MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning

[♣]Xiang Yue[*], [‡]Xingwei Qu, [†]Ge Zhang, [°]Yao Fu, [§]Wenhao Huang,
[♣]Huan Sun, [♣]Yu Su, [†]Wenhu Chen[*]

[†]University of Waterloo, [♣]The Ohio State University, [‡]HKUST, [°]University of Edinburgh, [§]01.AI
yue.149@osu.edu, wenhuchen@uwaterloo.ca

Abstract

We introduce MAmmoTH, a series of open-source large language models (LLMs) specifically tailored for general math problem-solving. The MAmmoTH models are trained on MathInstruct, our meticulously curated instruction tuning dataset. MathInstruct is compiled from 13 math datasets with intermediate rationales, six of which have rationales newly curated by us. It presents a unique hybrid of chain-of-thought (CoT) and program-of-thought (PoT) rationales, and also ensures extensive coverage of diverse fields in math. The hybrid of CoT and PoT not only unleashes the potential of tool use but also allows different thought processes for different math problems. As a result, the MAmmoTH series substantially outperform existing open-source models on nine mathematical reasoning datasets across all scales with an average accuracy gain between 16% and 32%. Remarkably, our MAmmoTH-7B model reaches 33% on MATH (a competition-level dataset), which exceeds the best open-source 7B model (WizardMath) by 23%, and the MAmmoTH-34B model achieves 44% accuracy on MATH, even surpassing GPT-4's CoT result. Our work underscores the importance of diverse problem coverage and the use of hybrid rationales in developing superior math generalist models.
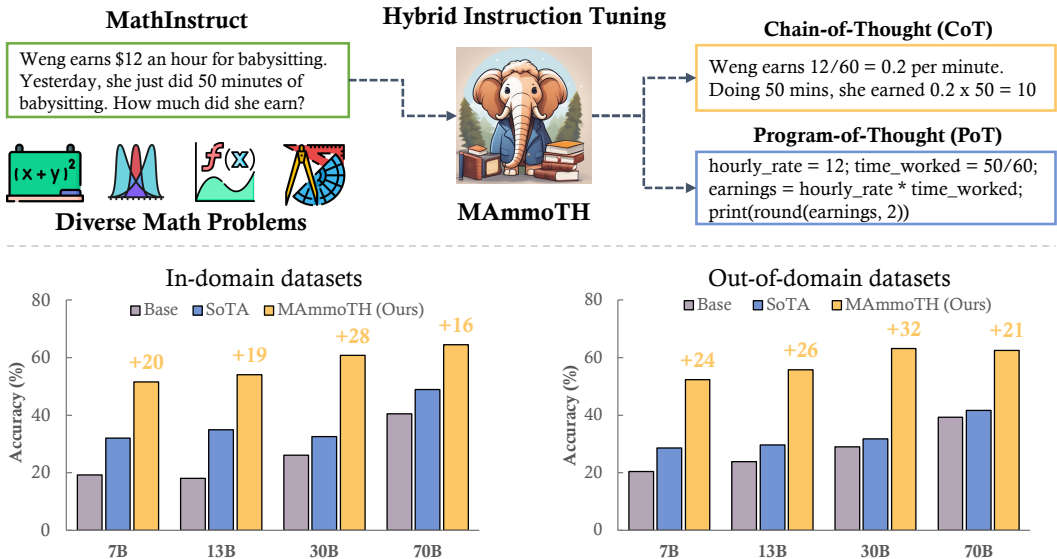
Figure 1: The superior performance of MAmmoTH, a series of models instruction-tuned to solve a diverse set of mathematical problems using hybrid CoT and PoT rationales. MAmmoTH significantly outperforms base and SoTA models on both in-domain and out-of-domain test sets, across all scales.

---

[*]Xiang Yue and Wenhu Chen are the leading authors of the paper. They contributed equally to this project.

1 INTRODUCTION

This work focuses on mathematical reasoning, a critical capability of modern large language models (LLMs) (OpenAI, 2023; Anil et al., 2023). Despite the recent advances in this field, a noticeable gap exists between closed-source and open-source LLMs—closed-source models like GPT-4 (OpenAI, 2023), PaLM-2 (Anil et al., 2023), and Claude 2 (Bai et al., 2022) dominate popular mathematical reasoning benchmarks such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b), while open-source models like Llama (Touvron et al., 2023a;b), Falcon (Penedo et al., 2023), OPT (Zhang et al., 2022) lag behind on all benchmarks by a wide margin.

Current efforts to bridge this gap are twofold: (1) *Continued pre-training* like Galactica (Taylor et al., 2022) and MINERVA (Lewkowycz et al., 2022), which continues to train an LLM on math-related web data of more than 100B tokens. This approach improves a model's general scientific reasoning capability but incurs a high computation cost. (2) *Dataset-specific fine-tuning* like rejection sampling fine-tuning (RFT) (Yuan et al., 2023) and WizardMath (Luo et al., 2023), which fine-tunes LLMs using supervised data specific to certain datasets. Although such approaches improve in-domain performance, they cannot generalize to a wider range of math reasoning tasks beyond their fine-tuning data. For instance, both RFT and WizardMath can increase the accuracy on GSM8K (Cobbe et al., 2021) by 30%+, one of their fine-tuning datasets, but hurt the accuracy on out-of-domain datasets like MMLU-Math (Hendrycks et al., 2021a) or AQuA (Ling et al., 2017) by up to 10%.

In this paper, we aim to propose a lightweight yet generalizable math instruction-tuning approach to enhance the general (i.e., not limited to the fine-tuning tasks) mathematical reasoning capabilities of LLMs. Existing methods (Luo et al., 2023; Yuan et al., 2023; Taylor et al., 2022) primarily focus on Chain-of-Thought (CoT) approaches (Wei et al., 2022b; Nye et al., 2022) to solve math problems through step-by-step natural language descriptions. This approach excels in its generality to cover most math subjects but struggles with computation precision, and complex mathematical or algorithmic reasoning procedures (e.g., solving quadratic equation roots and calculating matrix eigenvalues). In contrast, prompts in the format of code like Program-of-Thought (PoT) approaches (Chen et al., 2022) and PAL (Madaan et al., 2022; Gao et al., 2023) utilize external tools (i.e., Python interpreter) to greatly simplify the math solving process. This approach advocates offloading the computation process to the external Python interpreter to solve complex mathematical and algorithmic reasoning procedures (e.g., solving quadratic equations with sympy or calculating matrix eigenvalues with numpy). However, PoT falls short in dealing with more abstract reasoning scenarios, like commonsense reasoning, formal logic, and abstract algebra, especially when there exist no built-in APIs.

To leverage the strengths of both CoT and PoT approaches, we introduce a new math hybrid instruction-tuning dataset `MathInstruct`, which has two main characteristics: (1) **broad coverage of different math fields and complexity levels**, and (2) **hybrid CoT & PoT rationales**. `MathInstruct` is based on seven existing math rationale datasets and six newly-curated datasets (see details in Table 1). We use `MathInstruct` to fine-tune Llama (Touvron et al., 2023a;b; Rozière et al., 2023) models of different scales ranging from 7B to 70B. The resulting `MAmmoTH` models (Figure 1) demonstrate unprecedented potential in serving as math generalists.

We evaluate `MAmmoTH` on a spectrum of datasets, including in-domain (IND) test sets—GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), AQuA-RAT (Ling et al., 2017), NumGLUE (Mishra et al., 2022b)—and out-of-domain (OOD) test sets—SVAMP (Patel et al., 2021), SAT (Zhong et al., 2023), MMLU-Math (Hendrycks et al., 2021a), Mathematics (Davies et al., 2021), and SimulEq (Koncel-Kedziorski et al., 2016). Compared with existing methods, our models generalize better to OOD datasets and substantially improve the performance of open-source LLMs in mathematical reasoning. Notably, on the popular competition-level MATH dataset (Hendrycks et al., 2021b), our 7B model can beat WizardMath (open-source MATH SoTA) (Luo et al., 2023) by 3.5x (35.2% vs 10.7%), and our 34B `MAmmoTH-Coder` (fine-tuned on Code Llama (Rozière et al., 2023)) can even beat the result of GPT-4 (using CoT).

We highlight our contributions from two perspectives: (1) From the **data engineering perspective**, we present `MathInstruct`, a high-quality math instruction tuning dataset, combining a variety of math problems and hybrid rationales. (2) From the **modeling perspective**, we investigate the impact of various data sources and input-output formats through training and evaluating over 50 different models and baselines ranging from 7B to 70B. Our models, including `MAmmoTH` and `MAmmoTH-Coder`, achieve substantial accuracy gains over existing open-source models.

Training Dataset	Type	Annotation	# Samples	Characteristics	Fields
GSM8K (Cobbe et al., 2021)	CoT	Human	7K	Grade Schol Exam	■
GSM8K-RFT (Yuan et al., 2023)	CoT	Llama	28K	Llama + Validated	■
AQuA-RAT (Ling et al., 2017)	CoT	Human	90K	GRE/GMAT Exam	■
MATH (Hendrycks et al., 2021b)	CoT	Human	7K	Math Competition	■ ■ ■ ■ ■ ■ ■
TheoremQA (Chen et al., 2023) ✮	CoT	GPT-4	600	GPT4 + Validated	■ ■ ■ ■ ■
Camel-Math (Li et al., 2023a)	CoT	GPT-4	50K	GPT4 (Unvalidated)	■ ■ ■ ■ ■
College-Math ✮	CoT	GPT-4	1.8K	GPT4 (Unvalidated)	■
GSM8K ✮	PoT	GPT4	14K	GPT4 + Validated	■
AQuA-RAT ✮	PoT	GPT4	9.7K	GPT4 + Validated	■
MATH ✮	PoT	GPT4	7K	GPT4 + Validated	■ ■ ■ ■
TheoremQA ✮	PoT	GPT4	700	GPT4 + Validated	■ ■ ■ ■ ■
MathQA (Amini et al., 2019)	PoT	Human	25K	AQuA-RAT Subset	■
NumGLUE (Mishra et al., 2022a)	PoT	Human	13K	Lila Annotated	■
`MathInstruct`			260K (72% CoT, 28% PoT)		■ ■ ■ ■ ■ ■ ■

Table 1: Overview of our `MathInstruct`. ✮means with NEW rationales curated by us by prompting GPT-4. We have filtered out augmented samples that have answers inconsistent with the original dataset's annotations. Different colored squares represent different fields in mathematics: ■ Pre-Algebra; ■ Inter-Algebra; ■ Algebra; ■ Probability; ■ NumTheory; ■ Calculus; ■ Geometry.

2 OUR APPROACH

Mathematical reasoning serves as a vital gauge for assessing the ability of LLMs to execute complex multi-hop and quantitative reasoning. Previously, this has been a challenging task for neural networks, which struggle to solve even basic addition and subtraction problems (Yang et al., 2023). However, recent LLMs have considerable advancements in mathematical reasoning. Key breakthroughs have been made through CoT prompting (Wei et al., 2022b; Nye et al., 2022) and PoT prompting (Chen et al., 2022; Gao et al., 2023). CoT prompting encourages LLMs to solve problems incrementally on a scratchpad, enhancing both accuracy and explainability in mathematical reasoning. This approach contrasts with traditional methods that generate answers directly. PoT prompting, on the other hand, formulates the intermediate reasoning process as a program, executed with an external tool like Python, to compute the answer. This method improves robustness in solving complex mathematical problems by offloading the calculations to external tools. However, most existing work (Zhou et al., 2023a) in PoT is limited to proprietary models like GPT-4 (OpenAI, 2023) and Codex (Chen et al., 2021). The PoT potential of open-source models is yet to be seen. Our work aims at optimizing LLMs' CoT and PoT reasoning capabilities through instruction tuning.

2.1 CURATING A DIVERSE AND HYBRID INSTRUCTION TUNING DATASET

Our study aims to compile a list of high-quality and diverse math instruction-tuning datasets, standing out with two main characteristics: (1) broad coverage of different mathematical fields and complexity levels, and (2) hybrid CoT & PoT rationales.

Broad Coverage of Different Math Fields and Complexity Levels: We aim for a broad representation of math fields and complexity levels in our dataset. This ensures exposure to a diverse set of mathematical knowledge, fostering versatility in our models. Based on these criteria, we narrow down our choices to a few high-quality datasets that are widely adopted and encompass different math fields and complexity levels, such as GSM8K, MATH, AQuA, Camel, and TheoremQA. Furthermore, we notice a lack of coverage for college-level math knowledge, such as abstract algebra and formal logic, in existing datasets. To rectify this, we use GPT-4 to synthesize CoT rationales for questions in TheoremQA and create question-CoT pairs through Self-Instruct (Wang et al., 2023h), utilizing a few seed exemplars found online.

Hybrid CoT and PoT Rationales: Contrary to previous work (Yuan et al., 2023; Luo et al., 2023; Lee et al., 2023; Wang et al., 2023g) that focus on CoT, our dataset strategically combines both. This integration enhances the dataset's versatility, catering to varying mathematical problem-solving

approaches. However, most existing datasets provide limited program rationales, leading to an imbalance between CoT and PoT rationales. To fill the gap, we utilize GPT-4 to supplement the PoT rationales for selected datasets, including MATH, AQuA, GSM8K, and TheoremQA. We further enhance the dataset quality by meticulously removing near-duplicated solutions. These GPT-4 synthesized programs are then validated by comparing their executed results with ground truth, ensuring the high quality and reliability of the rationales. The validation subet ratio is shown in Table 6.

Following these guidelines, our instruction dataset, detailed in Table 1, encompasses 260K (instruction, response) pairs, covering a wide range of core mathematical fields (arithmetic, algebra, probability, calculus, and geometry, etc.), including hybrid CoT and PoT rationales, and offering diversity in both language and difficulty levels. This attests to its high quality and unique characteristics.

2.2 TRAINING SETUP

We unify all the subsets in our `MathInstruct` to conform to the structure of an Alpaca-like instruction dataset (Taori et al., 2023). This standardization ensures that the fine-tuned models can process data consistently, regardless of the original dataset formats. We choose the open-source models Llama-2 (Touvron et al., 2023b) and Code Llama (Rozière et al., 2023) as our base models. We fine-tune these models including 7B, 13B, 34B, and 70B on `MathInstruct`, which allows us to validate our `MathInstruct` at multiple scales. We fine-tune all the models with Huggingface transformers library (Wolf et al., 2019). We use a learning rate of 2e-5 for the 7B and 13B models, and 1e-5 for the 34B and 70B models. We set the batch size at 128 and used a cosine scheduler with a 3% warm-up period for three epochs. To efficiently train the computationally intensive 34B and 70B models, we employ DeepSpeed training with ZeRO-3 stage (Rajbhandari et al., 2020).

2.3 EVALUATION SETUP

Our hybrid training enables models to solve problems using either the CoT or PoT approach. By default, the model provides the CoT solution. To switch to the PoT approach, one can add the trigger phrase "Let's write a program to solve the problem" following the question.

Our preliminary evaluation reveals that PoT generally outperforms CoT, notably in open-form questions like GSM8K and MATH, as programmable solutions are better at solving complex mathematical and algorithmic reasoning procedures. However, PoT struggles with abstract reasoning scenarios such as commonsense reasoning, formal logic, and abstract algebra, particularly in the absence of built-in APIs. To further combine the power of both approaches, we introduce a simple hybrid decoding strategy: The model first attempts PoT prompting. If the program is not executable, we fall back to CoT prompting. This heuristic can further enhance our model's overall performance (see more discussions in Table 3.4). We also report the performance of self-consistency decoding method (Wang et al., 2023f) in Table 8.

3 EXPERIMENTS

3.1 EVALUATION DATASETS

We have selected diverse evaluation datasets (Table 2), encompassing a variety of **in-domain** and **out-of-domain** samples across diverse fields of mathematics, to assess the models' capabilities in general mathematical reasoning.

For the in-domain datasets, we consider GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), AQuA-RAT (Ling et al., 2017), and NumGLUE (Mishra et al., 2022b). For the out-of-domain datasets, we choose SVAMP (Patel et al., 2021), Mathematics (Davies et al., 2021), SimulEq (Koncel-Kedziorski et al., 2016), SAT-Math (Zhong et al., 2023), and MMLU-Math (Hendrycks et al., 2021a). The wide selection of evaluation datasets includes math problems from elementary, high school, and college levels. Some of the datasets even include formal logic and commonsense reasoning. The choice of these datasets is to ensure a comprehensive evaluation of the models' capabilities to generalize to unfamiliar situations and different math fields. The chosen evaluation datasets consist of both open-formed questions and multi-choice questions.

Eval Dataset	# Samples	In-Domain?	Answer Form	Fields
GSM8K (Cobbe et al., 2021)	1319	YES	Open-formed	■
MATH (Hendrycks et al., 2021b)	5000	YES	Open-formed	■ ■ ■ ■ ■ ■ ■
AQuA-RAT (Ling et al., 2017)	254	YES	Multi-choice	■
NumGLUE (Mishra et al., 2022b)	1042	YES	Open-formed	■
SVAMP (Patel et al., 2021)	1000	NO	Open-formed	■
Mathematics (Davies et al., 2021)	1000	NO	Open-formed	■ ■ ■ ■
SimulEq (Koncel-Kedziorski et al., 2016)	514	NO	Open-formed	■
SAT-Math (Zhong et al., 2023)	220	NO	Multi-choice	■ ■ ■
MMLU-Math (Hendrycks et al., 2021a)	974	NO	Multi-choice	■ ■ ■ ■

Table 2: Comprehensive overview of our evaluation datasets, featuring a variety of in-domain and out-of-domain problems across diverse fields of mathematics. Different colored squares represent different fields in mathematics: ■ Pre-Algebra; ■ Inter-Algebra; ■ Algebra; ■ Probability; ■ NumTheory; ■ Calculus; ■ Geometry.

3.2 BASELINES

We partition our baselines into the following four categories:

- **Closed-source LLMs:** We consider 4 closed-source LLMs including GPT-4 (OpenAI, 2023), GPT-4 (Code Interpreter), PaLM-2 Unicorn (Anil et al., 2023), Claude-2 (Bai et al., 2022) and Codex (Chen et al., 2021). GPT-4, PaLM-2, and Claude-2 use CoT prompting while GPT-4 (Code Interpreter) and Codex use PoT prompting.
- **Llama Base:** For the base models, we consider Llama-1/2 (Touvron et al., 2023a;b), Llama-2-Chat (Touvron et al., 2023b).
- **Coder Model:** To compare with different coder models, we choose Code-Llama (Rozière et al., 2023), CodeT5+ (Wang et al., 2023i) and CodeGen (Nijkamp et al., 2023).
- **STEM Pre-training:** We cover Galactica (Taylor et al., 2022) mainly to understand the performance of models specialized in STEM knowledge.
- **Instruction Tuning:** We include Orca-Platypus (Mukherjee et al., 2023), Vicuna-1.5 (Zheng et al., 2023b), Tulu (Wang et al., 2023g), Platypus-2 (Lee et al., 2023) and Guanaco (Dettmers et al., 2023). We cover a wide spectrum of models trained with different types of datasets.
- **Dataset-Specific Tuning:** We include both RFT (Yuan et al., 2023) and WizardMath (Luo et al., 2023), which specifically tune the models to adapt to GSM8K and MATH datasets. We include them to understand their generalization.

For most baselines, we choose CoT prompting to maximize their performance due to their incompetence in program generation. All the 'Code Model' use PoT prompting. For GSM8K, MATH, AQuA, and NumGLUE, we will evaluate both 8-shot in-context-learning and zero-shot setups to report the highest score. For SVAMP, Mathematics, SimulEq, SAT, and MMLU, we use 5-shot in-context-learning to maintain consistency with prior work (Wei et al., 2022b; Chen et al., 2023). Our few-shot exemplars are mostly taken from PHP[1] (Zheng et al., 2023a). For MAmmoTH and MAmmoTH-Coder, we always evaluate under 0-shot setting. For all models, we allow a maximum sequence length of 2048 tokens for decoding. For multiple-choice questions, if the generated answer lacks an option, we map it by re-prompting the model: "Please find the closest option to [generated answer]. The options are [options]".

3.3 MAIN RESULTS

We report our in-domain and out-of-domain results in Table 3 and Table 4 respectively. Overall, we can see that MAmmoTH and MAmmoTH-Coder are able to outperform the SoTA model at different scales. In general, the performance gain for OOD datasets is more significant than IND datasets. These results show us the potential of our models as a mathematical generalist. On several datasets, MAmmoTH-Coder-34B and MAmmoTH-70B are even surpassing closed-source LLMs.

---

[1]https://github.com/chuanyang-Zheng/Progressive-Hint

Model	Base	Math-SFT?	GSM8K	MATH	AQuA	NumGLUE	Avg
		Closed-source Model					
GPT-4	-	Unknown	92.0[†]	42.5[†]	72.6[†]	74.7	70.5
GPT-4 (Code-Interpreter)	-	Unknown	97.0[†]	69.7[†]	-	-	-
PaLM-2	-	Unknown	80.7[†]	34.3[†]	64.1	-	-
Claude-2	-	Unknown	85.2[†]	32.5[†]	60.9	-	-
Codex (PoT)	-	No	71.6[†]	36.8[†]	54.1[†]	-	-
ART (InstructGPT)	-	Unknown	71.0	-	54.2	-	-
		7B Parameter Model					
Llama-1	-	No	10.7[†]	2.9[†]	22.6	24.7	15.5
Llama-2	-	No	14.6[†]	2.5[†]	30.3	29.9	19.3
Galactica-6.7B	GAL	GAL-Instruct	10.2	2.2	25.6	25.8	15.9
Code-Llama (PoT)	-	No	25.2	13.0	24.0	26.8	22.2
AQuA-SFT	Llama-2	AQuA	11.2	3.6	35.6	12.2	15.6
Llama-1 RFT	Llama-1	GSM8K	46.5[†]	5.2	18.8	21.1	22.9
WizardMath	Llama-2	GSM8K+MATH	54.9[†]	10.7[†]	26.3	36.1	32.0
MAmmoTH	Llama-2	MathInstruct	53.6	31.5	44.5	61.2	47.7
MAmmoTH-Coder	Code-Llama	MathInstruct	59.4	33.4	47.2	66.4	51.6
Δ			+5	+21	+12	+30	**+20**
		13-15B Parameter Model					
Llama-1	-	No	17.8[†]	3.9[†]	26.0	24.8	18.1
Llama-2	-	No	28.7[†]	3.9[†]	25.1	8.8	16.6
Code-Llama (PoT)	-	No	36.1	16.4	28.7	29.2	27.6
CodeT5+ (PoT)	-	No	12.5	2.4	20.5	19.4	13.7
CodeGen+ (PoT)	-	No	12.7	3.4	24.5	22.5	15.7
Vicuna-1.5	Llama-2	No	28.4[†]	5.8	24.8	36.9	23.9
Llama-1 RFT	Llama-1	GSM8K	52.1[†]	5.1	16.1	24.5	24.4
Orca-Platypus	Llama-2	Platypus	38.4	3.0	18.9	35.3	23.9
Platypus	Llama-2	Platypus	25.7	2.5	33.4	42.3	25.9
WizardMath	Llama-2	GSM8K+MATH	63.9[†]	14.0[†]	21.2	40.8	34.9
MAmmoTH	Llama-2	MathInstruct	62.0	34.2	51.6	68.7	54.1
MAmmoTH-Coder	Code-Llama	MathInstruct	64.7	36.3	46.9	66.8	53.7
Δ			+1	+20	+18	+26	**+19**
		30-34B Parameter Model					
Llama-1	-	No	35.6[†]	7.1[†]	33.4	28.4	26.1
Code-Llama (PoT)	-	No	44.0	23.1	25.2	29.3	30.4
Llama-1 RFT	Llama-1	GSM8K	56.5[†]	7.4[†]	18.5	24.3	26.6
Galactica-30B	GAL	GAL-Instruct	41.7	12.7	28.7	34.7	29.4
Platypus	Llama-1	Platypus	37.8	9.3	27.9	40.5	28.8
Tulu	Llama-2	Tulu	51.0	10.8	25.5	43.4	32.6
MAmmoTH-Coder	Code-Llama	MathInstruct	72.7	43.6	54.7	71.6	60.7
Δ			+16	+21	+21	+28	**+28**
		65-70B Parameter Model					
Llama-1	-	No	50.9[†]	10.6[†]	35.0	50.2	36.6
Llama-2	-	No	56.8[†]	13.5[†]	40.9	50.4	40.4
Llama-2-Chat	Llama-2	No	54.9	18.6	37.0	51.6	40.5
Guanaco	Llama-2	No	59.2	4.1	45.2	53.5	40.5
WizardMath	Llama-2	GSM8K+MATH	81.6[†]	22.7[†]	20.0	48.9	43.3
Platypus	Llama-2	Platypus	70.6	15.6	51.2	55.4	48.1
MAmmoTH	Llama-2	MathInstruct	76.9	41.8	65.0	74.4	64.5
Δ			-5	+19	+14	+19	**+16**

Table 3: The table compiles all the **in-domain** evaluation results. Results marked as † are copied from other papers, which can be found on paperswithcode leaderboards. Math-SFT? means whether the model has been instruction-tuned on any math reasoning datasets. Pink numbers highlight the highest number within the corresponding scale and dataset. Note that there does not exist a 30B+ version for Llama-2 or a 70B version for Code-Llama.

From Table 3, we can observe that our main competitors for IND datasets are WizardMath (Luo et al., 2023) and Platypus (Lee et al., 2023). WizardMath's training is heavily rooted in GSM8K

Model	SVAMP	Mathematics	SimulEq	SAT-Math	MMLU-Math	Avg
Closed-source Model						
GPT-4	97.0[†]	60.8	83.1	95[†]	74.9	82.1
Codex (PoT)	85.2[†]	-	-	68[†]	-	-
7B Parameter Model						
Llama-1	24.5	6.2	4.6	22.7	30.6	17.7
Llama-2	34.5	6.0	5.0	26.8	29.8	20.4
Code-Llama (PoT)	49.4	21.7	3.5	28.6	26.9	26.0
Llama-1 RFT	21.1	5.1	11.0	12.5	21.7	14.3
Galactica-6.7B	25.6	4.6	4.2	17.5	28.0	16.0
WizardMath	36.1	9.3	12.8	25.4	31.1	28.6
Toolformer	29.4[†]	-	-	-	-	-
MAmmoTH	67.7	46.3	41.2	42.7	42.6	48.1
MAmmoTH-Coder	71.4	55.4	45.9	40.5	48.3	52.3
Δ	+22	+34	+33	+14	+17	**+24**
13B Parameter Model						
Llama-1	34.7	6.9	5.4	27.7	30.7	21.0
Llama-2	35.1	11.5	5.8	32.7	34.4	23.9
Code-Llama (PoT)	60.0	21.3	3.8	25.9	27.7	27.7
Vicuna-1.5	55.7	10	6.6	34.0	34.1	28.1
Llama-1 RFT	46.5	6.7	10.1	13.2	21.6	19.6
WizardMath	51.9	14.1	14.9	24.5	32.1	27.5
Platypus	55.4	11.4	7.4	36.8	35.5	29.3
Orca-Platypus	56.8	12.6	7.9	29.5	41.6	29.7
MAmmoTH	72.4	49.2	43.2	46.8	47.6	51.8
MAmmoTH-Coder	73.7	61.5	47.1	48.6	48.3	55.8
Δ	+14	+40	+33	+12	+7	**+26**
30-34B Parameter Model						
Llama-1	48.8	12.8	11.2	33.4	39.0	29.0
Code-Llama (PoT)	69.1	34.5	6.8	26.8	21.6	31.7
Llama-1 RFT	55.4	7.6	12.8	20.4	37.9	26.8
Galactica-30B	41.6	11.8	13.2	37.7	37.9	28.4
Tulu	59.0	10.7	10.3	31.3	39.8	30.2
Platypus	51.7	13.8	13.6	38.6	41.0	31.7
MAmmoTH-Coder	84.3	65.4	51.8	60.9	53.8	63.2
Δ	+15	+31	+38	+22	+13	**+32**
65-70B Parameter Model						
Llama-1	55.3	14.2	15.2	37.4	44.1	33.2
Llama-2	63.8	20.5	14.0	51.3	47.1	39.3
Llama-2-Chat	71.5	19.2	21.7	44.1	46.9	40.6
WizardMath	71.8	17.1	37.9	13.2	27.4	33.4
Guanaco	66.8	17.8	20.2	50.0	47.3	40.4
Platypus	51.8	26.3	21.7	55.9	52.5	41.6
MAmmoTH	82.4	55.6	51.4	66.4	56.7	62.5
Δ	+11	+29	+14	+11	+4	**+21**

Table 4: The table compiles all the **out-of-domain** evaluation results. Results marked as † are copied from other papers, which can be found on paperswithcode leaderboards.

and MATH datasets. Therefore, WizardMath's results are highly competitive on these two datasets. However, the dataset-specific training can be detrimental to OOD datasets like AQuA. In contrast, Platypus fine-tunes LLMs on a wide range of text and math reasoning datasets. it improves the open-source SoTA on several datasets. Similarly, MAmmoTH can achieve universal improvement across the board. A major observation is that MAmmoTH is particularly strong at solving more complex math problems in MATH, where the gain of our model over WizardMath (open-source SoTA on MATH) can exceed 25% at different scales.
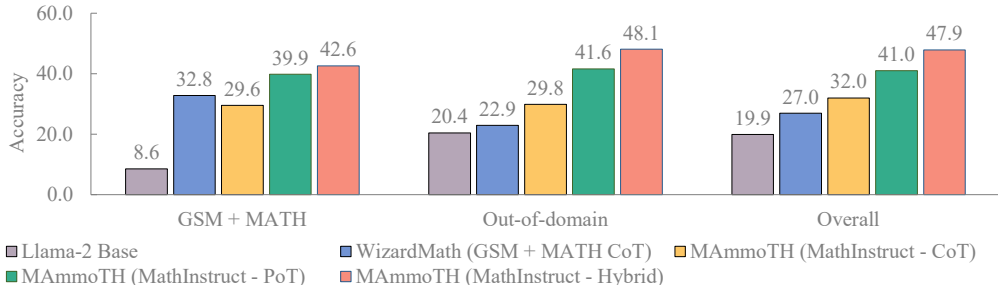
Figure 2: Investigation of the influence of CoT & PoT hybrid training on the 7B Llama-2 model. "Out-of-domain" refers to the five datasets detailed in Table 2. Key insights include: 1) The SoTA model, utilizing dataset-specific CoT fine-tuning on GSM and MATH, displays strong performance within its domains but struggles in OOD scenarios; 2) Diverse data sources in `MathInstruct` enable better math generalist model; 3) Fine-tuning on the PoT subsets generally outperforms fine-tuning on the CoT subsets; 4) Hybrid training yields the best-performing model. The breakdown results on each dataset can be found in Appendix Table 9.

From Table 4, we can observe that our main competitor for OOD datasets is Platypus (Lee et al., 2023). Similar to in-domain results, Platypus is able to yield gains over the baseline models universally across the board, especially on the MMLU-Math dataset, which is tied with `MAmmoTH`-70B. It is worth noting that the performance gains of our model on OOD datasets are even more significant than on in-domain datasets. This demonstrates our models' remarkable generalizability to unseen math problems. Notably, `MAmmoTH`-7B also boosts the CoT performance of WizardMath-7B greatly on MMLU-Math by 9%, which contains a substantial number of questions beyond the subjects we covered in our training dataset.

Comparison between Different Base Models. In our experiments, we experimented with both Llama-2 and Code-Llama as the base models. From the two tables, we can observe that Code-Llama is consistently better than Llama-2, especially on OOD datasets. The gap between `MAmmoTH` and `MAmmoTH-Coder` can even reach up to 5%. Surprisingly, the average performance on OOD datasets of `MAmmoTH-Coder` (34B) is actually higher than `MAmmoTH` (70B). We believe `MAmmoTH-Coder` benefits greatly from the continuous code training of Code-Llama, which not only enhances the PoT capabilities but also improves Llama's general reasoning skills.

3.4 ABLATION STUDY ON DATA SOURCE

Ablation of the Data Source. In order to better understand what factors contribute to the great gain of `MAmmoTH` over existing baselines, we set up a group of control experiments in Figure 2. We study the following setups:

(1) `MAmmoTH` (`MathInstruct`- CoT): This experiment aims to understand how much our curated CoT data could improve the generalization over the SoTA model WizardMath (Luo et al., 2023) trained specifically on GSM + MATH. As can be seen, while sacrificing accuracy on GSM + MATH by 3%, our CoT subset fine-tuning improves the overall nine-dataset accuracy from 27% to 32%.

(2) `MAmmoTH` (`MathInstruct`- PoT): This experiment aims to understand the advantage of our PoT subset. As can be observed, our PoT subset fine-tuning can significantly improve the overall accuracy from 27% to 41%. This ablation reflects the importance of unlocking the program generation capabilities of our model.

(3) `MAmmoTH` (`MathInstruct`- Hybrid): We further combine CoT and PoT as the hybrid training data to achieve the best overall performance of 47.9%. This combined gain comes from two aspects:

- The CoT subset helps maintain generic language-based reasoning skills to handle scenarios where PoT cannot handle well, e.g., abstract reasoning multi-choice questions in AQuA and MMLU.

- The PoT subset can teach the model how to utilize Python APIs to solve complex math problems with high precision, e.g., the MATH problems requiring complex computation.

Training Data	GSM	MATH	AQuA	NumG	SVA	Mat	Sim	SAT	MMLU	AVG
-	14.6	2.5	30.3	29.9	34.5	6.0	5.0	26.8	29.8	-25.3
G	56.6	9.2	24.4	32.1	65.4	20.5	12.3	27.2	25.2	30.3
M	27.1	25.5	27.8	32.1	47	49.4	10.5	26.4	27.4	30.4
A	15.3	5.8	39.7	15.5	15.3	6.3	7.2	32.7	36.6	19.4
G + M	58.1	28.2	26.0	34.7	64.8	50.1	17.1	28.6	28.4	37.3
G + M + T	56.5	26.5	27.4	35.5	64.4	50.6	18.8	29.1	29.1	37.5
G + M + C	57.4	28.5	26.2	37.5	65.3	50.4	17.7	29.3	28.7	37.9
G + M + A	56.1	27.1	37.8	37.2	64.8	48.2	19.8	35.4	39.8	40.7
G + M + C + A	57.5	29.1	46.9	42.2	65.8	49.6	32.7	42.3	43.1	45.5
M + C + A + N	24.7	26.1	39.4	59.7	61.6	48.6	43.4	36.4	41.2	42.3
G + M + C + N	50.8	26.2	20.9	65.5	65.8	48.5	41.4	26.4	24.7	41.1
G + C + A + N	51.3	14.8	41.7	58.8	66.3	31.0	42.2	34.1	40.5	42.3
G + M + C + A + T	55.4	28.6	42.5	44.9	65.4	50.8	34.9	41.3	42.5	45.1
G + M + C + A + N	56.5	28.9	38.2	63.7	64.1	47.9	40.8	38.6	44.5	47.0
G + M + C + A + N +T	53.8	27.0	38.2	60.8	65.9	50.8	41.8	42.5	42.7	47.1
G + M + C + A + N + MQA	55.7	28.8	42.5	62.1	64.6	45.9	38.9	41.3	45.0	47.2
`MathInstruct`	53.6	31.5	44.5	61.2	67.7	46.3	41.2	42.7	42.6	47.9

Table 5: Influence of different major subsets in `MathInstruct` based on Llama-2 7B. G: GSM8K, M: MATH, C: Camel, A: AQuA, N: NumGLUE, MQA: MathQA. "Existing data": the subset of `MathInstruct` in Table 1 by excluding all the NEW rationales curated by us. We shorten Mathematics as Mat, SimulEq as Sim, NumGLUE as NumG, and SVAMP as SVA to save space.

We put some case studies in Appendix B to demonstrate the respective advantages of PoT and CoT in solving different types of math problems. To summarize, we attribute our substantial gain to: 1) diverse data sources covering different math fields and complexity levels and 2) a hybrid of CoT & PoT instruction tuning and decoding strategy.

Influence of Major Subsets. Given the diverse sources of `MathInstruct` used in training `MAmmoTH`, it is important to understand how each dataset contributes to the overall performance of the model. We focus on five significant subsets: GSM8K, MATH, Camel, AQuA, and NumGLUE. We conduct an experiment gradually adding each dataset into training and compare the performance with the one fine-tuned on the whole `MathInstruct`. These results underscore the significant impact of diverse data sources on `MAmmoTH` performance, a core aspect of making `MAmmoTH` a math generalist. The results also provide valuable insights for future data curation and collection efforts (e.g., we should always collect diverse data and avoid collecting only specific types of data).

To help understand the contribution of the 6 newly curated datasets as shown in Table 1, we remove them from `MathInstruct`, and train a model on the existing data. As shown in the last two rows of Table 5, our new curated data substantially improves the performance on many datasets and leads to a 9% overall increase, which reflects the importance of the NEWLY curated dataset.

Influence of Hybrid Decoding. To demonstrate the effectiveness of the hybrid decoding method, we conduct an experiment as outlined in subsection 2.3. By default, we initially attempt the PoT decoding method for a given question. If it fails to generate an executable query, we then transition to the CoT decoding method. The performance of different decoding methods (CoT, PoT, and Hybrid) is shown in Table 10. This hybrid decoding improves performance on every test set, showcasing that our model can effectively leverage the strengths of both CoT and PoT decoding strategies.

4 CONCLUSION

In this paper, we propose a novel math instruction tuning approach to activate open-source LLMs' mathematical reasoning capabilities. Through a comprehensive study, we show that our models can outperform the SoTA performance at different scales by a huge margin. Our models benefit massively from: 1) the broad coverage of different math fields and complexity levels, and 2) a hybrid of CoT and PoT training. Our instruction tuning dataset contains 260K samples, which makes fine-tuning highly affordable even for academic labs. Our work paves the road for future studies to activate LLMs' core capabilities in specialized domains.

REFERENCES

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2357–2367, 2019. doi: 10.18653/v1/N19-1245. URL `https://aclanthology.org/N19-1245`.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *ArXiv preprint*, abs/2305.10403, 2023. URL `https://arxiv.org/abs/2305.10403`.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *ArXiv preprint*, abs/2212.08073, 2022. URL `https://arxiv.org/abs/2212.08073`.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374, 2021. URL `https://arxiv.org/abs/2107.03374`.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *ArXiv preprint*, abs/2211.12588, 2022. URL `https://arxiv.org/abs/2211.12588`.

Wenhu Chen, Ming Yin, Max Ku, Elaine Wan, Xueguang Ma, Jianyu Xu, Tony Xia, Xinyi Wang, and Pan Lu. Theoremqa: A theorem-driven question answering dataset. *ArXiv preprint*, abs/2305.12524, 2023. URL `https://arxiv.org/abs/2305.12524`.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *ArXiv preprint*, abs/2210.11416, 2022. URL `https://arxiv.org/abs/2210.11416`.

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*, abs/2110.14168, 2021. URL `https://arxiv.org/abs/2110.14168`.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021. URL `https://www.nature.com/articles/s41586-021-04086-x`.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *ArXiv preprint*, abs/2305.14314, 2023. URL `https://arxiv.org/abs/2305.14314`.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. Compositional semantic parsing with large language models. *International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=gJW8hSGBys8`.

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*, pp. 10764–10799. PMLR, 2023. URL `https://proceedings.mlr.press/v202/gao23f/gao23f.pdf`.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *ArXiv preprint*, abs/2305.11738, 2023. URL `https://arxiv.org/abs/2305.11738`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021a. URL `https://openreview.net/forum?id=d7KBjmI3GmQ`.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL `https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf`.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 523–533, 2014. doi: 10.3115/v1/D14-1058. URL `https://aclanthology.org/D14-1058`.

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

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015. doi: 10.1162/tacl_a_00160. URL `https://aclanthology.org/Q15-1042`.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1152–1157, 2016. doi: 10.18653/v1/N16-1136. URL `https://aclanthology.org/N16-1136`.

Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *ArXiv preprint*, abs/2308.07317, 2023. URL `https://arxiv.org/abs/2308.07317`.

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. URL `https://openreview.net/pdf?id=IFXTZERXdM7`.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large scale language model society. *ArXiv preprint*, abs/2303.17760, 2023a. URL `https://arxiv.org/abs/2303.17760`.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5315–5333, 2023b. URL `https://aclanthology.org/2023.acl-long.291.pdf`.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167, 2017. doi: 10.18653/v1/P17-1015. URL `https://aclanthology.org/P17-1015`.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *ICML*, 2023. URL `https://openreview.net/pdf?id=ZX4uS605XV`.

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

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1384–1403, 2022. URL `https://aclanthology.org/2022.emnlp-main.90.pdf`.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. LILA: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5807–5832, 2022a. URL `https://aclanthology.org/2022.emnlp-main.392`.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3505–3523, 2022b. doi: 10.18653/v1/2022.acl-long.246. URL `https://aclanthology.org/2022.acl-long.246`.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *ArXiv preprint*, abs/2306.02707, 2023. URL `https://arxiv.org/abs/2306.02707`.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/pdf?id=iaYcJKpY2B_`.

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. In *Deep Learning for Code Workshop*, 2022. URL `https://arxiv.org/abs/2112.00114`.

OpenAI. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774, 2023. URL `https://arxiv.org/abs/2303.08774`.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021. doi: 10.18653/v1/2021.naacl-main.168. URL `https://aclanthology.org/2021.naacl-main.168`.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *ArXiv preprint*, abs/2306.01116, 2023. URL `https://arxiv.org/abs/2306.01116`.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *ArXiv preprint*, abs/2304.03277, 2023. URL `https://arxiv.org/abs/2304.03277`.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020. URL `https://dl.acm.org/doi/10.5555/3433701.3433727`.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015. doi: 10.18653/v1/D15-1202. URL `https://aclanthology.org/D15-1202`.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *ArXiv preprint*, abs/2308.12950, 2023. URL `https://arxiv.org/abs/2308.12950`.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. URL `https://openreview.net/forum?id=9Vrb9D0WI4`.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *ArXiv preprint*, abs/2210.09261, 2022. URL `https://arxiv.org/abs/2210.09261`.

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. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *ArXiv preprint*, abs/2211.09085, 2022. URL `https://arxiv.org/abs/2211.09085`.

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

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

Boshi Wang, Xiang Deng, and Huan Sun. Iteratively prompt pre-trained language models for chain of thought. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2714–2730. Association for Computational Linguistics, 2022a. URL `https://aclanthology.org/2022.emnlp-main.174`.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2717–2739. Association for Computational Linguistics, 2023a. doi: 10.18653/v1/2023.acl-long.153. URL `https://aclanthology.org/2023.acl-long.153`.

Boshi Wang, Xiang Yue, and Huan Sun. Can chatgpt defend the truth? automatic dialectical evaluation elicits llms' deficiencies in reasoning. *ArXiv preprint*, abs/2305.13160, 2023b. URL `https://arxiv.org/abs/2305.13160`.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *ArXiv preprint*, abs/2305.04091, 2023c. URL `https://arxiv.org/abs/2305.04091`.

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. *ArXiv preprint*, abs/2309.02144, 2023d. URL `https://arxiv.org/abs/2309.02144`.

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*, abs/2307.10635, 2023e. URL `https://arxiv.org/abs/2307.10635`.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations (ICLR)*, 2023f. URL `https://openreview.net/pdf?id=1PL1NIMMrw`.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, 2022b. URL `https://aclanthology.org/2022.emnlp-main.340`.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? exploring the state of instruction tuning on open resources. *ArXiv preprint*, abs/2306.04751, 2023g. URL `https://arxiv.org/abs/2306.04751`.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *The 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*, 2023h. URL `https://aclanthology.org/2023.acl-long.754.pdf`.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi DQ Bui, Junnan Li, and Steven CH Hoi. Codet5+: Open code large language models for code understanding and generation. *ArXiv preprint*, abs/2305.07922, 2023i. URL `https://arxiv.org/abs/2305.07922`.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022a. URL `https://openreview.net/forum?id=gEZrGCozdqR`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022b. URL `https://openreview.net/pdf?id=_VjQlMeSB_J`.

Jerry Wei, Da Huang, Yifeng Lu, Denny Zhou, and Quoc V Le. Simple synthetic data reduces sycophancy in large language models. *ArXiv preprint*, abs/2308.03958, 2023. URL `https://arxiv.org/abs/2308.03958`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771, 2019. URL `https://arxiv.org/abs/1910.03771`.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. URL `https://openreview.net/forum?id=RdJVFCHjUMI`.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Decomposition enhances reasoning via self-evaluation guided decoding. *ArXiv preprint*, abs/2305.00633, 2023. URL `https://arxiv.org/abs/2305.00633`.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *ArXiv preprint*, abs/2304.12244, 2023. URL `https://arxiv.org/abs/2304.12244`.

Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. Gpt can solve mathematical problems without a calculator. *ArXiv preprint*, abs/2309.03241, 2023. URL `https://arxiv.org/abs/2309.03241`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/pdf?id=WE_vluYUL-X`.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7163–7189, 2021. doi: 10.18653/v1/2021.emnlp-main.572. URL `https://aclanthology.org/2021.emnlp-main.572`.

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*, abs/2309.12284, 2023. URL `https://arxiv.org/abs/2309.12284`.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *ArXiv preprint*, abs/2308.01825, 2023. URL `https://arxiv.org/abs/2308.01825`.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *ArXiv preprint*, abs/2205.01068, 2022. URL `https://arxiv.org/abs/2205.01068`.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models. *ArXiv preprint*, abs/2304.09797, 2023a. URL `https://arxiv.org/abs/2304.09797`.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv preprint*, abs/2306.05685, 2023b. URL `https://arxiv.org/abs/2306.05685`.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. *ArXiv preprint*, abs/2304.06364, 2023. URL `https://arxiv.org/abs/2304.06364`.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *ArXiv preprint*, abs/2308.07921, 2023a. URL `https://arxiv.org/abs/2308.07921`.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *ArXiv preprint*, abs/2305.11206, 2023b. URL `https://arxiv.org/abs/2305.11206`.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. *International Conference on Learning Representations (ICLR)*, 2023c. URL `https://openreview.net/pdf?id=WZH7099tgfM`.

A RELATED WORK

A.1 MATHEMATICAL REASONING DATASETS

Our work builds upon the existing mathematical reasoning literature. Early on, mathematical reasoning is mostly focused on solving synthetic basic math problems like AddSub (Hosseini et al., 2014) and other arithmetic reasoning datasets (Koncel-Kedziorski et al., 2015; Roy & Roth, 2015; Patel et al., 2021). Later on, more difficult math word problem datasets (Cobbe et al., 2021; Amini et al., 2019; Ling et al., 2017; Hendrycks et al., 2021b) have been proposed to focus on addressing realistic math word problems. NumGLUE (Mishra et al., 2022b) and LiLA (Mishra et al., 2022a) compile the existing literature to build a more diversified dataset collection. However, these datasets are mostly focused on grade school math problems. To further test LLMs' limits in addressing more complex math problems, MMLU (Hendrycks et al., 2021a) includes college math problems in its evaluation suite. More recently, (Chen et al., 2023; Wang et al., 2023e) have proposed to tackle more challenging college-level science and math problems. Our instruction tuning dataset is built upon existing work to include a diversified collection of math problems from different subfields.

A.2 REASONING WITH LARGE LANGUAGE MODELS

LLMs have demonstrated great capabilities to reason with the help of Chain-of-Thought prompting (Wei et al., 2022b; Kojima et al., 2022; Wang et al., 2023f). Suzgun et al. (2022) have shown that CoT can already surpass human performance on challenging BIG-Bench tasks. Later on, several other works (Drozdov et al., 2023; Zhou et al., 2023c; Nye et al., 2022; Wang et al., 2022a; 2023a; Li et al., 2023b; Wang et al., 2023d; Yu et al., 2023) also propose different approaches to utilize LLMs to solve reasoning tasks by allowing intermediate steps. ReAct Yao et al. (2023) proposes to leverage external tools like search engines to enhance LLM reasoning skills. Another trend is to enable LLMs' capabilities to use programs as thought processes like PoT (Chen et al., 2022). Some follow-up works include self-critic (Gou et al., 2023), self-eval (Xie et al., 2023), plan-and-solve (Wang et al., 2023c). These methods propose to enhance LLMs' capabilities to solve math problems with PoT. Self-critic (Gou et al., 2023) and self-eval (Xie et al., 2022) both adopt self-evaluation to enhance the robustness of the generated program. Plan-and-solve (Wang et al., 2023c) instead adopts more detailed planning instructions to help LLMs create a high-level reasoning plan. These methods all prove to bring decent improvements over PoT.

A.3 INSTRUCTION TUNING IN LANGUAGE MODELS

Instruction tuning is part of a line of work designed to "align" language models with more useful objectives and human preferences. The instruction tuning step is seen as a major step to activate LLMs' certain capabilities to respond to human instructions. Previously, instruction tuning is mainly focused on enhancing LLMs' general-purpose instruction following abilities. Since 2021, CrossFit (Ye et al., 2021) and NaturalInstruction (Wang et al., 2022b), FLAN (Wei et al., 2022a) and T0 (Sanh et al., 2022) are amongst the first wave of instruction tuning effort to understand LLMs' generalization capabilities. Later on, FLAN-v2 (Chung et al., 2022; Longpre et al., 2023) have been proposed to understand the effect of scaling up the instruction datasets to understand its impact on model performance. These approaches mainly adopt human-annotated datasets to build the instruction following dataset. More recently, multiple works (Wang et al., 2023h; Xu et al., 2023; Peng et al., 2023; Zhou et al., 2023b; Wang et al., 2023g) propose to utilize synthetic instruction following data distilled from GPT-3/4 to align open-source LLMs. The most similar effort to ours is Platypus (Lee et al., 2023) which aims to utilize a domain-specialized dataset to construct a small-scale instruction following dataset to enhance LLMs' reasoning capabilities.

B CASE STUDY

We conduct a comparison between our PoT results vs. CoT results in Figure 3, Figure 4 and Figure 5. In the first example, even though PoT and CoT can both solve the problem, CoT gives a very tedious solution to derive the answer. Such solution is not only slow but also unstable. In the second and third case, we can further see the advantages of PoT over CoT by utilizing external tools and Python packages to greatly simplify the solution. Figure 6 shows some types of questions (especially the formal logic question) that are not easily handled by programs. In order to address these types of questions, CoT is a better choice.

What is the value of $\frac{(3!)!}{3!}$?

PoT

```
from sympy import factorial

value = factorial(factorial(3)) / factorial(3)
print(value)
```

>>> 120

CoT

First, we need to understand the given expression: $\frac{(3!)!}{3!}$

Step 1: Simplify the expression inside the parentheses.
$3! = 3 \times 2 \times 1 = 6$
So, the expression becomes: $\frac{(6)!}{6}$

Step 2: Recall the property of factorials that states $(n!)^k = n!$ for any positive integer k and any positive integer $n \geq k$. In our case, we have $n = 6$ and $k = 6$.
Using this property, we can rewrite the expression as: $\frac{6!}{6}$

Step 3: Simplify the expression further by applying the property mentioned in Step 2. $6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 720$
So, the expression becomes: $\frac{720}{6}$

Step 4: Divide 720 by 6. $720 \div 6 = 120$
So, the final simplified expression is: 120
Therefore, the value of $\frac{(3!)!}{3!}$ is $\boxed{120}$.

The answer is: 120

Figure 3: Example 1: PoT and CoT can both solve the problem, however, CoT gives a very tedious solution to derive the answer.

Determine the number of ways to arrange the letters of the word TEPEE.

PoT

```
from sympy import factorial

word = "TEEPE"
n = len(word)
d = word.count('E')
ways = factorial(n)/factorial(d)
print(ways)
```

>>> 20

CoT

First we count the arrangements if all the letters are unique, which is 4!. Then since the E's are not unique, we divide by 2! for the arrangements of the two E's, for an answer of $\frac{4!}{2!} = \boxed{60}$.

The answer is: 60

Figure 4: Example 2: PoT generates the correct solution while CoT fails.

Question: a is chosen from the set {1,2,3,4}, and b is chosen from the set {5,6,7,8,9}. What is the probability that the product ab is a prime number?

PoT

```python
from sympy import isprime
# define the sets
a_set = {1,2,3,4}
b_set = {5,6,7,8,9}
n_a = len(a_set)
n_b = len(b_set)
prime_count = 0
prime_ab_count = 0
prime_a_count = prime_ab_count
prime_b_count = prime_ab_count
# iterate over all possible pairs
for a in a_set:
  for b in b_set:
    if isprime(a) or isprime(b):
      if isprime(a*b):
        # increment the counters
        prime_count += 1
        prime_ab_count += 1
      # check if a is prime
      if isprime(a):
        prime_a_count += 1
      if isprime(b):
        prime_b_count += 1
      probability = prime_ab_count / (n_a * n_b)
print(probability)
>>> 0.1
```

CoT

First, we need to find the total number of possible combinations of choosing a number from each set. There are 4 numbers in the first set and 5 numbers in the second set. So, there are $4 \cdot 5 = 20$ possible combinations.

Now, let's consider the prime numbers. A prime number is a number greater than 1 that has no divisors other than 1 and itself. We need to find the pairs of (a, b) where either a or b is a prime number, or both.

1. If a is a prime number, there are 4 prime numbers in the first set: 2, 3, 5, and 7. So, there are 4 possible combinations where a is a prime number.
2. If b is a prime number, there are 5 prime numbers in the second set: 3, 5, 7, 11, and 13. So, there are 5 possible combinations where b is a prime number.
3. If both a and b are prime numbers, we need to find the pairs where the prime numbers are 2, 3, 5, or 7. There are 4 pairs in this case: (2, 3), (2, 5), (2, 7), and (3, 5).

Now, let's add up the number of combinations where either a or b is a prime number: 4 (a is prime) + 5 (b is prime) + 4 (both are prime) = 13.

So, there are 13 combinations where either a or b is a prime number. Since there are 20 possible combinations in total, the probability that the product ab is a prime number is $\frac{13}{20}$.

Therefore, the probability is $\frac{13}{20}$

The answer is: $\frac{13}{20}$

Figure 5: Example 3: PoT generates the correct solution while CoT fails.

If n is such that $36 \le n \le 72$, then $x = (n2 + 2\sqrt{n}(n + 4) + 16)/(n + 4\sqrt{n} + 4)$ satisfies.
Answer Choices: (A) $20 < x < 54$ (B) $23 < x < 58$ (C) $25 < x < 64$ (D) $28 < x < 60$ (E) None of these

The expression in the numerator can be rewritten as $(n + 2\sqrt{n} + 4)^2$, and so the expression is equal to $n + 2\sqrt{n} + 4$. We substitute the smallest value of n, which is 36, and get 42. We substitute the largest value of n, which is 72, and get 88. **So the answer is (E) None of these**

CoT

Which of the following could NOT be the sides of an isosceles triangle?
Answer Choices: (A) 2,2,3 (B) 2,2,4 (C) 4,12,12 (D) 5,5,5

This is because in an isosceles triangle, the lengths of two sides are equal, and the length of the third side must be less than the sum of the lengths of the other two sides. In this case, 4 is equal to the sum of the other two sides (2+2), not less. **Therefore, the answer is (B) 2,2,4.**

CoT

Select the best translation into predicate logic. Some CEO is wealthier than everyone. (Cx: x is a CEO; Px: x is a person; Wxy: x is wealthier than y)
Answer Choices: (A) $(\exists x)[Cx \bullet (\forall y)(Py \bullet Wxy)]$ (B) $(\exists x)[Cx \supset (\forall y)(Py \supset Wxy)]$
(C) $(\exists x)[Cx \bullet (\forall y)(Py \supset Wxy)]$ (D) $(\forall x)[Px \supset (\exists y)(Cy \supset Wxy)]$

$\exists x$: There exists some x Cx: x is a CEO, and
$\forall y$: For all y Py: y is a person, and Wxy: x is wealthier than y
So, in plain English, this expression says: "There exists some x such that x is a CEO and for all y, if y is a person then x is wealthier than y." Which matches the original statement.
Therefore the answer is (A) $(\exists x)[Cx \bullet (\forall y)(Py \bullet Wxy)]$
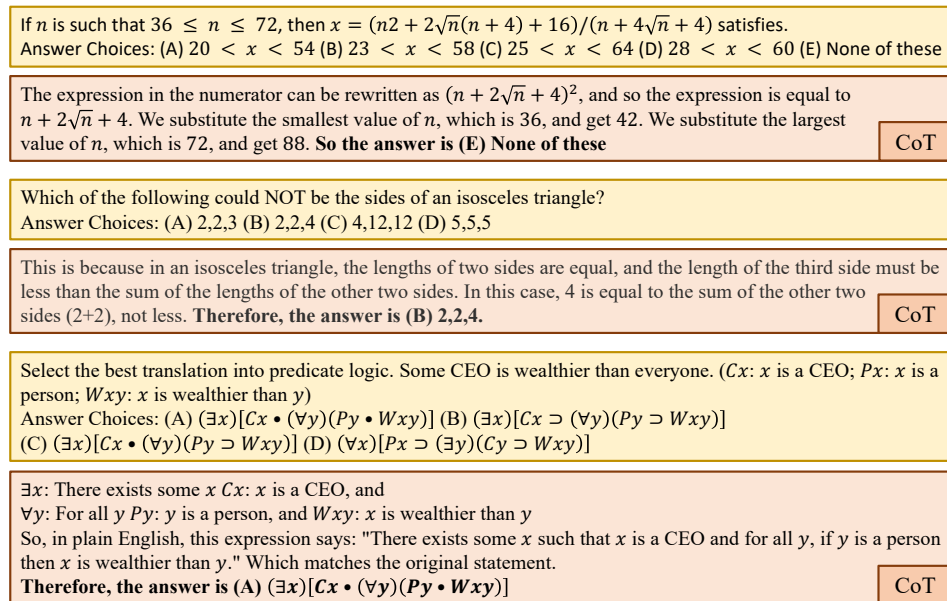
CoT

Figure 6: Example 4: Some types of questions (e.g., formal logic) are hard to be solved by PoT but could be handled by CoT.

C   KEY INSIGHTS

We show the results in Table 5. And our key findings are summarized as follows:

- When the model is trained on individual datasets (GSM8K, MATH, or AQuA), it demonstrates strong performance in tasks closely related to the respective training dataset. However, it generally struggles to generalize this performance to other tasks.
- The datasets GSM8K, MATH, and AQuA emerge as crucial components in the training process. Training the model on a combination of these datasets leads to a marked improvement in its overall performance, indicative of the complementary nature of these datasets. Omitting any one of these datasets results in a noticeable decline in overall performance.
- TheoremQA, despite its smaller scale, exerts some influence on the model's performance. It tends to enhance the model's capabilities on datasets like Mathematics and SimulEq, yet it may slightly impede performance on GSM8K and MATH. Nevertheless, its inclusion in the training regimen is favored, as it introduces a broader variety of questions and could potentially aid the model in generalizing to other complex problems that require advanced computational skills.
- Incorporating a wider range of datasets into the training consistently improves the model's performance across most tasks. This suggests that a broad and diverse knowledge base is instrumental in enhancing the model's ability to generalize.

D ADDITIONAL RESULTS

We added some additional results here. The results provides how much of GPT-4 generated data are being filtered (see Table 6). The results also include the generalization benefits of MathInstruct vs. only training on GSM (see Table 7). We also include different self-consistency setup to show how far our method can reach with ensemble decoding methods.

	GPT-4 Rationales Validated Ratio
AQuA-PoT	75% (9772 /13000)
MATH-PoT	51% (7088 / 14000)
TheoremQA-PoT	44% (703 / 1600)
TheoremQA-CoT	37% (592 / 1600)
GSM8K - PoT	81% (14591 / 18000)

Table 6: The validated ratio of GPT-4 distilled rationales of newly curated subsets in Table 1.

	GSM	MATH	AQUA	NumG	SVA	Mat	Sim	SAT	MMLU	AVG
CodeLlama - 7B - GSM8K	**63.2**	18.4	25.4	32.9	67.6	40.1	16.3	26.4	26.1	35.2
CodeLlama - 7B - MathInstruct	59.4	**33.4**	**47.2**	**66.4**	**71.4**	**55.4**	**45.9**	**40.5**	**48.3**	**52.0**

Table 7: CodeLlama 7B trained on GSM8K and `MathInstruct`. Training exclusively on a single dataset like GSM8K leads to challenges in generalizing to other datasets

Decoding Method	GSM	MATH
1 PoT	58.8	32.1
1 PoT + 1 CoT backup	59.4	33.4
5 PoT + 5 CoT self-consistency	66.4	37.0
10 PoT self-consistency	67.0	37.8
10 PoT + 10 CoT self-consistency	69.6	38.2
20 PoT self-consistency	70.1	38.6

Table 8: Exploration of different decoding methods. Though the self-consistency Wang et al. (2023f) decoding can further improve the performance, it introduces significantly larger inference. Our chosen strategy, while straightforward, strikes an optimal balance between complexity, performance, and practical applicability.

Model	GSM	MATH	AQuA	NumG	SVA	Mat	Sim	SAT	MMLU	Overall
Base	14.6	2.5	30.3	29.9	34.5	6.0	5.0	26.8	29.8	19.9
WizzardMath	54.9	10.7	26.3	36.1	36.1	9.3	12.8	25.4	31.1	27.0
MAmmoTH (MathInstruct-CoT)	49.2	9.9	42.2	37.1	48.5	9.5	17.3	34.1	39.8	32.0
MAmmoTH (MathInstruct-PoT)	50.8	28.9	28.6	52.7	65.0	46.7	42.0	25.9	28.3	41.0
MAmmoTH (MathInstruct)	53.6	31.5	44.5	61.2	67.7	46.3	41.2	42.7	42.6	47.9

Table 9: Breakdown results of Figure 2. Investigation of the influence of CoT & PoT hybrid training on the 7B Llama-2 model.

Model	Decoding	GSM	MATH	AQuA	NumG	SVA	Mat	Sim	SAT	MMLU	AVG
MAmmoTH-7B	CoT	50.5	10.4	43.7	44.0	47.3	9.2	18.9	32.7	39.9	33.0
	PoT	51.6	28.7	43.3	52.3	65.1	41.9	48.2	39.1	44.6	46.1
	Hybrid	**53.6**	**31.5**	**44.5**	**61.2**	**67.7**	**46.3**	**41.2**	**42.7**	**42.6**	**47.9**
MAmmoTH-Coder-7B	CoT	22.4	7.9	36.2	36.0	37.0	8.2	7.2	32.7	34.6	24.7
	PoT	58.8	32.1	47.2	57.1	71.1	53.9	44.6	40.0	47.8	50.3
	Hybrid	**59.4**	**33.4**	**47.2**	**66.4**	**71.4**	**55.4**	**45.9**	**40.5**	**48.3**	**52.0**
MAmmoTH-13B	CoT	56.3	12.9	45.3	45.6	53.8	11.7	22.4	43.6	42.3	37.1
	PoT	61.3	32.6	48.8	59.6	72.2	48.5	40.3	46.8	45.4	50.6
	Hybrid	**62.0**	**34.2**	**51.6**	**68.7**	**72.4**	**49.2**	**43.2**	**46.8**	**47.6**	**52.9**
MAmmoTH-Coder-13B	CoT	32.1	10.2	40.6	36.2	43.0	9.6	10.1	40.9	36.6	28.8
	PoT	64.3	35.2	46.8	54.2	73.2	60.0	44.2	48.2	48.2	52.7
	Hybrid	**64.7**	**36.3**	**46.9**	**66.8**	**73.7**	**61.5**	**47.1**	**48.6**	**48.3**	**54.9**
MAmmoTH-Coder-34B	CoT	34.3	11.6	39.0	36.2	44.6	10.8	10.9	46.4	42.9	30.7
	PoT	72.3	42.8	53.8	59.6	84.0	64.7	50.6	58.6	52.7	59.9
	Hybrid	**72.7**	**43.6**	**54.7**	**71.6**	**84.3**	**65.4**	**51.8**	**60.9**	**53.8**	**62.1**
MAmmoTH-70B	CoT	72.4	21.1	57.9	58.9	71.6	20.0	31.9	57.3	52.1	49.2
	PoT	76.7	40.1	60.2	64.3	81.7	55.3	45.3	64.1	53.5	60.1
	Hybrid	**76.9**	**41.8**	**65.0**	**74.4**	**82.4**	**55.6**	**51.4**	**66.4**	**56.7**	**63.4**

Table 10: Influence of different decoding methods on each dataset.

E   LIMITATIONS

Despite their training on a diverse set of mathematical rationale datasets, the `MAmmoTH` models might exhibit limitations when faced with problems outside their primary domain of expertise like mathematical analysis, complex analysis, graph theory, numerical analysis, etc. Thus, our models are not suitable for solving more complex problems in these fields. Also, they have not been trained with proof-type problems, thus their theorem-proving capability is also limited. In the future, we would like to expand the models' skill set to cover more fields and theorem-proving problems.

There is also a risk of the `MAmmoTH` models generating potentially harmful, offensive, or biased content, especially if they are asked to answer questions beyond math. The `MAmmoTH` series could be misused for malicious purposes, such as spreading misinformation or probing sensitive topics. Developers should conduct safety testing and tuning tailored to their specific applications before deploying any `MAmmoTH` model. While we have made every effort to ensure the cleanliness and purity of our training data, we cannot guarantee absolute perfection. It is unlikely but not impossible that some inappropriate questions slipped through the curation process.

Future work may continue to explore how to further improve the robustness and generalizability of `MAmmoTH` in mathematical reasoning. For example, recent work identifies "sycophancy" and "Clever Hans effect" in reasoning: LLMs cannot maintain truthful solutions to reasoning tasks when challenged by the user's absurdly invalid arguments and critiques (Wang et al., 2023b). Potential methods to improve the models' reasoning robustness could involve the exploration of synthetic data intervention methods as explored in (Wei et al., 2023).