy on RV questions with their accuracy on the corresponding original problems, we observe a significant accuracy drop, indicating the unreliability of existing benchmark designs.
- Combining the accuracy of LLMs in RV-Bench and their robustness during accuracy dropping, we propose our findings that: LLMs obtain certain proficiency in mathematical reasoning from their training, which is partially dependent on the data domain. The generalization of this proficiency is limited but can be elicited by test-time scaling.

2 RV-Bench

Figure 2 provides workflows for RV-Bench in both the MATH (Hendrycks et al., 2021b) and LeetCode data sources. In this section, we introduce the process of constructing RV-Bench, from the data sources to the annotation process.

2.1 Data Sources

The proposed RV-Bench comprises question functions constructed based two selective data sources: the MATH (Hendrycks et al., 2021b) test set and the LeetCode-Math branch.

MATH is a well-known dataset that covers 12,500 challenging mathematics problems targeted at high-school mathematics competitions. It includes annotated answers with full step-by-step reasoning processes, frequently used to enhance LLMs' capabilities in complex mathematical reasoning. Following the processing settings of the

[1] https://leetcode.com/problem-list/math/

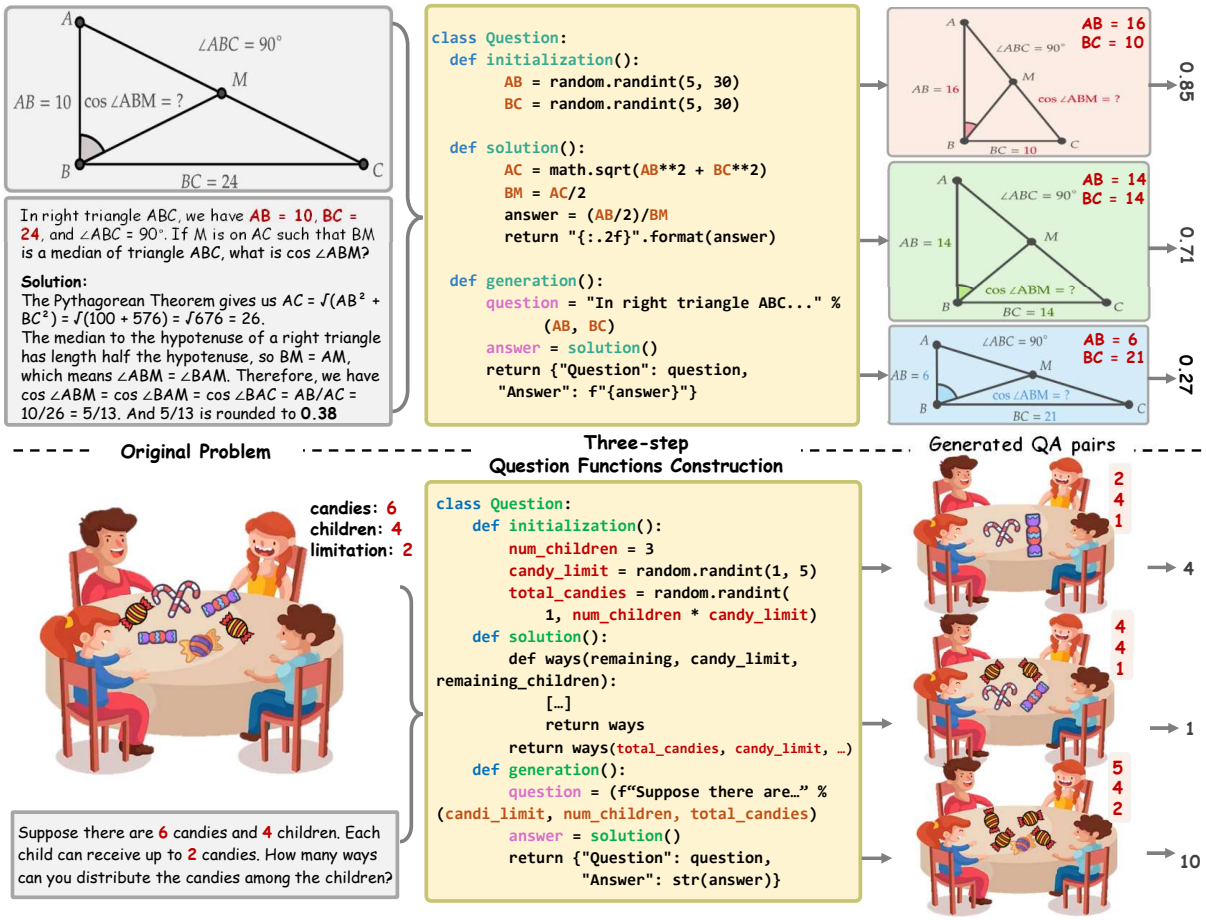

Figure 2: Two workflows of RV-Bench are shown for the MATH (above) and LeetCode (below) data sources. The question function `Question` comprises three modules: Initially, the `initialization` module randomizes a variable combination. Subsequently, the `solution` module returns a corresponding answer. Finally, the `generation` module outputs the instantiated question along with the corresponding answer, forming a QA pair for RV-Bench.

MATH-split in another widely-adopted dataset, PRM800K (Lightman et al., 2024), we construct 120 question functions by uniformly selecting problems at random from the test split for the following process. LeetCode is a widely recognized platform providing coding and algorithmic problems for users to practice coding skills (Coignion et al., 2024). As a branch of the coding questions, LeetCode-Math includes algorithmic questions whose content is designed based on mathematical reasoning and computation. Our motivation for selecting LeetCode as one of our data sources is derived from its original focus on coding problems. By transforming these problems into mathematical formats, we ensure these problems are unlikely to have been encountered during the LLMs' training. Through a careful review of all candidate solutions for each question, we construct 130 question functions by reformatting the question content with random variables, selected at random. Consequently, the question functions in RV-Bench are randomly sampled from their respective data sources, maintaining similar distributions of difficulty, type, and bias as the original problems.

2.2 Question Functions

As illustrated in Figure 2, a complete question function consists of three modules: `initialization`, `solution`, and `generation`. These modules are responsible for instantiating the random variables, solving any RV questions, and generating the QA pairs for RV-Bench, respectively. The construction details of RV-Bench is given in Appendix B.

3 Experimental Setups

Datasets. After the calibration and post-filtering processes, RV-Bench consists of 230 question functions, with 115 derived from the MATH test set and 115 from LeetCode-Math. To compare the LLM performance on the random variables and

the original setting, we sample the corresponding original problems of question functions from the MATH test set (**MATH-Sp**) and LeetCode-Math (**LeetCode-Sp**). Specifically, we define the problem that instantiated with the variable combinations and the answer provided by examples found below the official description as the original problem for LeetCode-Sp[2]. In this paper, for each question function, we generate an RV question group with **five** RV questions that are instantiated with unique variable combinations. In total, 575 RV questions from 115 RV question groups are generated by the MATH question functions (**MATH-RV**) and also the same number by the LeetCode question functions (**LeetCode-RV**). These two question sets are utilized in the all the experiments in our study.

Evaluation Metrics. We define four metrics for our RV-Bench evaluation. Given a set of RV question groups $\mathcal{Q}_{\text{RV}} = \{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(m)}\}$, where $m = |\mathcal{Q}_{\text{RV}}|$, their corresponding original problems set is denoted as $\mathcal{Q}_{\text{Sp}}$. The RV question group generated by the i-th question function is denoted as $\mathcal{G}^{(i)} = \left(q_1^{(i)}, q_2^{(i)}, \ldots, q_n^{(i)}\right) \in \mathcal{Q}_{\text{RV}}$, where n is the number of generated RV questions $q_j^{(i)}$ in each $\mathcal{G}$. The original problem of $\mathcal{G}^{(i)}$ is denoted as $q_{\text{Sp}}^{(i)} \in \mathcal{Q}_{\text{Sp}}$. Let $\hat{a}_j^{(i)}$, $a_j^{(i)}$, $\hat{a}_{\text{Sp}}^{(i)}$, and $a_{\text{Sp}}^{(i)}$ denote the predicted answers and label answers of RV question $q_j^{(i)}$ and the original problem $q_{\text{Sp}}^{(i)}$, respectively. We further define $N_{\mathcal{G}^{(i)}}$ as the number of correctly answered RV questions in RV question group $\mathcal{G}^{(i)}$.

1) Exact Match Accuracy (Acc): Measures the correctness of the answer for each RV question through strict string matching, representing the approximation of the expectation over $\mathcal{Q}_{\text{RV}}$:

$$\text{Acc} = \frac{\sum_i^m \sum_j^{|\mathcal{G}^{(i)}|} \mathbb{1}\left(\hat{a}_j^{(i)} = a_j^{(i)}\right)}{m \cdot n}. \quad (1)$$

2) Group Accuracy@n (GA@n): Indicates that all n generated questions are answered correctly in $\mathcal{G}^{(i)}$, representing aggregate correctness of the model on the RV question group:

$$\text{GA@}n = \frac{\sum_{i=1}^m \mathbb{1}\left(\forall q_j^{(i)} \in \mathcal{G}^{(i)}, \hat{a}_j^{(i)} = a_j^{(i)}\right)}{m}. \quad (2)$$

3) Complete Ratio (CR): Assess whether the original problem is answered correctly and at least 80% of the generated RV questions are also correctly answered. It represents the ratio of questions where the model can completely solve both the original and random variable versions:

$$\text{CR} = \frac{\sum_{i=1}^m \mathbb{1}\left(\hat{a}_{\text{Sp}}^{(i)} = a_{\text{Sp}}^{(i)} \wedge N_{\mathcal{G}^{(i)}} \geq \lceil 0.8 \cdot n \rceil\right)}{m}. \quad (3)$$

4) Original Only Ratio (OOR): Evaluates whether the original problem is answered correctly, while at least 80% of the answers to the RV questions are incorrect. It represents the proportion of questions where the model can only solve the original problem but fails to solve the RV questions:

$$\text{OOR} = \frac{\sum_{i=1}^m \mathbb{1}\left(\hat{a}_{\text{Sp}}^{(i)} = a_{\text{Sp}}^{(i)} \wedge N_{\mathcal{G}^{(i)}} \leq \lceil 0.2 \cdot n \rceil\right)}{m}. \quad (4)$$

Implementations. Following the evaluation setting of LLaMA-3 (Dubey et al., 2024), we employ 4-shot prompting using problems from Minerva (Lewkowycz et al., 2022) as the few-shot examples during inference on MATH-RV and MATH-Sp. Similarly, for LeetCode-RV and LeetCode-Sp, we randomly select four problems from LeetCode-Math out of LeetCode-RV and manually craft step-by-step solutions to serve as the few-shot examples. All experiments on open-source LLMs are conducted on an NVIDIA server with 8 A100 GPUs, while proprietary LLMs are accessed via APIs provided by their respective official platforms.

Model Selection. The selected models include a diverse range of LLMs, covering various model sizes and families to draw comprehensive conclusions across different aspects. Given the current focus on open-source LLMs, we select widely-used representative models such as LLaMA (Dubey et al., 2024), Qwen (Bai et al., 2023; Yang et al., 2024a), Phi (Abdin et al., 2024), Yi (Young et al., 2024), Gemma (Team et al., 2024b), and DeepSeek (Liu et al., 2024a) We also include math-specific open-source models tailored for mathematical domain expertise: Qwen-Math (Yang et al., 2024b), and DeepSeek-Math (Shao et al., 2024). For proprietary LLMs, we incorporate well-known models like GPT-4o (Achiam et al., 2023), Claude-3 (Anthropic, 2023), GLM-4-Plus (GLM et al., 2024), Gemini-2-Pro (Team et al., 2024a), and DeepSeek-V3 (Liu et al., 2024b). Finally, given the rising interest in large reasoning models (LRMs)

[2]Annotators are required to select the appropriate example with reasonable variables; an instance can be found here.

#	Models	Size	MATH-RV				LeetCode-RV				Overall Acc (↑)
			Acc (↑)	GA@5 (↑)	CR (↑)	OOR (↓)	Acc (↑)	GA@5 (↑)	CR (↑)	OOR (↓)	
1	o3-mini	∼	**92.52**	82.61	87.83	6.09	**77.57**	**61.74**	**67.83**	6.09	**85.05**
2	DeepSeek-R1	671B	**92.52**	**85.22**	**88.70**	6.09	72.17	52.17	57.39	**5.22**	82.35
3	o1-mini	∼	84.00	67.83	80.87	5.22	66.09	41.74	51.30	6.09	75.05
4	Gemini-2.0-Pro	∼	84.17	71.30	78.26	8.70	60.17	34.78	42.61	8.70	72.17
5	DeepSeek-v3	671B	85.04	72.17	76.52	5.22	58.26	34.78	37.39	12.17	71.65
6	GLM-Zero-Preview	∼	83.13	65.22	77.39	6.09	60.00	35.65	44.35	9.57	71.57
7	QwQ-32B-Preview	32B	83.83	60.87	79.13	5.22	58.96	30.43	42.61	7.83	71.40
8	Claude-3.5-Sonnet	∼	80.35	63.48	73.04	6.09	61.39	35.65	42.61	8.70	70.87
9	Qwen2.5-Max	∼	81.39	63.48	74.78	6.96	58.43	33.04	42.61	12.17	69.91
10	Qwen2.5-72B-It	72B	81.04	62.61	76.52	6.09	58.43	29.57	40.00	10.43	69.74
11	Qwen2.5-32B-It	32B	80.00	61.74	73.91	**4.35**	55.48	26.09	39.13	12.17	67.74
12	GLM-4-Plus	∼	77.91	53.91	71.30	6.96	55.30	26.96	38.26	14.78	66.61
13	o1-preview	∼	75.83	42.61	59.13	6.96	54.78	32.17	40.87	9.57	65.31
14	GPT-4o	∼	76.70	57.39	63.48	6.09	50.09	20.00	32.17	13.04	63.40
15	Phi-4	14B	72.00	53.04	61.74	8.70	54.78	26.96	34.78	9.57	63.39
16	Llama3.3-70B-It	70B	74.43	52.17	62.61	9.57	45.57	18.26	22.61	15.65	60.00
17	Qwen2.5-7B-It	7B	71.65	52.17	60.00	8.70	46.78	20.87	26.09	13.04	59.22
18	Qwen2.5-Math-It	7B	72.70	51.30	62.61	12.17	37.91	10.43	17.39	14.78	55.31
19	Qwen2.5-3B-It	3B	67.65	43.48	60.00	8.70	37.04	12.17	19.13	14.78	52.35
20	Llama3.1-70B-It	70B	62.78	39.13	50.43	9.57	40.35	14.78	23.48	15.65	51.57
21	Gemma2-27B-It	27B	59.13	34.78	46.96	6.09	35.65	13.04	17.39	13.91	47.39
22	Phi-3-medium-4k-It	14B	53.04	24.35	35.65	11.30	37.22	8.70	19.13	13.91	45.13
23	Yi-1.5-Chat	34B	50.96	21.74	31.30	11.30	33.74	8.70	13.04	13.04	42.35
24	Phi-3-mini-4k-It	3.8B	50.26	26.09	37.39	14.78	34.26	9.57	16.52	11.30	42.26
25	Qwen2.5-7B-Base	7B	53.22	25.22	36.52	13.04	31.13	7.83	12.17	21.74	42.18
26	Gemma2-9B-It	9B	51.30	30.43	36.52	13.04	29.91	5.22	12.17	13.91	40.61
27	GPT-3.5-turbo	∼	48.35	20.87	30.43	11.30	31.48	9.57	14.78	12.17	39.92
28	Mathstral-7B	7B	45.22	19.13	29.57	14.78	28.70	6.96	12.17	11.30	36.96
29	Llama3.1-8B-It	8B	46.43	25.22	30.43	16.52	27.13	6.96	10.43	15.65	36.78
30	DeepSeek-Math-It	7B	48.17	18.26	33.04	11.30	24.70	6.09	7.83	12.17	36.44
31	Mixtral-8x7B-It-v0.1	46.7B	33.22	11.30	13.91	17.39	27.65	6.09	9.57	20.00	30.44
32	Llama3.2-3B-It	3B	36.70	15.65	22.61	14.78	23.83	5.22	9.57	20.00	30.27
33	Llama3.1-8B-Base	8B	24.52	6.09	12.17	17.39	21.57	5.22	7.83	16.52	23.05

Table 1: The **RV-Bench** leaderboard across various LLMs comprises the RV-questions generated from MATH (**MATH-RV**) and LeetCode (**LeetCode-RV**) question functions. The leaderboard is ranked by **Overall Acc**, which denotes the exact match accuracy for all generated RV-questions in both MATH-RV and LeetCode-RV. The rank of each model is listed in the column **#**, with the best and second-best results for each column highlighted in bold and underlined, respectively. An ∼ in column Size indicates that the model is proprietary that the model size in not publicly available. (↑) indicates a higher value is better for this metric, while (↓) indicates a lower value is better.

in both academia and industry, we also include QwQ (Team, 2024; Yang et al., 2024a), OpenAI o1-preview/mini (Qin et al., 2024) and o3-mini.

4 RV-Bench Learderboard for LLMs

Table 1 summarizes the performance of various LLMs on our proposed RV-Bench. Given the definitions of the metrics in Section 3, it is intuitive that, in most cases, the order of metric values for a specific LLM follows Acc $\geq$ CR $\geq$ GA@5. Specifically, higher Acc and GA@5 indicate the model's greater performance on RV questions and correctness on RV question groups. The higher CR shows that the models completely understand the pattern when they correctly answer the original problem. In contrast, a higher OOR reveals that even though the models correctly answer the original problem, they fail to sufficiently understand the pattern of the question content, leading to difficulties in solving the same question with random variables.

LLMs are expected to demonstrate superior performance across metrics such as Acc, CR, and GA@5, and are preferably expected to have lower OOR. *Models that meet this expectation are recognized as having completely understood the questions and possessing genuine mathematical reasoning capabilities.* Furthermore, the generally lower GA@5 suggests that while models can solve individual instances correctly, they struggle to maintain consistency across various variable combinations. This indicates that current LLMs still face challenges in thoroughly solving certain types of mathematical problems, regardless of the potential perturbations introduced by the variable combinations. Additionally, LLMs may suffer from non-integer intermediate computation when replacing well-designed original variables with random variable combinations in mathematic problems.

In detail, o3-mini and DeepSeek-R1 achieve disruptive leading performance on RV-Bench, excelling not only in overall accuracy at 92.52% but also in GA@5, CA, and OOR metrics, highlighting their outstanding mathematical reasoning capabilities. Additionally, proprietary LRMs like the

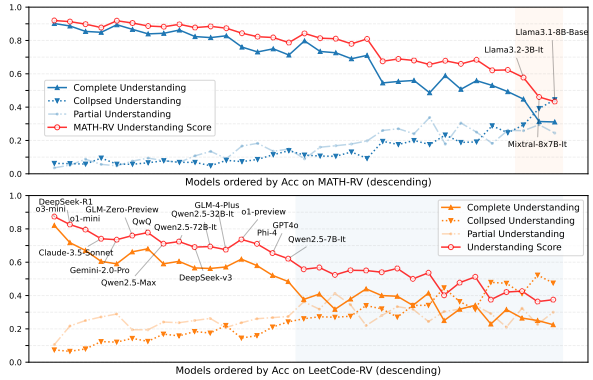

Figure 3: The blue and orange lines show average understanding degrees for correctly answered questions. The red line with circular markers represents the average pattern understanding score for each RV question group. Inconsistencies in LLMs' question pattern understanding are highlighted in the color shaded area.

o1-mini and GLM-Zero-Preview demonstrate reliable mathematical reasoning abilities. Despite having a model size of approximately 30B, the open-source LRM QwQ-32B also achieves promising results. Its performance, enhanced by computation scaling during test-time, even surpasses that of renowned advanced LLMs such as GPT-4o and Claude-3.5. Large-scale chat LLMs such as Gemini-2.0-Pro, DeepSeek-V3, and Claude-3.5-Sonnet obtain solid results, verifying the benefits of scaling in model size. Furthermore, the Phi-4 model achieves impressive results with just 14B parameters, validating the effectiveness of training with synthetic data (Abdin et al., 2024). Comparatively, open-source LLMs, especially those with sizes around 7B, exhibit mediocre accuracy.

4.1 Macroscopic Analysis of RV-Bench

We further advanced the analysis from a macroscopic perspective, considering the model's accuracy on both RV questions from MATH-RV and LeetCode-RV, as well as the original problems from MATH-Sp and LeetCode-Sp. Table 1 reports the CR and OOR metrics, which measure the model's understanding of the question pattern by verifying the consistency of accuracy. Apparently, the higher overall accuracy a model achieves, the higher CR it will have. Leading models like o3-mini and DeepSeek-R1 achieve nearly 90% of CR, demonstrating that they completely understand most of the question patterns behind the original problems and can expertly handle the associated RV questions with various variable combinations. For well-performing models, they maintain a consistent OOR and a slight difference between CR and Acc, indicating that a small portion of the question patterns are not sufficiently understood, which is evidenced by correct answers to only the original problems. In contrast, models with comparatively worse performance possess a higher OOR and greater variance between CR and Acc.

As CR and OOR reveal both complete and insufficient understanding behaviors based on the inconsistency in accuracy, when LLMs correctly answer the original problems, we further quantify the degree of the LLMs' understanding of question patterns to verify their genuine mathematical reasoning capability on RV-Bench. Specifically, we assign a pattern understanding score S to each RV question group $\mathcal{G}^{(i)}$. Derived from Section 3, the score is formulated as:

$$S_{\mathcal{G}^{(i)}} = \begin{cases} 1, & N_{\mathcal{G}^{(i)}} \geq \lceil 0.8 \cdot n \rceil \\ 0, & N_{\mathcal{G}^{(i)}} \leq \lceil 0.2 \cdot n \rceil \\ 0.5, & \text{otherwise} \end{cases} . \quad (5)$$

Different values of $S_{\mathcal{G}^{(i)}}$ reflect different degrees of LLM's understanding of the question pattern corresponding to $q_{\text{Sp}}^{(i)}$. Moreover, the degrees are categorized as **complete understanding** ($S_{\mathcal{G}^{(i)}} = 1$), **partial understanding** ($S_{\mathcal{G}^{(i)}} = 0.5$), and **collapsed understanding** ($S_{\mathcal{G}^{(i)}} = 0$) of $q_{\text{Sp}}^{(i)}$.

Figure 3 presents the average frequency of various understandings for each correctly answered original problem and the average pattern understanding score obtained by the corresponding RV question group across different LLMs. Furthermore, we highlight the models in the colored shaded area with an average score below 0.6, we consider that these models demonstrate inconsistency in their question pattern understanding. In other words, these models do not perform genuine mathematical reasoning capability on RV-Bench.

What can be concluded from the previous observation is that: the performance of nearly all LLMs on MATH-RV is significantly better than their performance on LeetCode-RV. One possible reason for this discrepancy is the higher difficulty and complexity of LeetCode-RV and LeetCode-Sp. Beyond this, we introduce another potential explanation based on our findings: ***the mathematical reasoning accuracy of LLMs partially depend on the data-domain involved in their training, which does not generalize across mathematical reasoning tasks.*** As mentioned in Section 2.1, LeetCode-related data is primarily utilized for enhancing

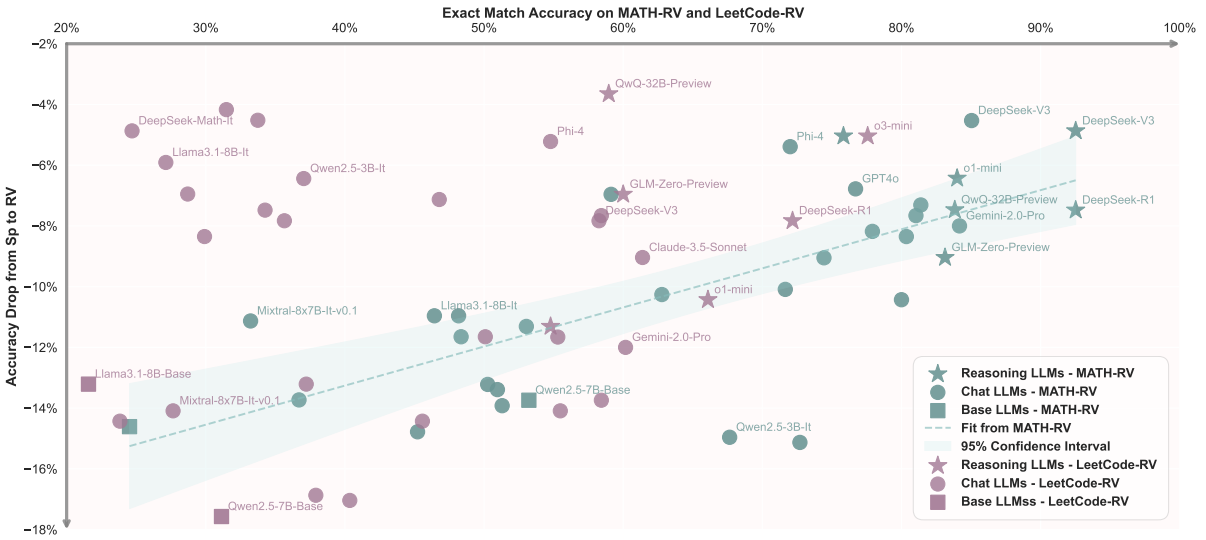

Figure 4: Accuracy drop from answering original problems to the corresponding RV questions is illustrated. Green data points represent the accuracy drop from MATH-Sp to MATH-RV, and pink data points represent the accuracy drop from LeetCode-Sp to LeetCode-RV. Different types of LLMs are presented with different shapes of markers, and some of the representative model names are provided. A dotted line is fitted from the data points of MATH-RV, the corresponding 95% confidence interval are given. We further calculate the correlation coefficient of green data points, which is $r_{\text{M}} = -0.72$, and the correlation coefficient of pink data points is $r_{\text{L}} = -0.14$.

coding skills and kept "unseen" for mathematical reasoning tasks. For questions in MATH-RV, although these questions remain new to the LLMs, it is highly likely that they have encountered MATH training sets within the same data domain to enhance their mathematical reasoning capabilities. Through this, LLMs can develop specific proficiency in MATH-domain data. However, such proficiency is scarce on LeetCode. Deducing from the performance variance, this proficiency does not generalize well, even when directly applied to similar mathematical reasoning tasks.

4.2 Accuracy Dropping in RV

Figure 4 illustrates the accuracy drop of various LLMs when transitioning from answering the original problems in MATH-Sp and LeetCode-Sp to solving the same questions with various variable combinations in MATH-RV and LeetCode-RV. Each data point in the scatter plot represents the accuracy drop of a specific LLM on a particular question set. Significantly, all models exhibited varying degrees of accuracy drop introduced by random variable perturbation, ranging from 4% to 16%. The widespread occurrence of this dropping phenomenon supports our previous concern, namely that the existing benchmark design is overly simplistic for current LLMs. We consider that ***matching a single answer only for a fixed problem is unreliable, as it may neglect influences such as data contamination and inherent randomness, and introduce potential bias into the final results.*** In our proposed random variable setting, replacing variables in mathematical problems can lead to significant accuracy deviations.

When observing the data points representing different question sets, we fit a line using MATH-related data points that indicating. Further calculating the correlation coefficient between accuracy on MATH-RV and accuracy drop from MATH-Sp to MATH-RV, we obtained $r_{\text{M}} = -0.72$, indicating a high negative correlation: the poorer the model's performance, the more significant the accuracy drop it suffers. In other words, the higher the accuracy of the LLM on MATH-RV, the better its robustness and consistency across various variable combinations. In contrast, the correlation coefficient computed with LeetCode-related data points is $r_{\text{L}} = -0.14$, indicating that there is no convincing relationship between the model's accuracy on LeetCode-RV and its robustness and consistency.

Similarly, we can conclude that: the consistency and robustness of LLMs on random variable settings in LeetCode-RV are significantly poorer than those in MATH-RV. Apart from the possible reason of varying difficulties, we extend the potential explanation we introduced at the end of Section 4.1:

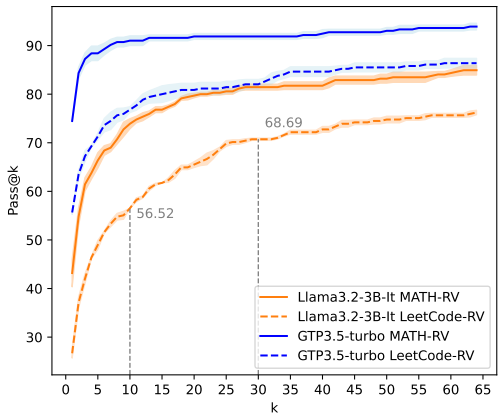

Figure 5: Results using Pass@k metric of Llama3.2-3B-It and GPT-3.5-Turbo, where each line stands for one model's test-time scaling on one data domain.

the mathematical reasoning robustness and consistency of LLMs are also partially data-domain-dependent. The proficiency in a specific data domain does not generalize well in terms of robustness and consistency to similar mathematical reasoning tasks. In conclusion, we merge these explanations as the potential underlying reason for this inconsistent phenomenon: ***LLMs obtain certain proficiency in mathematical reasoning from their training, but this proficiency is partially dependent on the data domain. It works for similar questions within the same domain but does not generalize well.*** As a result, it is questionable whether this data-domain-dependent proficiency can truly constitute LLMs' genuine mathematical reasoning capability.

4.3 Test-time Scaling Elicits Proficiency

The previous section introduces two potential reasons for the model's inconsistent accuracy and robustness between MATH-RV and LeetCode-RV: the possible variance in difficulty level and the potential data-domain-dependent proficiency of LLMs. We extend the experimental setting by using test-time scaling to allow LLMs to answer the questions with multiple attempts (Brown et al., 2024) for further exploration. Specifically, we evaluate the LLMs using pass@k metrics following the setting of Codex (Chen et al., 2021). For every mathematical question, we let the LLMs generate P independent answers. For $1 \leq k \leq P$, the pass@k metric is formulated as:

$$\text{pass@}k = \mathbb{E}_{\text{Questions}}\left[1 - \frac{\binom{P-c}{k}}{\binom{P}{k}}\right], \qquad (6)$$

where c is the number of correctly answered questions. By setting P = 100, we re-evaluate two selective LLMs from Table 1: Llama3.2-3B-It and GPT-3.5-Turbo that using pass@k.

Figure 5 displays the pass@k with multiple attempts. Taking LeetCode-RV as an example, with a single attempt, the llama's accuracy is about 26.67%. However, with up to 10 attempts, the model's pass@10 increases to 56.52%. Notably, the upper bounds of pass@k by increasing k in LeetCode-RV are consistent with those in MATH-RV, reaching around 70% at pass@30. The remaining 30% of questions are considered high difficulty-level questions that LLMs cannot correctly solve due to their inherent mathematical reasoning limitations. Apart from the potential reason for difficulties, the extent of pass@k scaling in the LeetCode domain data is observable larger than the scaling in the MATH domain. We consider this phenomenon an **"elicitation of proficiency generalization in mathematics reasoning tasks."** As a result, these findings indirectly support the potential reason for the inconsistency between the MATH-RV and LeetCode-RV being more likely due to ***LLMs' imbalance in proficiency between encountered and "unseen" data domains. The generalization of proficiency is not well-established across similar mathematical reasoning tasks but can be elicited by test-time scaling.***

5 Conclusion

Motivated by significant limitations in existing mathematical reasoning benchmarks, such as their simplistic design and potential data contamination, we introduce RV-Bench, a novel benchmark that utilizes RV questions to more accurately evaluate the capabilities of LLMs. Our findings reveal that there are significant drops in accuracy when LLMs encounter variable combinations that are "unseen" during training, underscoring the unreliability of existing benchmarks in truly capturing LLM performance. Additionally, while LLMs do gain mathematical proficiency during their training phase, this proficiency is typically tied to specific data domains and exhibits limited generalizability across broader mathematical contexts. However, we further demonstrate that employing test-time scaling can enhance this generalization. RV-Bench provides a more reliable and effective framework for evaluating LLMs, offering insightful findings to advance mathematical reasoning applications.

6 Limitations

A potential limitation of this study is that our findings and conclusions regarding LLMs in mathematical reasoning rely on experimental analysis and empirical studies; theoretical analysis remains a subject for future work. Another potential limitation is that, as RV-Bench is a fully open-source benchmark, over time, more RV questions may experience data contamination during LLM training, similar to their original counterparts from existing datasets. In such cases, RV-Bench may not maintain the reliability described in this paper.

Ethics Statement

We confirm that we have fully complied with the ACL Ethics Policy in this study. All research in this paper is based on publicly available datasets extensively used in studies related to LLMs in mathematical reasoning, and all annotation parts of our study will also be made publicly available.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

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*.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *EACL 2024 Student Research Workshop (EACL)*.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Anthropic. 2023. The claude 3 model family: Opus, sonnet, haiku. *Anthropic*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *European Chapter of the Association for Computational Linguistics (EACL)*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard (2023-2024). *Hugging Face*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Steffi Chern, Zhulin Hu, Yuqing Yang, Ethan Chern, Yuan Guo, Jiahe Jin, Binjie Wang, and Pengfei Liu. 2024. Behonest: Benchmarking honesty of large language models. *arXiv preprint arXiv:2406.13261*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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*.

Tristan Coignion, Clément Quinton, and Romain Rouvoy. 2024. A performance study of llm-generated code on leetcode. In *International Conference on Evaluation and Assessment in Software Engineering (EASE)*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. *GitHub repository*.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024. Investigating data contamination in modern benchmarks for large language models. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Junnan Dong, Zijin Hong, Yuanchen Bei, Feiran Huang, Xinrun Wang, and Xiao Huang. 2024. Clr-bench: Evaluating large language models in college-level reasoning. *arXiv preprint arXiv:2410.17558*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. 2024. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *arXiv preprint arXiv:2406.18321*.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. *Hugging Face*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zijin Hong and Jian Liu. 2024. Towards better question generation in QA-based event extraction. In *Findings of Association for Computational Linguistics (ACL)*.

Zijin Hong, Zheng Yuan, Hao Chen, Qinggang Zhang, Feiran Huang, and Xiao Huang. 2024a. Knowledge-to-SQL: Enhancing SQL generation with data expert LLM. In *Findings of Association for Computational Linguistics (ACL)*.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024b. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint arXiv:2406.08426*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of Association for Computational Linguistics (ACL)*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024a. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Zhen Huang, Zengzhi Wang, Shijie Xia, Xuefeng Li, Haoyang Zou, Ruijie Xu, Run-Ze Fan, Lyumanshan Ye, Ethan Chern, Yixin Ye, et al. 2024b. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent ai. *arXiv preprint arXiv:2406.12753*.

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*.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo Jose Taylor, and Dan Roth. 2024. A peek into token bias: Large language models are not yet genuine reasoners. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Association for Computational Linguistics (ACL)*.

Sayash Kapoor and Arvind Narayanan. 2023. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Linyi Li, Shijie Geng, Zhenwen Li, Yibo He, Hao Yu, Ziyue Hua, Guanghan Ning, Siwei Wang, Tao Xie, and Hongxia Yang. 2024. Infibench: Evaluating the question-answering capabilities of code large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, yelong shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. 2024. Not all tokens are what you need for pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024a. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024b. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024c. MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark. In *Findings of Association for Computational Linguistics (ACL)*.

Yixin Liu, Kejian Shi, Katherine He, Longtian Ye, Alexander Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. 2024d. On learning to summarize with large language models as references. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

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*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Shiwen Ni, Xiangtao Kong, Chengming Li, Xiping Hu, Ruifeng Xu, Jia Zhu, and Min Yang. 2024. Training on the benchmark is not all you need. *arXiv preprint arXiv:2409.01790*.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. 2024. Mathematical discoveries from program search with large language models. *Nature*.

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

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning (ICML)*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown. *Hugging Face*.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: a stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP 2018 Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (EMNLP)*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce LLMs

step-by-step without human annotations. In *Association for Computational Linguistics (ACL)*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024b. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, et al. 2024a. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*.

Yifan Xu, Xiao Liu, Xinghan Liu, Zhenyu Hou, Yueyan Li, Xiaohan Zhang, Zihan Wang, Aohan Zeng, Zhengxiao Du, Zhao Wenyi, Jie Tang, and Yuxiao Dong. 2024b. ChatGLM-math: Improving math problem-solving in large language models with a self-critique pipeline. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, 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 *International Conference on Learning Representations (ICLR)*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting large language model for machine translation: a case study. In *International Conference on Machine Learning (ICML)*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, et al. 2024. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. In *Findings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Donglin Zhuang, Xingyao Zhang, Shuaiwen Song, and Sara Hooker. 2022. Randomness in neural network training: Characterizing the impact of tooling. In *Machine Learning and Systems (MLSys)*.

A Related Work

A.1 Benchmarking LLMs

The rapid development of LLMs has significantly advanced the evaluation of their capabilities (Chang et al., 2024). Well-designed benchmarks such as MMLU (Hendrycks et al., 2021a), GLUE (Wang et al., 2018), MMLU-Pro (Wang et al., 2024b), SuperGLUE (Wang et al., 2019), CommonSenseQA (Talmor et al., 2019), and ARC (Clark et al., 2018) have pioneered the evaluation of general tasks like question answering (QA), natural language understanding (NLU), and commonsense reasoning. As LLMs have demonstrated success across various domains, there has been a growing demand to evaluate their performance on task-specific benchmarks (Chang et al., 2024). As a result, an increasing number of domain-specific datasets have been introduced. For example, BoolQ (Clark et al., 2019) and SQuAD (Rajpurkar et al., 2016) evaluate reading comprehension (Hong and Liu, 2024) and language-based complex reasoning, while GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), and MathQA (Amini et al., 2019) focus on mathematical problem-solving. With the growing number of LLMs, leaderboards such as the OpenLLM Leaderboard (Beeching et al., 2023; Fourrier et al., 2024) and OpenCompass (Contributors, 2023) now provide comprehensive evaluations across various mainstream benchmarks. For more complex reasoning tasks, benchmarks like InfiBench (Li et al., 2024), MathBench (Liu et al., 2024c), and OlympicArena (Huang et al., 2024b) have been successively released.

Our proposed RV-Bench is also a domain-specific benchmark for evaluating LLMs' mathematical reasoning capabilities. The well-designed

random variables framework effectively reflects LLMs' genuine performance in understanding mathematical problems.

A.2 LLMs in Mathematical Reasoning

Mathematical reasoning is a task that effectively showcases the capabilities of LLMs and has garnered significant attention within the community, achieving remarkable advancements (Ahn et al., 2024). Techniques such as continual pre-training (Lin et al., 2024), fine-tuning (Yuan et al., 2023), and reinforcement learning (Wang et al., 2024a) are extensively employed to enhance LLMs' mathematical reasoning. This has led to the development of math-specific LLMs like Qwen-Math (Yang et al., 2024b), DeepSeek-Math (Shao et al., 2024), and MetaMath (Yu et al., 2024). Consequently, these methodologies have attained near-perfect performance on complex mathematical reasoning datasets (Fourrier et al., 2024). OpenMath (Toshniwal et al., 2024) highlighted the importance of question diversity during fine-tuning, achieving a 96% accuracy on the grade-school arithmetic reasoning benchmark GSM8K (Cobbe et al., 2021). Likewise, o1 (Zhong et al., 2024) improved performance on the high-school-level mathematical competition benchmark MATH (Hendrycks et al., 2021b), attaining nearly 95% accuracy.

Most recently, GSM-Symbolic (Mirzadeh et al., 2024) uncovered a limitation regarding LLMs' genuine capabilities in arithmetic reasoning by evaluating them with diverse questions generated from symbolic templates of GSM8K. The study revealed that LLMs tend to replicate reasoning steps observed during training, rather than genuinely reasoning through specific problems, which poses a critical challenge to the current evaluation methodologies for LLMs. To further address this issue, our proposed RV-Bench employs "unseen" RV questions to effectively assess LLMs' mathematical reasoning. We provide a re-ranking and comparison with the original MATH benchmark, demonstrating that LLMs still face significant challenges in understanding and solving complex mathematical reasoning questions, thereby revealing their genuine capabilities.

B Construction of RV-Bench

In this section, we detail the construction of the question function by thoroughly describing the annotation process for each module. This section also serves as an guideline of RV-Bench annotation.

Original Problem:
Suppose that we roll 2 fair 6 -sided dice. What is the probability that the 2 numbers rolled sum to 4 ?
`Initialization:`
`num_dice` $\in \{2, 3, 4\}$
`target_sum` $\in [$ `num_dice` $, 6 \times$ `num_dice` $]$

Table 2: **Step 1.** Annotators first review the problem and identify the variables, assigning each variable a semantic-based name. Then, a random range is set for each variable. The ranges are required to maintain the difficulty level of the original problem.

Preliminaries. To ensure the quality of our benchmark, we recruited 10 candidate annotators, all of whom are graduate students with strong backgrounds in mathematics and computer science (majoring in one domain and achieving an "A" grade in the other through examination). Candidates were required to solve a selection of mathematical and coding problems (100 in total) sampled from the MATH test set and LeetCode for qualification purposes. Ultimately, six candidates were chosen to serve as the professional annotators for RV-Bench.

For a specific problem, as exemplified in Figure 2, the annotator initially reviews the problem meticulously to ensure a thorough understanding of the original content and the text-based solution provided. If the problem is found to be unclear or insufficiently comprehended by the annotator, it is subjected to a post-calibration process for further discussion, details of which are provided in Section B.2. Problems that are well comprehended proceed to the annotation process as described below.

Step 1: Identify and Initialize the Variables. For a problem meeting the criteria, the annotator begins by identifying the variables. Typically, key numbers, names, and equations are potential candidates for random variables. In practice, a problem may contain multiple variables, but to maintain a consistent difficulty level, only those variables that align with the original problem's intent are selected for randomization. As illustrated in Tab. 2, the original problem involves calculating the probability for a specific sum with two dice. The number of dice and the target sum, which are integral to the problem's intent, are identified as variables,

Original Solution:

There are 3 ways to roll a sum of 4: 3 on the first die and ... 3 on the second die. There are 36 total possibilities, so the probability is \frac{3}{36} = \frac{1}{12}.

Solution:

```
total_outcomes = 6**num_dice
```

---

enumerate the possible outcomes

```
prob = outcomes / total_outcomes
```

Table 3: **Step 2.** Annotators convert the original text-based step-by-step solution into code implementation. Due to the close coupling of some solutions with specific problems, annotators may need to revise the solutions to ensure they are generalizable.

Step 1 & Step 2:

num_dice = 3

target_sum = 9

prob = \frac{25}{216}

Generation:

Q: Suppose that we roll 3 fair 6-sided dice. What is the probability that the 3 numbers rolled sum to 9?

A: \frac{25}{216}

Table 4: **Step 3.** Based on the initialized variables from Step 1 and the corresponding answer from Step 2, the QA pair is generated as an instantiated question.

highlighted in green and yellow, respectively. The description "6-sided dice" which serves as a characteristic of the objective content, is not identified as a variable and is shown in gray. Once the variables are identified, the annotator locates their numerical and symbolic elements in the original problem and replaces them with slots.

Next, the annotator assigns a range to the identified variables, accompanied by semantically related variable names to define the `initialization` module. For variables that involve interdependent calculations, their ranges must be mutually constrained to ensure the question remains solvable (details are discussed in Sec. B.1). For example, in Tab. 2, the range of the target sum depends on the number of dice. To initialize the question function, each variable is assigned a random value selected from its range.

Step 2: Construct the General Solution. For RV questions, the numerical outcomes will vary with different combinations of variable values. Therefore, a general solution must be constructed to solve the problem irrespective of these values.

In the case of problems from the MATH test set, the provided solutions include detailed text-based step-by-step problem-solving process. The annotator is required to convert these text-based solutions into code implementations. In certain instances, the annotator may need to revise the code implementation because some text-based solutions are coupled tightly to the specific problem and lack generalization. The process of constructing a general solution for LeetCode-Math problems differs slightly. For each problem in LeetCode, some code-based solutions are available from the community[3], which have been validated through successful execution. The annotator must identify and comprehend a `Python` solution and transform it into the appropriate format, therefore defining the `solution` module based on the community-provided solutions.

Once the variable combination has been initialized in Step 1, the `solution` module takes it as input and returns a correct corresponding answer. This module undergoes further validation for correctness and effectiveness through calibration among annotators, as detailed in Sec. B.2.

Step 3: Generate the QA Pairs. Following the completion of the previous steps, where the initialized variables and the general solution for various variable combinations were defined, annotators proceed to Step 3. In this step, annotators utilize the problem content with slots identified in Step 1 to define the `generation` module. This involves filling the slots with randomized variable values and formatting the output into question-answer (QA) pairs. Importantly, the original problem may include content that is extraneous to the problem (e.g., restrictions related to computational environments). Annotators are required to remove these irrelevant sections to ensure the question content focuses solely on the pertinent details. All generated questions from different question functions are compiled into a comprehensive question set for RV-Bench.

[3] An solution instance can be found here.

B.1 Conditions

To maintain the difficulty level of the RV questions consistent with the original problems, we incorporate difficulty control conditions when defining the random ranges for variables. We establish three conditions/criteria for setting the random range in Step 1: 1) The fluctuation range of variables should remain uniform across different questions; 2) Variables that significantly affect the problem's complexity may be fixed as constants; 3) The random range for simpler questions can be broader, whereas for more challenging questions, it should be narrower to prevent considerable variations in difficulty.

By controlling the difficulty level, we ensure that LLMs are fairly compared on RV questions relative to the original problem, minimizing performance differences that could arise from variations in difficulty.

B.2 Calibration and Post-Filtering

Following the annotation process, we undertake a calibration and post-filtering step (Li et al., 2024) to enhance the consistency and objectivity of the question functions in RV-Bench. During Step 1, any problematic question that is not well-comprehended is promptly subjected to calibration and discussion. Confusing problems are collaboratively reevaluated and re-entered into the annotation process. If a problem cannot guarantee solvability or generalization for random variables, it is removed from the dataset. After all question functions have been annotated, a cross-calibration process is conducted. Annotators review each other's annotations, verifying the correctness of the question functions and testing the runtime of the `solution` module across a broad spectrum of variable combinations. This runtime testing helps identify potential issues, such as exceeding maximum recursion depth, to ensure that each unique variable combination remains correctly solvable. Additionally, question functions derived from LeetCode-Math that do not closely relate to mathematical reasoning or computation are filtered out.