NuminaMath: The largest public dataset in AI4Maths with 860k pairs of competition math problems and solutions

Jia LI[1], Edward Beeching[2], Lewis Tunstall[2], Ben Lipkin[3], Roman Soletskyi[1], Shengyi Huang[2], Kashif Rasul[2], Longhui Yu[1], Albert Q. Jiang[4], Ziju Shen[5], Zihan Qin[5], Bin Dong[5], Li Zhou[6], Yann Fleureau[1], Guillaume Lample[4], and Stanislas Polu[1]

[1]Numina [2]Huggingface [3]MIT [4]Mistral AI [5]Peking University [6]Answer AI

July 19, 2024

Abstract

Numina is an open AI4Maths initiative dedicated to advancing both artificial and human intelligence in the field of mathematics. In this paper, we present the NuminaMath dataset, a comprehensive collection of 860,000 pairs of competition math problems and solutions. This dataset is designed to enhance the mathematical reasoning capabilities of large language models (LLMs) and stands as the largest math dataset ever released in the field. The NuminaMath dataset includes problems ranging from high-school-level to advanced-competition-level, all meticulously annotated with accompanying chain-of-thought traces. We detail the construction, composition, and potential applications of the dataset, and provide experimental results that demonstrate its effectiveness. The power of this dataset is underscored by its role in fine-tuning a model to win the 1st AIMO Progress Prize, showcasing its significant impact on the development of state-of-the-art mathematical reasoning models.

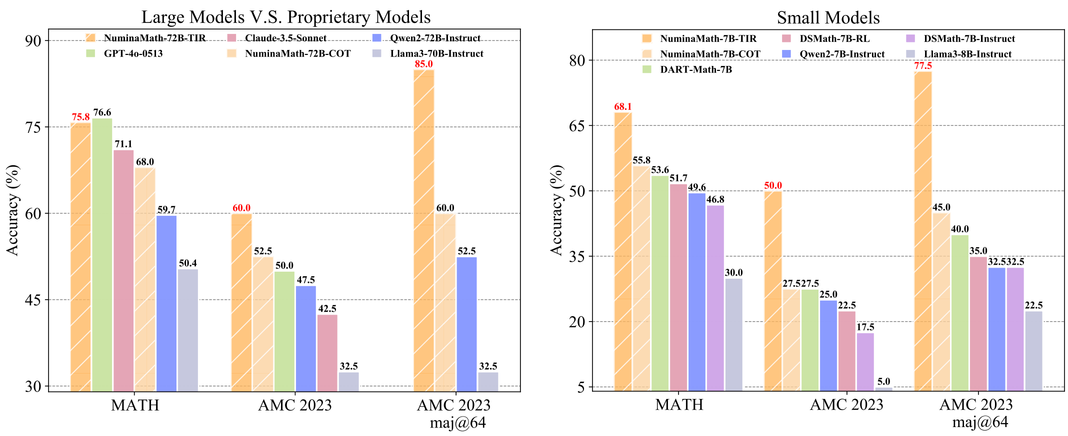


Figure 1: numina.

1 Introduction

Mathematics is a universal language, fundamental to various domains including science, engineering, and economics. The complexity and beauty of mathematical problems make them an excellent testbed for developing and evaluating artificial intelligence (AI) models. Despite the significance of competition-level mathematics, there are very few publicly available datasets. Notable datasets include MATH [Hendrycks et al., 2021], with a training set of 7.5k problems, and OrcaMath [Mitra et al., 2024], focused on grade-school mathematics with a subset of problems similar to GSM8K [Cobbe et al., 2021]. High-performing models like DeepSeekMath [Shao et al., 2024] and others often achieve state-of-the-art performance without publishing their datasets, creating a gap in the availability of resources for the broader AI research community.

The NuminaMath dataset, an open-source initiative, addresses this gap by providing a comprehensive resource for training AI models to solve complex mathematical problems. This dataset consists of nearly one million pairs of math problems and solutions, annotated with a unified chain-of-thought (CoT) format to facilitate reasoning and problem-solving by current AI systems. This dataset is the biggest Math dataset ever released and has allowed the Numina team to win the 1st AIMO progress prize.

The primary motivation for focusing on competition-level mathematics is its potential to serve as a precursor for advanced reasoning tasks in AI. Achieving a gold medal in the International Mathematical Olympiad (IMO) is a benchmark for exceptional mathematical achievement and a strong predictor of future success. IMO gold medalists are 50 times more likely to win a Fields Medal compared to a typical Cambridge PhD graduate. Furthermore, half of all Fields medalists participated in the IMO during their youth. Thus, by developing AI models capable of solving IMO-level problems, we can build intuition for what works in high-level mathematical reasoning.

To push the limits of open large language models (LLMs), it is crucial to combine chain-of-thought reasoning with tool use, such as the integration of code interpreters. Our research utilized this dataset to win the Progress Prize, demonstrating the practical application and success of the Numina dataset in advancing AI capabilities.

This paper also explores the performance of various open-weights models at 7B and 70B parameter scale, as well as evaluating how well closed models like GPT-4 [OpenAI, 2024] and Claude 3.5 [Anthropic, 2024] perform on recent competitions, such as AMC 2023 and AIME 2024, using code interpreters. Additionally, we investigate whether our models can outperform these large models.

We provide a detailed analysis of the dataset, breaking it down by source and visualizing the distribution of problems per country for Olympiad-level problems. We also include token statistics (input/output), the number of Python blocks in the tool-integrated reasoning (TIR) format, and the distribution of problem types, particularly those with integer outputs.

In summary, the Numina dataset represents a significant step forward in developing AI models capable of tackling complex mathematical problems. The dataset facilitates deeper understanding and reasoning by providing detailed, step-by-step solutions. We invite contributions from the community to further enhance the dataset and its applications.

2 Related Works

Several datasets have been developed to aid in the training of AI models for mathematical problem solving. The Mathematics Dataset by DeepMind [Saxton et al., 2019], the MATH [Hendrycks et al., 2021] dataset, and the GSM8K [Cobbe et al., 2021] Dataset are notable examples. These datasets provide a range of problems from elementary arithmetic to advanced mathematics, annotated in various formats.

- **Math LLMs:** Numerous excellent models have emerged in the world of LLMs for math. In this report, we will distinguish between closed models (hidden weight and undisclosed data), open weights models (public weights but private data), and open data models (both weights and data are public). In the world of closed models, Minerva [Lewkowycz et al., 2022] was among the first to pursue mathematics training at scale. Recently, GPT-4 [OpenAI, 2024], Gemini 1.5 [GeminiTeam, 2024], and Claude 3.5 [Anthropic, 2024] have been among the strongest contenders. Other impressive models in this space have been open-weight but trained at least partially on private data, including Llemma [Azerbayev et al., 2024], MetaMath [Yu et al., 2023], WizardMath [Luo et al., 2023], and DeepSeekMath [Shao et al., 2024].

- **Chain of thought (CoT) / Scratchpads:** CoT [Wei et al., 2023] and scratchpads [Nye et al., 2021] were first introduced as prompting strategies to augment the problem-solving capacities of LLMs by incrementally decomposing problems into smaller steps. While transformers are tightly bound in the complexity of problems solvable in a single forward pass [Merrill and Sabharwal, 2024], looped transformers can solve a much more expressive set of problems [Merrill and Sabharwal, 2023, Giannou et al., 2023].

- **Finetuning with bootstrapped CoT trajectories:** One successful approach for distilling the performance gains of CoT has been to leverage in-context-learning (ICL) to sample well-formed CoT trajectories for existing problem solution pairs, filter based on correctness, and then utilize these intermediate samples for fine-tuning. Such an approach can push models to sample from increasingly effective problem-solving strategies. In the math domain, successful works including ToRA [Gou et al., 2024], WizardMath [Luo et al., 2023], and DeepSeekMath [Shao et al., 2024] all utilized variants of this strategy to bootstrap synthetic training data for fine-tuning. In addition, Chen et al. [2024] have characterized the boundaries of this type of approach, as well as presenting strategies for determining optimal fine-tuning data mixtures.

- **Use of synthetic problems:** Even with the amazing collection of existing math datasets, public data availability remains a limiting factor in scaling up training. As such, there has been a recent focus on harnessing LLMs to generate novel problems, which can then be used to bootstrap training data. Li et al. [2024], as well as MetaMath [Yu et al., 2023] and MuMath [You et al., 2024] have explored this approach, fine-tuning on synthetic data, revealing strong performance gains. Microsoft's Phi family of models [Gunasekar et al., 2023, Li et al., 2023, Javaheripi et al., 2023, Abdin et al., 2024] have also presented substantial evidence for the effectiveness of synthetic data.

- **Leveraging external tools:** In approaching problems with underlying symbolic structure including mathematical and logical reasoning, augmenting LLMs with external "tools", such as access to a Python interpreter [Chen et al., 2022, Gao et al., 2023, He-Yueya et al., 2023] or SAT-solver [Olausson et al., 2023, Pan et al., 2023, Ye et al., 2024], has proved a powerful tactic. In such settings, an LLM operates as a declarative programmer, generating a formal specification of a given problem in code, that is then offloaded to an external solver, which can return an exact solution.

- **Tool-integrated reasoning:** In drawing from the complementary strengths of reasoning over text and programs, recent works have adopted combined strategies that train on both CoT and program trajectories [Yue et al., 2023]. These works, especially ToRA [Gou et al., 2024] and MuMath-Code [Yin et al., 2024], which interleaves planning over free-form text with program generation and execution in joint trajectories, formed the basis of the most successful strategies explored and adopted in the development of this dataset and corresponding models.

- **Decoding algorithms:** While the base sampling distributions of LLMs may contain correct solutions to problems, it is often the case that greedy generation is a sub-optimal decoding strategy. Instead, it is often preferable to sample, steer, and filter multiple candidate trajectories, and use a scoring function and decision rule to rerank or reconcile the diverse responses. Self-consistency [Wang et al., 2022], self-refine [Madaan et al., 2024], and CoRe [Zhu et al., 2022], among other works, can be construed as variants of such a general strategy. The outputs of such online inference can also be used to bootstrap fine-tuning data for distillation, as in Yu et al. [2024] and STaR [Zelikman et al., 2022].

3 Datasets

A significant challenge in the development of open math LLMs has been the lack of substantial high-quality data. The training splits of the popular MATH [Hendrycks et al., 2021] and GSM8K [Cobbe et al., 2021] datasets have been frequently used, but their scale is limited, with less than 20k samples combined. The Microsoft orca-math-word-problems-200k (orca-math) dataset [Mitra et al., 2024] has been a valuable contribution to this space, adding 200K samples of grade level synthetic math problems.

With this report, we contribute the NuminaMath-CoT dataset, which consists of 860K problem-solution pairs covering a wide range of mathematical topics, from high school exercises to competition-level problems. The data sources include Chinese high school math exercises, US and international mathematics olympiad problems, and problems collected from online forums. It is the largest math problems dataset ever released.

Dataset Name	**Number of Samples for Training**	**Number of Competition Level Samples**
MATH	7,500	7,500
GSM8K	7,470	0
orca-math	200K	0
NuminaMath-CoT	860K	400K

Table 1: Datasets and sample sizes.

3.1 Data Collection and Processing

The data collection process involved acquiring materials from the web, extracting problems from PDFs using optical character recognition (OCR), segmenting samples into problem-solution pairs, translating text into English, and realigning pairs to produce accompanying chain-of-thought reasoning traces. The final answers were formatted to ensure consistency and accuracy. Here we describe the data collection and processing method by data source. The specific processing method will be detailed in the next subsection.

- **MATH and GSM8K:** We follow the recommendations of DeepseekMath [Shao et al., 2024] and ReAlign [Fan et al., 2024], using GPT-4 to reformat reference solutions from original problems to utilize our CoT format.

- **Orca-Math:** Orca-Math is an excellent large-scale grade-school-level synthetic dataset. We match regular expressions to simplify and identify answers from the provided solution text in the original Orca-Math dataset, then wrap each answer in `\boxed{}`. If you already have a copy of the Orca-Math dataset in your training data, or a better formatted version, it is not necessary to include this part of the NuminaMath dataset. For example, https://huggingface.co/datasets/PawanKrd/math-GPT-4o-200k could be a viable alternative, where the final solution is written by GPT-4o.

- **AMC and AIME:** AMC and AIME are among the most popular competition problem sets. Each covers different levels of difficulties, from grade school competition to near IMO level; Our subset spans various topics, including arithmetic, geometry, combinatorics, and algebra. Problems from AMC or AIME have clear value output, either being a multiple-choice (AMC) or an integer (AIME). We collected AMC and AIME problem statements from the AoPS wiki website. Problems are already in latex format. Several good solutions have been proposed by the AoPS community for each problem. We select the first candidate with a `\boxed{}` symbol in the solution. This dataset accounts for roughly 6,500 problems. However, not all can be used in the training set. Part of this data has been used to create the MATH dataset [Hendrycks et al., 2021]. As such, we performed a decontamination process using embeddings (see 3.4), which retains roughly 4,300 problems for the training set. Finally, we apply the same method as above to realign reference solutions into CoT format using GPT-4o.

- **AoPS Forum:** The AoPS forum also has a large collection of competition problems across the world. We have crawled all the data from the Contest Collection page of the website. However, the reference solution is not collected in the forum. Instead, replies might contain one good solution or at least hint at the solution. We pick all replies with a `\boxed{}` or $\blacksquare$ symbol, then choose the one with the most latex symbols. Once we pick the best reply, we consider it as a reference solution and feed it into the realignment prompt for GPT-4o to rewrite the solution as before.

- **Chinese K-12 Exam:** We follow Shao et al. [2024]'s fine tuning method, to collect Chinese K-12 education math exercises. These exercises are not competition level but will provide a solid base knowledge and good arithmetic skills for the model to learn. We retrieve all our exercises from public exam papers. Most of them can be found and downloaded from https://www.shijuan1.com/ or He et al. [2023]. When the exam papers come in a PDF format, we apply OCR and regex segmentation to extract problem-solution pairs, and then use GPT-4o to translate and realign the reference solutions.

- **Synthetic Data:** We follow Li et al. [2024] to create synthetic math data using the MATH dataset and the training split of AMC-AIME dataset. In Li et al. [2024] synthetic problems are first sampled using a seed problem by GPT-4, with a non-zero temperature (0.8). In the second stage, a new solution is created with greedy decoding. In our approach, we directly use the solution of the first stage to reduce costs.

- **World Olympiads Data:** We collect 152k problem-solution pairs from the following sources
  - International contests and their respective shortlists (IMO, APMO, BMO, etc.)
  - National and regional contests with a breakdown per country shown at Figure 2
  - Problem-solving forums, puzzle and olympiad books, summer school materials

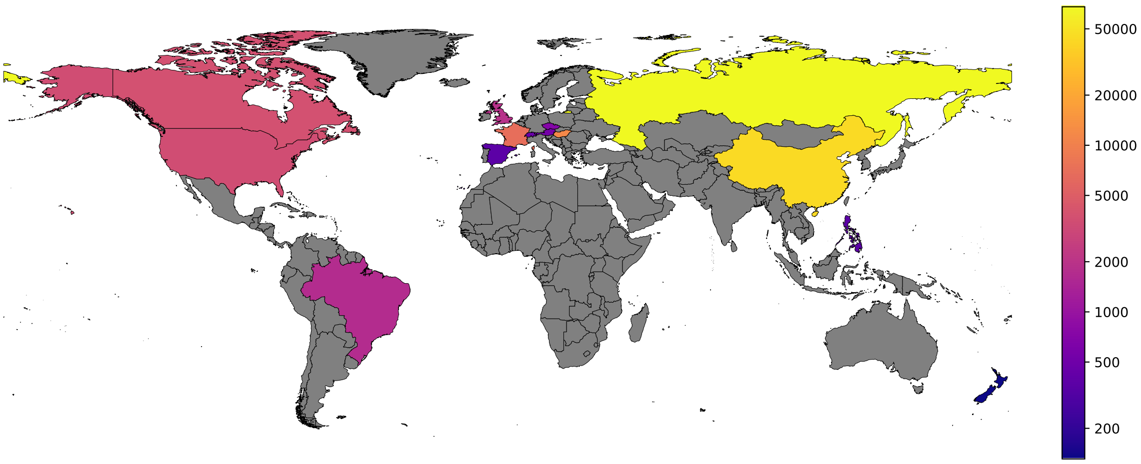


Figure 2: Number of national contest problems collected per country.

Typically, files containing problems were found in PDF or scraped from sites in HTML format. In the latter case, they were converted to PDF for the correct formula representation. Then the full pipeline described in the next section was employed.

Combining all these data sets together, we then get the NuminaMath-CoT dataset. The breakdown by source is shown in Figure 3.

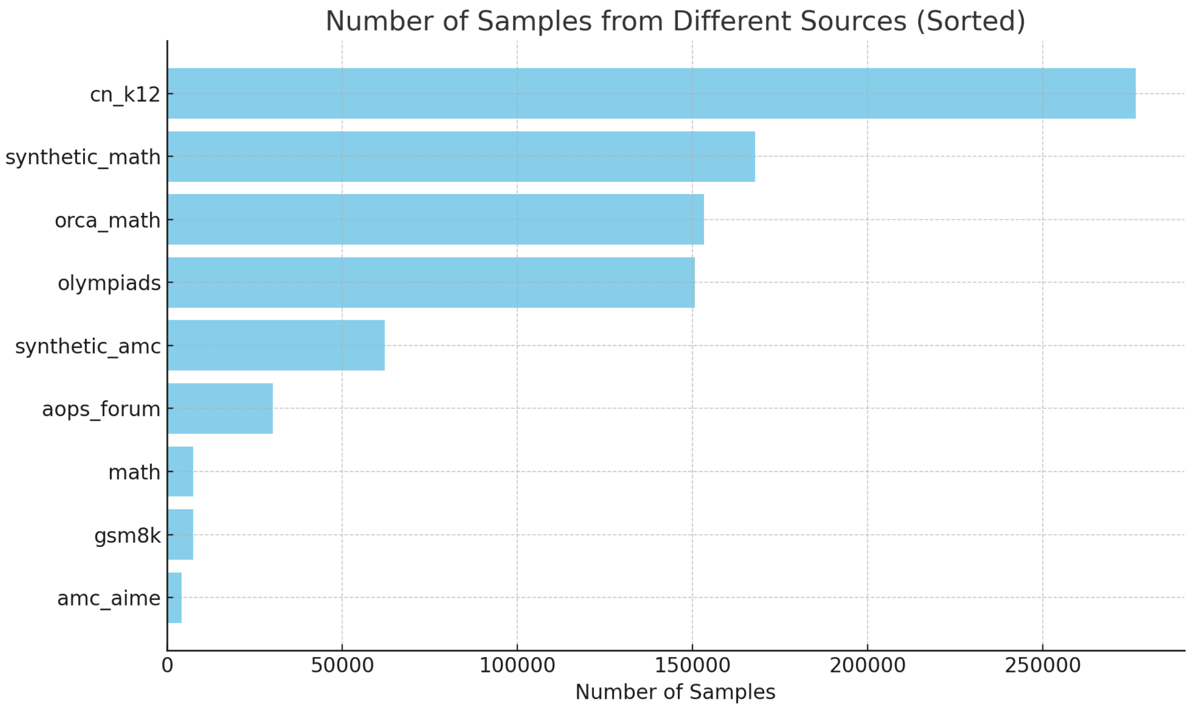


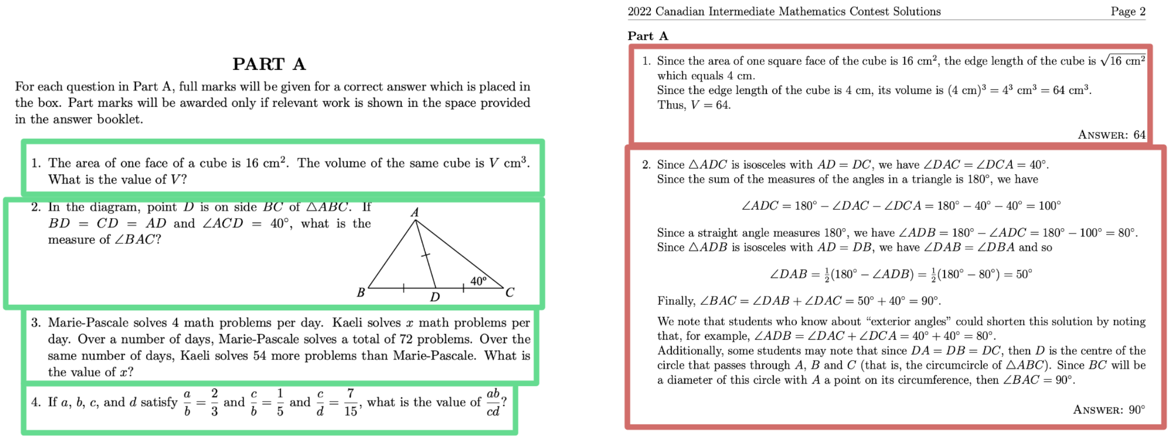
Figure 3: Number of samples per data source.

3.2 Processing Pipeline

The purpose of the processing pipeline is to convert the original solution to a uniform format: English, CoT, and a conclusion either with `\boxed{}` or with ■.

The data processing pipeline contains multiple steps, namely:

- **OCR from PDFs:** Most of the raw data comes in PDF format. We use the API of https://mathpix.com/ to convert PDF to markdown format. Depending on the segmentation strategy, we either convert the whole PDF (a sequence of problems and solutions) into text or convert a well segmented problem (single problem) into markdown text.


PART A

For each question in Part A, full marks will be given for a correct answer which is placed in the box. Part marks will be awarded only if relevant work is shown in the space provided in the answer booklet.

1. The area of one face of a cube is 16 cm^2. The volume of the same cube is V cm^3. What is the value of V?
2. In the diagram, point D is on side BC of $\triangle ABC$. If $BD = CD = AD$ and $\angle ACD = 40°$, what is the measure of $\angle BAC$?
3. Marie-Pascale solves 4 math problems per day. Kaeli solves x math problems per day. Over a number of days, Marie-Pascale solves a total of 72 problems. Over the same number of days, Kaeli solves 54 more problems than Marie-Pascale. What is the value of x?
4. If a, b, c, and d satisfy $\frac{a}{b} = \frac{2}{3}$ and $\frac{c}{b} = \frac{1}{5}$ and $\frac{c}{d} = \frac{7}{15}$, what is the value of $\frac{ab}{cd}$?

2022 Canadian Intermediate Mathematics Contest Solutions Page 2

Part A

1. Since the area of one square face of the cube is 16 cm^2, the edge length of the cube is $\sqrt{16 \text{ cm}^2}$ which equals 4 cm.
Since the edge length of the cube is 4 cm, its volume is $(4 \text{ cm})^3 = 4^3 \text{ cm}^3 = 64 \text{ cm}^3$.
Thus, $V = 64$.
ANSWER: 64

2. Since $\triangle ADC$ is isosceles with $AD = DC$, we have $\angle DAC = \angle DCA = 40°$.
Since the sum of the measures of the angles in a triangle is 180°, we have
$$\angle ADC = 180° - \angle DAC - \angle DCA = 180° - 40° - 40° = 100°$$
Since a straight angle measures 180°, we have $\angle ADB = 180° - \angle ADC = 180° - 100° = 80°$.
Since $\triangle ADB$ is isosceles with $AD = DB$, we have $\angle DAB = \angle DBA$ and so
$$\angle DAB = \tfrac{1}{2}(180° - \angle ADB) = \tfrac{1}{2}(180° - 80°) = 50°$$
Finally, $\angle BAC = \angle DAB + \angle DAC = 50° + 40° = 90°$.
We note that students who know about "exterior angles" could shorten this solution by noting that, for example, $\angle ADB = \angle DAC + \angle DCA = 40° + 40° = 80°$.
Additionally, some students may note that since $DA = DB = DC$, then D is the centre of the circle that passes through A, B and C (that is, the circumcircle of $\triangle ABC$). Since BC will be a diameter of this circle with A a point on its circumference, then $\angle BAC = 90°$.
ANSWER: 90°

Figure 4: Illustration of manual segmentation.

- **Problem Solution Segmentation:** Part of World Olympiads Data was segmented by matching the starts of problems and solutions with a corresponding regex that was engineered manually for the chosen contests. Another part, that was hard to match with regex, was segmented manually at the PDF level and then converted to text with OCR.

- **Reformatting:** We use GPT-4o to reformat the extracted text solution. We use the batch API of GPT-4o to perform the following steps:

  1. Translate the solution to English.
  2. CoT realignment.
  3. Box the final answer.

3.3 NuminaMath-TIR

After having first created the NuminaMath-CoT dataset, it was straightforward to extend to TIR (tool integrated reasoning). We follow the same approach as Gou et al. [2024], in particular their prompt, to sample TIR data from GPT-4o. Here is the process to create this TIR dataset:

1. Extract a subset of roughly 100K problems with value output from the NuminaMath-CoT dataset.
2. Sample a solution using the GPT-4o assistant API for each problem with a temperature of 0.8.
3. Filter the negative samples where model generated answers don't match the reference answer. For integer output problems exact match is used. For other expressions, a match is determined using GPT-4o as a judge.
4. Repeat the same process on the negative problems.

3.4 Decontamination

A good internal validation set was one of the keys to our success in winning the AIMO Progress Prize. To reflect unbiased model performance, it was necessary to undergo a careful decontamination procedure. The well-known MATH and GSM8K datasets, as well as AIME 2024 and AMC12 2023, were selected to evaluate our models. We used the following two-step decontamination strategy.

- **Exact string match:** Following Shao et al. [2024], we removed all 10-gram exact match from all datasets, except the synthetic dataset, the MATH train set, which was developed from seed datasets where decontamination was already conducted. We noticed that there was a fraction of contaminated problems in the MATH test set that were just the paraphrasing of samples from the MATH train set with different numbers. Since this was not flagged for contamination by MATH, we too do not flag this as contamination.

- **Nearest neighbor search using embedding:** We have seen increasing use of the AIME and AMC problems as part of training and fine-tuning datasets in the development of math LLMs [Zhou et al., 2024]. Here it was critical to be particularly cautious in decontaminating these datasets. The MATH test set itself was constructed using a subset of the AIME-AMC exercises [Hendrycks et al., 2021], where some problems were modifications of the original problems to facilitate parsing. The MATH test set will add statements such as "express the final result as $\frac{m}{n}$, compute $m+n$" which makes exact string matches unreliable. To make matters more difficult, in our case, the Olympiad dataset was sourced globally, and so some of the problems that we had collected reflected translated versions of AMC-AIME problems. To better decontaminate, we computed Mistral embeddings for each of our problems except for MATH train and the synthetic datasets, then removed all problems with an embedding distance < 0.15. This value was derived empirically, above which we did not observe contamination in our internal tests.

  Here is an example:

Problem from AIME:	Problem from MATH test:
A sphere is inscribed in the tetrahedron whose vertices are $A = (6,0,0), B = (0,4,0), C = (0,0,2)$, and $D = (0,0,0)$. The radius of the sphere is $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $m+n$.	A sphere is inscribed in the tetrahedron whose vertices are $A = (6,0,0)$, $B = (0,4,0)$, $C = (0,0,2)$, and $D = (0,0,0)$. Find the radius of the sphere.
The original problem in 2001 AIME I Problems/Problem 12.	The modified version in the MATH test set.

4 Experiments

We conducted several experiments to evaluate the effectiveness of the Numina dataset in training AI models. Our models were fine-tuned using a two-stage process inspired by the MuMath-Code paper.

4.1 Fine-Tuning Stages

- **Stage 1:** Fine-tuning on a large, diverse dataset of natural language math problems and solutions, with Chain of Thought (CoT) annotations to facilitate reasoning.

- **Stage 2:** Fine-tuning on a synthetic dataset of tool-integrated reasoning, where problems are decomposed into rationales, Python programs, and outputs.

We trained models at two scales:

- 7B, based on DeepSeekMath-Base 7B; and
- 72B, based on Qwen2-72B.

All models were trained with a global batch size of 32, a learning rate of 2×10^{-5}, and a cosine scheduler. The training was run on 8-32 H100 GPUs, depending on the model size, and took less than 24 hours for both stages. Due to time constraints, the NuminaMath 72B model was only trained for 2 epochs in Stage 1. Further details of hyper-parameters are shown in Table 2.

Table 2: Hyper-parameters used in the experiments.

	NuminaMath-7B		**NuminaMath-72B**	
stage	CoT	TIR	CoT	TIR
block size	2048	1024	2048	2048
epochs	3	4	1	3
warmup ratio	0	0.1	0.1	0.1

4.2 Tool-integrated reasoning (TIR)

As in ToRA [Gou et al., 2024] and MuMath-Code [Yin et al., 2024], at inference time, Numina TIR models have been trained to leverage interleaved CoT rationales alongside Python code generation, execution, and repair. Our approach is outlined in Algorithm 1 and described below. The complete implementation can be found in the linked public GitHub repository[1].

Each run of TIR begins with a problem, x, from a dataset, $\mathcal{X}$. The goal of TIR is to sample a candidate solution, y, where the valid solution space may vary between datasets, e.g., $\mathcal{Y}_{\texttt{AIMO}} = \{i \in \mathbb{N} \mid 0 \leq i \leq 999\}$. TIR begins by initializing a context, $\boldsymbol{c}$, with an initial prompt $\boldsymbol{c}_0$ containing only x. This context is then extended through up to k rounds of interaction. On each iteration, i, TIR uses a sampler, `S`, and an LLM, θ, to sample text containing CoT and Python source code, $\boldsymbol{z}_i$, until reaching the stop keyword $w_{\texttt{stop}} = \texttt{```output}$. After sampling $\boldsymbol{z}_i$, TIR first checks if a candidate solution has been generated, which would be wrapped in the keyword $w_{\texttt{answer}} = \texttt{\textbackslash boxed\{\}}$. If an answer is present, TIR applies a response parser, `R`, to the output, which acts to sanitize the text and return only the final numerical response with any units and other formatting removed. If no valid response is present, TIR assesses whether any code has been generated by matching a regular expression with the python region keyword $w_{\texttt{python}} = \texttt{```python(.*)```}$. If no such region is available, $\boldsymbol{z}_i$ is discarded, and TIR proceeds to the next iteration, resampling a fresh block of text. If such a region is available, the Python source code is passed to the Python interpreter, `I`, which parses and executes the source code. The result, $\boldsymbol{r}_i$ from running $I(\boldsymbol{z}_i)$ may include the output of print statements, or a truncated `Traceback` if an exception was raised. The running context is then extended, proceeding to the next round of interaction, via setting $\boldsymbol{c}_i$ to $\boldsymbol{c}_{i-1} \oplus \boldsymbol{z}_i \oplus \boldsymbol{r}_i$, where $\oplus$ denotes concatenation. Thus, by the end of interaction, $\boldsymbol{c} = \boldsymbol{c}_0\boldsymbol{z}_1\boldsymbol{r}_1\boldsymbol{z}_2\boldsymbol{r}_2 \ldots \boldsymbol{z}_{\leq k}$, where either a candidate answer, y, is successfully extracted from $\boldsymbol{z}_{\leq k}$ else an error keyword $w_{\texttt{error}}$, e.g., -1 for $\mathcal{Y}_{\texttt{AIMO}}$, is returned.

[1] https://github.com/project-numina/aimo-progress-prize

Algorithm 1 Tool-integrated reasoning (TIR)

Require: LLM θ, sampler $\mathtt{S}$, interpreter $\mathtt{I}$, response parser $\mathtt{R}$, problem x, blocks k, keywords w

$c_0 \leftarrow x$ ▷ Initialize context to problem text
for $i \leftarrow 1 : k$ **do**
  $z_i \sim \mathtt{S}\left(p_\theta\left(\cdot \mid c_{i-1}\right), w_{\mathtt{stop}}\right)$ ▷ Sample CoT and code until stop token
  if $w_{\mathtt{answer}}$ in z_i **then** ▷ Parse response and return answer if generated
    $y \leftarrow \mathtt{R}\left(z_i\right)$
    return y
  else if $w_{\mathtt{python}}$ not in z_i **then** ▷ Regenerate block if code missing
    continue
  end if
  $r_i \leftarrow \mathtt{I}\left(z_i\right)$ ▷ Strip code, run interpreter, and return output or traceback
  $c_i \leftarrow c_{i-1} \oplus z_i \oplus r_i$ ▷ Update context with generated text and interpreter result
end for
return $w_{\mathtt{error}}$ ▷ If can't generate an answer in k blocks

4.2.1 Incorporating self-consistency (SC-TIR)

TIR (Algorithm 1) describes the process for sampling a single candidate solution y for an initial problem x. However, TIR does not proceed through direct samples from the conditional distribution $p_\theta\left(y \mid x\right)$, which would correspond to a direct prediction of the solution. Rather, TIR jointly draws an answer alongside a sequence of samples from an auxiliary latent variable, the trace of generated CoT planning and Python source code. This sequence is denoted $\boldsymbol{z}$. Generations drawn from TIR thus correspond to samples from $p_\theta\left(y, \boldsymbol{z} \mid x\right)$. To effectively calculate $p_\theta\left(y \mid x\right)$, it is thus necessary to marginalize out the generated traces, which can be achieved via the following summation:

$$p_\theta\left(y \mid x\right) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p_\theta\left(y, \boldsymbol{z} \mid x\right)$$

In the context of latent variable modeling, this equation is typically referred to as the marginal likelihood and is often computationally infeasible when $\mathcal{Z}$ is large, as is the case here. This type of scenario occurs frequently in practice, and various approximation strategies exist. Most notably in the context of marginalizing LLM reasoning traces, Wang et al. [2022] propose self-consistency (SC), which approximates the marginalization and application of a maximum a-posteriori (MAP) decision rule by drawing a finite number n of samples $\boldsymbol{y}$ from $p_\theta\left(y, \boldsymbol{z} \mid x\right)$ and then applying a majority voting procedure:

$$y_{SC} = \underset{y}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbb{1}\left(\boldsymbol{y}_i = y\right)$$

In the case of SC-TIR, we generate n samples from TIR, then apply a filter, $\mathtt{F}$, which removes ill-formed responses outside the support of $\mathcal{Y}$, and finally apply self-consistency majority voting, as outlined in Algorithm 2.

4.3 Evaluation Results

We evaluated our models using various metrics, including accuracy on the MATH benchmark and performance on internal validation sets consisting of AMC and AIME problems. The results demonstrated significant improvements in problem-solving capabilities, particularly in handling complex reasoning tasks.

Table 3 presents comparisons of various 7B and 8B parameter models and shows that with TIR one can achieve SoTA performance. This conclusion also holds on larger 70B parameter

Algorithm 2 Self-consistent TIR (SC-TIR)

Require: subroutine `TIR`, response filter `F` , problem x, number of samples n

$\boldsymbol{y} \leftarrow \texttt{empty}\,(n)$

for $i \leftarrow 1 : n$ **do** ▷ Sample n independent TIR trajectories from problem

  $\boldsymbol{y}_i \sim \texttt{TIR}\,(x)$

end for

$y_{SC} \leftarrow \texttt{maj}\,(\texttt{F}\,(\boldsymbol{y}))$ ▷ Filter invalid answers and apply majority vote decision rule

return y_{SC}

variants as in Table 4 where the metrics beat Claude 3.5 and get close to GPT-4o on MATH and outperform them on the other competition benchmarks.

		NuminaMath 7B		Qwen2 7B	Llama3 8B	DeepSeekMath 7B		DART-Math 7B
		CoT	TIR	Instruct	Instruct	Instruct	RL	CoT
GSM8k Grade school math	0-shot	76.3%	84.6%	82.3%	79.6%	82.8%	88.2%	86.6%
MATH Math problem solving	0-shot	55.8%	68.1%	49.6%	30.0%	46.8%	51.7%	53.6%
AMC 2023 Competition-level math	0-shot maj@64	11/40 18/40	20/40 31/40	10/40 13/40	2/40 9/40	7/40 13/40	9/40 14/40	11/40 16/40
AIME 2024 Competition-level math	0-shot maj@64	0/30 1/30	5/30 10/30	1/30 4/30	0/30 2/30	1/30 1/30	1/30 1/30	1/30 1/30

Table 3: Comparison of various 7B and 8B parameter language models on different math benchmarks. All scores except those for NuminaMath 7B TIR are reported without tool-integrated reasoning.

		NuminaMath 72B		Qwen2 72B	Llama3 70B	Claude 3.5	GPT-4o
		CoT	TIR	Instruct	Instruct	Sonnet	0513
GSM8k Grade school math	0-shot	91.4%	91.5%	91.1%	93.0%	96.4%	95.8%
MATH Math problem solving	0-shot	68.0%	75.8%	59.7%	50.4%	71.1%	76.6%
AMC 2023 Competition-level math	0-shot maj@64	21/40 24/40	24/40 34/40	19/40 21/40	13/40 13/40	17/40 -	20/40 -
AIME 2024 Competition-level math	0-shot maj@64	1/30 3/30	5/30 12/30	3/30 4/30	0/30 2/30	2/30 -	2/30 -

Table 4: Comparison of various open weight and proprietary language models on different math benchmarks. All scores except those for NuminaMath 72B TIR are reported without tool-integrated reasoning.

5 Conclusion and Future Works

The NuminaMath dataset is the largest math dataset ever released. It represents a significant step forward in the development of AI models capable of solving complex mathematical problems. By providing detailed, step-by-step solutions, the dataset facilitates deeper understanding and reasoning. The successful use of this dataset in winning the Progress Prize underscores its potential in advancing the capabilities of AI in mathematical problem-solving.

Future work will focus on several key areas to enhance the dataset and its applications:

- **Expanding the Dataset:** We aim to increase the diversity and quantity of problems in the dataset, particularly by incorporating more problems from various international mathematics competitions and higher education levels. This expansion will help in creating a more robust training resource for AI models.
- **Refining the Annotation Process:** Improving the quality and accuracy of annotations, especially for complex problems, is crucial. Further refinement of CoT annotations will enhance the reasoning capabilities of the models trained on this dataset.
- **Integrating Tool-Based Reasoning:** Exploring new methods for integrating tool-based reasoning, such as incorporating more advanced code interpreters and mathematical solvers, will push the boundaries of what AI models can achieve in terms of problem-solving.
- **Synthetic Data Validation:** In our synthetic data generation pipeline, we use a single-stage approach with GPT-4o at nonzero temperature (0.8). While this generates diverse problems and solutions, it is challenging to validate their correctness. Future work will involve developing methods to further annotate or modify these synthetic problems to improve fine-tuning performance.
- **Improving CoT Solutions for Complex Problems:** For some very challenging problems, the current CoT solutions may lack sufficient detail. Further cleaning up and validating these annotations, possibly with the assistance of domain experts, will improve model performance.
- **Enhancing Instruction Following:** Models fine-tuned on the NuminaMath dataset may lose some capacity for general instruction following. Addressing this issue will involve balancing the dataset to maintain the ability for few-shot learning and general instruction compliance. Another promising avenue is the development of novel alignment objectives.
- **Minimizing Dependency on Proprietary Models:** The dataset's creation heavily relies on GPT-4o, which introduces some restrictions. Future efforts will focus on utilizing more open models to perform necessary tasks, thereby reducing dependency on proprietary solutions.
- **Leveraging Proof Problems:** Approximately 40% of the dataset consists of proof-based problems, particularly from the IMO. Future work will explore translating these problems into formal mathematical proofs, fully leveraging their potential to enhance the reasoning capabilities of AI models.
- **Preventing Data Contamination:** To prevent training data contamination, we included canary strings in the dataset. This practice will be continued and refined in future iterations to ensure the integrity of the dataset.

We invite contributions from the community to further enhance the dataset and its applications. Collaboration will be key to overcoming the current limitations and advancing the state of AI in mathematical reasoning.

Acknowledgments

We express our gratitude to Mistral.ai, General Catalyst, Answer.ai, and the Beijing International Center for Mathematical Research at Peking University for their support from the beginning.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Anthropic. Claude 3.5 sonnet, 2024. URL https://www.anthropic.com/news/claude-3-5-sonnet.

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, 2024. URL https://arxiv.org/abs/2310.10631.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Zui Chen, Yezeng Chen, Jiaqi Han, Zhijie Huang, Ji Qi, and Yi Zhou. An empirical study of data ability boundary in llms' math reasoning, 2024. URL https://arxiv.org/abs/2403.00799.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Run-Ze Fan, Xuefeng Li, Haoyang Zou, Junlong Li, Shwai He, Ethan Chern, Jiewen Hu, and Pengfei Liu. Reformatted alignment, 2024. URL https://arxiv.org/abs/2402.12219.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

GeminiTeam. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pages 11398–11442. PMLR, 2023.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving, 2024. URL https://arxiv.org/abs/2309.17452.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

Conghui He, Zhenjiang Jin, Chao Xu, Jiantao Qiu, Bin Wang, Wei Li, Hang Yan, Jiaqi Wang, and Dahua Lin. Wanjuan: A comprehensive multimodal dataset for advancing english and chinese large models, 2023.

Joy He-Yueya, Gabriel Poesia, Rose E Wang, and Noah D Goodman. Solving math word problems by combining language models with symbolic solvers. *arXiv preprint arXiv:2304.09102*, 2023.

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. *NeurIPS*, 2021.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay 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, 2022. URL https://arxiv.org/abs/2206.14858.

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, 2024. URL https://arxiv.org/abs/2403.04706.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

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, 2023. URL https://arxiv.org/abs/2308.09583.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

William Merrill and Ashish Sabharwal. The expresssive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

William Merrill and Ashish Sabharwal. A logic for expressing log-precision transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math, 2024.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv preprint arXiv:2310.15164*, 2023.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. *Advances in Neural Information Processing Systems*, 36, 2024.

Shuo Yin, Weihao You, Zhilong Ji, Guoqiang Zhong, and Jinfeng Bai. Mumath-code: Combining tool-use large language models with multi-perspective data augmentation for mathematical reasoning. *arXiv preprint arXiv:2405.07551*, 2024.

Weihao You, Shuo Yin, Xudong Zhao, Zhilong Ji, Guoqiang Zhong, and Jinfeng Bai. Mumath: Multi-perspective data augmentation for mathematical reasoning in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2932–2958, 2024.

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.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

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. *arXiv preprint arXiv:2309.05653*, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. Jiuzhang3.0: Efficiently improving mathematical reasoning by training small data synthesis models, 2024. URL https://arxiv.org/abs/2405.14365.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. Solving math word problems via cooperative reasoning induced language models. *arXiv preprint arXiv:2210.16257*, 2022.