

PyraMathBench: Evaluating and Improving Mathematical Capability in Large Language Models

Zetian Ouyang¹, Linlin Wang^{1*}, Gerard de Melo², Liang He¹

¹East China Normal University ²Hasso Plattner Institute, University of Potsdam

{51265901102}@stu.ecnu.edu.cn, {llwang, lhe}@cs.ecnu.edu.cn, gdm@demelo.org

Abstract

Despite the pivotal role of numerical reasoning as the cornerstone of mathematical capabilities in large language models (LLMs) across applications, few benchmarks evaluate LLMs by integrating numerical processing and mathematical reasoning, hindering the interpretability of failures in math tasks. We introduce PyraMathBench¹, a comprehensive hierarchical benchmark with 32,505 questions derived from 7,404 math word problems, spanning 4 key cognitive aspects, 14 subcategories, and 2 modalities. Experiments reveal that LLMs’ performance is severely compromised by inadequate numerical computation and weak handling of abstract numerical questions. To address this, we propose the Smart Optimization & Learning-based VErSatile module (SOLVE) and Interactive Relative Policy Optimization (IRPO), which enhance LLMs’ numerical-mathematical synergy via efficient tool calls (fuzzy matching and low-quality call rejection). Comparative experiments show Qwen-2.5 achieves a 5.0 score improvement with SOLVE and IRPO training.

1 Introduction

Numerical reasoning is ubiquitous in scientific research (Spithourakis and Riedel, 2018), financial analysis (Chen et al., 2019; Jiang et al., 2020), and integral to text understanding (Yuan et al., 2023; Sundararaman et al., 2020). Despite advances in data and compute, large language models (LLMs) like GPT-4 and Llama still struggle with math tasks (Patel et al., 2021; Zhao et al., 2023), largely due to inadequate numerical processing (e.g., flawed number tokenization (Liu and Low, 2023; Yuan et al., 2023)) and numerical hallucination (Ji et al., 2023; Chen et al., 2023).

These issues highlight the pressing need to continually enhance the numerical processing abilities

of these models, as these capabilities essentially reflect the model’s proficiency in integrating numerical and mathematical reasoning, directly impacting its ability to solve real-world problems and engage in abstract thinking (Wei et al., 2022). However, current answer-centric benchmarks fail to disentangle numerical processing flaws from mathematical reasoning errors, lacking mechanisms to diagnose failure modes related to numerical competence. While external tools show promise for boosting capabilities (Schick et al., 2023; Shen et al., 2023; Qin et al., 2024), this gap also hinders performance (Yuan et al., 2025) due to little interpretability. This creates an urgent need for a high-quality benchmark that enables fine-grained assessment of numerical-mathematical synergy.

Current benchmarks predominantly assess the mathematical reasoning abilities of language models through math word problems (MWP). Datasets like GSM8K (Cobbe et al., 2021) and APE210K (Zhao et al., 2020), based on elementary-level problems, and benchmarks such as MATH (Hendrycks et al., 2021), ARB (Sawada et al., 2023), and FrontierMath (Glazer et al., 2024), which involve competition-level problems like the IMO and AMC, are widely used. However, these benchmarks do not fully capture the limitations of LLMs’ capabilities. For example, when models provide incorrect answers, it remains unclear whether the failure stems from computational errors or misinterpretation of the question.

Some efforts, such as LILA (Mishra et al., 2022), attempt to address this by breaking down tasks into subtasks. Akhtar et al. (Akhtar et al., 2023) introduced a framework to probe LLMs’ numerical reasoning at various levels. But these frameworks lack cross-task correlations, testing LLMs’ abilities in isolation without exploring how the models’ abilities to solve simpler tasks may influence their performance on more complex tasks. In fact, mathematical ability can be thought of as a hierarchy,

*Corresponding author.

¹<https://github.com/optifine233-ship-it/PyraMathBench>

akin to a pyramid structure, where complex tasks are broken down into simpler foundational parts. By isolating and evaluating these core tasks, we can better understand the interplay between foundational skills and higher-level reasoning. This hierarchical approach not only assesses complex tasks but also identifies how weaknesses in basic skills can affect the overall performance.

Based on these, we propose the PyraMathBench (PMB), a comprehensive hierarchical benchmark providing 32,505 questions derived from 7,404 math word problems, covering 4 key cognitive aspects, 14 subcategories, and 2 modalities. Additionally, the subtasks are decomposed from real math word problems rather than generated, enhancing their relevance for real-world applications. PMB also incorporates the compositional relationships between tasks, decomposing math problems into smaller, modular subtasks that can be assessed individually and in combination. Using PMB, we evaluate a variety of state-of-the-art (SOTA) LLMs, identifying areas for improvement and offering valuable insights into the factors that influence performance.

In our analysis, a key weakness observed across the LLMs is their limited capacity for abstraction, equation solving, and factual retrieval. To address this issue, we introduce the Smart Optimization & Learning-based VERSatile module (SOLVE) to enhance the flexibility and adaptability of LLMs to tool calling, as well as a new RL framework Interactive Relative Policy Optimization (IRPO). Our method yields a 5.0% improvement against the vanilla LLM.

Our contributions can be summarized as:

- We propose the PyraMathBench, a comprehensive hierarchical benchmark that includes 32,505 questions derived from 7,404 math word problems, covering 4 key cognitive aspects, 14 subcategories, and 2 modalities, ensuring a comprehensive evaluation.
- We evaluate a variety of SOTA LLMs with this benchmark, identifying areas for improvement and offering valuable insights into the factors that influence performance.
- To address the challenges of mathematical reasoning, we introduce SOLVE module and IRPO algorithm, designed to mitigate tool call parsing failures and improve the efficiency of tool calling.

2 The Taxonomy of PyraMathBench

The core motivation behind PMB’s taxonomy is the recognition that mathematical tasks often require multiple layers of cognitive aspects and computational skills, ranging from simple numerical parsing to intricate logical reasoning. An LLM’s ability to solve a high-level math word problem is contingent upon its proficiency in handling lower-level subcomponents. Previous benchmarks lack an explicit framework for isolating different cognitive aspects, making it difficult to diagnose specific failure points. By decomposing complex mathematical tasks into distinct hierarchical aspects, PMB provides a systematic method to evaluate capability at each stage of mathematical cognition, allowing for a more interpretable assessment of LLM performance.

Inspired by previous research (Xu et al., 2022; Akhtar et al., 2023) and Piaget’s cognitive theory, taking into account both the reasoning paradigm and the feasibility of annotation, our benchmark taxonomizes tasks into four hierarchical aspects (A1–A4), encompassing 14 distinct tasks. Figure 2 shows the composition of subtasks at each aspect and examples of subtask annotation. **Complex Reasoning** evaluates the model’s ability to integrate multiple cognitive processes and mathematical principles. It requires sophisticated logical deductions, image interpretation, and multistep problem-solving. **Understanding** focuses on the model’s ability to comprehend and interpret mathematical content, transforming unstructured text into mathematical representations. **Calculation** primarily tests computational efficiency and correctness in solving mathematical problems. Finally, **Numerical Parsing** assesses the model’s ability to parse and process numerical data in various formats, thereby evaluating its capacity to recognize, interpret, and extract numerical information for further computation. The specific descriptions, prompts, and examples for 14 subtasks are provided in Appendix A.

3 Construction and Statistics

Data Sources. The PMB dataset incorporates six existing evaluation datasets and practice question collections. The data collection adheres to the following guidelines: 1) It includes common mathematical problems and visual reasoning tasks to represent the typical problem distribution. 2) Each problem is structured to allow clear decomposi-

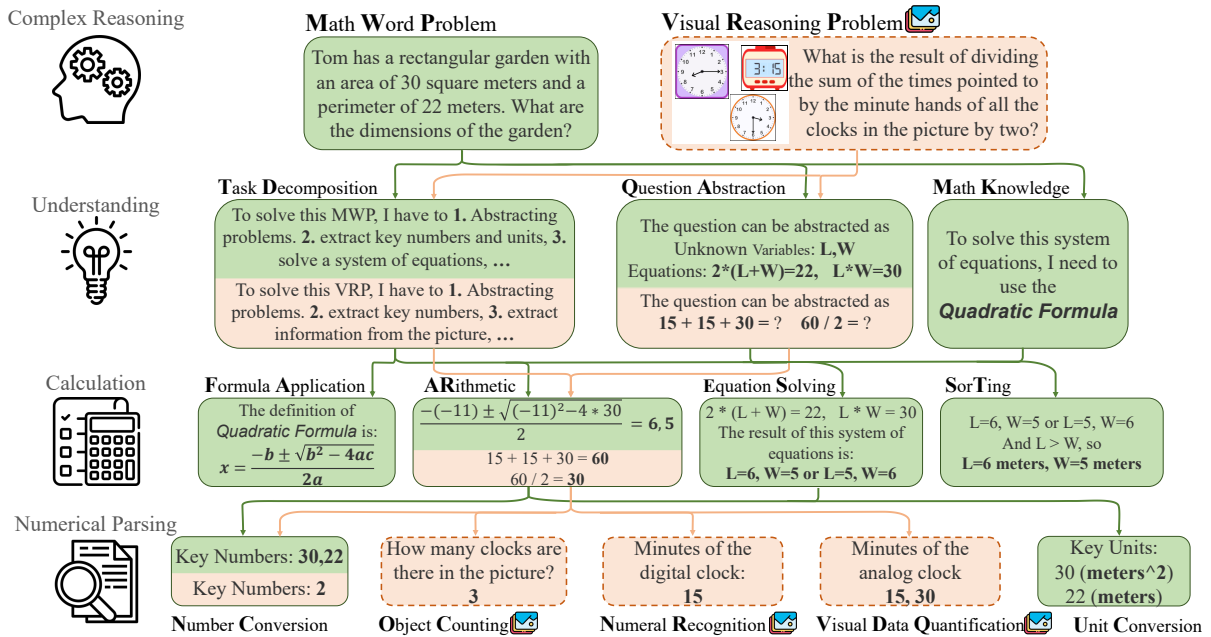


Figure 1: Taxonomy of PyraMathBench and two examples of decomposing complex reasoning problems (green and pink) into subtasks, dashed lines representing multimodal tasks.

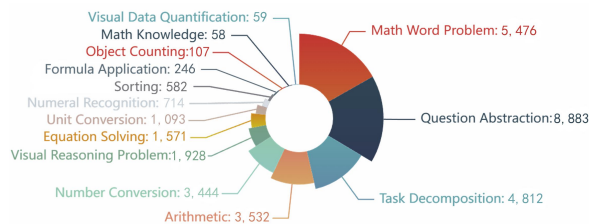


Figure 2: Data distribution of 14 subtasks.

tion into subtasks, facilitating unambiguous labeling. 3) The dataset is varied in difficulty, ensuring the inclusion of challenging tasks to effectively evaluate the performance of LLMs. We excluded non-mathematical content from the data. Based on this, we considered 6 datasets as data sources: ASDiv (Miao et al., 2020), alg514 (Kushman et al., 2014), Dolphin 18K (Shi et al., 2015), SVAMP (Patel et al., 2021), TAT-QA (Zhu et al., 2021), and MathVista (Lu et al., 2024). Another feature of these datasets is that they provide well-structured answer inference processes or automated question generation tools, facilitating the extraction of sub-task questions.

Subtasks Annotation. The dataset annotation is conducted by six experts proficient in high school-level mathematics. The subtask questions are evenly distributed among the six experts for annotation. For validation the answers annotated by different experts are evaluated using the met-

rics outlined in Section 4. If the score falls below 90, the question is deemed ambiguous and subsequently discarded. We also utilized the tabular data from TAT-QA to create images to expand the variety of multimodal tasks.

Additionally, we standardize the mathematical representations across different datasets, ensuring compatibility with both Python interpreters and LaTeX (the latter being used for more complex expressions). For floating-point answers, numerical values are rounded to six decimal places. The detailed data construction process can be found in Appendix G.

Statistics. Figure 1 presents the distribution of subtasks. PyraMathBench offers several advantages over existing evaluation methods: (1) **Comprehensive Coverage** – PMB includes a diverse array of tasks, spanning four primary areas of mathematical reasoning and 14 subcategories, derived from 32,505 questions across 7,404 Math Word Problems. This extensive dataset facilitates a thorough assessment of models across a wide range of topics and difficulty levels, ensuring broad coverage of mathematical challenges. (2) **Compositionality of Subtasks** – PMB structures subtasks derived from the same Math Word Problem, allowing for detailed performance analysis. This compositional approach enables the isolation and evaluation of a model’s ability to break down complex problems

into simpler components, providing insights into foundational skill deficiencies and their impact on overall performance. (3) **Multimodal Tasks** – By incorporating both unimodal and multimodal tasks, PMB enables a more comprehensive evaluation of LLMs. This allows assessing models’ ability to process different input types and engage in complex forms of reasoning.

4 Evaluation

We conduct an evaluation of 11 representative LLMs using the PMB dataset, including GPT-4o, GPT-4o-mini², Claude-opus-4.6³, LLaVA 13B (Liu et al., 2023), DeepSeek-R1 (DeepSeek-AI et al., 2025), DeepSeek-Math 7B (Shao et al., 2025), Qwen-2.5 14B (Yang et al., 2024a), Qwen-2.5-Math 7B (Yang et al., 2024b), Llama3.1 8B (Grattafiori et al., 2024), Gemma2 9B (Team et al., 2024), and Mistral 7B (Jiang et al., 2023). To enhance the robustness and reliability of the results with sampling-based decoding, we sample three outputs for each question and calculate the median and deviation of the scores. To simulate real-world mathematical question-answering scenarios, we employed zero-shot settings with Chain of Thought (CoT) prompting (Wei et al., 2022). The detailed experiment settings are provided in Appendix B.

$$\text{Score}(y, \hat{y}) = \begin{cases} 100 & \text{if } |\hat{y} - y| < 10^{-4} \\ 0 & \text{if } \hat{y} \text{ is UNDEFINED} \\ \max\left(0, 100 - \frac{|\hat{y} - y| \cdot 10^4}{\max(1, y, \hat{y})}\right) & \text{otherwise} \end{cases} \quad (1)$$

The answer types include four formats: 1) a number or list of numbers, 2) expressions, 3) brief text, and 4) multiple-choice options. Numerical answers are evaluated using Equation 1, where y represents the reference answer and \hat{y} represents the model response. The equation is adapted from the relative error definition of Yuan et al. (2023) and transforms the relative error into a 0–100 score such that a relative error of 1% yields a score of 0, enabling us to distinguish answer deviations stemming from numerical precision from completely erroneous reasoning. We employ Math-Verify⁴ to handle diverse mathematical expressions, and normalize structurally different but mathematically equivalent forms (e.g., factored vs. expanded, fractional vs. decimal) prior to embedding compari-

²<https://platform.openai.com/docs/models>

³<https://www.anthropic.com/api>

⁴<https://github.com/huggingface/Math-Verify>

son. For text answers, we embed the response via MathBERT (Shen et al., 2021) and calculate the cosine similarity with the answer; the 0.9 matching threshold was calibrated on a held-out 30% subset of open-ended text questions and achieves near-optimal F1 against expert annotations, with robustness verified against human judgment (98.5% accuracy, Pearson $r = 0.94$; see Appendix C). For multiple-choice questions, we calculate the perfect match rate. Finally, all scores are normalized to a range of 0 to 100. All reported results reflect model versions current as of the dates listed in Appendix B.

5 Results

In this section, we analyzed the performance of 11 LLMs on PMB, more analyze through case studies can be found in Appendix D. The results indicate that DeepSeek-R1, Claude-opus-4.6, GPT-4o, and GPT-4o-mini are the top performers, with DeepSeek-R1 exhibiting the highest overall performance. As shown in Table 1, DeepSeek-R1 excelled in seven out of eleven text-only tasks, while Claude-opus-4.6 leads in TD and UC. We note that DeepSeek-R1 (671B) is substantially larger than the other open-weight models (7B–14B), so its inclusion is intended as an upper-bound reference for mathematical reasoning rather than a direct head-to-head comparison. Within the comparable closed-source group, Claude-opus-4.6 demonstrated particularly strong understanding and numerical parsing capabilities.

Contrary to expectations, **math-specific LLMs underperformed on PMB**. Their average score is lower than Llama-3.1, which has a similar number of parameters. This is primarily due to the LLMs’ poor instruction-following ability. For example, in the QA and NR subtasks, these models often ignore prompts to extract and abstract from important numerical data, instead directly calculating answers. Additionally, DeepSeek-Math’s inconsistent answer formats contributed to its lower performance. These findings suggest that excessive finetuning or reinforcement learning may even hinder an LLM’s generalization ability in a specific field.

Furthermore, no MLLM has scored more than 20 on the visual Numerical Parsing subtasks, which is far below their score on the VRP subtask. To assess the statistical reliability of these findings, we conducted bootstrap resampling (1,000 iterations)

Model	Size	MWP	QA	TD	MK	Arithmetic
Aspect		Reasoning	Understanding			Calculation
GPT-4o	-	92.1±0.9	66.1±1.6	66.6±1.9	75.6±2.1	96.0±0.4
GPT-4o mini	-	89.6±2.4	64.7±2.2	61.2±2.4	80.8±1.3	95.7±0.9
LLaVA	13B	34.4±0.8	11.4±1.7	36.1±2.4	61.0±1.8	60.9±0.7
DeepSeek-R1	671B	93.1±1.1	87.7±2.6	75.6±1.6	100±0	96.2±1.1
DeepSeek-Math	7B	59.3±0.7	34.7±0.6	54.7±0.7	40.1±6.8	84.8±0.4
Qwen-2.5	14B	91.1±2.5	64.5±2.7	62.7±2.9	83.5±3.0	90.4±0.7
Qwen-2.5-Math	7B	82.5±0.6	48.4±2.5	45.9±1.3	19.2±1.6	95.1±0.3
Llama3.1	8B	72.3±1.1	54.1±2.2	76.7±0.1	55.2±7.9	89.3±0.4
Gemma2	9B	87.4±0.8	53.1±1.3	77.7±0.6	78.3±2.5	81.4±0.9
Mistral	7B	42.3±1.4	6.6±1.0	78.8±0.3	71.0±3.6	66.5±0.1
Claude-opus-4.6	-	91.8±4.2	87.4±3.4	97.8±0.6	84.6±0	95.1±1.2

Model	Size	ES	Sorting	FA	NC	UC
Aspect		Calculation			Numerical Parsing	
GPT-4o	-	97.5±1.1	95.8±3.7	76.7±1.3	83.8±2.7	69.7±5.6
GPT-4o mini	-	98.3±0.5	96.6±3.4	60.5±0.5	72.8±2.9	73.5±2.4
LLaVA	13B	21.6±5.1	55.2±1.4	34.6±3.5	65.1±0.5	29.5±1.9
DeepSeek-R1	671B	99.5±0.5	97.4±2.6	100±0	81.9±1.1	78.6±2.6
DeepSeek-Math	7B	90.6±2.9	64.6±2.9	21.5±0.6	74.9±2.7	55.4±2.0
Qwen-2.5	14B	98.6±0.5	96.0±1.5	79.4±0.7	80.9±0.8	36.6±1.5
Qwen-2.5-Math	7B	97.4±0.7	91.5±1.6	77.3±0.9	9.7±0.3	3.5±0.3
Llama3.1	8B	86.8±1.1	81.2±2.5	59.5±0.8	72.7±0.9	24.3±1.7
Gemma2	9B	95.6±0.1	93.9±1.4	71.7±1.0	69.0±0.6	14.9±1.1
Mistral	7B	21.9±1.1	90.4±2.0	51.7±0.2	64.3±1.4	53.9±2.6
Claude-opus-4.6	-	99.1±0.9	94.7±5.3	86.7±0	69.8±4.8	82.7±4.6

Table 1: Main results of 11 LLMs on text-only subtasks of PyraMathBench.

Model	VRP	NR	VDQ	OC
Aspect	Reasoning	Numerical Parsing		
GPT-4o	73.0±0.7	12.3±3.2	6.6±2.2	1.9±1.8
GPT-4o mini	68.3±1.6	14.6±5.6	16.7±4.3	0±0
LLaVA	25.5±2.8	2.8±1.2	8.5±2.5	7.0±3.1

Table 2: Results of 3 MLLMs on multi-modal subtasks of PyraMathBench.

on GPT-4o across the visual subtasks (NR: 714, VDQ: 59, OC: 107), yielding 95% confidence intervals that remain well below the VRP performance level, supporting the robustness of the conclusion. This indicates that in fact, **MLLMs rarely obtain information from images** when solving mathematical image problems and predominantly rely on textual information. Their ability to extract and process mathematical information from images remains fairly underdeveloped, even when the images involved are simple in nature (e.g., counting a certain item in figures or identifying an equation in the diagrams).

In complex reasoning tasks, DeepSeek-R1 achieved the highest score of 93.1 on the MWP task, followed by GPT-4o (92.1) and Claude-opus-4.6 (91.8). Notably, LLaVA shows a rather low performance on the VRP task at 25.5, in contrast to GPT-4o (73.0) and GPT-4o mini (68.3). Overall, **LLMs exhibited significant weaknesses in the understanding aspect**. Claude-opus-4.6 outper-

formed the others with an average understanding score of 89.9 (QA: 87.4, TD: 97.8, MK: 84.6), surpassing DeepSeek-R1 (87.8) and far ahead of the third-ranked Qwen-2.5 (70.2). Regarding calculation, the leading LLMs scored around 90–99 on arithmetic and equation-solving tasks. In the Formula Application subtask, DeepSeek-R1 leads with a score of 100, followed by Claude-opus-4.6 (86.7) and Qwen-2.5 (79.4). Notably, Claude-opus-4.6 achieved a near-perfect score of 99.1 on the Equation Solving task and the highest score of 82.7 on the Unit Conversion task, suggesting strong algebraic manipulation and dimensional reasoning capabilities. However, Claude-opus-4.6 underperforms on the Numerical Categorization task (69.8) relative to its overall capability level, falling behind GPT-4o (83.8) and GPT-4o mini (72.8). This contrast indicates that **Claude-opus-4.6 excels at structured calculation and language understanding, but exhibits relative weakness in categorizing and classifying numerical quantities**, which represents a different capability profile from DeepSeek-R1’s more uniformly high performance across tasks. This task requires selecting the correct formula among several variations. The unsatisfactory performance highlights the importance of eliminating hallucinations in mathematical reasoning. The most notable insight regarding Numerical Parsing is the poor performance

of MLLMs. However, despite an average score of 9.9 on the Numerical Recognition subtask, the MLLMs do actually recognize a considerable number of digits in the image, but **fail to determine which digits are useful for solving the problem**, particularly in the presence of redundant data. As a result, MLLMs may also exhibit serious hallucinations in the presence of redundant information in the image. This suggests that previous work (Liu et al., 2024) aimed at improving feature extraction through region-of-interest identification in images may not be sufficiently effective in mathematical contexts.

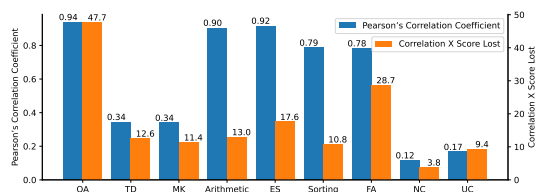


Figure 3: The Pearson’s correlation with MWP and (Correlation X Score Lost) of each subtask.

To quantify the influence of various abilities on performance, we computed the Pearson Correlation Coefficient between MWP scores and each subtask. The results, shown in Figure 3, reveal that QA (0.94), ES (0.92), AR (0.90), ST (0.79), and FA (0.78) are strongly correlated with MWP performance. When considering score losses, QA (47.7), ES (17.6), and FA (28.7) emerged as the key weaknesses of LLMs. This analysis suggests that **LLMs’ performance on complex mathematical reasoning tasks is strongly influenced by their ability to handle calculations and abstract questions**. To address these issues, we propose a method in Section 6 to enhance LLMs’ mathematical capabilities through tool calls.

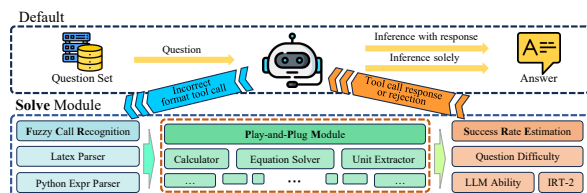


Figure 4: The workflow of SOLVE. The call parsing module is used to extract expressions from tool calls, and the difficulty assess module is used to assess tool call difficulty, when the difficulty is below the threshold, SOLVE will bypass the tool call.

6 The proposed SOLVE module and IRPO

Enhancing LLMs with external tools has emerged as a promising approach to improve their computational capabilities (Schick et al., 2023; Shen et al., 2023; Qin et al., 2024). To assess the effectiveness of the tool calling approach in mitigating the issues identified in Section 5, we first implemented a simple Play-and-Plug tool calling Module (PPM). Then we compared the accuracy of Qwen-2.5 with and without PPM on the PMB/MWP datasets, as shown in Figure 4. The results reveal a slight decrease in accuracy when using tools. Upon conducting a case analysis of the model’s output, we identified the primary reasons for the increase in error rate as: (1) LLMs failing to generate tool call requests in the instructed format, (2) tool call iterations increasing the context length, posing a challenge to LLM’s understanding, and (3) LLMs tending to call tools for simple questions, which were unnecessary for the reasoning process.

To address these issues, we introduce the Smart Optimization & Learning-based VErSatile module (SOLVE) as well as Interactive Relative Policy Optimization (IRPO), designed to mitigate tool call parsing failures and improve the efficiency of tool calling. The specific implementation can be found in the Appendix E and Appendix F, respectively.

6.1 The Smart Optimization & Learning-based VErSatile module

SOLVE enhances the flexibility and adaptability of tool integration to make it superior to simple PPM. Figure 4 illustrates the workflow of SOLVE. While preserving plug-and-play compatibility, SOLVE improves tool call efficiency through fuzzy call recognition, i.e. the LLMs can call tools in various styles, not limited to a certain format. SOLVE offloads the responsibility of formatting tool call outputs from the LLM, allowing the model to produce unstructured or loosely formatted tool calls. SOLVE then standardizes these outputs expressed in LaTeX, Python syntax, or informal handwritten styles, substantially increasing the success rate of tool invocation. Given that different LLM series’ tool call ability may be trained in different formats, this feature becomes particularly important.

Additionally, SOLVE employs a rule-based mechanism to estimate the difficulty and discrimination of tool calls. These parameters are incorporated into a two-parameter logistic Item Response

$$\begin{aligned}
\mathcal{J}_{IRPO}(\theta) = & \mathbb{E} \left[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{ref}}(O | q), \left\{ \{o_{i,j}\}_{j=1}^{N_i} \right\}_{i=1}^G \sim \pi_{\theta_{ref}}^\tau(O | q) \right] \\
& \frac{1}{G} \sum_{i=1}^G \left\{ \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\frac{\pi_\theta(o_{i,t} | q, o_{i,<t})}{\pi_{\theta_{ref}}(o_{i,t} | q, o_{i,<t})} \hat{A}_{i,t} - \beta \mathbb{D}_{KL}[\pi_\theta || \pi_{\theta_{ref}}] \right) + \right. \\
& \left. \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left[\frac{1}{N_{i,t}} \sum_{j=1}^{N_{i,t}} \left(\frac{\pi_\theta^\tau(o_{i,j,t} | q, o_{i,j,<t})}{\pi_{\theta_{ref}}^\tau(o_{i,j,t} | q, o_{i,j,<t})} \hat{A}_{i,j,t} - \beta \mathbb{D}_{KL}[\pi_\theta^\tau || \pi_{\theta_{ref}}^\tau] \right) \right] \right\}
\end{aligned} \tag{2}$$

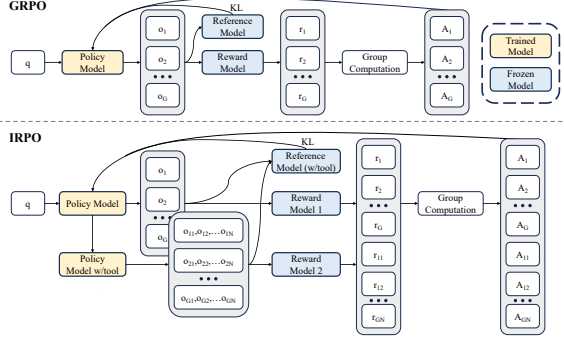


Figure 5: Demonstration of GRPO and our IRPO. IRPO samples two sets of model outputs and calculates the advantages using different reward models/functions.

Theory model (IRT-2PL) (Cai et al., 2016) to estimate the probability of the LLM autonomously solving the tool call task correctly, formulated as $P(X_{ij} = 1 | \theta_j) = \frac{1}{1 + e^{-a_i(\theta - b_i)}}$, where a_i and b_i denote the discrimination and difficulty of question i , respectively, and θ_j represents the model’s proficiency, which SOLVE will dynamically update based on recent performance. When $P(X_{ij} = 1 | \theta_j)$ exceeds a predefined confidence threshold (0.95 in our experiments), SOLVE bypasses tool calls and delegates the task directly to the LLM, optimizing both reliability and computational efficiency.

6.2 Interactive Relative Policy Optimization

6.2.1 Optimization Objective

Reinforcement learning (RL) has demonstrated efficacy in guiding LLMs to align with desired behaviors (Wang et al., 2024; Luo et al., 2023). In tool-augmented inference, existing RL approaches like GRPO (Shao et al., 2025) optimize the LLM’s performance purely from the model’s perspective. These methods primarily reward the model based on its answers without considering the involvement of external tools. To address this gap, we propose a new RL framework that extends GRPO by incorporating tool usage, as depicted in Figure 5.

For a given question q , the IRPO simultaneously

samples two sets of outputs, one from the base policy model π_θ , and another from the tool-augmented policy model π_θ^τ , which include both intermediate tool-related responses and final LLM responses. The outputs are denoted as $\{o_1, o_2, \dots, o_G\}$ and $\{\{o_{11}, o_{12}, \dots, o_{1N_1}\}, \dots, \{o_{G1}, o_{G2}, \dots, o_{GN_G}\}\}$, where N_i is the response round of the i -th response, and $o_{ij}, j < N_i$ represents the j -th round to the i -th response, which includes a tool call request and the result of that request, and $o_{ij}, j = N_i$ represents the final response of the LLM when using external tools. The IRPO optimization objective is given by: where π_θ and $\pi_{\theta_{ref}}$ represent the current and reference policy models, respectively, and q denote the questions sampled from the reference policy model $\pi_{\theta_{ref}}$. The KL divergence, \mathbb{D}_{KL} , is estimated using an unbiased estimator (Schulman, 2020).

The IRPO objective is tailored for tool-involving scenarios. The advantage of this formulation is that it simultaneously optimizes the model’s decision-making ability and the tool invocation strategy. For simplicity, we retain only the KL divergence term to stabilize training and omit the truncation for parameter updates.

6.2.2 Advantage and Reward Function

To calculate the advantages for individual output tokens in base and tool-involving scenarios, denoted as $A_{i,t}$ and $A_{i,j,t}$, we propose rule-based reward functions to evaluate the model outputs. Specifically, given the model answer \hat{y} and the reference y , the reward for the final answer is determined by its accuracy, calculated with Equation 1 as $r_i = \text{Score}(y, \hat{y})$ and $r_{i,j=N_i} = \text{Score}(y_{iN_i}, \hat{y}_i)$. For multi-round tool-call rewards, we use the Equation 3:

$$\begin{aligned}
r_{i,j < N_i} = & \frac{1}{3} \left(\mathbb{I}_{q_{sub}}(q_{ij}) \times \sqrt{1 - \text{Acc}^2(q_{ij})} \right. \\
& \left. + \frac{e^{-\text{Mean}(r_i)} - e^{-1}}{1 - e^{-1}} + \frac{1}{1 + e^{-a_i(\theta - b_i)}} \right)
\end{aligned} \tag{3}$$

where q_{ij} represents the function call part of o_{ij} , $\mathbb{I}_{q_{sub}}$ is the indicator function of the PMB subquestion set, and $\text{Acc}(q_{ij})$ is the accuracy of q_{ij} calculated in the main experiment. The formula has three components:

- If the tool call is a PMB subtask, the score decreases with higher accuracy, implying the model can independently solve the problem.
- The second term calculates the average score output from the default policy model, where higher scores suggest direct computation without tool calls.
- The third part estimates the probability that the LLM with proficiency θ generates a correct response. The parameters for this component are stored in the SOLVE module and are dynamically updated across tool invocations. The rewards from this component are dependent on the problem and model proficiency, enabling the SOLVE component to decide when to call a tool based on the model’s capabilities and the problem’s complexity.

Finally, based on the collected rewards $\mathbf{r} = \{r_1, r_2, \dots, r_G, r_{11}, r_{12}, \dots, r_{GN_G}\}$, we normalize the output tokens in each round as the advantages $\hat{A}_{ij} = \tilde{r}_i = \frac{r_i - \text{Mean}(\mathbf{r})}{\text{Std}(\mathbf{r})}$ and $A_{i,j,t} = \tilde{r}_{ij} = \frac{r_{ij} - \text{Mean}(\mathbf{r})}{\text{Std}(\mathbf{r})}$. The policy is then optimized by maximizing the objective function outlined in Equation 2. Algorithm 1 provides a summary of the IRPO workflow.

Algorithm 1 IRPO

- 1: **Input:** policy model $\pi_{\theta_{init}}$; reward functions r_{func} ; task prompts \mathcal{D}
 - 2: $\pi_{\theta} \leftarrow \pi_{\theta_{init}}; \pi_{\theta_{ref}} \leftarrow \pi_{\theta_{init}}; \pi_{\theta_{ref}}^{\pi} \leftarrow \pi_{\theta_{init}}$
 - 3: **for** step = 1, ..., M **do**
 - 4: Sample a batch \mathcal{D}_l from \mathcal{D}
 - 5: Sample G outputs $\{o_i\}_{i=1}^G \sim \pi_{\theta_{ref}}(\cdot|q)$ for each question $q \in \mathcal{D}_l$
 - 6: Sample G outputs $\{o_{i1}, o_{i2}, \dots, o_{iN_i}\}_{i=1}^G \sim \pi_{\theta_{ref}}^{\pi}(\cdot|q)$ for each question $q \in \mathcal{D}_l$
 - 7: Compute \mathbf{r} for each sample output by r_{func}
 - 8: Compute $\hat{A}_{i,t}$ and $\hat{A}_{i,j,t}$ for each token
 - 9: **for** IRPO iteration = 1, ..., μ **do**
 - 10: Update the policy model π_{θ}
 - 11: **end for**
 - 12: **end for**
 - 13: **Output:** π_{θ}
-

6.3 Experimental Results

We conduct experiments to evaluate the performance of Qwen-2.5 and Llama-3.1-8B with dif-

Model	Training	No	PPM	SOLVE
Qwen-2.5	Default LLM	91.1	82.8	95.8
	IRPO	N/A	90.2	96.1
	GRPO	91.3	84.3	95.5
Llama-3.1-8B	Default LLM	72.3	66.5	74.8
	IRPO	N/A	69.0	76.1
	GRPO	73.0	70.1	72.9

Table 3: Comparison of score on the MWP dataset for Qwen-2.5 and Llama-3.1-8B with different tool call modules and reinforcement learning algorithms.

ferent tool call modules and reinforcement learning algorithms. These configurations are assessed on the MWP dataset to gauge their impact on the model’s performance in complex reasoning tasks. The experimental results are presented in Table 3.

For both models, the PPM configuration resulted in degraded performance compared to the default LLM baseline (Qwen-2.5: 91.1 \rightarrow 82.8; Llama-3.1-8B: 72.3 \rightarrow 66.5). Upon examining the output, we identified numerous tool call failures due to issues such as parameter errors and parsing errors, primarily attributed to the challenges of using tool documentation to guide LLMs in tool invocation.

In contrast, the integration of the SOLVE module consistently improved performance across both models (Qwen-2.5: 95.8; Llama-3.1-8B: 74.8). This improvement underscores SOLVE’s enhanced flexibility and adaptability through fuzzy call recognition and efficient tool invocation strategies. Further, applying IRPO alongside the SOLVE module led to additional gains, achieving the highest scores for both models (Qwen-2.5: 96.1; Llama-3.1-8B: 76.1). GRPO, however, exhibited inconsistent behavior: while it matched IRPO’s trend for Qwen-2.5 (95.5 with SOLVE), it underperformed even the default baseline for Llama-3.1-8B when paired with SOLVE (72.9), falling below the default LLM+SOLVE result (74.8). This suggests that GRPO’s inability to model multi-step tool invocation relationships is more pronounced in weaker models, whereas IRPO’s explicit multi-round optimization remains effective regardless of model capacity.

7 Related Work

The evaluation of LLMs in mathematical reasoning has seen significant advancements through the development of various benchmarks targeting distinct cognitive tasks and problem-solving abilities. MWPs have been a central focus, as they mir-

ror real-world applications of mathematical reasoning and knowledge integration. Datasets like GSM8K (Cobbe et al., 2021), APE210K (Zhao et al., 2020), MATH401 (Yuan et al., 2023), and Math23K (Wang et al., 2017) provide diverse problem sets ranging from elementary to undergraduate levels, assessing foundational to advanced reasoning skills. In pursuit of more rigorous assessments, the Advanced Reasoning Benchmark (Sawada et al., 2023) sourced from graduate-level exams and professional resources, covering topics from undergraduate to early graduate curricula. Olympiad-Bench (He et al., 2024), FrontierMath (Glazer et al., 2024), PutnamBench (Tsoukalas et al., 2024), and OmniMATH (Gao et al., 2025) focus on olympiad-level mathematics, curating problems from international competitions like IMO and AMC. However, these benchmarks do not fully capture the limitations of LLMs’ capabilities. For example, when models provide incorrect answers, it remains unclear whether the failure stems from computational errors or misinterpretation of the question. Some efforts, such as LILA (Mishra et al., 2022), attempt to address this by breaking down tasks into sub-tasks. Akhtar et al. (Akhtar et al., 2023) introduced a framework to probe LLMs’ numerical reasoning at various levels. However, these frameworks lack cross-task correlations, testing LLMs’ abilities in isolation without exploring how the models’ abilities to solve simpler tasks may influence their performance on more complex tasks.

The mathematical ability of MLLMs is also a focus in both academia and industry, MathVista (Lu et al., 2024) is a benchmark designed to combine challenges from diverse mathematical and visual tasks and systematically analyze the mathematical reasoning capabilities of SOTA MLLMs in visually complex scenarios. MathVerse (Zhang et al., 2024) meticulously collects 2,612 high-quality, multi-subject math problems with diagrams to assess whether and how much MLLMs can truly understand the visual diagrams for mathematical reasoning. However, these evaluation benchmarks take two modalities as inputs and assess the model’s image understanding ability, ignoring the impact of textual information on the model’s accuracy, as better answers may be due to sufficient information provided in the text.

In terms of equipping LLMs with tools, two primary approaches have been explored: (1) fine-tuning open-source LLMs (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024; Parisi et al., 2022;

Hao et al., 2023), and (2) providing tool documentation and few-shot demonstrations (Shen et al., 2023; Song et al., 2023; Lu et al., 2023; Xu et al., 2023). However, these methods either introduce additional data collection and model training overhead or are limited by issues such as document inconsistency, redundancy, and incompleteness, which hinder effective tool utilization (Yuan et al., 2025). Hsieh et al. (Hsieh et al., 2023) proposed a modified tool documentation method to support zero-shot tool calling, which, however, did not resolve the inherent challenges of tool calling. Goldie et al. (Goldie et al., 2025) proposed a synthetic data generation and RL methodology targeting multi-step optimization scenarios, but without aligning each round of tool calling, and not specifically designed for mathematics.

8 Conclusion

This paper proposes PyraMathBench, a comprehensive hierarchical benchmark that includes 32,505 questions derived from 7,404 math word problems, covering 4 key cognitive aspects, 14 subcategories, and 2 modalities. Our evaluation of multiple LLMs and MLLMs highlights their limitations in question abstraction, equation solving, and image-based information extraction, which impede accurate inferences on complex mathematical tasks. We also propose the SOLVE module and IRPO algorithm, both designed to mitigate tool call parsing failures and improve the efficiency of tool calling. Experimental results demonstrate that this model significantly improves LLMs’ performance in solving math word problems.

9 Limitations

This study annotates subtasks by decomposing the MWP and VRQ problems, though it is important to note that there could be multiple versions of decomposition regarding task types and content. While various strategies have been employed to mitigate the impact of this issue during evaluation (e.g., arithmetics and univariate equations share the same evaluation status), it might still influence the results, particularly in the **Understanding** aspect. Furthermore, our task decomposition method does not independently evaluate the full range of LLM language capabilities except for the 14 subtasks, which means our taxonomy does not include all possible atomic tasks. This is a direction for our future work. Moreover, the current study focuses

on English only. Additional research could be conducted on a diverse range of further languages.

While the SOLVE module and IRPO are designed to enhance LLMs' performance on MWP-related tasks, their effectiveness may not be as pronounced in other mathematical domains, such as formula proofs or algebraic calculations, which have a fundamentally different solution path from MWP.

Acknowledgments

This work was supported by the Computational Biology Program (Grant No. 25JS2830400 & 25JS2830402) of Science and Technology Commission of Shanghai Municipality, and Shanghai Municipal Science and Technology Major Project (Grant No. 2025SHZDZX025G06).

References

- Mubashara Akhtar, Abhilash Shankarampeta, Vivek Gupta, Arpit Patil, Oana Cocarascu, and Elena Simperl. 2023. Exploring the numerical reasoning capabilities of language models: A comprehensive analysis on tabular data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15391–15405.
- Li Cai, Kilchan Choi, Mark Hansen, and Lauren Harrell. 2016. Item response theory. *Annual Review of Statistics and Its Application*, 3(1):297–321.
- Anthony Chen, Panupong Pasupat, Sameer Singh, Hongrae Lee, and Kelvin Guu. 2023. Purr: Efficiently editing language model hallucinations by denoising language model corruptions. *arXiv preprint arXiv:2305.14908*.
- Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2019. Numeracy-600k: Learning numeracy for detecting exaggerated information in market comments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6307–6313.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- DeepSeek-AI, Daya Guo, and Dejian Yang et al. 2025. *Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning*. Preprint, arXiv:2501.12948.
- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2025. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*.
- Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, and 1 others. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*.
- Anna Goldie, Azalia Mirhoseini, Hao Zhou, Irene Cai, and Christopher D Manning. 2025. Synthetic data generation & multi-step rl for reasoning & tool use. *arXiv preprint arXiv:2504.04736*.
- Aaron Grattafiori, Abhimanyu Dubey, and Abhinav Jauhri et al. 2024. *The llama 3 herd of models*. Preprint, arXiv:2407.21783.
- Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36:45870–45894.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.
- 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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *arXiv preprint arXiv:2308.00675*.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Chengyue Jiang, Zhonglin Nian, Kaihao Guo, Shanbo Chu, Yinggong Zhao, Libin Shen, and Kewei Tu. 2020. *Learning numeral embedding*. In *Findings of the Association for Computational Linguistics*:

- EMNLP 2020*, pages 2586–2599, Online. Association for Computational Linguistics.
- Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. [Learning to automatically solve algebra word problems](#). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning.
- Tiedong Liu and Bryan Kian Hsiang Low. 2023. GOAT: Fine-tuned LLaMA outperforms GPT-4 on arithmetic tasks. *arXiv preprint arXiv:2305.14201*.
- Zuyan Liu, Yuhao Dong, Yongming Rao, Jie Zhou, and Jiwen Lu. 2024. [Chain-of-spot: Interactive reasoning improves large vision-language models](#). *Preprint*, arXiv:2403.12966.
- 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*.
- Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36:43447–43478.
- 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, and Ashwin Kalyan. 2022. [LILA: A unified benchmark for mathematical reasoning](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5807–5832, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.
- 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, Online. Association for Computational Linguistics.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.
- Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J Nay, Kshitij Gupta, and Aran Komatsuzaki. 2023. Arb: Advanced reasoning benchmark for large language models. *arXiv preprint arXiv:2307.13692*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- John Schulman. 2020. Approximating kl divergence. *John Schulman’s Homepage*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2025. DeepSeek-Math: Pushing the limits of mathematical reasoning in open language models. In *The Thirteenth International Conference on Learning Representations*.
- Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. 2021. Mathbert: A pre-trained language model for general nlp tasks in mathematics education. *arXiv preprint arXiv:2106.07340*.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-gpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.
- Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. [Automatically solving number word problems by semantic parsing and reasoning](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142, Lisbon, Portugal. Association for Computational Linguistics.
- Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, and 1 others. 2023. Restgpt:

- Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*.
- Georgios Spithourakis and Sebastian Riedel. 2018. **Numeracy for language models: Evaluating and improving their ability to predict numbers**. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2104–2115, Melbourne, Australia. Association for Computational Linguistics.
- Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. 2020. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4742–4753.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.
- George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. PutnamBench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37.
- Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 845–854.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Jialiang Xu, Mengyu Zhou, Xinyi He, Shi Han, and Dongmei Zhang. 2022. Towards robust numerical question answering: Diagnosing numerical capabilities of NLP systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.
- Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 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, and 1 others. 2024b. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. In *Advances in Neural Information Processing Systems*, volume 37.
- Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. 2025. EasyTool: Enhancing LLM-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*.
- Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, and 1 others. 2024. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pages 169–186. Springer.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.
- Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*.
- Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. **TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance**. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Detailed Description of each Subtask

In this section, we provide detailed information on each subtask in Table 8~21, including 1) aspects, 2) whether it is a multimodal task, 3) size, 4) design rationale and description, and 5) all versions of the prompt we used.

B Experiment Settings

In our evaluation experiments, We conduct an evaluation of 10 representative LLMs using the PMB dataset, we adopt the model versions and sampling parameters provided in Table 4. The version dates listed in Table 4 (e.g., GPT-4o 2024-11-20, DeepSeek-R1 2025-01-20) correspond to the API or checkpoint releases available at the time of evaluation, and reported scores should be interpreted relative to these snapshots. For experiments in Section The SOLVE module and Interactive Relative Policy Optimization, the PMB was split into training, validation, and test sets in a ratio of 8:1:1. The model was trained with the hyperparameters listed in Table 7.

C Validation of the Text-Answer Scoring Metric

For open-ended text answers, responses are scored by the cosine similarity between MathBERT embeddings of the model output and the reference, with a match threshold of 0.9. This section documents the threshold selection procedure, its agreement with human scoring, and the treatment of mathematically equivalent expressions.

Threshold calibration. A random 30% split of the open-ended text questions was set aside as a calibration subset, with each item independently labeled correct or incorrect by two domain experts. Candidate thresholds were swept over $[0.75, 0.95]$, and the value yielding the highest F1 against expert labels was retained. F1 differences within $[0.85, 0.92]$ are below 0.9%, and the relative ranking of models on the Understanding aspect is preserved across this interval, indicating that downstream conclusions are insensitive to the exact threshold choice.

Agreement with human judgment. On a separately drawn validation subset, the scorer agreed with expert grading at 98.5% accuracy, with a Pearson correlation of $r = 0.94$ between embedding-based and human scores. Replacing embedding scores with human scores on this subset does not change the comparative conclusions reported in Section 5.

Mathematically equivalent expressions. Because embedding similarity is not a symbolic prover, a canonicalization step is applied before similarity is computed: fraction and decimal forms are unified, factored expressions are expanded, and variable orderings are standardized. A manual inspection

of structurally different but mathematically equivalent reference/response pairs confirms that most such pairs exceed the 0.9 threshold after canonicalization. Cases requiring deep symbolic identity transformations are rare and do not alter rankings on the affected subtasks.

D In-depth Model Analysis

PyraMathBench (PMB), which builds upon Piaget’s cognitive theory, offers a more comprehensive evaluation framework by organizing mathematical tasks into a pyramid of four capability levels, specifically designed to address and evaluate these deficiencies by rigorously assessing LLMs’ performance on both foundational and advanced tasks. Introducing the subtask decomposition mechanism in PMB enhances its ability to pinpoint logical reasoning errors with precision. For example, in Figure 6, GPT-4-o failed to answer an MWP. Upon reviewing its response in the corresponding Task Decomposition subtask, it became apparent that GPT-4-o overlooked the significance of numerical information identification.

A closer examination of its reasoning process revealed that the error stemmed from a misinterpretation of the relationship between the diameter and the distance, leading to an information extraction error, so this error could be attributed to the wrong identification of information. This case demonstrates PMB’s effectiveness in detecting inference errors or omissions.

The figure shows two screenshots from a task decomposition interface. The top screenshot, titled 'Math Word Problem', contains a question about walking around a circular lake, a reference answer of 1.28, and a GPT-4o response that incorrectly calculates the circumference as 6.28 miles. The bottom screenshot, titled 'Task Decomposition', shows the GPT-4o response to a subtask asking for mathematical operations, where it fails to identify the numerical information from the question.

Math Word Problem

[Question] To get to school, Chad must either walk around a circular lake or canoe across it. The diameter of the lake is 2 miles. How much shorter is his trip if he canoes across the lake rather than walks around it? (Use the value of $\pi = 3.14$)

[Reference] 1.28

[GPT-4o] To solve this problem, we need to calculate both the walking and canoeing distances and then find the difference. Walking around the lake: The path around the lake is the circumference of the circle. The formula for the circumference of a circle is $C = \pi \times d$, where d is the diameter. Given the diameter is 2 miles, the circumference is $C = 3.14 \times 2 = 6.28$ miles. Canoeing across the lake: The path across the lake is the diameter of the circle. We are given that the diameter is 2 miles. To find out how much shorter the trip is when canoeing, subtract the canoe distance from the walking distance: $6.28 - 2 = 4.28$ miles. Thus, Chad's trip is 4.28 miles shorter if he canoes across the lake.

Task Decomposition

[Question] Based on the MWP provided below, choose what mathematical operations are needed to solve it. [...options and MWP...]

[Reference] Additional information such as mathematical formulas, constants, theorems, etc. that are not directly provided in the question. Perform mathematical arithmetic. Identify and only identify the numbers in various formats provided in the information that are needed to solve the problem.

[GPT-4o] 1. We know the ratio of the lengths of line AB to AC is 2:5, and the length of AC is given as 25. This indicates a proportional relationship between the segments. 2. By using the ratio, we can set up the equation: $(AB / AC) = 2 / 5$. Substituting the given length of AC (25) into the equation gives: $(AB / 25) = 2 / 5$. Solve this equation for AB by cross-multiplying to obtain $AB = (2/5) * 25 = 10$. Relevant mathematical operations include setting up an equation based on the proportional relationship and performing arithmetic to solve for AB. <Answer>AC</Answer>

Figure 6: An example illustrating GPT-4o failed to answer an MWP question, and this can be oriented to overlook the significance of numerical information identification in its TD subtask.

In Figure 7, GPT-4o appears to solve a Vi-

sual Reasoning Problem from MathVista correctly. However, a closer inspection of the corresponding Numerical Recognition subtask reveals that GPT-4o’s solution is based entirely on textual information, with no contribution from the visual elements of the problem. Existing multimodal datasets often lead to overestimating MLLMs’ visual reasoning abilities, as humans may incorrectly assume that the model is leveraging visual cues. PMB addresses this gap by quantifying the extent to which MLLMs rely on various modalities by segmenting the task into separate multimodal subtasks. This approach allows PMB to effectively identify and highlight the erroneous over-reliance on text, thereby providing a more rigorous assessment of the model’s true multimodal capabilities.

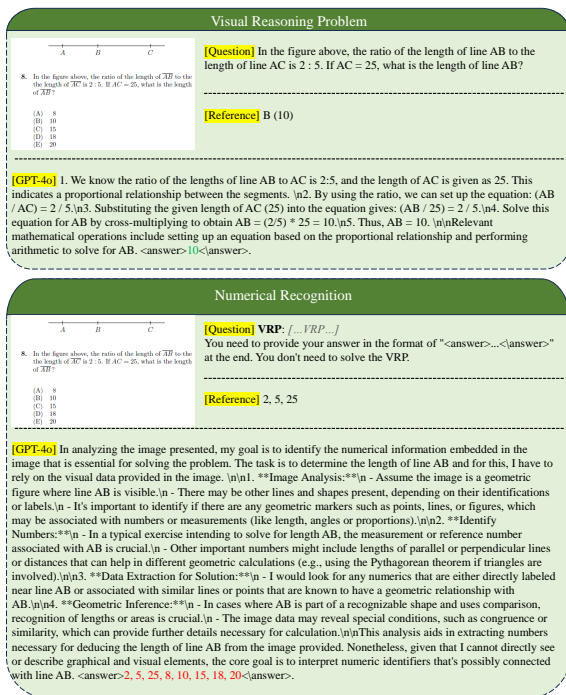


Figure 7: An example illustrating GPT-4o mainly relies on text information to solve multimodal problems instead of visual information.

For LLMs that were fine-tuned on mathematical tasks like DeepSeek-Math and Qwen2.5-Math, inadequate instruction following ability is one of the reasons for their poor performance on multiple subtasks. Additionally, DeepSeek-Math’s inconsistent answer formats contributed to its lower performance. In Figure 8, DeepSeek-Math ignores prompts asking to abstract the question into an expression and doesn’t answer the MWP, instead directly calculating answers. And in the two QA questions, DeepSeek-Math responded in different

formats, neither of which matches the indication of the prompt.

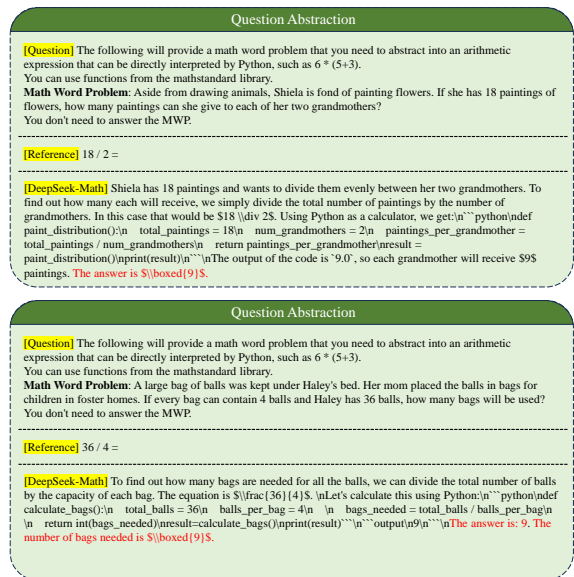


Figure 8: Two examples illustrating DeepSeek-Math failed to answer QA questions due to inadequate instruction following ability, and its answers are given in different formats.

Through the case analysis in Figure 9, we identified that the failure of MLLMs in NR tasks stems primarily from their inability to extract only the required numbers. Although GPT-4o recognizes numbers with relatively high accuracy, the numbers in the answer are randomly selected from the image without focusing on the relevant areas necessary for solving the problem. GPT-4o’s response to the corresponding visual reasoning problem revealed that while LLMs struggle to identify relevant data in lower-level tasks, they effectively discard incorrect answers through logical reasoning in higher-level tasks, leading to a lesser performance degradation in those cases.

In VDQ tasks, MLLMs exhibit prominent hallucinations, resulting in inferences and analyses that deviate from the actual content of the image. In OC tasks, MLLMs fail not only due to their inability to select the correct objects based on instructions but also due to poor performance in counting large, patterned groups. Hence, MLLMs struggle to extract meaningful information from images when addressing visual reasoning problems, relying primarily on text-based data. This suggests that some previous work (Liu et al., 2024) focused on enhancing feature extraction through key region-of-interest identification in images may fail to yield sufficiently satisfactory results in mathematical contexts.

Model	Hyperparameters
GPT-4o	model=GPT-4O 2024-11-20 version, temperature=0.7, top_p=0.9, max_tokens=1000
GPT-4o-mini	model=GPT-4O-mini 2024-7-18 version, temperature=0.7, top_p=0.9, max_tokens=1000
LLaVA	model=LLava-v1.5-7B, temperature=0.7, top_p=0.9, max_tokens=1000, quantization=Q4_0
DeepSeek-R1	model=DeepSeek-R1 2025-1-20 version, temperature=0.7, top_p=0.9, max_tokens=1000
DeepSeek-MATH	model=DeepSeek-Math-7B-rl, temperature=0.7, top_p=0.9, max_tokens=1000, quantization=Q8_0
Qwen	model=Qwen2.5-14B-Instruct, temperature=0.7, top_p=0.9, max_tokens=1000, quantization=Q8_0
Qwen-MATH	model=Qwen2.5-Math-7B-Instruct, temperature=0.7, top_p=0.9, max_tokens=1000, quantization=Q8_0
Llama3.1	model=Llama3.1-8B-Instruct, temperature=0.7, top_p=0.9, max_tokens=1000, quantization=Q8_0
Gemma2	model=Gemma2-9B-Instruct, temperature=0.7, top_p=0.9, max_tokens=1000, quantization=Q8_0
Mistral	model=Mistral-9B-Instruct, temperature=0.7, top_p=0.9, max_tokens=1000, quantization=Q8_0

Table 4: Hyperparameter for LLMs and MLLMs in main experiments.

Visual Reasoning Problem

Baseball runs scored last season	
Runs scored	Frequency
0	8
1	0
2	7
3	14
4	10

[Question] A statistician analyzed the number of runs scored by players last season. How many players scored more than 2 runs last season?

[Reference] 24

[GPT-4o] To find out how many players scored more than 2 runs, we only need to consider the frequency of players who scored 3 and 4 runs. Players who scored 3 runs: 14. Players who scored 4 runs: 10. Adding these together gives us the total number of players who scored more than 2 runs. $14 + 10 = 24$.
`<answer>24</answer>`

Numerical Recognition

Baseball runs scored last season	
Runs scored	Frequency
0	8
1	0
2	7
3	14
4	10

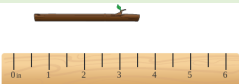
[Question] VRP: [...VRP...]
 You need to provide your answer in the format of "`<answer>...</answer>`" at the end. You don't need to solve the VRP.

[Reference] 3, 14, 4, 10

[GPT-4o] 1. The task is to find out the number of players who scored more than 2 runs last season. We should identify the numbers associated with the 'number of runs' scored by players in the image. Focus only on the individual numbers/statistics related to runs that appear in the data visualization. Numbers indicating the runs that are greater than 2 are relevant. `<answer>3, 4</answer>`.

Figure 9: An example illustrating GPT-4o failed to extract the target numbers from the visual table but solved the corresponding visual reasoning problem correctly through logical reasoning.

Visual Data Quantification

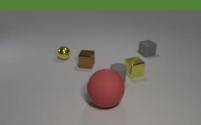


[Question] Move the ruler to measure the length of the twig to the nearest inch. The twig is about () inches long.

[Reference] 3

[GPT-4o] To infer the length of the twig from the image, I need to assess the visual reference of a standard measurement. Using the image provided in a base64 encoded format along with the description, analyze the twig's position against the ruler. Convert this visual estimation into numerical inches by identifying closest to a known reference length in the image. `<answer>6</answer>`.

Object Counting



[Question] Subtract all brown cubes. Subtract all gray cylinders. How many objects are left?

[Reference] 4

[GPT-4o] To determine the number of objects left, we must identify and subtract the specified objects from the image. Brown cubes: We need to subtract these from the total number of objects. There are 5 brown cubes based on the color and shape descriptions. Gray cylinders: Subtract these objects as well. Assuming there are 3 gray cylinders based on visual recognition. Total objects in the image (presumed for exercise): Let's presume the image contains a total of 20 objects consisting of various shapes and colors. The subtraction process will be: Total objects = 20. Brown cubes (5) = Subtract 5. Gray cylinders (3) = Subtract 3. Thus, $20 - (5 + 3) = 20 - 8 = 12$. Hence, the number of objects left after subtraction of brown cubes and gray cylinders is 12. `<answer>12</answer>`.

Figure 10: Two examples illustrating GPT-4o failed to answer a VDQ question and an OC question.

E Design of SOLVE Module

E.1 Functions of the SOLVE Module

The SOLVE supports several functions, including 1) Arithmetic, 2) Equation Solving, 3) Sorting, and 4) Knowledge Explanation. Here are the detailed descriptions of each function:

- **Arithmetic.** Used to calculate expressions, input a math expression in python expression format or latex format, return the corresponding result, with 6 decimal places retained.
- **Equation Solving.** A equation solver. Input one or more unknowns and their corresponding number of equations, return the solution to this equation or system of equations, with 6 decimal places retained.
- **Sorting.** Sorts a set of numbers, enter a sequence of numbers and arrange them in ascending or descending order as required.
- **Knowledge Explanation.** Supplies mathematical knowledge (e.g., formulas, definitions, and theorem proofs) in response to LLM queries from vector database.

E.2 Algorithm for Dynamic Assessment of LLM Ability

SOLVE employs a rule-based mechanism to estimate the difficulty and discrimination of questions, and adaptively adjusting their estimated ability level based on performance across questions of varying difficulty. The algorithm is grounded in psychometric principles, drawing inspiration from adaptive testing frameworks and item response theory (IRT). Key design principles include: 1) **Adaptive Updates:** The LLM's ability estimate (denoted as θ) is adjusted iteratively based on its performance, with larger updates for unexpected outcomes (e.g., solving a far more difficult problem or failing an easier one). 2) **Difficulty-Aware Feedback:** Each problem's discrimination α and difficulty β is predefined via rule-based methods depicted in Subsection E.3, enabling systematic comparisons between (θ), α , and β) using the equation $P(X = 1|\theta) = \frac{1}{1+e^{-\alpha(\theta-\beta)}}$, discrimination controls the sensitivity of the expectation to the $\theta - \beta$ difference. 3) **Nonlinear Response:** Updates are scaled according to the discrepancy between (θ) and β) and the accuracy of LLM response, ensuring proportional adjustments—significant for surprising results, minor for expected ones. Specifically, the ability estimate θ is updated using a prediction error-driven adjustment: $\theta_{t+1} = \theta + \lambda \cdot (\theta_t - \beta)$,

Here, λ is a learning rate modulating the step size, we adopt 0.01 in experiments. Steps 2–4 repeat across multiple problems, progressively refining θ to converge toward the LLM’s true ability level.

E.3 Estimation of Question Difficulty and Discrimination

The estimation of difficulty and discrimination of mathematical expressions is designed to quantify computational complexity through a multi-dimensional evaluation framework. This methodology combines structural analysis of the expression syntax with semantic evaluation of computational outcomes, implemented through three primary assessment dimensions:

- **Operator and Function Complexity.** Arithmetic operations receive lowest weights. Common mathematical functions are assigned customized weights reflecting their computational demands. Advanced operations or functions (e.g., calculus) are assigned with highest weights.
- **Structural Complexity.** Expressions with multiple distinct variables and higher recursive depth are assigned with higher scores.
- **Result-Type Evaluation.** Scores increase along the complexity spectrum: Integer < Rational < Floating-point < Irrational.

The coefficients assigned to each assessment dimension depend on the specific task and actual usage scenario. In this work, we extracted 1000 math word problems from PMB and initialized the coefficients based on the accuracy of each LLMs.

F Detailed Implementation of IRPO

IRPO is a new RL framework that extends GRPO by incorporating tool usage. In this section, we explain the implementation of IRPO reward functions.

For a given question q , the IRPO simultaneously samples two sets of outputs, one from the base policy model π_θ , and another from the tool-augmented policy model π_θ^τ , which include both intermediate tool-related responses and final LLM responses. Therefore, there are two kinds of reward functions, one is to score the response that provides the final answer, and the other is to score the tool call request of the large model. We have explained the first kind of reward function in main content.

For the second kind of reward function (i.e., Equation 4), the reward function is structured to address three critical challenges in guiding LLMs to appropriately invoke external tools for mathematical problem-solving: (1) incentivizing tool usage only when necessary, (2) discouraging redundant tool calls when the model can independently solve sub-problems, and (3) balancing tool reliance with the LLM’s inherent proficiency.

The first term, $\mathbb{I}_{q_{sub}}(q_{ij}) \times \sqrt{1 - \text{Acc}^2(q_{ij})}$, penalizes unnecessary tool calls based on LLM’s performance on subtasks within the PMB. Here, $\text{Acc}(q_{ij})$ measures the LLM’s standalone accuracy on q_{ij} subtask, precomputed from a diagnostic evaluation phase. The square root function ensures the penalty scales non-linearly: as $\text{Acc}(q_{ij})$ increases (indicating strong intrinsic capability), the reward decreases sharply, discouraging redundant tool usage.

The second term, $\frac{e^{-\text{Mean}(r_i)} - e^{-1}}{1 - e^{-1}}$, introduces a dynamic baseline derived from the average reward of the default policy. During training, this term normalizes rewards to the range $[0, 1]$ using exponential smoothing. A high $\text{Mean}(r_i)$ (indicating competent standalone performance) reduces the reward for tool invocation, steering the model toward self-reliance. Conversely, low baseline rewards amplify incentives for tool calls.

The third term, $\frac{1}{1 + e^{-a_q(\theta - b_q)}}$, employs a logistic function to model the likelihood of the LLM (with proficiency θ) generating correct answers without tools. Parameters (discrimination) and (difficulty) are task-specific, calibrated via Item Response Theory. This term ensures the reward accounts for the LLM’s evolving skill level: as θ surpasses β , the reward for tool-free solutions increases, promoting autonomous problem-solving aligned for LLMs with high proficiency.

G Data Construction Process

G.1 Human Annotation Detail

The dataset annotation is conducted by six experts proficient in high school-level mathematics. The subtask questions are evenly distributed among the six experts for annotation, while the corresponding answers require validation by at least two experts. Table 5 and 6 provides specific annotation guidance for each expert.

To ensure annotation consistency, we measured inter-annotator agreement using macro-averaged Cohen’s Kappa, obtaining a substantial agreement

score of $\kappa = 0.86$. For disagreement resolution, any unresolved discrepancies between the two validating experts were escalated to a third expert annotator who served as an impartial arbitrator to determine the final annotation. This multi-stage validation ensures high-quality and consistent sub-task decomposition across the entire dataset.

This document provides a comprehensive guide for annotating Math Word Problems (MWP) into the 9 specified subtasks. These guidelines are designed to assist experts in evaluating the mathematical complexity of MWPs and determining how each of the 9 subtasks is incorporated or not in the problem. Experts should follow this process step-by-step to ensure a consistent and thorough annotation of each problem. You can refer to the following example question and the example annotations in each subtasks.

Example Question: Tom has a rectangular garden with an area of 30 square meters and a perimeter of 22 meters. What are the dimensions of the garden?

1. Question Abstraction

Goal: To convert the problem into a structured mathematical representation.

Steps to Annotate:

Check for mathematical operations or relationships: Look for direct operations or relationships that can be mathematically represented (e.g., addition, multiplication, algebraic expressions).

Identify variables or unknowns: Look for words that indicate variables or unknowns that need to be solved for (e.g., “x”, “y”, “total”, “amount”).

Extract mathematical expressions: If there are relationships in the problem, translate them into equations, arithmetic expressions, or sorting list.

Example Annotation:

Unknown variables: L, W. Equations: $2 \times (L + W) = 22$, $L \times W = 30$

2. Task Decomposition

Goal: To break down the problem into steps for solving.

Steps to Annotate:

Identify the main goal: What is the problem asking for (e.g., finding an unknown, calculating a total)?

Determine substeps: Select the sub steps required from A-I below:

A: Additional information such as mathematical formulas, constants, theorems, etc. that are not directly provided in the question.

B: Solve an equation or system of equations.

C: Perform mathematical arithmetic.

D: Sort or compare the data in the question.

E: Identify and only identify the numbers in various formats provided in the information that are needed to solve the problem.

F: Identify the numerical unit(s) required to obtain the answer

G: Identify and only identify the numbers in various formats provided in the image that are needed to solve the problem.

H: Quantify data in images that are not directly presented in numerical terms.

I: Count the number of certain objects in the picture.

Suggest completing it at the end.

Example Annotation:

ABCDEFGHI

3. Math Knowledge

Goal: To evaluate whether the model needs to apply advanced mathematical concepts not explicitly stated in the problem.

Steps to Annotate:

Check for implicit knowledge: Look for problems that require knowledge of constants, special numbers (like pi, e), or advanced mathematical formulas (e.g., quadratic formula, trigonometric identities).

Select formula: Select the index of the desired formula or constant from the formula library.

Example Annotation:

5(Quadratic Formula)

4. Arithmetic

Goal: To evaluate the basic arithmetic operations ability of LLMs.

Steps to Annotate:

Check for simple operations: Identify basic arithmetic operations such as addition, subtraction, multiplication, division, exponentiation, or square roots.

Check for complex arithmetic expressions: Some problems may involve multiple operations that need to be solved step by step.

Example Annotation:

Arithmetic: $\frac{-(-11) \pm \sqrt{(-11)^2 - 4 \times 30}}{2}$

Answer: 6

5. Equation Solving

Goal: To solve equations involving one or more variables.

Steps to Annotate:

Identify equations: Look for sentences that imply an equation that can be solved for an unknown (e.g., "x + 5 = 10").

Identify types of equations: Distinguish between linear, quadratic, or higher-degree equations, as well as systems of equations.

Equation abstraction: Extract the unknown variables and corresponding equations from the problem, and it is recommended to use xyz as the unknown variable name.

Example Annotation:

Unknown variables: L, W. Equations: $2 \times (L + W) = 22$, $L \times W = 30$

Answer: $L = 6$, $W = 5$

6. Sorting Goal: To arrange numbers or objects in a specific order.

Steps to Annotate:

Identify ordering criteria: Look for instructions that require the arrangement of numbers(ascending/descending), objects, or values based on a given condition.

Check for number: Extract the numbers that need to be sorted from the question and use the sorted result as the answer. Example Annotation:

Sort: 6, 5, descending(Sort the solution results of the equation and determine the specific values of L and W.) Answer: 6, 5

7. Number Conversion

Goal: To convert numbers between formats.

Steps to Annotate:

Identify number types: Check for numbers expressed in various forms (e.g., words, scientific notation, fractions) and convert to Arabic numerals.

Check for required numbers: Identify which numbers are truly needed to solve the problem and eliminate irrelevant numbers.

Example Annotation:

30, 22

8. Unit Conversion Goal: To convert between different units of measurement.

Steps to Annotate:

Identify units: Look for units mentioned in the problem (e.g., meters, kilograms, degrees Celsius).

Check for necessary conversions: Determine if the problem requires converting between units (e.g., from kilometers to miles, Celsius to Fahrenheit).

Apply conversion factors: Use known conversion rates or formulas to convert units.

Example Annotation:

30 (square meters), 22 (meters)

Final Notes:

Cross-check each problem: Not all problems will involve every subtask. Some MWPs may require only one or a few subtasks, while others may require a combination.

Record only relevant subtasks: If a subtask is not applicable to the problem, do not annotate it. Focus on what is essential for solving the problem.

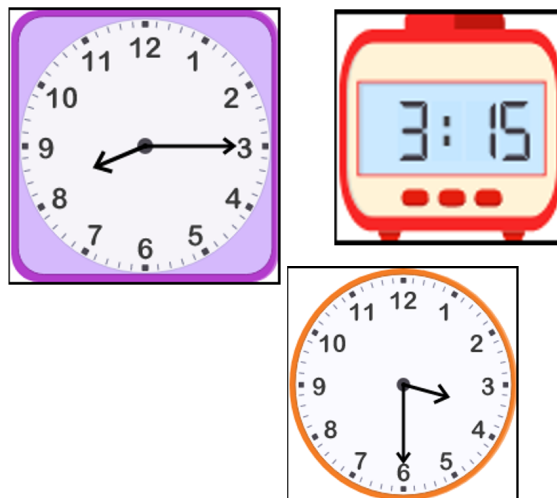
Be Objective: Extract only the data that is relevant to the specific subtask. Avoid over-annotating or adding interpretations that go beyond the task requirements.

Table 5: Guidance for annotate 9 text-only subtasks.

This document provides a comprehensive guide for annotating Math Word Problems (MWPs) into the 9 specified subtasks. These guidelines are designed to assist experts in evaluating the mathematical complexity of MWPs and determining how each of the 9 subtasks is incorporated or not in the problem. Experts should follow this process step-by-step to ensure a consistent and thorough annotation of each problem. You can refer to the following example question and the example annotations in each subtask.

Example Question: What is the result of dividing the sum of the times pointed to by the minute hands of all the clocks in the picture by two?

Example Image:



1. Numeral Recognition

Goal: Look for visual content in the image that contains numbers, symbols, or mathematical expressions. (e.g. Numbers, Variables/Constants, Formulas or Expressions)

Steps to Annotate:

Extracting Numbers and Symbols: Carefully select all visible numerical elements, ensuring that only those that are directly relevant to the problem are included. For example, in a mathematical formula within an image, extract the numbers and symbols that make up the equation.

Exclude Invalid Information: If the image contains irrelevant or distracting elements that might confuse the LLM, such as extra digits or misplaced symbols.

Example Annotation:

15(minutes of the digital clock, 3 shouldn't be annotated since it is not used for solving the problem.)

2. Visual Data Quantification

Goal: Look for elements in the image that involve visual representations of data that aren't explicitly presented as numbers.

Steps to Annotate:

Check for quantified data: Check if the image contain any visual data (like clocks, rulers, or diagrams) that requires interpretation and transformation into a numerical value.

Extract useful data: Mark the quantified numbers required for problem-solving from the image and record their corresponding labels.

Example Annotation:

Minutes of the left top analog clock: 15

Minutes of the bottom analog clock: 30

3. Object Counting

Goal: Look for descriptions or instructions that request a specific count of objects or items within the image. These objects can vary from physical items to abstract representations.

Steps to Annotate:

Identify and Isolate Objects: Find all objects that match the description and isolate them visually. If objects are grouped or clustered together, break down the grouping to count each one individually.

Handle Ambiguities: If objects are obscured or difficult to distinguish, note the possible error margins or uncertainties, ensuring clear documentation of assumptions.

Count the Objects: Count each distinct object and record the total. Be mindful of specific instructions that may request a subset of objects (e.g., only counting certain colors or shapes).

Example Annotation:

How many clocks are there in the picture?: 3

Table 6: Guidance for annotate 3 multi-modal subtasks.

G.2 Visualization of Tabular Data

The Visual Reasoning Problems subtask of PMB includes table tasks visualized using table data. The conversion process works by first preparing and structuring the table data, then visually rendering it into an image format using a variety of stylistic choices. The detailed processes are as follows:

- **Initial Table Processing:** The process begins by determining the maximum width and overall height of the columns within the table, which is essential for correctly formatting the table when converted into an image. Then we initiate the metadata of the table, including width, height, labels, index, and caption.
- **Styling Selection:** The appearance of the table image is controlled by various randomly assigned style parameters. This includes decisions on the color and style of the divider lines between columns and rows, as well as the background colors for the cells. For example, the divider lines can be styled with different patterns (solid, dashed, dotted, etc.), and their color is either black or a randomly generated similar color depending on the selected style.
- **Column Definitions and Layout:** Then we define how each column of the table should be displayed, including the alignment of text (left-aligned for the index column). The column widths are adjusted based on the length of the content in each column, ensuring that the text does not overflow or become misaligned when rendered in the image. The table is then defined using the Table object, which includes all of these layout and stylistic parameters.
- **Table Rendering:** Using the defined table, we proceed to render it into an image. This step involves integrating the column definitions, coloring, and divider styles into the visual representation of the table. The final table image is generated with a specified size and saved as a PNG file, named according to the question index.

Parameter	Value
Learning rate	5e-6
Adam β_1	0.9
Adam β_2	0.99
Weight decay	0.1
Warmup ratio	0.1
LR scheduler	Cosine
Train batch size (per device)	1
Gradient accumulation steps	4
Number of generations	4
Number of generations with tools	4
Max prompt length	256
Max completion length	1024
Number of training epochs	2
Max gradient norm	0.1
Temperature	0.7
Top-k	10
Top-p	0.9

Table 7: Training hyperparameters used for experiments in Section The SOLVE module and Interactive Relative Policy Optimization.

Subtask	Aspect	Multi-modal	Size
Math Word Problem (MWP)	Complex Reasoning	No	5476
Description			
Assesses the model’s ability to solve mathematical problems presented in natural language and reasoning through complex, real-world problems and translating them into mathematical solutions.			
Prompt			
System		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\nThink step by step and answer the following math word problem.		**Question** : \n{question}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. If the result is a floating-point number, round to six decimal places.	

Table 8: Detailed description of subtask Math Word Problem.

Subtask	Aspect	Multi-modal	Size
Object Counting (OC)	Numerical Parsing	Yes	107
Description			
This subtask requires models to count specified objects in an image based on a given description. It tests the models’ visual reasoning and object recognition skills.			
Prompt			
System		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\nIdentify the number of specified objects from the following image.		**Target Object** : \n{question}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. You don’t need to solve the VRP.	

Table 9: Detailed description of subtask Object Counting.

Subtask	Aspect	Multi-modal	Size
Visual Reasoning Problem (VRP)	Complex Reasoning	Yes	1928
Description			
<p>This subtask evaluates the model’s ability to combine textual and visual information for solving mathematical problems. MLLMs need to reason across multiple modalities and extract relevant insights from both text and images. The tasks include various types such as geometry problems, VQA, and statistic reasoning.</p>			
Prompt			
System		Human	
<p>You are a helpful AI robot, you can solve mathematical problem accurately. Think step by step and answer the following visual reasoning problem based on the following image.</p>		<p>**Question**: $\{question\}$ You need to provide your answer in the format of "$\langle answer \rangle \dots \langle answer \rangle$" at the end. If the result is a floating-point number, round to six decimal places.</p>	

Table 10: Detailed description of subtask Visual Reasoning Problem.

Subtask	Aspect	Multi-modal	Size
Question Abstraction (QA)	Understanding	No	8883
Description			
This subtask requires LLMs to convert natural language problems into solvable structured mathematical representations, including arithmetic, equations, and sorting numbers.			
Prompt			
System (arithmetic)	Human		
You are a helpful AI robot, you can solve mathematical problem accurately. The Math Word Problem(MWP), as a manifestation of questions, can be understood as the process of solving it by computing an operational expression. The following will provide a math word problem that you need to abstract into an arithmetic expression that can be directly interpreted by Python, such as $6 * (5+3)$. You can use functions from the <code>math</code> standard library.	**Math Word Problem**: $\{MWP\}$ You need to provide your answer in the format of " <code><answer>...<answer></code> " at the end. You don't need to answer the MWP.		
System (equation)	Human		
You are a helpful AI robot, you can solve mathematical problem accurately. You are a helpful AI robot, you can solve mathematical problem accurately, The Math Word Problem(MWP), as a manifestation of questions, can be understood as the process of solving a equation or system of equations. The following will provide a math word problem that you need to abstract into a equation or system of equations. Specifically, you need to first list the unknown variable(s) that need to be used after abstraction. If there are multiple unknown variables, use commas to separate them. Then list the abstract equation, and if there are multiple equations, list them in multiple lines. You can use functions from the <code>math</code> standard library.	**Math Word Problem**: $\{MWP\}$ You need to provide your answer in the format of " <code><Unknown Variables>...<Unknown Variables></code> " <code><Equations></code> <code>...<Equations></code> " at the end. You don't need to answer the MWP.		
System (sorting)	Human		
You are a helpful AI robot, you can solve mathematical problem accurately. The following will provide a math word problem(MWP) that you have to compare or sort some numbers in the MWP to solve it. Extract the numbers that need to be compared or sorted from the questions.	**Math Word Problem**: $\{MWP\}$ $\{MWP\}$ You need to provide your answer in the format of " <code><answer>...<answer></code> " at the end. You don't need to answer the MWP.		

Table 11: Detailed description of subtask Question Abstraction.

Subtask	Aspect	Multi-modal	Size
Task Decomposition (TD)	Understanding	No	4812
Description			
<p>The LLMs are required to analyze the MWP and determine the necessary steps for solving it. The LLMs must have a sufficient understanding of the text and mathematical logic to answer correctly. This subtask has a certain degree of openness.</p>			
Prompt			
System	Human		
<p>You are a helpful AI robot, you can solve mathematical problem accurately. The Math Word Problem(MWP), as a type of comprehensive mathematical problem, it may require various mathematical operations such as calculations and solving equations during to solve it. Based on the MWP provided below, choose what mathematical operations are needed to solve it.</p>	<p>Math Word Problem: $\{MWP\}$ Mathematical operation list: A: Additional information such as mathematical formulas, constants, theorems, etc. that are not directly provided in the question. B: Solve an equation or system of equations. C: Perform mathematical arithmetic. D: Sort or compare the data in the question. E: Identify and only identify the numbers in various formats provided in the information that are needed to solve the problem. F: Identify the numerical unit(s) required to obtain the answer. G: Identify and only identify the numbers in various formats provided in the image that are needed to solve the problem. H: Quantify data in images that are not directly presented in numerical terms. I: Count the number of certain objects in the picture. You need to provide your answer in the format of "$\langle answer \rangle \dots \langle answer \rangle$" at the end. You only need to reply with the index letter. If multiple operations are selected, separate them with commas. You don't need to answer the MWP.</p>		

Table 12: Detailed description of subtask Task Decomposition.

Subtask	Aspect	Multi-modal	Size
Math Knowledge (MK)	Understanding	No	58
Description			
<p>This subtask evaluates the model's ability to leverage fundamental mathematical knowledge, such as approximations of constants and geometric formulas that are not explicitly provided in the problem. For example, the approximation of e or applying the quadratic formula for the root of an equation.</p>			
Prompt			
System	Human		
<p>You are a helpful AI robot, you can solve mathematical problem accurately. The following will provide a math word problem(MWP). To solve this MWP, an additional knowledge point, such as a theorem or formula, is required. Please answer the name of this knowledge point.</p>	<p>Math Word Problem: $\{question\}$ You need to provide your answer in the format of "$\langle answer \rangle \dots \langle answer \rangle$" at the end. You don't need to answer the MWP.</p>		

Table 13: Detailed description of subtask Math Knowledge.

Subtask	Aspect	Multi-modal	Size
Arithmetic	Calculation	No	3532
Description			
This subtask evaluates the model's proficiency in performing basic mathematical expressions including operations such as four operations, root operation, exponential operation, etc.			
Prompt			
System		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\nPlease calculate the provided arithmetic expression.		**Arithmetic Expression**:\n{expression}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. If the result is a floating-point number, round to six decimal places.	

Table 14: Detailed description of subtask Arithmetic.

Subtask	Aspect	Multi-modal	Size
Equation Solving (EQ)	Calculation	No	1571
Description			
Require LLMs to solve both single-variable and systems of equations. It evaluates the model's algebraic skills and its capacity for handling more advanced mathematical structures.			
Prompt			
System (equation)		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\n"Please solve the provided equation.		**Unknown Variable**:\n{variable}\n**Equation**:\n{equation}"	
System (system of equations)		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\n"Please solve the provided system of equations.		**Unknown Variable**:\n{variables}\n**System of Equations**:\n{equations}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. For example, "<answer>["x", "5"]<answer>". If the result is a floating-point number, round to six decimal places.	

Table 15: Detailed description of subtask Equation Solving.

Subtask	Aspect	Multi-modal	Size
Sorting	Calculation	No	582
Description			
This subtask evaluates a model’s ability to arrange numbers or objects in a specific order, assesses its understanding of order relationships and computational reasoning.			
Prompt			
System		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\nPlease sort the following numbers in ascending order.		**Numbers**:\n\n{numbers}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end.	

Table 16: Detailed description of subtask Sorting.

Subtask	Aspect	Multi-modal	Size
Formula Application (FA)	Calculation	No	246
Description			
This subtask requires the LLMs to recognize and apply specific formulas to solve problems and tests the LLMs’ familiarity with mathematical relationships.			
Prompt			
System		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\nChoose the correct definition for the following theorem or formula.		**Theorem or Formula**:\n\nformula\n\n**Options**:\noptions\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end.	

Table 17: Detailed description of subtask Formula Application.

Subtask	Aspect	Multi-modal	Size
Number Conversion (NC)	Numerical Parsing	No	3444
Description			
<p>This subtask evaluates an LLM’s ability to recognize and interpret important numbers in different formats, such as Arabic numerals, written words, and scientific notation. For example, "one hundred and three" or "1.13e+2" should be converted into "113". LLM also needs to avoid identifying invalid information.</p>			
Prompt			
System		Human	
<p>You are a helpful AI robot, you can solve mathematical problem accurately.\nThe Math Word Problem(MWP), as a type of comprehensive mathematical problem, it requires identify important information in the question to solve the problem.\nThe following will provide a math word problem, and you need to identify and **ONLY** identify "the numbers in various formats provided in the information that are needed to solve the problem.</p>		<p>**MWP**: {MWP}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. You don’t need to solve the MWP.</p>	

Table 18: Detailed description of subtask Number Conversion.

Subtask	Aspect	Multi-modal	Size
Unit Conversion (UC)	Numerical Parsing	No	1093
Description			
<p>In MWP, especially in physics-related problems, unit conversion is extremely important. This subtask measures an LLM’s understanding of various units of measurement and its ability to convert between them. For example, converting "5 kW-h" to J or "100°C" to Fahrenheit.</p>			
Prompt			
System		Human	
<p>You are a helpful AI robot, you can solve mathematical problem accurately.\nThe Math Word Problem(MWP), as a type of comprehensive mathematical problem, it requires identify important information in the question to solve the problem.\nThe following will provide a math word problem, and you need to identify the number with unit(s) required to solve the MWP. (Ignore numbers without units.)</p>		<p>**MWP**: {MWP}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. You don’t need to solve the MWP.</p>	

Table 19: Detailed description of subtask Unit Conversion.

Subtask	Aspect	Multi-modal	Size
Numeral Recognition (NR)	Numerical Parsing	Yes	714
Description			
This task assesses an LLM’s ability to extract mathematical content like numbers, variables, and formulas from images. LLM also needs to avoid identifying invalid information.			
Prompt			
System		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\n\nThe Visual Reasoning Problem(VRP), as a type of comprehensive mathematical problem, it requires identify important information in the image to solve the problem.\n\nThe following will provide a visual reasoning problem and a image, you need to identify and **ONLY** identify the numbers in the image that are needed to solve the problem.		**VRP** : {VRP}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. You don’t need to solve the VRP.	

Table 20: Detailed description of subtask Numeral Recognition.

Subtask	Aspect	Multi-modal	Size
Visual Data Quantification (VDQ)	Numerical Parsing	Yes	59
Description			
In the image, some data is not directly presented in numerical form, such as the time pointed by the clock or the length of an object. This subtask evaluates the model’s ability to understand instructions and quantify nonvalue data in images.			
Prompt			
System		Human	
You are a helpful AI robot, you can solve mathematical problem accurately.\n\nIdentify the specified data from the following image. If the data is not presented directly in numerical form, you need to quantify it.		**Question** : \n{question}\n\nYou need to provide your answer in the format of "<answer>...<answer>" at the end. You don’t need to solve the VRP.	

Table 21: Detailed description of subtask Visual Data Quantification.