Neuro-Symbolic Data Generation for Math Reasoning

Zenan Li[1*] **Zhi Zhou**[1*] **Yuan Yao**[1] **Yu-Feng Li**[1]
Chun Cao[1] **Fan Yang**[2] **Xian Zhang**[2] **Xiaoxing Ma**[1]
[1]State Key Lab of Novel Software Technology, Nanjing University, China
[2]Microsoft Research Asia
lizn@smail.nju.edu.cn, zhouz@lamda.nju.edu.cn,
zhxian@microsoft.com, xxm@nju.edu.cn

Abstract

A critical question about Large Language Models (LLMs) is whether their apparent deficiency in mathematical reasoning is inherent, or merely a result of insufficient exposure to high-quality mathematical data. To explore this, we developed an automated method for generating high-quality, supervised mathematical datasets. The method carefully mutates existing math problems, ensuring both diversity and validity of the newly generated problems. This is achieved by a neuro-symbolic data generation framework combining the intuitive informalization strengths of LLMs, and the precise symbolic reasoning of math solvers along with projected Markov chain Monte Carlo sampling in the highly-irregular symbolic space. Empirical experiments demonstrate the high quality of data generated by the proposed method, and that the LLMs, specifically LLaMA-2 and Mistral, when realigned with the generated data, surpass their state-of-the-art counterparts.

1 Introduction

Despite recent progress [1–6], both proprietary and open-source LLMs are still far from satisfactory in mathematical reasoning [7–9]. It is an open question whether LLM's subpar reasoning capability is inherent or due to the the extreme scarcity of high-quality mathematical datasets [10–13]. As an initial step towards answer this question, a data generation framework that could create high-quality math datasets is required. To this end, current two lines of research struggle in the *diversity-validity* dilemma: (1) to produce diverse math data, the prompt-based method effectively rephrases math problems using LLMs, but may induce errors thus ruining the *validity*, especially considering the rigor of maths; (2) to ensure the validity, template-based methods are often used by rewriting math problems with certain rules, sacrificing the *diversity* and thus confining data scale.

To address this dilemma, we propose a novel *neuro-symbolic* framework that automatically generates high-quality, supervised mathematical data. The merit of this paradigm lies in leveraging both neural and symbolic strengths: (1) the math problem is generated in the symbolic space, achieving diversity through systematic sampling, while maintaining validity through symbolic solvers; (2) the translation from the symbolic space back to the natural language space can be effectively supported by LLMs, ensuring the consistency between newly generated formal problems and their corresponding natural language versions.

Our framework, as illustrated in Figure 1, initiates with the formalization of the original problem expressed via the math symbolic tools. Next, it *mutates* the formal problem into an evolved version, and then derives a new natural language problem by *informalization*. Specifically, we design a mutation mechanism, including various simplification and complication strategies, such that the new problems can be generated with a controllable complexity. As shown in Figure 2, our mutation

[*]Equal contribution. This work was partially done during Zenan's internship at MSRA.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

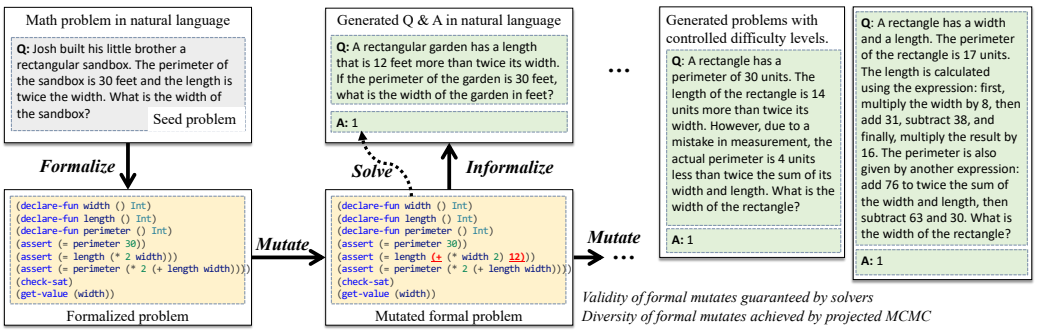


Figure 1: The overview of our neuro-symbolic data generation framework. The framework comprises three steps: (1) Formalize the seed problem into its symbolic version. (2) Mutate the symbolic problem to create new variants. (3) Translate the variants in symbolic form back to the natural language version. Additionally, we prompt GPT-4 to generate reasoning paths, which are verified by symbolic solvers, as part of the supervision.

mechanism can properly adjust the complexity of generated problems, and the exposure to more complex math problems can improve the LLM's reasoning capability. Moreover, to ensure the data validity and achieve higher generation diversity, we combine the symbolic solving with the random sampling through the projected Markov chain Monte Carlo technique [14, 15].

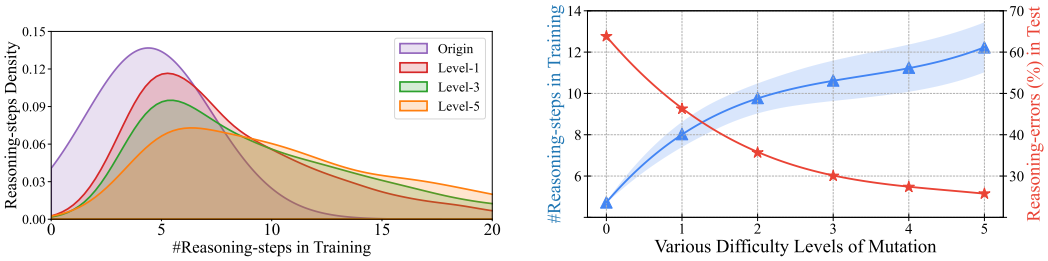


Figure 2: The performance of our proposed mutation mechanism. The first figure illustrates that the generated problems with higher difficulty levels lead to more reasoning steps of GPT-4. The second figure shows that the gradual incorporation of more difficult problems consistently improves the LLM's reasoning capability.

Empirical evaluation on GSM8K [10] and MATH [11] demonstrates the effectiveness of the proposed method. Particularly, we use the proposed framework to generate a mathematical dataset of 620K examples for supervised fine-tuning. The experimental results show that, the fine-tuned models on LLaMA-2 [16] and Mistral-7B [17] significantly outperform the existing open-source counterparts on both GSM8K and MATH datasets, as well as two out-of-domain datasets SVAMP [12] and ASDiv [18]. On the GSM8K dataset, the model fine-tuned on Mistral-7B even outperforms GPT-3.5-Turbo (by 2.4%), a proprietary model with an order of magnitude larger parameters. Additionally, we evaluate the scalability of our method, and observe consistent performance improvements, as the size of training data increases. This upward trajectory suggests a promising avenue for further enhancing LLMs' mathematical capabilities.

2 Mutation

Compared to existing data generation methods, the key feature of our framework lies in the mutation of math problems within the symbolic space, enabling systematic sampling and symbolic solving. Technically, our mutation mechanism includes several simplification and complication strategies, to control the complexity of the generated math problems. The overall framework of our problem mutation method is summarized in Algorithm 1.

Algorithm 1 The overall framework of problem mutation

Input: A seed problem expressed by goal g and constraints $h_1, \ldots, h_n$.
Output: A new problem expressed by goal g' and constraint $h'_1, \ldots, h'_n$.

for $i = 1, \ldots,$ **do** ▷ *Complicate the problem using Projected MCMC*
  Initalize random operations $\oplus_i$ and interpreted functions e'_i.
  Mutate $g' = g \oplus_0 e'(z_0)$, $h'_i = h_i \oplus_i e'(z_i)$ with parameter z_i.
  Randomly perturb a subset $\{z_1, \ldots, z_j\}$ for $j < n$.
  Solve the rest subset $\{z_{j+1}, \ldots, z_n\}$.
  if $\{z_1, \ldots, z_n\}$ is solvable **then**
    Instantiate the new problem by $\{z_1, \ldots, z_n\}$.
    Stop the complication process
  end if
end for
for $i = 1, \ldots$ **do** ▷ *Simplicate the problem using SMT tactics*
  Initalize a random tactic from {*simplify*, *qe*, *simplify*, ... }.
  Apply the tactic to the problem g' and constraints $h'_1, \ldots, h'_n$.
end for

2.1 Formalization

We first provide the formalization of math problems, based on which the mutation mechanism is operated. Specifically, we adopt the SMT-LIB language [19], a standard notation compatible with prevalent SMT solvers (e.g., Z3 [20], CVC5 [21], and MathSAT [22]). It can also be easily extended for symbolic calculators (e.g., SymPy [23]) and numerical solvers (e.g., SciPy [24]). With SMT-LIB language, the math problem in the following structure is enabled:

$$
\begin{aligned}
&\textit{Goal} && g := \min \mid \max \mid \text{solve } \ f(\mathbf{x}) \\
&\textit{Constraints} && h := h_1 \wedge h_2 \mid h_1 \vee h_2 \mid \text{ite}(h_1, h_2, h_3) \mid \\
& && \quad \forall \mathbf{x}.\, e_1(\mathbf{x}) \bowtie e_2(\mathbf{x}) \mid \exists \mathbf{x}.\, e_1(\mathbf{x}) \bowtie e_2(\mathbf{x}) \mid \\
& && \quad e_1 \bowtie e_2, \quad \bowtie \in \{\geq, \leq, >, <, =, \neq\} \\
&\textit{Expressions} && e := c \mid \mathbf{x} := (x_1, \ldots, x_n) \mid \text{foo}(\mathbf{x}) \mid \\
& && \quad e_1 \oplus e_2, \ \ \oplus \in \{+, -, \times, \div\} \\
&\textit{Domains} && \mathcal{D} := \mathbb{N} \mid \mathbb{N}^+ \mid \mathbb{R} \mid \mathbb{C}
\end{aligned}
$$

where c denotes a constant, $\mathbf{x}$ denotes an *n*-dimensional variable, ite denotes the if-then-else structure, foo refers to an *interpreted* function (e.g., trigonometric, logarithmic or user-defined ones) on the domain, and g and h represent any function of interest (can include quantifiers). In particular, we pre-defined a series of interpreted functions, such as `summation`, `binomial`, `gcd`, `lcm`, `derivate`, and `integral`, which facilitate the formalization of most high-school level mathematical problems (excluding geometry) within the above SMT-LIB language.

2.2 Simplification

We perform simplification by systematically considering *expression reduction* and *constraint reduction*, which can be attained through heuristic tactics provided by standard symbolic solvers [25].

Specifically, we apply the *simplify* tactic for expression reduction, which involes operations such as constant or variable folding (e.g., $x + 0 \Rightarrow x$ or $y + x - x \Rightarrow y$), expression expansion (e.g., $(x+1)^2 \Rightarrow x^2 + 2x + 1$), and function application (e.g., $(x = 2) \wedge (y = \log(x)) \Rightarrow y = \log(2)$); we also perform symbolic and numerical computations for further reductions (e.g., $\gcd(2x, 6y) \Rightarrow 2\gcd(x, 3y)$ and $\sin(\pi/6) \Rightarrow 0.5$).

For constraint reduction, we mainly employ the Gaussian elimination tactic *gaussian_elim* (e.g., $x = 2 \wedge y \leq x + z \Rightarrow y \leq 2 + z$). To handle the if-then-else term, we apply the *elim_term_ite* tactic to decompose it by introducing a fresh variable (e.g., $\text{ite}(x > y, x, y) > z \Rightarrow (k > z) \wedge (x > y \rightarrow k = x) \wedge (x \leq y \rightarrow k = y)$). For constraints involving quantifiers, we strive to eliminate them using the *qe* tactic (e.g., $\exists y.(y > 0) \wedge (x = y + 2) \Rightarrow x > 2$). Appendix B provides more examples illustrating these simplifications.

2.3 Complication

To *complicate the expressions*, a straightforward strategy is to incorporate additional operators. For example, given an atomic constraint $h = e_1 \bowtie e_2$, we can introduce an additional expression, denoted by e', and derive a more complex constraint $\tilde{h} = e_1 \bowtie (e_2 \oplus e')$.

$$(\mathrm{M}_1) \begin{cases} a(b+c) = 152 \\ b(c+a) = 162 \\ c(a+b) = 170 \\ a, b, c \in \mathbb{N}^+ \end{cases} \Rightarrow (\mathrm{M}_2) \begin{cases} a(b+c) = 152 \oplus_1 e_1'(z_1) \\ b(c+a) = 162 \oplus_2 e_2'(z_2) \\ c(a+b) = 170 \oplus_3 e_3'(z_3) \\ a, b, c \in \mathbb{N}^+, z_1, z_2, z_3 \in \mathbb{R} \end{cases}$$

$$\Rightarrow (\mathrm{M}_3) \begin{cases} a(b+c) + z_1 = 152 \\ b(c+a) - z_2 = 162 \\ c(a+b) = 170 \\ z_1 = 114 \\ z_2 = 36 \\ a, b, c \in \mathbb{N}^+ \end{cases} \Rightarrow (\mathrm{M}_4) \begin{cases} a(b+c) + d = 152 \\ b(c+a) - e = 162 \\ c(a+b) = 170 \\ d + e = 150 \\ d - e = 78 \\ a, b, c, d, e \in \mathbb{N}^+ \end{cases}$$

However, such a strategy is non-trivial in practice. The first challenge lies in the *validity* aspect, i.e., the math problem is often carefully designed, and thus a random mutation may ruin their well-defined structure. Consider the running example problem (M_1), which has been normalized for simplicity. In this problem, a reckless mutation can easily violate the positive integer constraints, causing the problem ill-defined and unsolvable.

To address this issue, we equip each mutation with an auxiliary variable, followed by symbolic solvers to ensure the problem remains well-defined. Continuing with the previous example, we introduce three auxiliary variables, denoted by z_1, z_2, z_3, and then mutate the problem as (M_2), where $\oplus_1, \oplus_2, \oplus_3 \in \{+, -, \times, \div\}$ represent three random operators. Furthermore, we instantiate e_1', e_2', e_3' by interpreted functions, i.e., $e_1' = \mathrm{foo}_1(z_1)$, $e_2' = \mathrm{foo}_2(z_2)$, and $e_3' = \mathrm{foo}_3(z_3)$, where $\mathrm{foo}_1, \mathrm{foo}_2, \mathrm{foo}_3$ are randomly selected from $\mathrm{foo}(z) = z \mid \log(z) \mid \exp(z) \mid \arcsin(z) \mid \cdots$. For our running example, we simply choose the identity function for $\mathrm{foo}_1, \mathrm{foo}_2, \mathrm{foo}_3$, and set $\oplus_1 = -, \oplus_2 = +, \oplus_3 = \times$. Using symbolic solvers to compute a feasible solution of (z_1, z_2, z_3), we derive a new and well-defined problem (M_3).

The subsequent challenge is to ensure the *diversity* of the mutated problems, which now becomes how to make the solutions of auxiliary variables sufficiently diverse. This is essentially a model counting problem [26, 27], and current symbolic solvers still underperform in this regard [28]. To this end, we instead opt for auxiliary variable solution generation via the *projected Markov chain Monte Carlo* (projected MCMC) [14, 15]. Simply put, projected MCMC first perturbs a subset of variables (projected random walk), and then resolves the remaining part (inverse projection via symbolic solvers), which ensures both diversity and validity of the variable solutions.

Finally, to *complicate the constraints*, one can easily reverse the process of simplification. For our running example, we can reverse the Gaussian elimination with refreshed variables, obtaining the final form (M_4), which is then included as a new problem in the dataset.

3 Informalization

Informalization aims to translate a formal problem back to natural language without the loss of soundness [29]. As shown in Example 1, a simple, one-line instruction follows the formally posed SMT-LIB problem, serving as the input. Then, GPT-4 interprets the formal problem as a new math word problem.

The key challenge of informalization lies in ensuring a *consistent* conversion, i.e., the natural language problem informalized by GPT-4 should align with the formal solution given by symbolic solvers. Since it is difficult to directly measure this consistency, we instead use GPT-4 to generate a solution for each informalized problem, and then calculate the *consistency rate* between the solutions from GPT-4 and those from symbolic solvers as a surrogate metric. Furthermore, we observe that, if the problem is *incorrectly* informalized, GPT-4's solutions almostly cannot be confirmed by symbolic solvers (i.e., zero false positive). Therefore, the surrogate consistency rate can be regarded as a lower bound to the true consistency rate.

Example 1: Informalization

```
(declare-fun sara_shoes_cost () Real)
(declare-fun sara_dress_cost () Real)
(declare-fun sara_total_cost () Real)
(declare-fun rachel_budget () Real)
(assert (= sara_shoes_cost 50.0))
(assert (= sara_dress_cost 200.0))
(assert (= sara_total_cost (+ sara_shoes_cost sara_dress_cost)))
(assert (= rachel_budget (* 2 sara_total_cost)))
(check-sat)
(get-value (rachel_budget))
```

Translate the math problem formulated with SMT-LIB back to a natural language problem.

GPT-4 output:
Sara bought a pair of shoes for $50.00 and a dress for $200.00. If Rachel has twice the amount that Sara spent in total, how much is Rachel's budget?

To further improve conversion consistency, in addition to the basic zero-shot learning template in Example 1, we investigate the effects of the following operations, whose detailed examples are available in Appendix C.

(1) **Mutation**. Mutation complicates the problem, making the informalization more difficult. Therefore, we first analyze the informalization error caused by the mutation.

(2) **Few-shot learning**. Few-shot examples offer a stronger instruction to the LLM, and also introduce the randomness when aided by random retrieval.

(3) **Comment generation**. Recognizing that GPT-4 is unfamiliar with SMT-LIB's prefix expressions, we automatically convert these into the infix format, included as comments.

(4) **Math-word instruction**. We simply append one more sentence "Ensure to be a math word problem" in the prompt. With this instruction, informalization tends to imbue digits with some practical meaning (e.g., 7 ⇒ one week).

Table 1: Consistency rate of six different operations used in informalization: (1) Mutation; (2) Few-shot learning; (3) Comment generation; (4) Math-word instruction; (5) Problem modification; (6) Variable refresh. We recommend two patterns (i.e., P1: 1-5 and P2: 1-3&6), both of which can achieve satisfactory results.

Ops	(1)	(2)	(3)	(4)	(5)	(6)	Rate (%)
–	✗	✗	✗	✗	✗	✗	75.6 (–)
	✓	✗	✗	✗	✗	✗	41.6 (↓)
–	✓	✓	✗	✗	✗	✗	76.2 (↑)
	✓	✓	✓	✗	✗	✗	87.6 (↑)
P1	✓	✓	✓	✓	✓	✗	90.5 (↑)
P2	✓	✓	✓	✗	✗	✓	97.1 (↑)

(5) **Problem modification**. For mutated problems, rather than generating a new informalization, we prompt GPT-4 to modify the original informalization result.

(6) **Variable refresh**. We standardize the naming of all introduced variables (e.g., `rachel_budget` ⇒ `x_1`), to eliminate the impact of math word problems.

Different combinations of the above operations result in different patterns. The effects of some typical patterns are shown in Table 1, where the results are evaluated on 1,000 problems randomly sampled from GSM8K. The basic pattern in Example 1 yields a consistency rate of 75.6%. The mutation operation alone indeed degrades the consistency, but its combination with other operators can further boost the informalization performance. In practice, we use two different patterns for different informalization styles: the first pattern (P1) tends to generate math word problems, whereas the problems generated by the second pattern (P2) tend to be pure math problems.

4 Experiments

In this section, we conduct a series of experiments to answer the following four research questions:

RQ1: Efficacy – Using our data generation framework, can the fine-tuned model achieve better performance compared with existing models?

RQ2: Efficiency – Given the same data generation budget, is the generated data from our framework better than that from the state-of-the-art data generation framework?

RQ3: Generability – Is the effecitveness achieved by our framework due to potential data contamination introduced during the generation process?

RQ4: Scalability – With more data generated, can our approach be continually effective in further improving model performance?

4.1 Experimental Setup

Dataset. We conduct our data generation on the training sets of two popular mathematical reasoning benchmarks: GSM8K [10] and MATH [11]. GSM8K is a dataset comprising high-quality grade school math problems, which contains 7,473 training data and 1,319 testing data. MATH is a dataset comprised of challenging competition math problems, spanning seven subjects including Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus. There are 7,500 training data and 5,000 testing data in the MATH dataset. Additionally, we include two mathematical reasoning datasets, i.e., SVAMP [12] and ASDiv [18], to evaluate the out-of-domain generalizability of the models fine-tuned on the data generated from GSM8K and MATH datasets.

Comparison Methods. In our experiments, we compare the models trained using our generated data with existing state-of-the-art open-source mathematical reasoning models, including WizardMath [30], MuggleMATH [31], MAmmoTH [32], and MetaMath [33]. We also conduct a thorough comparison between our math generation method and the bootstrapping method employed in MetaMathQA [33], which is presently the most extensive open-source dataset for mathematical reasoning.

Data Generation Details. We use our mutation mechanism to generate a series of problems with varying levels of difficulty, and the specifics are as follows. Starting with a problem from the original dataset as a seed, we first perform simplification to the problem, and define this new version as level-0. Then, we randomly apply one expression complication step and one constraint complication step to the level-0 version, deriving a more difficult problem (level-1 version); and such complications can be repeated to obtain more difficult problems. For the GSM8K dataset, we create datasets across five levels of difficulty, with 30K examples at level-0 and 100K examples for the remaining four levels. As for the MATH dataset, we establish four levels of difficulty, where level-0, level-1, level-2, and level-3 contain 70K, 120K, 120K, and 120K examples, respectively. Particularly, for some problems in the MATH dataset that cannot be solved by symbolic solvers, we directly prompt GPT-4 to rephrase the problem and ignore the solution verification. The number of generated problems without solution verification varies across problem categories, and the details can be referred to Appendix D. In total, we generated 860K math problems based on the proposed framework to construct our dataset.

Each generated math problem consists of a natural language problem description informalized by GPT-4 (version 0710), a final answer outputted by the symbolic solver, and a reasoning path from the problem to the final answer. The reasoning path for each problem is also generated by GPT-4, which is further verified by the corresponding answer derived from symbolic solvers.

More implementation details about training hyperparameters, instruction prompts, and symbolic solver integration, are included in Appendix D.

4.2 Empirical Results

RQ1: Efficacy. Using the generated math dataset, we fine-tune the LLaMA-2 base models of 7B and 13B parameter sizes, as well as the Mistral 7B base model. The fine-tuned models, as well as the comparison methods, are evaluated on the GSM8K and MATH datasets.

Table 2: Performance comparison among existing mathematical reasoning models fine-tuned on three base models (LLaMA-2 7B, LLaMA-13B, and Mistral 7B). The best performance is in bold. The delta performance between our model and other SOTA LLMs on each dataset is also reported.

Model	#Dataset	LLaMA-2 7B Base		LLaMA-2 13B Base		Mistral 7B Base	
		GSM8K	MATH	GSM8K	MATH	GSM8K	MATH
WizardMath	>240K	54.9	10.7	63.9	14.0	83.2	33.0
MuggleMATH	157K	68.4	8.4	74.0	9.4	-	-
MAmmoTH†	260K	50.5	10.4	56.3	12.9	61.9	17.5
MetaMath	395K	66.5	19.8	72.3	22.4	77.7	28.2
Ours	860K	**79.0**	**30.4**	**84.1**	**33.7**	**86.8**	**37.3**
Δ		↑ 10.6	↑ 10.6	↑ 10.1	↑ 11.3	↑ 3.6	↑ 4.3

† Model performance is re-evaluated using Pass@1 of CoT prompt.

As shown in Table 2, our approach achieves the best performance among the baseline models across different model scales. Compared to LLMs with the LLAMA-2 7B base model, our model demonstrates a fair improvement in accuracy, surpassing them by at least 10.6% on the two datasets. For our model fine-tuned using the LLAMA-2 13B base model, our model achieves an accuracy of 84.1% and 33.7%, outperforming existing SOTA model by 10.1% and 11.3%. When fine-tuned on the Mistral 7B base model, our model still attains the best performance with an increase in accuracy of 3.6% and 4.3%, respectively. Notably, our model even slightly outperforms GPT-3.5-Turbo (80.8% and 34.1%) by 6.0% on the GSM8K dataset and 3.2% on the MATH dataset.

In addition to the above competitors, we also compare our models with tool-based models, and provide the results in Appendix E.2. We summarize two observations here. First, tool-based models tend to over-rely on external tools, and thus do not necessarily improve the inherent reasoning ability of LLMs. Second, although the tool-based models perform better on the MATH dataset (which frequently entails complex calculations), they still underperform on the GSM8K dataset (which emphasizes knowledge reasoning but involves simpler calculations).

RQ2: Efficiency. To illustrate the data efficiency of our method, we carry out a comparative experiment with current SOTA method MetaMathQA [33]. The MetaMathQA dataset comprises 240K data bootstrapped from the GSM8K training dataset and 155K data from the MATH training dataset. To ensure a fair comparison with MetaMathQA, we use the same data budget. Additionally, we expand the MATH data of the MetaMathQA dataset to 430K, aligning its size with that of our generated data. Then, we fine-tune LLaMA-2-7B models on the 240K GSM8K augmented data, as well as the 155K and 430K MATH augmented data, respectively.

The performance of fine-tuned models are given in Table 3. We also evaluate the models on two out-of-domain datasets, SVAMP and ASDiv. The results confirm the efficiency of our framework. With an equal generation budget of 240K for the GSM8K dataset and 155K for the MATH dataset, our method exhibits accuracy improvements, ranging from 2.1% to 24.4% across the four datasets. This superiority is consistent with the model trained on 430K MATH generation data, with improvements of 19.9%, 3.2%, 12.7%, and 19.2%, respectively.

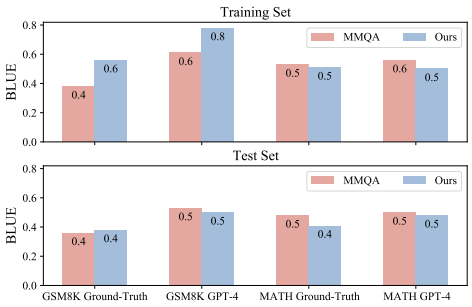
RQ3: Generability. Despite we carefully ensure that our mutations on the training set do not access the test set, we still provide a series of analysis about the potential data contamination or overfitting issues. We first use the memorization detection method introduced in Minerva [34]. Specifically, we select 150 problems with the highest majority vote score and then compute the BLUE score [35] on solutions of trained Mistral 7B, GPT-4, and the ground-truth. The results of our method and MetaMath are provided in Figure 3, which show that our BLUE score on the test set is (1) much lower than our BLUE score on the training set; (2) is consistent with that of MetaMath. Hence, there is no evidence that the mutation contaminates the test set.

Second, in addition to the two out-of-distribution datasets SVAMP and ASDiv, we conduct experiments on another benchmark DyVal [9], which avoids the leak of test set through dynamically generating new benchmarks. The results of our models, alongside those of the comparison models, are provided in Appendix E.3. In summary, our models demonstrated superior performance in 11 out

Table 3: Comparison between our method and MetaMathQA (MMQA) with the same data budgets. The models are fine-tuned using LLaMA-2-7B base model, and evaluated on GSM8K, MATH, SVAMP, and ASDiv datasets. The results illustrate the high quality of our generated data.

Method	Training Dataset GSM8K	Math	Performance GSM8K	Math	SVAMP	ASDiv
MMQA	240K	0K	66.1	5.8	61.7	72.5
Ours	240K	0K	72.7	8.2	78.8	79.2
	Δ		↑ 6.6	↑ 2.3	↑ 17.1	↑ 6.7
MMQA	0K	155K	28.6	19.9	49.0	61.0
Ours	0K	155K	42.1	22.0	73.4	64.1
	Δ		↑ 13.5	↑ 2.1	↑ 24.4	↑ 3.1
MMQA	240K	155K	67.5	21.7	64.1	76.0
Ours	240K	155K	73.7	23.4	85.2	81.1
	Δ		↑ 6.2	↑ 1.7	↑ 21.1	↑ 5.0
MMQA*	0K	430K	35.0	25.6	66.4	51.1
Ours	0K	430K	54.9	28.8	79.1	70.3
	Δ		↑ 19.9	↑ 3.2	↑ 12.7	↑ 19.2

Figure 3: BLUE scores between the output of our fine-tuned model, versus the ground-truth solution and GPT-4 output. The model is fine-tuned on the Mistral 7B base model, and MetaMath Mistral 7B model is also included as a reference. The results show that our method does not induce data contamination.



of 12 cases and delivered competitive results in the remaining case. Particularly, as the complexity of the tasks increased, our models exhibited a relatively robust performance compared to other models.

Finally, we include an additional experiment on the Hungarian High School National Finals Exam dataset [36], whose problems are newly collected at 2023. We manually check 33 testing problems based on the provided ground-truth answer, and our model correctly solved 14 problems and partially solved 6 problems, resulting in an exam score of 44. The result is comparable to GPT3.5-turbo (i.e., 41 exam score), conforming the generalizability of our method.

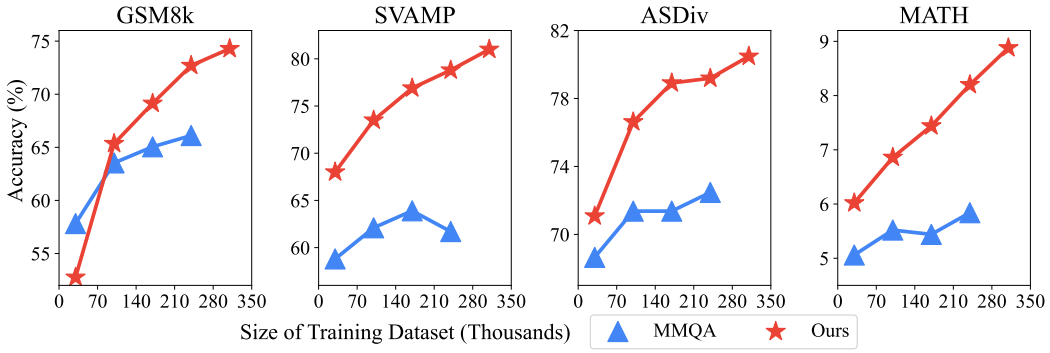


Figure 4: Performance curves of the LLaMA-2-7B models fine-tuned on various scales of datasets. The two datasets are generated by our approach and MetaMath (MMQA). The performance can be consistently enhanced by increasing the amount of data generated using the proposed framework.

RQ4: Scability. To explore the scalability of our framework, we fine-tune the LLaMA-2 7B model using our generated datasets of various sizes and difficulties. To be specific, we progressively incorporate a 30K dataset, along with four additional 70K datasets generated from GSM8K. Note that these five datasets are randomly sampled from five different levels of difficulty. Five LLaMA-2-7B models are fine-tuned based on these datasets, and the scalability curves are shown in Figure 4. We also include MetaMathQA with the same data settings as a reference. Since MetaMathQA cannot inherently group the dataset into various difficult levels, we construct five datasets by incrementally random sampling from the GSM8K subset of the MetaMathQA dataset.

The results presented in Figure 4 indicate the promising scalability of our method. That is, as the size of data increases, the accuracy of the model consistently improves. In contrast, the performance enhancement observed in MetaMathQA is limited and starts to diminish as the data size reaches 70K. We also present the models' performance on the other three out-of-domain datasets in this case, i.e., SVAMP, ASDiv, and MATH datasets. The results demonstrate that the scalability of our method is robust and generalizable, while MetaMathQA hardly guarantees such consistency.

We also investigate the diversity gain relative to the original dataset for each difficulty level. The results are provided in Appendix E.4. It is observed that the dataset consisting of the same difficulty level cannot further improve the diversity with a larger data budget. On the contrary, the diversity gain of the dataset comprising all difficulty levels continues to increase as the data budget grows.

5 Related Work

Recent surveys [37–39] have comprehensively discussed the current advances in the mathematical reasoning of LLMs. Here, we review three main lines of existing work on enhancing the mathematical reasoning for LLMs related to our study: prompt-based methods, rephrasing-based methods, and tool-based methods.

Prompt-based Method. Prompt-based methods aim to harness the inherent capabilities of LLMs by carefully designing appropriate input prompts without tuning the model parameters. This line of work starts from the observation that LLMs can effectively tackle more math problems when provided with a simple Chain-of-Thought (CoT) prompt, i.e., "Let's think step by step" [40]. Building upon the CoT prompt, Wang et al. [41] further propose to consolidate multiple reasoning paths based on the self-consistency of correct answers. Later, several researchers propose to prompt LLMs to decompose complex problems. For example, Zhou et al. [42] introduce the least-to-most strategy that prompts LLMs to break down the original problem into a series of sub-problems. Khot et al. [43] further boost this strategy, by assigning each sub-problem to the corresponding LLM that is specifically optimized for it. Finally, few-shot prompting, e.g., Few-shot CoT [44] and Complex CoT [45], has also been studied to enhance the reasoning performance of LLMs. To further improve the few-shot prompting, the prompt retrieval is proposed to automatically select high-quality examples [46, 47], while the prompt compression is explored to include more examples in restricted context by pruning each example [48].

Rephrasing-based Method. The second line of existing work aims to generate additional math data, based on which the mathematical reasoning capability of LLMs can be established via supervised fine-tuning. To address the data scarcity issue, current research mainly focuses on rephrasing the problem or the answer. For the answer rephrasing, Magister et al. [49] adopt the PaLM and GPT-3 to generate CoT math data, resulting in improved performance of the T5 model on math reasoning tasks. To mitigate the inclusion of incorrect answers during the supervised fine-tuning, RFT [50] introduces a rejection sampling strategy, whereas AFT [51] trains an LLM to categorize them. Regarding the problem rephrasing, WizardMath [30] proposes a reinforced evol-instruct method. It instructs ChatGPT and trains a new LLM to rephrase the problem, equipped by a reward model for evaluating the quality of generated problems. Combining the rephrasing of problems and answers together, MuggleMATH [31] builds the AugGSM8K dataset based on prompting GPT-3.5 and GPT-4. MetaMath [33] develops a question bootstrapping method based on LLMs, unifying rephrasing, self-verification [52], FOBAR [53], and answer augmentation strategies, obtaining the MetaMathQA. Xwin-Math [54] is a peer study that significantly enhances the reasoning capacity of LLMs using problems generated by GPT-4 Turbo. In contrast, our work focuses on generating verifiable problems through controllable mutations, rather than relying entirely on the GPT model.

Our proposed method also falls into this category. In contrast to existing methods directly prompting LLMs to rephrase the problem, we mutate the problem in the formal symbolic space, resulting in a more controllable mutation mechanism that ensures both the validity and diversity of the generated problems. Moreover, the quality of reasoning paths is also guaranteed by the symbolic solvers.

Tool-based Method. Tool-based methods aim to enhance the math solving performance of LLMs by instructing them to use external math tools. For instance, PoT (Program of Thought) [55] and PAL [56] propose to prompt the LLMs to delegate the computation to a program interpreter, which can be executed to obtain the final answer. To further improve the tool-using ability of LLMs, MathCoder [57] constructs a math dataset containing problems and their code-based solutions for the supervised fine-tuning; MAmmoTH [32] builds a dataset that combines CoT and PoT reasoning, enabling LLMs to perform hybrid inference. Since the interaction with math tools can further boost the performance of LLMs, TVA [58] includes the Isabelle theorem prover to check each reasoning step and guide the reflection of LLMs; Tora [59] generates interactive tool-use trajectories on mathematical datasets and then performs imitation learning on the annotations.

Our proposed method shares some similarities with tool-based approaches as both involve symbolic solvers. However, rather than using external tools to solve mathematical problems, our approach aims

to explore the inherent reasoning capability of LLMs. Therefore, symbolic solvers are only used to ensure the validity of the generated data as well as the correctness of the generated reasoning paths.

6 Limitations

The Capability of Symbolic Solvers. The effectiveness of our approach significantly hinges on the symbolic solvers. However, existing mathematical tools (e.g., Z3 [20], SymPy [23], and SciPy [24]) face limitations when it comes to expressing and solving a wide array of mathematical problems. For instance, the Z3 SMT solver struggles with expressing higher-order concepts like `gcd` and `lcm`, while the SymPy encounters difficulties in solving inequalities involving multiple variables. In our framework, we integrate five mathematical tools, i.e., Z3, CVC4 [60], MathSAT [22], SymPy, and SciPy, and employ SMT-LIB [19] as a unified formal language to enhance the performance of symbolic solving.

The Expressiveness of Mutations. The mutation operators used within our framework remain limited, especially in generating more difficult problems (e.g., college- and even IMO-level math problems). One of our future work is to introduce more mutation operators, further increasing the problem difficulty. A possible strategy is the problem fusion [61], which fuses two formal problems into a single, new problem, rather than merely modifying an individual problem. Moreover, the informalization facilitated by LLMs can effectively mitigate the unnaturalness issue stemming from brute-force fusion.

The Dependence on GPT-4. GPT-4 is involved in our framework to carry out the informalization and generate the reasoning paths. We also consider the possible solutions that the dependence on GPT-4 can be gradually removed. First, by leveraging our generated formal-informal pairs, we can fine-tune a new LLM specifically for the informalization. Second, it is possible to bypass the generation of reasoning paths, through curriculum learning [62, 63] instead of supervised fine-tuning. Particularly, the reward in the curriculum learning can be determined by whether the generated solution is consistent with symbolic solvers , and the curriculum progresses by incorporating problems of various difficulty levels.

Broader Impact

The paper aims to advance the field of math data generation. There are many potential societal consequences of our work, and we firmly believe that the majority of these impacts are positive and none which we feel must be highlighted here.

7 Conclusion

This paper explores the question of whether sufficient exposure to high-quality mathematical data could enhance LLMs' inherent mathematical reasoning capability. We identify a key challenge in balancing diversity and validity in current math problem generation methods. To tackle this challenge, we propose a neuro-symbolic framework that initially generates formal mathematical problems and then informalizes them back into natural language versions. By casting the data generation into the formal language space, the diversity and validity of the generated math problems can be effectively ensured by the systematic sampling and symbolic solvers. Building upon this, we carefully devise a mutation mechanism, establishing the math dataset encompassing various difficulty levels, and prompt the LLMs to accomplish informalization. Through empirical experiments, we demonstrate that our neuro-symbolic data generation framework significantly enhances the performance of various LLMs in mathematical reasoning tasks, surpassing the current state-of-the-art open-source models. The results also suggest a promising pathway for further enhancing LLMs' mathematical capabilities.

In future work, we intend to expand the expressiveness of mutations and enhance the capability of symbolic solvers to support more types of problems, such as inequality problems. Our goal is to offer a data generation framework to automatically generate high-quality, supervised datasets for LLMs. We expect that our neuro-symbolic data generation framework can provide a potential solution for LLMs to solve the problem of data scarcity, and thereby facilitate in building more LLMs in downstream tasks. Further, our framework has the potential to be integrated with recent studies [64], which only require problems and final answers.

Acknowledgment

We appreciate the anonymous reviewers for their valuable insights and helpful comments. This work is supported by the National Natural Science Foundation of China (Grants #62025202), the Frontier Technologies R&D Program of Jiangsu (BF2024059), and the Key Program of Jiangsu Science Foundation (BK20243012). Xian Zhang (zhxian@microsoft.com) and Xiaoxing Ma (xxm@nju.edu.cn) are the corresponding authors.

References

[1] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems 35*, 2022.

[2] Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic AI: the 3rd wave. *Artificial Intelligence Review*, 56(11):12387–12406, 2023.

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

[4] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.

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

[6] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 2023.

[7] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023.

[8] Paulo Shakarian, Abhinav Koyyalamudi, Noel Ngu, and Lakshmivihari Mareedu. An independent evaluation of chatgpt on mathematical word problems (mwp). *arXiv preprint arXiv:2302.13814*, 2023.

[9] Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. Dyval: Graph-informed dynamic evaluation of large language models. *CoRR*, abs/2309.17167, 2023.

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

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[12] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

[13] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

[14] Weiming Feng, Kun He, and Yitong Yin. Sampling constraint satisfaction solutions in the local lemma regime. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1565–1578, 2021.

[15] Zenan Li, Yuan Yao, Taolue Chen, Jingwei Xu, Chun Cao, Xiaoxing Ma, L Jian, et al. Softened symbol grounding for neuro-symbolic systems. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[17] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

[18] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 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, 2020.

[19] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2017. Available at `www.SMT-LIB.org`.

[20] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[21] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, et al. cvc5: A versatile and industrial-strength smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442. Springer, 2022.

[22] Roberto Bruttomesso, Alessandro Cimatti, Anders Franzén, Alberto Griggio, and Roberto Sebastiani. The mathsat 4 smt solver: Tool paper. In *Computer Aided Verification: 20th International Conference, CAV 2008 Princeton, NJ, USA, July 7-14, 2008 Proceedings 20*, pages 299–303. Springer, 2008.

[23] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[24] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[25] Leonardo De Moura and Grant Olney Passmore. The strategy challenge in smt solving. In *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*, pages 15–44. Springer, 2013.

[26] Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[27] Mark Jerrum and Alistair Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. *Approximation Algorithms for NP-hard problems, PWS Publishing*, 1996.

[28] Stefano Ermon, Carla Gomes, and Bart Selman. Uniform solution sampling using a constraint solver as an oracle. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 255–264, 2012.

[29] Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

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

[31] Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. Query and response augmentation cannot help out-of-domain math reasoning generalization. *CoRR*, abs/2310.05506, 2023.

[32] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *CoRR*, abs/2309.05653, 2023.

[33] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

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

[35] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[36] Keiran Paster. Testing language models on a held-out high school national finals exam. `https://huggingface.co/datasets/keirp/hungarian_national_hs_finals_exam`, 2023.

[37] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, pages 225–237, 2024.

[38] Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. A survey of deep learning for mathematical reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 14605–14631, 2023.

[39] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[40] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, pages 22199–22213, 2022.

[41] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[42] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[43] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[44] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, pages 24824–24837, 2022.

[45] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[46] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[47] Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[48] Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, and Mao Yang. Boosting llm reasoning: Push the limits of few-shot learning with reinforced in-context pruning. *arXiv preprint arXiv:2312.08901*, 2023.

[49] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adámek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 1773–1781, 2023.

[50] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023.

[51] Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. Making large language models better reasoners with alignment. *CoRR*, abs/2309.02144, 2023.

[52] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, 2023.

[53] Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James T Kwok. Forward-backward reasoning in large language models for mathematical verification. *arXiv preprint arXiv:2308.07758*, 3, 2023.

[54] Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. Common 7b language models already possess strong math capabilities. *CoRR*, abs/2403.04706, 2024.

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

[56] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10764–10799, 2023.

[57] Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*, 2023.

[58] Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. Don't trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*, 2024.

[59] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.

[60] Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. Cvc4. In *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*, pages 171–177. Springer, 2011.

[61] Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating smt solvers via semantic fusion. In *Proceedings of the 41st ACM SIGPLAN Conference on programming language design and implementation*, pages 718–730, 2020.

[62] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[63] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

[64] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*, 2024.

[65] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2023.

[66] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the 1th International Conference on Learning Representations*, 2022.

[67] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model. Technical report, 2023.

[68] Marco Gario and Andrea Micheli. Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In *SMT Workshop 2015*, 2015.

[69] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[70] Zhoulai Fu and Zhendong Su. Xsat: a fast floating-point satisfiability solver. In *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II 28*, pages 187–209. Springer, 2016.

[71] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. Dl2: training and querying neural networks with logic. In *International Conference on Machine Learning*, pages 1931–1941. PMLR, 2019.

[72] Zenan Li, Zehua Liu, Yuan Yao, Jingwei Xu, Taolue Chen, Xiaoxing Ma, L Jian, et al. Learning with logical constraints but without shortcut satisfaction. In *The Eleventh International Conference on Learning Representations*, 2022.

[73] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

A Broader Impacts

The paper aims to advance the field of math data generation. There are many potential societal consequences of our work, and we firmly believe that the majority of these impacts are positive and none which we feel must be highlighted here.

B Examples of Simplification

Example 2: Simplification strategy on Problem (No. 41) of the GSM8K dataset.

Autoformalization:
Translate the natural language problem into SMT-LIB language: "Sara bought a pair of shoes for $50.00 and a dress for $200.00. If Rachel has twice the amount that Sara spent in total, how much is Rachel's budget?"

GPT-4 output:

```
(declare-fun sara_shoes_cost () Real)
(declare-fun sara_dress_cost () Real)
(declare-fun sara_total_cost () Real)
(declare-fun rachel_budget () Real)
(assert (= sara_shoes_cost 50.0))
(assert (= sara_dress_cost 200.0))
(assert (= sara_total_cost (+ sara_shoes_cost sara_dress_cost)))
(assert (= rachel_budget (* 2 sara_total_cost)))
(check-sat)
(get-value (rachel_budget))
```

Mutation & Informalization:

```
(declare-fun sara_dress_cost () Real)
(declare-fun sara_shoes_cost () Real)
(declare-fun sara_total_cost () Real)
(assert (= sara_dress_cost 200.0))
(assert (= sara_shoes_cost 50.0))
(assert (= sara_total_cost (+ sara_shoes_cost sara_dress_cost)))
(check-sat)
(get-value (sara_total_cost))
```

Translate the math problem formulated with SMT-LIB back to a natural language problem.

GPT-4 output:
Sara went shopping and bought a dress for $200.00 and a pair of shoes for $50.00. What is the total amount Sara spent on her shopping trip?

We elucidate the simplification strategy through Example 2, which is selected from the GSM8K dataset. In this formal problem, we can conduct one step of Gaussian elimination, i.e., randomly solving and removing a variable (e.g., `rachel_budget` $= 500$). Then, we derive the new problem with its informalized version. Furthermore, we present another example, specifically Example 3, sourced from the MATH dataset. Here, the new problem is generated through the simplification strategy that randomly performs some calculations.

C Examples of Informalization

Various prompts yield different styles of informalization. To illustrate this, we provide two specific examples as follows. In Example 4, we do not refresh the variable, and instruct the informalization by adding the prompt "Ensure to be a math word problem." The result of GPT-4's informalization tends

Example 3: Simplification strategy on Problem (No. 496) of the MATH dataset.

Autoformalization:
Translate the natural language problem into SMT-LIB language: "Compute $\\displaystyle \\frac{2+4-8+16+32-64}{4+8-16+32+64-128}$."

GPT-4 output:

```
(declare-fun num () Real)
(declare-fun den () Real)
(declare-fun fraction () Real)
; Define the numerator and denominator expressions
(assert (= num (+ 2 4 (- 8) 16 32 (- 64))))
(assert (= den (+ 4 8 (- 16) 32 64 (- 128))))
; Define the fraction as the numerator divided by the denominator
(assert (= fraction (/ num den)))
(check-sat)
(get-value (fraction))
```

Mutation & Informalization:

```
(declare-fun num () Real)
(declare-fun den () Real)
(declare-fun fraction () Real)
; Define the numerator and denominator expressions
(assert (= num (+ (- 2) 16 (-32))))
(assert (= den (+ (- 4) 32 (- 64))))
; Define the fraction as the numerator divided by the denominator
(assert (= fraction (/ num den)))
(check-sat)
(get-value (fraction))
```

Translate the math problem formulated with SMT-LIB back to a natural language problem.

GPT-4 output: Calculate the value of the fraction \\(\\frac{-2 + 16 - 32}{-4 + 32 - 64}\\).

to envelop the problem with common knowledge, culminating in the creation of a math word problem. In contrast, Example 5 involves refreshing each variable and eliminating the prompt "Ensure it is a math word problem." Hence, its corresponding informalization renders the new problem more abstract and devoid of extraneous elements.

D Additional Experimental Details

Generation Details. In the GSM8K dataset, each problem can be accurately formalized into the SMT-LIB format and successfully mutated into a new version. However, some problems, particularly in the precalculus and geometry categories of the MATH dataset, cannot be formalized or mutated effectively. Overall, out of a total of 7,500 problems, 822 cannot be formalized into the SMT-LIB format, and approximately 3,600 formalizations are inaccurate although they remain usable for the data generation. To address this issue, we strategically added a proportional number of mutated problems derived by directly prompting GPT-4, bypassing solution verification. The detailed counts of these problems are presented in Table 4.

Training Details. In this paper, we fully fine-tune the LLAMA-2-7B and LLAMA-2-13B models using four H800 NVIDIA GPUs. Each model is trained for 3 epochs with a batch size of 128 and a learning rate of 2e-5. For the fine-tuning of the LLAMA-2-70B model, we adopt the QLoRA [65] method with a learning rate of 1e-4. The rank and alpha of LoRA [66] are set to 96 and 16, respectively, with a dropout rate of 0.05 between the two matrices. The LoRA modules are added to both the attention and MLP layers. The 70B model is fine-tuned using eight A800 NVIDIA GPUs.

Example 4: Informalization to be a math word problem (Problem No.14 in GSM8K).

```
(declare-fun pages_per_minute () Real)
(declare-fun total_pages () Int)
(declare-fun time_hours () Int)
(assert (= pages_per_minute (/ 2 5)))
(assert (= total_pages 144))
; (time_hours = (((total_pages / pages_per_minute) * 1/60) * 1/2))
(assert (= time_hours (* (* (/ total_pages pages_per_minute)
                (/ 1 60)) (/ 1 2))))
(check-sat)
(get-value (time_hours))
```

Translate the math problem formulated with SMT-LIB back to a natural language problem. Please ensure to be a math word problem.

GPT-4 output:
Jamie has a book with 144 pages that she wants to read. She reads at a pace of 2/5 pages per minute. If she reads for half the time it would normally take her to read the book at this pace, how many hours will she have read?

Example 5: Informalization to be a pure math problem (Problem No.14 in GSM8K).

```
(declare-fun x_0 () Real)
(declare-fun x_1 () Int)
(declare-fun x_2 () Int)
(assert (= x_0 (/ 2 5)))
(assert (= x_1 1))
; (x_2 = (2/5 * (x_1 / x_0)))
(assert (= x_2 (* (/ 2 5) (/ x_1 x_0))))
(check-sat)
(get-value (x_2))
```

Translate the math problem formulated with SMT-LIB back to a natural language problem.

GPT-4 output:
A fraction representing a number is 2/5. If you have the number 1 and you want to find out how many times this fraction fits into it, what would be the result?

Prompt 1: Training Prompt

Below is an instruction that describes a task. Write a response that appropriately completes the request.\n \n ### Instruction:\n{instruction}\n\n### Response:\n

Table 4: Detailed count of problems generated by GPT-4 and mutated during the generation process of the MATH dataset. The numbers 1-7 correspond to Algebra, Counting and Probability, Geometry, Intermediate Algebra, Number Theory, Prealgebra, and Precalculus, respectively.

Category	1	2	3	4	5	6	7	Total
# LLM generated	20.4K	9.2K	6.4K	15.4K	8.4K	12.4K	7.8K	80K
# Mutated	97.4K	36.2K	31.2K	60.3K	40.7K	59.2K	24.1K	350K

When fine-tuning the Mistral 7B model, the same training settings as LLAMA-2-7B are used, except

for the learning rate, which is set to 5e-6. Moreover, we adopt the instruction template Prompt 1 used in Alpaca [67] for fine-tuning each model.

Prompt 2: Testing Prompt

Below is an instruction that describes a task. Write a response that appropriately completes the request.\n\n### Instruction:\n{instruction}\n\n### Response:\n

Evaluation Details. We evaluate each fine-tuned model using a zero-shot evaluation protocol with the corresponding recommended instruction template. Our model is evaluated using the following instruction template Prompt 2, which is consistent with our training instruction template.

For answer extraction and accuracy calculation, we follow the code of WizardMath [30] to extract the answer after the phrase "The answer is".

Symbolic Solvers. We integrate five symbolic solvers, Z3, CVC4, MathSAT, SymPy, and SciPy based on the PySMT framework [68]. To be specific, the PySMT intrinsically includes Z3, CVC4, and MathSAT, and we further extend its support to encompass SMT-LIB version 2.5, which incorporates more commands like `define-rec`. Next, we proceed to serialize the SMT-LIB format into SymPy expressions, and attempt to find solutions using SymPy's `solve` function. In addition, the SymPy expressions are also encoded as NumPy [69] functions, thereby enabling the using of SciPy's optimization modules, such as `differential_evolution` and `minimize`. Note that we introduce a fuzzy-logic-like strategy [70–72] in the encoding, which combines the equalities and inequalities into a loss function, subsequently enabling optimization methods for problem-solving tasks.

E Additional Experimental Results

E.1 Detailed results on MATH dataset

We present detailed results across different categories in the MATH dataset in Table 5. The numbers 1-7 correspond to Algebra, Counting and Probability, Geometry, Intermediate Algebra, Number Theory, Prealgebra, and Precalculus, respectively. The results indicate that Algebra is easier to improve, as evidenced by its higher mutation success rate. It is worth noting that improvements in Counting and Number Theory are reasonable because related mutation operators (e.g., binomial, gcd, lcm, etc.) are included in our framework. However, Precalculus and Geometry are still not well-supported. For example, the concept of triangle cannot currently be correctly expressed in the SMT-LIB format, resulting in relatively low improvement rates.

Table 5: Comparison of performance between MetaMath and our method across different categories of MATH dataset. The used base model is Mistral-7B. The best performance is in bold.

Category	1	2	3	4	5	6	7
MetaMath	41.4	23.6	15.4	20.7	47.9	15.3	22.5
Ours	**59.6**	**35.4**	**16.4**	**32.8**	**58.5**	**18.5**	**25.0**
Δ	+18.2	+11.8	+1.0	+12.1	+10.6	+3.2	+2.5

E.2 Comparison to tool-based methods

We compare our model with the tool-based models in Table 6. Although tool-based models achieve good performance with tools, they meet severe performance degradation when tools are not available. This result indicates that training a model using code-based and language-based rationales does not necessarily enhance the intrinsic reasoning ability; instead, it often promotes excessive dependence on external tools. For datasets that involve complex calculations, such as the MATH dataset, tool-based methods offer certain advantages due to their utilization of the strong capabilities of external tools. For datasets that emphasize knowledge reasoning but involve simpler calculations, such as the

Table 6: Performance comparison among tool-based methods and our methods. We report the performance of tool-based methods w/ and w/o tools. The best performance is in bold.

Model	Base Model	GSM8k	MATH
Tora	LLaMA2-7B	68.8	40.1
MAmmoTH	LLaMA2-7B	53.6	31.5
MAmmoTH(w/o tools)	LLaMA2-7B	50.5	10.4
Tora	CodeLLaMA-7B	72.6	**44.6**
MAmmoTH	CodeLLaMA-7B	59.4	33.4
MAmmoTH(w/o tools)	CodeLLaMA-7B	22.1	7.6
MAmmoTH	Mistral-7B	75.0	40.0
MAmmoTH(w/o tools)	Mistral-7B	61.9	17.5
Ours	LLaMA2-7B	79.2	28.8
Ours	Mistral-7B	**86.8**	37.3

GSM8K dataset, they underperform our proposed methods, proving that their reasoning ability is still inadequate.

E.3 Experiments on DyVal Datasets

We also compare our models and existing mathematical reasoning models using DyVal [9] datasets to further evaluate the generalization ability of our models. DyVal is a flexible evaluation protocol for dynamic evaluation of LLMs, which generates evaluation samples with controllable complexities using directed acyclic graphs (DAGs). We focus on Arithmetic tasks using three DAGs' orders: topological (TOPO), reversed topological (REVERSED), and random orders (RAND). Following the same setting, we generate testing four increased complexity levels {D1, D2, D3, D3}, with tree depths and widths set to (2, 2), (3, 2), (3, 3), (4, 2). The performance comparison between models fine-tuned on LLaMA2-7B and Mistral-7B are respectively shown in Table 7 and Table 8. The results show that our models achieve the best performance in 11/12 cases and give a competitive performance in the remaining one case. As the complexity increases, our models achieves a relatively robust performance compared with the other models.

Table 7: Performance comparison of models fine-tuned on LLaMA2-7B using DyVal datasets with four complexity levels and three graph orders. The best performance is in bold.

Model	Arithmetic-Topo				Arithmetic-Reversed				Arithmetic-Rand			
	D1	D2	D3	D4	D1	D2	D3	D4	D1	D2	D3	D4
WizardMath	74.0	55.4	21.4	19.9	71.0	50.6	23.2	7.8	74.2	49.0	19.6	10.7
MAmmoTH	77.3	35.1	13.7	8.1	81.8	27.7	13.1	4.8	78.3	26.4	12.7	3.8
MetaMath	91.9	48.0	20.3	16.9	90.5	50.5	18.1	5.8	90.2	41.0	17.9	7.4
Ours	**99.7**	**85.7**	**66.8**	**62.5**	**99.1**	**75.8**	**39.8**	**18.1**	**98.8**	**71.4**	**41.2**	**19.0**

Table 8: Performance comparison of models fine-tuned on Mistral-7B using DyVal datasets with four complexity levels and three graph orders. The best performance is in bold.

Model	Arithmetic-Topo				Arithmetic-Reversed				Arithmetic-Rand			
	D1	D2	D3	D4	D1	D2	D3	D4	D1	D2	D3	D4
MAmmoTH	90.2	66.7	29.9	32.8	92.2	64.8	25.3	18.8	91.0	62.2	23.9	18.2
MetaMath	97.8	75.7	43.0	45.6	**99.5**	78.6	45.6	37.2	98.3	76.3	44.6	38.8
Ours	**99.6**	**91.3**	**74.0**	**71.5**	99.3	**85.4**	**56.8**	**49.2**	**98.4**	**87.5**	**68.6**	**47.2**

E.4 Diversity Gain across Various Difficulty Levels

To illustrate the need for various difficulty levels, we calculate the diversity gain relative to the original dataset for each difficulty level. We apply the same method as in MetaMath [33] to compute diversity

gain but use the BERT model [73] as a feature extractor instead. We select data budgets of 35K, 50K, and 100K, respectively, and investigate the diversity across four levels, from level-1 to level-4. Additionally, we create a mixed version by sampling data from all levels. The results are shown in Figure 5, and we can observe that: (1) The higher the difficulty level, the greater the diversity of generated data; (2) The mixed version increases with the growing data budget, and achieves the highest diversity gain.

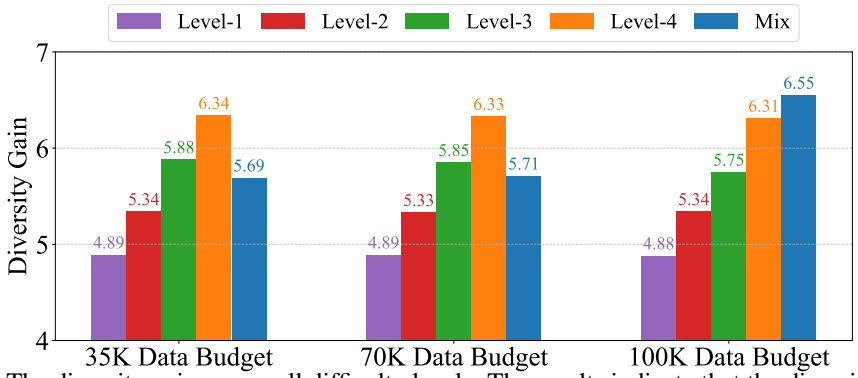


Figure 5: The diversity gain across all difficulty levels. The results indicate that the diversity gain of the Mix version continues to increase and reaches the highest compared with alternatives as the data budget increases.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation is thoroughly discussed in Appendix **??**.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: N/A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Details of data generation, training and inference process are discussed in Appendix D. We will public the code, as well as the fine-tuned models, for the reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code for our data generation framework, as well as a small part of our generated data, in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and test settings are detailed in Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We cannot provide statistical significance of the experiments due to the limited GPU resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details of the computer resources used in the experiments are provided in D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have carefully reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Social impacts are discussed in **??**.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not pose such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Code packages and datasets are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets up to now.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.