
Not All Votes Count! Translated Program for Verification Improves Self-Consistency of Language Models for Math Reasoning

Vernon Y.H. Toh¹ Deepanway Ghosal¹ Soujanya Poria¹

Abstract

Large language models (LLMs) have become increasingly capable of solving mathematical reasoning problems. However, many open-source LLMs still encounter issues with calculation errors and semantic misunderstandings during intermediate reasoning steps. In this work, we present PROVE, a simple yet effective framework that leverages translated Python programs derived from natural language solutions as a verification mechanism. This verification mechanism helps identify and filter out potentially incorrect paths before final answers are aggregated. Unlike basic majority voting, our approach rejects solutions whose program outputs do not align with the generated solution, only aggregating those that pass the verification step. We conducted extensive experiments with 13 open-source LLMs of various model sizes, ranging from 0.5B to 13B parameters, across eight mathematical benchmarks. Our findings demonstrate that PROVE consistently outperforms basic majority voting as a heuristic and other program-assisted reasoning baselines for solving mathematical reasoning tasks, achieving improvements of up to 18% on GSM8K and 8% on MATH-500. Our codes are available at <https://github.com/declare-lab/prove>.

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable success in mathematical reasoning tasks, especially with advanced models like GPT-4 (OpenAI et al., 2024). However, smaller open-source LLMs, such as Mistral, Gemma 2, and Llama 2 (Jiang et al., 2023; Team et al., 2024; Touvron et al., 2023) fall short in mathematical rea-

soning, often producing inaccurate content and failing to handle complex calculations. Many works since then have aimed to tackle those challenges by using different prompting methods such as Plan-and-Solve (PS) Prompting (Wang et al., 2023a) which aims to mitigate calculation errors and improve the overall quality of the generated reasoning by adopting a two-step process. Alternatively, non-prompting approaches, such as fine-tuning LLMs specifically for reasoning tasks, have also shown notable improvements. However, this often demands large amounts of distilled data from models like GPT-4 (Wang et al., 2024; Gou et al., 2024), resulting in significant computational and data generation costs.

Despite these advancements, fundamental challenges remain (Banerjee et al., 2024). Autoregressive models lack mechanisms to correct their own errors, meaning that once a mistake is made, it can propagate through the entire response. As a result, LLMs are still vulnerable to hallucinations which can lead to incorrect reasoning or calculation. Given these limitations, a promising direction is to enhance reasoning by leveraging the knowledge already embedded within LLMs themselves (Wang et al., 2023b; Hao et al., 2023; Madaan et al., 2023; Weng et al., 2023; Wang et al., 2023a). Following Wang et al. (2023b), one can sample multiple reasoning paths to solve a question and select the most common answer. While aggregating answers like this improves performance, sometimes the most common answer could still be wrong.

Recent work has explored alternative verification methods to enhance mathematical reasoning. One promising direction is verification through formalization (Lin et al., 2024), which translates problems and solutions into formal representations for verification. However, this approach is often challenging, requiring domain-specific formalization and few-shot prompting with more powerful models like GPT-3.5. A simpler yet effective alternative is to leverage LLMs' code generation capabilities for verification. Executing generated Python programs allows for automatic verification of intermediate calculations without requiring explicit formalization.

In this paper, we introduce PROVE, a *frustratingly simple* approach that uses *translated **PRO**grams for **VER**ification*. Our

¹Singapore University of Technology and Design. Correspondence to: Vernon Y.H. Toh <toh.vernon@gmail.com>.

The second AI for MATH Workshop at the 42nd International Conference on Machine Learning, Vancouver, Canada. Copyright 2025 by the author(s).

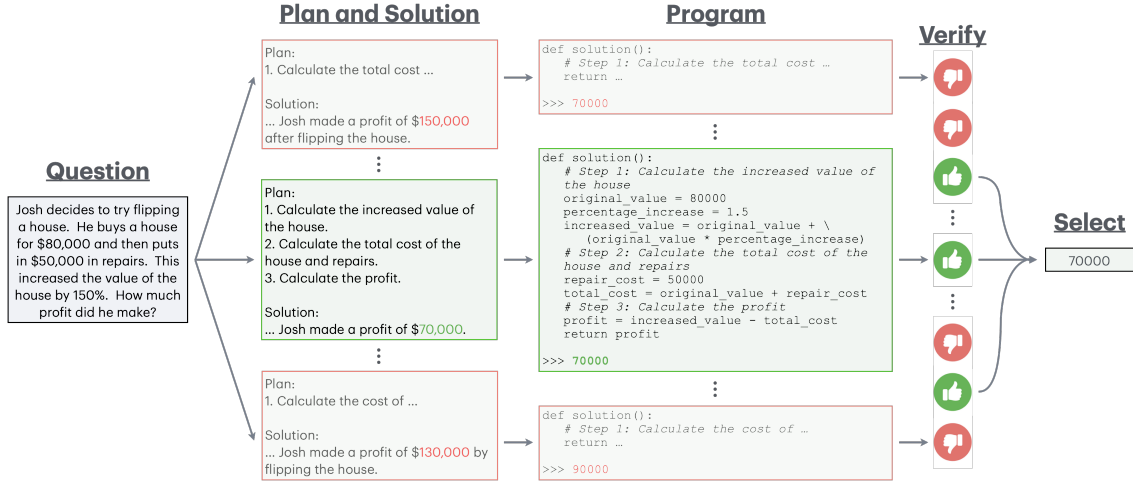


Figure 1: The PROVE Framework.

framework is model-agnostic, requiring no fine-tuning or few-shot prompting of LLMs. Instead, it leverages the capabilities of open-source LLMs to translate natural language plans and solutions into Python programs in a zero-shot setting for verification. Unlike methods that aggregate all generated natural language solutions through simple majority voting, we only consider solutions that are successfully verified by their corresponding Python programs. While natural language plans and solutions provide a high-level outline of the steps needed to solve a math word problem, verifying their correctness can be difficult. Python program implementations, however, offer a more precise representation. By using a program interpreter for computation, they can effectively detect errors in natural language solutions, such as miscalculations.

We evaluate PROVE using 13 LLMs from various model families and sizes (Abdin et al., 2024; Touvron et al., 2023; Dubey et al., 2024; Jiang et al., 2023; Team et al., 2024; Yang et al., 2024), with parameter sizes ranging from 0.5B to 13B, across eight mathematical reasoning datasets (Cobbe et al., 2021; Patel et al., 2021; Miao et al., 2020; Roy & Roth, 2015; Koncel-Kedziorski et al., 2015; Roy et al., 2015; Hosseini et al., 2014; Lightman et al., 2024). The results show that PROVE consistently outperforms standard majority voting (Wang et al., 2023b) across all model sizes and datasets, achieving up to an 18% improvement on GSM8K and 8% on MATH-500.

Contributions. We summarize the contributions of our paper as follows:

- PROVE, a model-agnostic framework enhancing mathematical reasoning by translating NL solutions to programs for verification before aggregation.
- Extensive evaluation on 13 open-source LLMs (0.5B-13B) across eight benchmarks, demonstrating PROVE

consistently outperforms strong baselines, including self-consistency with both natural language and program-based reasoning paths, as well as the natural language verifier technique.

- Detailed analysis showing PROVE significantly reduces calculation errors with more substantial gains for smaller LLMs.
- Ablation studies on the impact of translation models, prompting strategies, and an analysis of the program translation process.

2. PROVE

Overview. As illustrated in Figure 1, the PROVE framework begins by prompting a large language model (LLM) to generate plans and solutions for a given math word problem. Next, the generated plans and solutions are translated into a Python program using an LLM. The resulting program is then executed, and its output is compared against the initially generated solution. If the outputs match, the solution is considered valid; otherwise, it is discarded. Finally, the remaining valid solutions undergo a majority voting process to determine the final answer. Unlike traditional methods requiring fine-tuning or few-shot examples, PROVE operates entirely through zero-shot prompting with open-source LLMs, making it highly flexible and efficient.

Generating Plans and Solutions. The first step in PROVE involves prompting an LLM to generate a natural language plan and solution for solving a given math word problem. Given a problem x , the LLM is prompted using the zero-shot prompting method proposed by Wang et al. (2023a): "Let's first understand the problem and devise a plan to solve it. Then, let's carry out the plan and solve the problem step by step.". We sample multiple responses from the LLM,

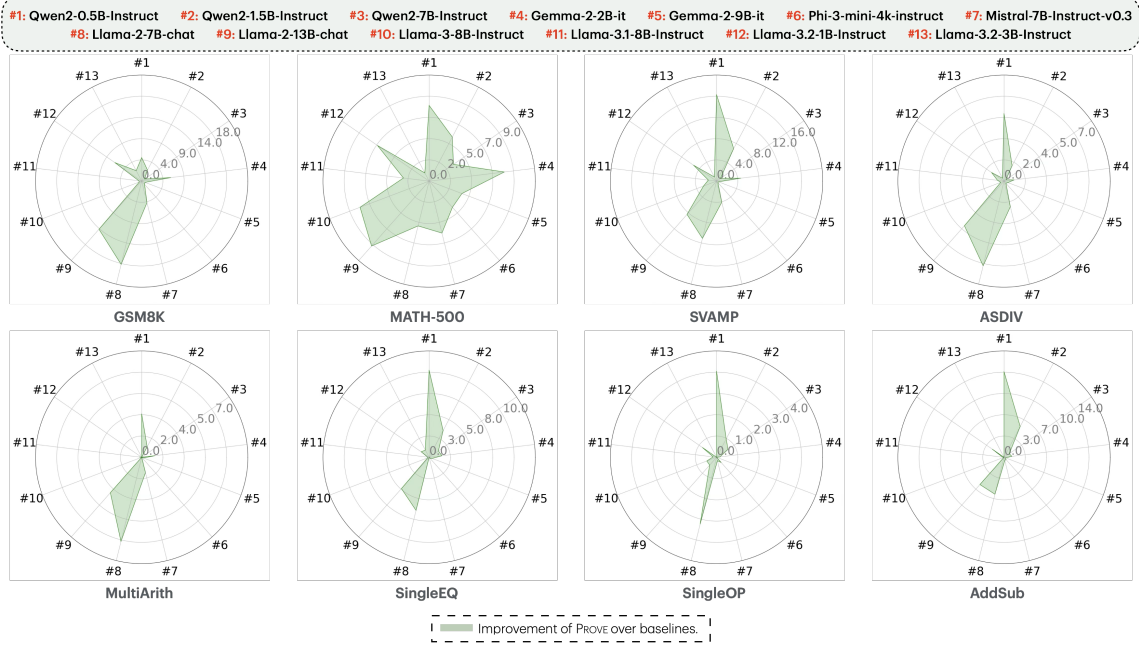


Figure 2: Performance improvement of PROVE over baseline methods across 13 LLMs evaluated on eight mathematical reasoning benchmarks.

producing a set of candidate plans and solutions, denoted as $\{s_1, \dots, s_n\}$.

Translating Plans and Solutions into Python Programs.

Once the candidate plans and solutions $\{s_1, \dots, s_n\}$ are obtained, each solution s_i is individually fed into another LLM to generate a corresponding Python program p_i . Specifically, the LLM is instructed to generate a corresponding Python program based on the plan in s_i to solve the given problem. This process results in a set of pairs $\{(s_1, p_1), \dots, (s_n, p_n)\}$, which are then used for verification.

Verification and Selection. To verify the correctness of each solution, we execute the Python program p_i and obtain its output $\hat{p}_i$. We then extract the corresponding answer $\hat{s}_i$ from the generated solution s_i using the answer extraction method proposed by Kojima et al. (2022): “Therefore, the answer (Arabic numerals) is...”. A solution s_i is considered valid if its extracted answer $\hat{s}_i$ matches the program output $\hat{p}_i$. After filtering out invalid solutions, majority voting is performed on the remaining valid answers to determine the final answer. If no candidate solutions are validated, majority voting is performed over all originally generated (invalidated) answers. This fallback ensures an answer is always produced.

3. Experiments

3.1. Setup

Models and datasets. PROVE is a model-agnostic framework designed to integrate with any LLM for mathematical reasoning tasks. We evaluated 13 LLMs from various families and sizes, ranging from 0.5B to 13B parameters. Specifically, we evaluated the following models: Phi-3-mini-4k-instruct (3.8B) (Abdin et al., 2024), Llama-2-7B-chat (Touvron et al., 2023), Llama-2-13B-chat, Llama-3-8B-Instruct (Dubey et al., 2024), Llama-3.1-8B-Instruct, Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), Gemma-2-2B-it (Team et al., 2024), Gemma-2-9B-it, Qwen2-0.5B-Instruct (Yang et al., 2024), Qwen2-1.5B-Instruct, and Qwen2-7B-Instruct. Our evaluations covers eight commonly used mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), ASDIV (Miao et al., 2020), MultiArith (Roy & Roth, 2015), SingleEQ (Koncel-Kedziorski et al., 2015), SingleOP (Roy et al., 2015), and AddSub (Hosseini et al., 2014). To speed up the evaluation process, we used MATH-500, a subset of problems from the MATH dataset, as suggested by Lightman et al. (2024).

Implementation details. In PROVE, we sample a total of 16 plans and solutions using a temperature setting of 0.7. To extract the answer from the generated solutions, we rely on Phi-3-mini-4k-instruct. We also standardize the



Figure 3: An actual example of the prompts used in our framework, along with the outputs generated by Phi3-mini-4k-instruct, for the GSM8K task.

use of Phi-3-mini-4k-instruct to translate the plan and solution into Python programs for verification. With only 3.8 billion parameters, Phi-3-mini-4k-instruct offers more efficient inference. Notably, when Phi-3-mini-4k-instruct is being evaluated, it performs self-verification by generating not only the plan and solution but also translating the plan and solution to a Python program to verify the results. During the selection phase, we perform majority voting on the remaining valid answer. If no valid answer remain, we perform majority voting over all invalid answers. For the more challenging MATH dataset, we use GPT-4o (2024-02-15-preview) to translate the plan and solution into Python code. Additional details on the MATH dataset setup can be found in Appendix B.

Baselines. We compare PROVE with two decoding strategies: single sampling (greedy decoding) and multiple sampling (self-consistency decoding) (Wang et al., 2023b). The comparison is conducted using three different prompting techniques: zero-shot Chain-of-Thought (CoT) (Kojima et al., 2022), zero-shot Program-of-Thought (PoT) (Chen et al., 2023), and zero-shot Plan-and-Solve (PS) (Wang et al., 2023a). Due to the similarity of Program-Aided Language Models (PAL) and PoT, we have just selected 1 of the 2 in our baselines, in this case we selected PoT. Performance is evaluated based on accuracy, where a problem is considered correct only if the final answer exactly matches the ground truth.

Not All Votes Count! Translated Program for Verification Improves Self-Consistency of Language Models for Math Reasoning

Method	Qwen-2			Gemma-2		Phi-3	Mistral	Llama-2		Llama-3	Llama-3.1	Llama-3.2	
	0.5B	1.5B	7B	2B	9B	3.8B	7B	7B	13B	8B	8B	1B	3B
GSM8K													
Zero-shot CoT	32.22	57.71	88.25	67.63	89.69	86.35	58.83	29.80	41.24	80.74	87.49	50.72	81.88
Zero-shot PoT	5.23	2.81	63.61	50.80	80.89	84.38	48.52	8.49	26.99	73.46	84.38	42.15	33.06
Zero-shot PS	25.63	51.02	85.90	63.00	89.01	86.50	58.07	28.35	39.80	80.36	86.66	49.05	81.12
Zero-shot CoT @maj16	48.85	71.57	91.66	72.94	91.05	91.96	75.66	38.97	54.44	87.72	92.49	66.11	87.87
Zero-shot PoT @maj16	28.43	37.00	85.67	59.21	86.43	91.36	70.96	27.22	40.86	84.07	90.37	58.45	83.32
Zero-shot PS @maj16	42.00	68.46	90.98	73.39	91.96	92.19	78.54	41.32	54.21	89.61	92.57	65.66	88.86
Zero-shot CoT @maj32	48.52	73.16	92.19	70.51	90.22	92.49	75.89	40.71	54.66	88.02	91.43	55.72	85.60
Zero-shot PoT @maj32	35.10	53.68	85.52	54.74	86.28	92.27	73.01	35.48	40.03	84.91	89.69	50.04	81.43
Zero-shot PS @maj32	44.73	70.36	91.96	71.72	91.58	92.72	79.83	41.55	57.77	88.55	92.57	59.14	87.04
PROVE	53.83 (+4.98)	74.22 (+1.06)	92.42 (+0.23)	79.61 (+6.22)	92.72 (+0.76)	93.10 (+0.38)	83.24 (+3.41)	59.51 (+17.96)	68.08 (+10.31)	90.14 (+0.53)	93.19 (+0.62)	73.01 (+6.90)	91.36 (+2.50)
MATH-500													
Zero-shot CoT	7.20	18.60	45.20	19.60	46.40	41.60	9.60	2.00	4.40	26.00	46.60	13.40	36.80
Zero-shot PS	7.00	13.40	37.40	20.80	42.60	42.40	10.80	2.20	4.00	24.60	47.60	16.60	35.60
Zero-shot CoT @maj16	11.60	29.40	64.00	29.20	53.20	54.00	18.60	3.20	8.00	34.80	62.40	34.60	54.80
Zero-shot PS @maj16	11.60	28.20	63.40	29.00	54.80	51.40	21.20	3.80	6.20	36.80	62.80	38.00	56.40
Zero-shot CoT @maj32	12.40	30.00	65.60	30.00	54.60	55.80	19.80	3.60	8.60	36.00	63.20	35.60	56.00
Zero-shot PS @maj32	12.20	29.00	64.40	30.00	55.60	53.20	22.00	4.20	7.20	37.60	63.40	39.00	56.80
PROVE	19.80 (+7.40)	34.80 (+4.80)	67.20 (+1.60)	37.40 (+7.40)	58.60 (+3.00)	57.80 (+2.00)	27.00 (+5.00)	8.80 (+4.60)	17.40 (+8.80)	44.80 (+7.20)	65.60 (+2.20)	44.80 (+5.80)	57.40 (+0.60)
SVAMP													
Zero-shot CoT	41.00	62.10	89.90	74.40	88.70	89.10	71.80	56.40	62.80	85.00	86.80	68.40	86.50
Zero-shot PoT	13.60	5.40	47.00	73.40	91.80	90.70	52.80	25.50	44.10	86.70	88.00	62.60	66.30
Zero-shot PS	35.60	61.00	90.70	73.60	89.80	89.80	73.20	56.00	57.30	83.40	88.30	65.60	87.50
Zero-shot CoT @maj16	57.10	78.30	93.60	80.00	91.30	94.00	84.40	69.70	77.80	90.50	92.30	82.30	91.90
Zero-shot PoT @maj16	47.80	39.20	74.50	79.30	92.50	94.70	80.50	59.70	66.00	90.60	91.50	76.80	92.00
Zero-shot PS @maj16	56.60	79.60	93.70	81.70	91.40	93.60	84.70	71.30	77.80	91.20	92.90	81.50	92.80
Zero-shot CoT @maj32	59.40	79.50	93.60	76.30	90.80	93.90	83.70	66.90	76.10	89.00	91.60	77.00	90.30
Zero-shot PoT @maj32	52.90	55.20	72.50	76.40	92.30	94.40	81.50	67.10	66.70	90.50	91.60	73.10	90.90
Zero-shot PS @maj32	60.90	82.40	94.20	80.30	91.00	94.90	84.30	71.50	79.00	90.40	92.60	77.60	91.00
PROVE	73.80 (+12.90)	86.70 (+4.30)	94.60 (+0.40)	86.40 (+4.70)	92.90 (+0.40)	95.10 (+0.20)	88.90 (+4.20)	82.70 (+11.20)	86.40 (+7.40)	93.90 (+2.70)	94.50 (+1.60)	87.70 (+5.40)	93.70 (+0.90)
ASDIV													
Zero-shot CoT	55.10	73.14	91.89	80.68	91.60	91.84	74.52	58.59	65.03	85.35	89.12	75.91	89.22
Zero-shot PoT	18.46	6.35	48.09	71.66	82.20	83.73	54.20	32.40	48.71	78.86	86.74	67.18	59.49
Zero-shot PS	45.99	70.04	91.32	81.82	91.51	91.98	74.62	57.63	62.12	84.40	88.98	72.47	87.74
Zero-shot CoT @maj16	69.18	82.68	93.56	85.35	92.56	94.51	83.92	66.46	74.86	89.46	92.27	84.45	92.50
Zero-shot PoT @maj16	57.49	59.02	77.24	78.24	86.26	92.89	78.77	63.36	66.56	86.74	91.89	79.53	91.27
Zero-shot PS @maj16	68.03	83.35	93.46	87.02	92.99	94.18	84.78	69.27	75.43	89.29	92.22	85.07	92.75
Zero-shot CoT @maj32	70.56	83.78	93.37	85.11	92.80	94.56	84.64	68.13	75.00	89.87	92.63	82.68	92.27
Zero-shot PoT @maj32	63.07	66.32	75.76	77.62	87.21	93.75	79.77	65.74	68.46	88.31	92.03	76.57	90.41
Zero-shot PS @maj32	69.99	83.30	93.51	86.74	93.23	94.45	85.73	70.47	76.86	89.49	92.75	83.25	92.37
PROVE	74.90 (+4.34)	84.83 (+1.05)	93.76 (+0.20)	87.88 (+0.86)	93.35 (+0.12)	94.75 (+0.19)	87.02 (+1.29)	76.67 (+6.20)	80.49 (+3.63)	89.98 (+0.11)	92.83 (+0.08)	86.35 (+1.28)	93.03 (+0.28)

Table 1: Main results showing the comparison of PROVE and baseline methods across 13 LLMs, ranging in size from 0.5B to 13B, on GSM8K, MATH-500, SVAMP, and ASDIV. Additional results are provided in Appendix A due to space constraints.

3.2. Main Results

We report the main results of our experiments in Table 1. We highlight several key observations:

PROVE consistently outperforms baselines across all evaluated LLMs and datasets. Our evaluation of PROVE across various model families and sizes shows consistent performance improvements over strong baseline methods. For instance, on the GSM8K dataset, Mistral-7B-Instruct-v0.3 achieves an accuracy of 83.24% with PROVE, showing a 3.41% improvement over the 79.83% accuracy of the Zero-shot PS @maj32 baseline. Similarly, on the SVAMP dataset, Mistral-7B-Instruct-v0.3 reaches an accuracy of 88.90%, surpassing the best baseline by 4.2%. Furthermore, Mistral-7B-Instruct-v0.3 achieves performance gains of up to 1.29% across other datasets, including ASDIV, MultiArith, SingleEq, SingleOp, and AddSub. These improvements are

consistently observed across all 13 LLMs we tested.

PROVE provides larger performance gains for smaller models. In our comparison of various model families, including Qwen-2, Gemma-2, Llama-2, and Llama-3.2, we found that smaller models typically show more significant performance improvements compared to their larger counterparts. For example, on the GSM8K dataset, the smaller Llama-2-7B-chat model achieved an 17.96% improvement over the baseline, while the larger Llama-2-13B-chat model showed a 10.31% gain. Similarly, the Gemma-2-2B-it model improved by 6.22%, whereas the larger Gemma-2-9B-it model had a 0.76% increase. A similar pattern is shown for Qwen-2 models where Qwen2-0.5B-Instruct outperformed its baseline by 4.98%, Qwen2-1.5B-Instruct improved by 1.06%, and the larger Qwen2-7B-Instruct saw just a 0.23% gain. Similarly, Llama-3.2-1B-Instruct demon-

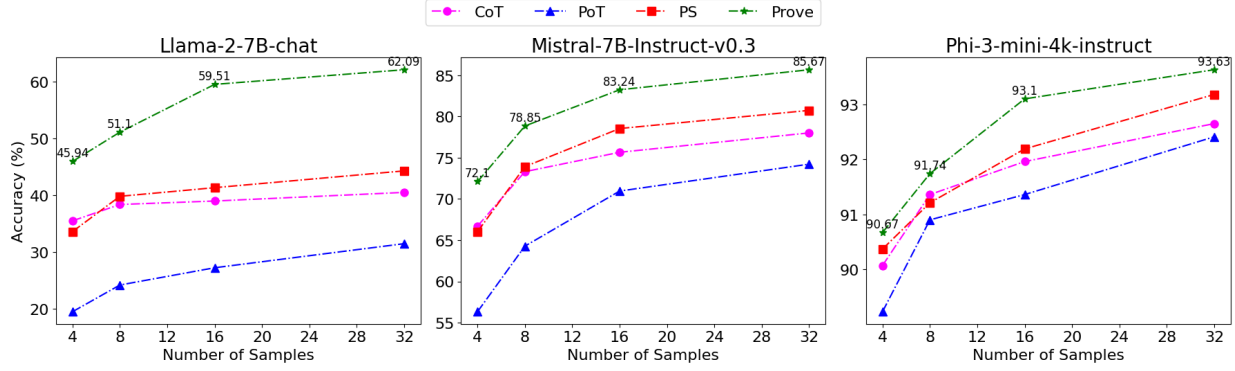


Figure 4: Performance comparison of PROVE and baseline methods across various LLMs evaluated on GSM8K with different numbers of samples.

strated a 6.9% improvement, while the larger Llama-3.2-3B-Instruct achieved only a 2.5% increase. These findings suggest that smaller models benefit more from PROVE, likely because they are more prone to errors, such as miscalculations, making filtering mechanisms particularly effective for them.

Improvement on challenging mathematical dataset.

We also assess the performance of PROVE on the MATH-500 dataset. Our results demonstrate that PROVE consistently improves performance across all models, with the largest improvement of 8.8% observed for Llama-2-13B-chat. Even for smaller models, such as Qwen2-0.5B-Instruct and Llama-3.2-1B-Instruct, we observe notable gains of 7.4% and 5.8%, respectively. These results suggest that even on more challenging datasets, models can generate the correct answer by considering multiple reasoning paths. However, when the correct answer is not the most frequent, leveraging better heuristics to aggregate reasoning paths can lead to significant performance improvements.

3.3. Ablation Study

Impact of the number of plans. In Figure 4, we compare the accuracy of CoT, PoT, and PS prompting methods with self-consistency and PROVE across different numbers of sampled solutions on GSM8K. As the number of sampled solutions increases from 4 to 32, PROVE consistently outperforms the baseline methods, highlighting its effectiveness. Notably, with just 4 samples, the Llama-2-7b-chat and Mistral-7B-Instruct-v0.3 models show notable accuracy improvements of 10.46% and 5.39%, respectively. With 32 samples, the Llama-2-7b-chat and Mistral-7B-Instruct-v0.3 models achieve gains of 17.81% and 4.93%, respectively.

Impact of translation model. We examine the impact of model selection for translating plans and solutions into Python programs on GSM8K. In our main experiments, the

Model	Translation Model	Accuracy (%)
Llama-3.2-1B-Instruct	Phi-3-mini-4k-instruct	73.01
	Llama-3-8B-Instruct	73.54
	Llama-3.1-8B-Instruct	74.53
Llama-2-7B-chat	Phi-3-mini-4k-instruct	59.51
	Llama-3-8B-Instruct	58.15
	Llama-3.1-8B-Instruct	61.94
Mistral-7B-Instruct-v0.3	Phi-3-mini-4k-instruct	83.24
	Llama-3-8B-Instruct	80.29
	Llama-3.1-8B-Instruct	83.32

Table 2: Performance of PROVE on GSM8K, evaluated across various translation models and LLMs.

default translation model is Phi-3-mini-4k-instruct. Additionally, we evaluate Llama-3-8B-Instruct and Llama-3.1-8B-Instruct as alternative translation models. Table 2 illustrates the performance differences across these translation models. Notably, Llama-3.1-8B-Instruct, which has more parameters than Phi-3-mini-4k-instruct, results in slight performance improvements across all tested LLMs. Specifically, it achieves performance gains of 1.52%, 2.43%, and 0.08% for Llama-3.2-1B-Instruct, Llama-2-7B-chat, and Mistral-7B-Instruct-v0.3, respectively.

Model	Prompting Method	Accuracy (%)
Llama-3.2-1B-Instruct	CoT	72.89
	PS	73.01
Llama-2-7B-chat	CoT	59.29
	PS	59.51
Mistral-7B-Instruct-v0.3	CoT	82.79
	PS	83.24

Table 3: Performance of PROVE on GSM8K, with different prompting strategies across various LLMs.

Impact of Plan and Solve (PS) prompting. We investigated the impact of PS prompting on our framework by comparing its effectiveness against CoT prompting in the

first stage of PROVE. We evaluated three different models, namely Llama-3.2-1B-Instruct, Llama-2-7B-Chat, and Mistral-7B-Instruct-v0.3, on GSM8K. As shown in Table 3, the performance differences between the two prompting methods were minimal, with PS outperforming CoT by 0.12% to 0.45% across the models tested. These results highlight the robustness of PROVE, demonstrating its ability to maintain strong performance regardless of the prompting method used.

Model	Verification Method	Accuracy (%)
Llama-3.2-1B-Instruct	Natural Language	53.45
	Program	73.01
Llama-2-7B-chat	Natural Language	37.00
	Program	59.51
Mistral-7B-Instruct-v0.3	Natural Language	68.23
	Program	83.23

Table 4: Performance of PROVE on GSM8K, evaluated across various verification methods.

PROVE vs Natural Language Verifier. We conduct an ablation study comparing our proposed method PROVE with a baseline that directly prompts an LLM to identify errors in generated solutions using natural language. For a fair comparison, we use the same model (Phi-3-mini-4k-instruct) as in PROVE, prompting it with the question: “Are there any errors in this solution (Yes/No)?” to serve as the natural language verification method. Table 4 presents accuracy results on GSM8K across three different models. In every case, PROVE outperforms natural language verification. For instance, Llama-3.2-1B-Instruct improves from 53.45% to 73.01%, and Mistral-7B-Instruct-v0.3 achieves 83.23% accuracy with over 15 percentage points higher than the natural language verification. Even models like Llama-2-7B-chat see substantial gains, with an improvement of 22.5 points. These findings underscore the limitations of natural language for precise verification and demonstrate the robustness of translating reasoning into executable programs for reliable, deterministic checking.

3.4. Analyses

PROVE gets correct when the vanilla majority vote fails. In Figure 5, we present two case study examples where PROVE successfully filters out incorrect PS solutions. In these instances, majority voting fails because the correct answer is not the most common response. In contrast, PROVE effectively identifies and removes incorrect reasoning paths, ensuring that the correct answer becomes the majority. In Figure 5 (left), the model incorrectly calculates the sum as “\$4.20 + \$9.45 + \$1.35 = \$15.10” when the correct sum should be “\$15.00.” Similarly, in Figure 5 (right), the model makes a semantic error by miscounting the total number of zebras as 3 instead of 4. However, with our PROVE

framework, the translated program detects these errors and effectively filters them out. These examples demonstrate how leveraging translated programs for verification allows us to efficiently filter out incorrect reasoning paths.

Method	Calculation	Missing	Semantic
Zero-shot PS @maj16	20%	12%	68%
PROVE	0%	18%	82%

Table 5: Distribution of error types (calculation error, missing step error, semantic understanding error) for 100 randomly sampled examples from GSM8K, where both PS and PROVE got incorrect.

Error analysis. We manually analyzed 100 randomly selected examples from GSM8K where both PS and PROVE provided incorrect answers, using Llama-3.2-1B-Instruct. Following Wang et al. (2023a), we classified the errors into three categories: calculation errors, missing step errors, and semantic understanding errors. The analysis results (Table 5) show that PROVE achieved a 0% calculation error rate, significantly outperforming PS, which had a 20% calculation error rate. This highlights PROVE’s effectiveness in minimizing calculation errors, as illustrated in the case study example in Figure 5 (left).

	Actual Positive: Correct PS Solution	Actual Negative Wrong PS Solution
Predicted Positive: Program Match	4865 (97.3%)	1695 (33.9%)
Predicted Negative: Program Mismatch	135 (2.7%)	3305 (66.1%)

Table 6: Confusion Matrix for program translation. We randomly selected 5,000 correct and incorrect PS solutions and translated them into Python program.

Analysis on Python program translation. To evaluate the program translation error rate, we randomly selected 5,000 correct and incorrect PS solutions and translated them into Python programs using the default translation model from our main experiments. The performance of the program translation is presented in Table 6. The results shows that the translation model achieves a high true positive rate and a low false negative rate, demonstrating its effectiveness in translating PS solutions into Python programs. Although the false positive and true negative rates are moderate, suggesting that some incorrect solutions are not filtered out, the majority of incorrect PS solutions are successfully filtered. These findings provide strong evidence of the effectiveness of PROVE in math reasoning.

4. Related Works

Recent advancements in Large Language Models (LLMs) have demonstrated remarkable capabilities in complex rea-

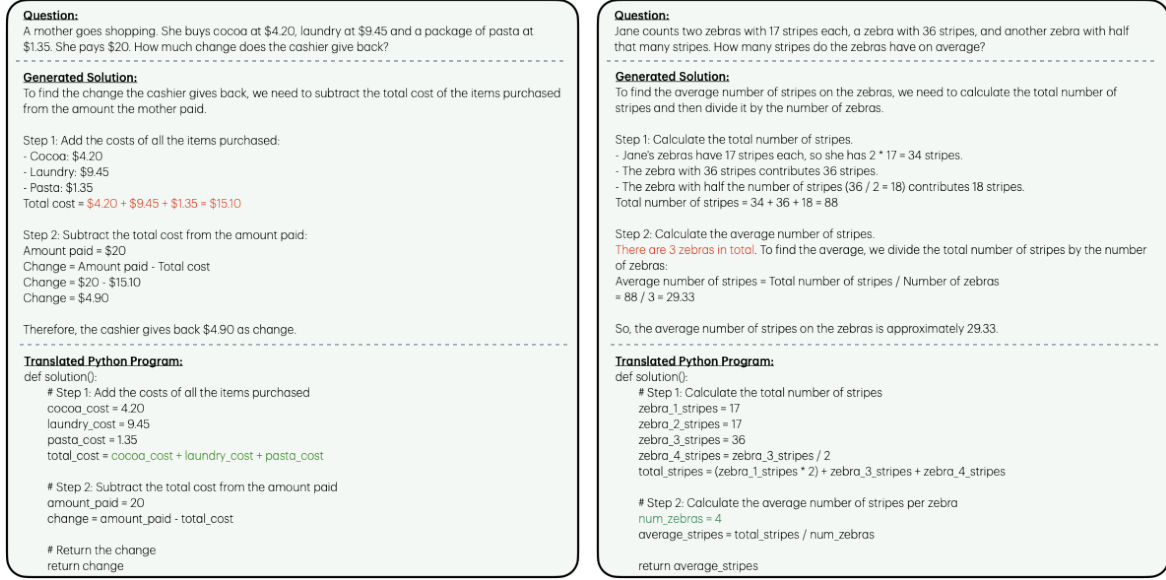


Figure 5: Case study examples where PROVE gets correct when vanilla majority voting fails on GSM8K type question. We show how the translated program can correctly filter out generated solutions that have errors in them. Left: highlighted in red shows a calculation error. Right: highlighted in red shows semantic understanding error.

soning tasks. Prior research shows that step-by-step reasoning prompts like Chain-of-Thought (CoT) (Kojima et al., 2022) and Plan-and-Solve (PS) (Wang et al., 2023a), significantly improve performance over direct answer generation. Concurrently, methods integrating Python programs directly into the reasoning process, such as Program-Aided Language Models (PAL) (Gao et al., 2023) and Program of Thoughts (PoT) (Chen et al., 2023), leverage code execution for precision. Techniques like multi-step decoding (e.g., Tree of Thoughts (Yao et al., 2023)) and multi-sample reasoning with self-consistency (Wang et al., 2023b) further enhance robustness by exploring diverse reasoning paths and aggregating solutions. However, a common challenge with these generative approaches, especially for smaller models, is their susceptibility to hallucination and error propagation; if the majority of generated paths are flawed, aggregation methods like self-consistency will still yield incorrect results. Our method, PROVE, complements these efforts by introducing a crucial verification step by translating generated natural language solutions into Python programs to filter out flawed reasoning paths before aggregation, aiming for more accurate final answers.

Using LLMs to evaluate model generations for verification is a well-established concept (Kushman et al., 2014; Roy & Roth, 2015; Shen et al., 2021). A common strategy involves training a separate verifier model to assess the correctness of generated outputs (Cobbe et al., 2021). Other techniques, such as step-by-step verification (Lightman et al., 2024) and ranking multiple reasoning paths to

choose the most accurate one (Weng et al., 2023), show promise but often depend heavily on the inherent reasoning abilities of the primary LLM or a similarly capable verifier model, which can limit their effectiveness, particularly for smaller models. Program-based verification approaches have also been proposed (Zhou et al., 2024a; Han et al., 2024; Zhou et al., 2024b). Other approaches focus on improving the faithfulness of the reasoning steps themselves. Lyu et al. (2023) (Faithful CoT) focused on aligning intermediate natural language steps in CoT with symbolic solver outputs to ensure step-wise correctness. Similarly, CoMAT (Leang et al., 2024) enhances mathematical reasoning by generating mathematically annotated thoughts, essentially creating a more structured and verifiable reasoning trace. While these methods demonstrate the value of programmatic checks or formalized reasoning, they can be complex, often requiring few-shot prompting or more capable base models, or focus on detailed step-wise annotation. In contrast, PROVE is designed to be model-agnostic and adaptable to smaller, open-source LLMs without needing few-shot exemplars for its verification pipeline. It focuses on leveraging a translation model to verify diverse, complete natural language solutions via program execution, differentiating it by its specific mechanism of post-hoc translation and filtering of entire reasoning paths within a self-consistency framework.

5. Conclusion

In this paper, we demonstrate that using translated programs for verification can effectively filter out low-quality reasoning paths, which may include calculation or semantic understanding errors. Our approach, PROVE, is model-agnostic and does not require fine-tuning or few-shot exemplars for prompting. PROVE consistently outperforms baseline methods across 13 LLMs and eight mathematical reasoning datasets.

References

- Abdin, M., Jacobs, S. A., Awan, A. A., Aneja, J., Awadallah, A., Awadalla, H., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., Benhaim, A., Bilenko, M., Bjorck, J., Bubeck, S., Cai, Q., Cai, M., Mendes, C. C. T., Chen, W., Chaudhary, V., Chen, D., Chen, D., Chen, Y.-C., Chen, Y.-L., Chopra, P., Dai, X., Giorno, A. D., de Rosa, G., Dixon, M., Eldan, R., Fragoso, V., Iter, D., Gao, M., Gao, M., Gao, J., Garg, A., Goswami, A., Gunasekar, S., Haider, E., Hao, J., Hewett, R. J., Huynh, J., Javaheripi, M., Jin, X., Kauffmann, P., Karampatziakis, N., Kim, D., Khademi, M., Kurilenko, L., Lee, J. R., Lee, Y. T., Li, Y., Li, Y., Liang, C., Liden, L., Liu, C., Liu, M., Liu, W., Lin, E., Lin, Z., Luo, C., Madan, P., Mazzola, M., Mitra, A., Modi, H., Nguyen, A., Norick, B., Patra, B., Perez-Becker, D., Portet, T., Pryzant, R., Qin, H., Radmilac, M., Rosset, C., Roy, S., Ruwase, O., Saarikivi, O., Saied, A., Salim, A., Santacroce, M., Shah, S., Shang, N., Sharma, H., Shukla, S., Song, X., Tanaka, M., Tupini, A., Wang, X., Wang, L., Wang, C., Wang, Y., Ward, R., Wang, G., Witte, P., Wu, H., Wyatt, M., Xiao, B., Xu, C., Xu, J., Xu, W., Yadav, S., Yang, F., Yang, J., Yang, Z., Yang, Y., Yu, D., Yuan, L., Zhang, C., Zhang, C., Zhang, J., Zhang, L. L., Zhang, Y., Zhang, Y., Zhang, Y., and Zhou, X. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL <https://arxiv.org/abs/2404.14219>.
- Banerjee, S., Agarwal, A., and Singla, S. LLMs will always hallucinate, and we need to live with this, 2024. URL <https://arxiv.org/abs/2409.05746>.
- Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Srivankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Al-lonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Syn-naeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang,

- B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzmán, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Damla, I., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Prasad, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhota, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabisa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Kenally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Albiero, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models, 2023. URL <https://arxiv.org/abs/2211.10435>.
- Gou, Z., Shao, Z., Gong, Y., yelong shen, Yang, Y., Huang, M., Duan, N., and Chen, W. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Ep0TtjVoap>.
- Han, V. T. Y., Puduppully, R., and Chen, N. F. Veritymath: Advancing mathematical reasoning by self-verification through unit consistency. In *AI for Math Workshop @ ICML 2024*, 2024. URL <https://openreview.net/forum?id=S9utaRXaZt>.
- Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z. Reasoning with language model is planning with world model. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL <https://aclanthology.org/2023.emnlp-main.507>.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- Hosseini, M. J., Hajishirzi, H., Etzioni, O., and Kushman, N. Learning to solve arithmetic word problems with verb categorization. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 523–533, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1058. URL <https://aclanthology.org/D14-1058>.
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023. URL <https://arxiv.org/abs/2310.06825>.

- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=e2TBb5y0yFf>.
- Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O., and Ang, S. D. 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>.
- Kushman, N., Artzi, Y., Zettlemoyer, L., and Barzilay, R. Learning to automatically solve algebra word problems. In Toutanova, K. and Wu, H. (eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 271–281, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1026. URL <https://aclanthology.org/P14-1026>.
- Leang, J. O. J., Gema, A. P., and Cohen, S. B. Co-mat: Chain of mathematically annotated thought improves mathematical reasoning, 2024. URL <https://arxiv.org/abs/2410.10336>.
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=v8L0pN6EOi>.
- Lin, L., Fu, J., Liu, P., Li, Q., Gong, Y., Wan, J., Zhang, F., Wang, Z., Zhang, D., and Gai, K. Just ask one more time! self-agreement improves reasoning of language models in (almost) all scenarios. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 3829–3852, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.230. URL <https://aclanthology.org/2024.findings-acl.230>.
- Lyu, Q., Havaldar, S., Stein, A., Zhang, L., Rao, D., Wong, E., Apidianaki, M., and Callison-Burch, C. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.
- Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.
- Miao, S.-y., Liang, C.-C., and Su, K.-Y. A diverse corpus for evaluating and developing English math word problem solvers. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.92. URL <https://aclanthology.org/2020.acl-main.92>.
- OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kopic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Měly, D., Nair, A., Nakano, R., Nayak, R., Neelakantan,

- A., Ngo, R., Noh, H., Ouyang, L., O’Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorný, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2024. URL <https://arxiv.org/abs/2303.08774>.
- Patel, A., Bhattamishra, S., and Goyal, N. Are NLP models really able to solve simple math word problems? In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL <https://aclanthology.org/2021.naacl-main.168>.
- Roy, S. and Roth, D. Solving general arithmetic word problems. In Márquez, L., Callison-Burch, C., and Su, J. (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202. URL <https://aclanthology.org/D15-1202>.
- Roy, S., Vieira, T., and Roth, D. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13, 2015. doi: 10.1162/tacl.a.00118. URL <https://aclanthology.org/Q15-1001>.
- Shen, J., Yin, Y., Li, L., Shang, L., Jiang, X., Zhang, M., and Liu, Q. Generate & rank: A multi-task framework for math word problems. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2269–2279, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.195. URL <https://aclanthology.org/2021.findings-emnlp.195>.
- Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., Tsitsulin, A., Vieillard, N., Stanczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J.-B., Neyshabur, B., Bachem, O., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagici, A., Carl, A., Shen, A., Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C., Choquette-Choo, C. A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozińska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Plucińska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J. P., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., van Amersfoort, J., Gordon, J., Lipschultz, J., Newlan, J., yeong Ji, J., Mohamed, K., Badola, K., Black, K., Milligan, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjoesund, L. L., Usui, L., Sifre, L., Heuermann, L., Lago, L., McNealus, L., Soares, L. B., Kilpatrick, L., Dixon, L., Martins, L., Reid, M., Singh, M., Iverson, M., Görner, M., Velloso, M., Wirth, M., Davidow, M., Miller, M., Rahtz, M., Watson, M., Risdal, M., Kazemi, M., Moynihan, M., Zhang, M., Kahng, M., Park, M., Rahman, M., Khatwani, M., Dao, N., Bardoliwalla, N., Devanathan, N., Dumai, N., Chauhan, N., Wahltinez, O., Botarda, P., Barnes, P., Barham, P., Michel, P., Jin, P., Georgiev, P., Culliton, P., Kuppala, P., Comanescu, R., Merhej, R., Jana, R., Rokni, R. A., Agarwal, R., Mullins, R., Saadat, S., Carthy, S. M., Perrin, S., Arnold, S. M. R., Krause, S., Dai, S., Garg, S., Sheth, S., Ronstrom, S., Chan, S., Jordan, T., Yu, T., Eccles, T., Hennigan, T., Kocisky, T., Doshi, T., Jain, V., Yadav, V., Meshram, V., Dharmadhikari, V., Barkley, W., Wei, W., Ye, W., Han, W., Kwon, W., Xu, X., Shen, Z., Gong, Z., Wei, Z., Cotruta, V., Kirk, P., Rao, A., Giang, M., Peran, L., Warkentin, T., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Sculley, D., Banks, J., Dragan, A., Petrov, S., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Borgeaud, S., Fiedel, N., Joulin, A., Kenealy, K., Dadashi, R., and Andreev, A. Gemma 2: Improving open language models at a practical size, 2024. URL <https://arxiv.org/abs/2408.00118>.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi,

- A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. URL <https://arxiv.org/abs/2307.09288>.
- Wang, K., Ren, H., Zhou, A., Lu, Z., Luo, S., Shi, W., Zhang, R., Song, L., Zhan, M., and Li, H. Mathcoder: Seamless code integration in LLMs for enhanced mathematical reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=z8TW0ttBPp>.
- Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.147. URL <https://aclanthology.org/2023.acl-long.147>.
- Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., and Zhao, J. Large language models are better reasoners with self-verification. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2550–2575, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.167. URL <https://aclanthology.org/2023.findings-emnlp.167>.
- Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Yang, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Liu, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., Guo, Z., and Fan, Z. Qwen2 technical report, 2024. URL <https://arxiv.org/abs/2407.10671>.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=5Xclecx0lh>.
- Zhou, A., Wang, K., Lu, Z., Shi, W., Luo, S., Qin, Z., Lu, S., Jia, A., Song, L., Zhan, M., and Li, H. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *The Twelfth International Conference on Learning Representations*, 2024a. URL <https://openreview.net/forum?id=c8McWs4Av0>.
- Zhou, J. P., Staats, C. E., Li, W., Szegedy, C., Weinberger, K. Q., and Wu, Y. Don’t trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*, 2024b. URL <https://openreview.net/forum?id=V5tdi14ple>.

A. Additional Main Results

Method	Qwen-2			Gemma-2		Phi-3	Mistral	Llama-2		Llama-3	Llama-3.1	Llama-3.2	
	0.5B	1.5B	7B	2B	9B	3.8B	7B	7B	13B	8B	8B	1B	3B
MULTIARITH													
Zero-shot CoT	78.17	93.50	98.67	94.00	98.17	98.17	83.67	74.33	76.67	96.67	97.83	85.67	97.83
Zero-shot PoT	18.00	4.33	59.67	88.67	98.00	96.67	61.67	29.83	60.17	97.33	97.83	78.67	62.33
Zero-shot PS	62.33	87.50	97.83	94.33	98.67	98.50	83.50	70.67	70.00	97.83	97.33	83.67	97.00
Zero-shot CoT @maj16	91.83	98.00	98.33	97.17	98.33	98.83	91.50	85.67	92.17	98.83	98.00	97.00	99.00
Zero-shot PoT @maj16	78.33	49.67	90.33	95.00	98.67	98.67	92.83	77.83	79.00	99.00	99.00	95.83	99.00
Zero-shot PS @maj16	91.33	98.33	98.17	97.00	98.67	98.83	94.33	90.33	92.17	98.17	98.50	97.83	98.50
Zero-shot CoT @maj32	92.83	98.00	98.50	96.33	98.33	98.83	92.67	89.17	94.00	98.67	98.33	86.17	98.50
Zero-shot PoT @maj32	88.33	70.33	86.50	94.50	98.67	98.83	94.50	83.67	83.67	98.67	99.17	85.83	98.67
Zero-shot PS @maj32	92.00	98.50	98.83	96.67	98.83	98.83	95.00	91.67	92.33	98.50	98.83	96.67	98.67
PROVE	95.67 (+2.84)	99.50 (+1.00)	98.83 (+0.00)	98.50 (+1.33)	99.00 (+0.17)	99.00 (+0.17)	95.83 (+0.83)	97.84 (+6.17)	96.34 (+2.34)	99.17 (+0.17)	99.34 (+0.17)	98.00 (+0.17)	99.17 (+0.17)
SINGLEEQ													
Zero-shot CoT	67.72	85.24	97.44	92.32	96.65	98.23	83.86	78.54	83.07	93.70	94.49	87.40	95.47
Zero-shot PoT	25.00	5.91	49.61	94.49	98.23	98.03	69.49	41.34	69.69	94.49	96.26	84.06	62.40
Zero-shot PS	62.80	81.69	96.65	92.13	97.24	98.03	85.83	75.00	76.57	91.14	95.28	84.45	93.90
Zero-shot CoT @maj16	81.50	90.75	96.85	94.49	97.24	98.02	89.37	83.27	92.13	96.85	97.24	94.09	97.83
Zero-shot PoT @maj16	80.31	71.06	79.33	96.06	98.82	99.02	93.69	86.22	88.78	98.03	98.43	94.49	97.64
Zero-shot PS @maj16	80.71	93.11	97.44	95.08	98.03	98.23	91.14	85.24	91.73	97.24	97.83	94.88	97.24
Zero-shot CoT @maj32	83.07	90.94	97.44	93.70	97.24	98.62	88.98	84.25	93.11	97.05	97.05	91.73	97.05
Zero-shot PoT @maj32	85.43	81.89	76.77	95.47	98.23	98.02	93.87	88.58	88.39	98.23	98.02	92.72	97.83
Zero-shot PS @maj32	82.48	93.31	98.03	95.28	97.83	98.82	90.75	85.24	92.13	97.24	98.03	92.52	97.05
PROVE	92.13 (+6.70)	96.80 (+3.49)	98.82 (+0.79)	97.64 (+1.58)	99.21 (+0.39)	99.21 (+0.19)	94.69 (+0.82)	92.91 (+4.33)	97.24 (+4.13)	98.23 (+0.00)	98.62 (+0.19)	96.06 (+1.18)	98.82 (+0.99)
SINGLEOP													
Zero-shot CoT	6.33	89.32	96.26	91.64	95.91	96.26	90.93	85.41	85.94	94.13	96.09	88.43	95.91
Zero-shot PoT	29.54	10.5	44.84	90.57	95.73	95.55	72.78	51.25	68.51	93.24	95.91	84.70	74.02
Zero-shot PS	66.19	88.26	96.26	93.42	96.26	96.80	90.75	81.67	81.32	91.10	95.55	86.83	94.13
Zero-shot CoT @maj16	87.37	92.88	96.26	94.13	96.09	96.80	94.66	88.79	92.70	96.62	96.80	93.77	96.80
Zero-shot PoT @maj16	84.52	77.22	80.60	92.88	95.73	96.80	93.59	89.86	85.77	95.37	96.44	91.99	96.44
Zero-shot PS @maj16	88.08	93.59	96.26	94.66	96.62	97.15	94.84	91.10	91.28	96.44	96.62	93.06	96.62
Zero-shot CoT @maj32	87.37	93.59	96.26	93.95	96.26	96.98	95.02	89.50	92.35	96.44	96.62	94.20	96.62
Zero-shot PoT @maj32	88.79	86.83	78.47	93.42	95.91	97.33	94.13	92.35	88.26	95.91	96.44	93.24	96.09
Zero-shot PS @maj32	87.01	93.77	96.44	93.59	96.62	97.15	94.66	91.46	92.13	96.80	96.62	94.66	96.80
PROVE	92.53 (+3.74)	94.66 (+0.89)	96.98 (+0.54)	94.84 (+0.18)	96.62 (+0.00)	97.51 (+0.18)	95.02 (+0.00)	94.66 (+2.31)	93.24 (+0.54)	97.16 (+0.36)	96.98 (+0.18)	94.66 (+0.00)	96.80 (+0.00)
ADDSUB													
Zero-shot CoT	58.73	79.75	94.94	86.33	94.94	95.95	82.28	73.67	81.01	89.37	92.66	84.56	93.42
Zero-shot PoT	49.62	7.34	28.61	53.92	95.95	95.7	70.38	42.53	65.57	93.67	95.44	78.73	88.86
Zero-shot PS	51.39	77.47	92.91	89.11	95.19	94.43	83.29	73.42	75.19	85.32	92.66	77.97	90.89
Zero-shot CoT @maj16	70.89	86.58	95.44	90.89	94.43	96.2	85.82	80.51	86.08	93.42	95.19	89.62	95.95
Zero-shot PoT @maj16	71.39	64.81	63.54	90.89	95.95	96.46	91.14	81.52	85.06	95.19	95.95	82.53	95.95
Zero-shot PS @maj16	71.14	85.82	94.94	91.14	95.7	96.46	88.35	83.29	87.34	93.16	94.43	90.13	94.68
Zero-shot CoT @maj32	72.41	87.34	94.94	89.62	95.70	96.71	86.08	81.27	85.32	91.14	94.94	87.59	95.95
Zero-shot PoT @maj32	73.42	73.42	59.49	90.63	95.95	96.71	90.38	85.57	86.08	95.19	94.68	87.09	95.70
Zero-shot PS @maj32	75.19	87.85	94.94	90.63	95.70	96.71	87.09	83.04	87.59	91.39	94.94	89.87	95.95
PROVE	85.06 (+9.87)	92.15 (+4.30)	96.46 (+1.02)	92.41 (+1.27)	96.20 (+0.25)	96.96 (+0.25)	91.65 (+0.51)	89.37 (+3.80)	93.16 (+5.57)	95.19 (+0.00)	96.20 (+0.25)	92.41 (+2.28)	95.95 (+0.00)

Table 7: Main results showing the comparison of PROVE and baseline methods across 13 LLMs, ranging in size from 0.5B to 13B, on MultiArith, SingleEQ, SingleOP, AddSub.

B. Implementation Details for MATH

The overall framework for the MATH dataset closely follows the structure outlined in Section 2, with minor adjustments in the prompts to accommodate answers in LaTeX format. Figure 6 provides an actual example of these prompts, along with the outputs generated by Phi3-mini-4k-instruct.

Generating plan and solution. We prompt an LLM to generate a plan and solution using the zero-shot prompt proposed by Wang et al. (2023a): “Let’s first understand the problem and devise a plan to solve the problem. Then, let’s carry out the plan and solve the problem step by step”. Additionally, we extend this prompt by adding “Present the final answer enclosed in `\boxed{}`.”, instructing the model to present the final answer within `\boxed{}` to facilitate easy extraction of the answer.

Translating plan and solution to Python programs. We use each plan and solution individually as input to GPT-4o (2024-02-15-preview), prompting it to generate a Python code block that implements the plan to solve the math word problem and printing the final answer in `\\boxed{}`. We append “*Convert the following plan and solution to a math problem into Python code. Print the final answer enclosed in \\boxed{}*.” to the prompt for generation.

Verification and selection. Finally, we execute the Python code block and retrieve the output answer by extracting the value enclosed in `\\boxed{}` from the standard output produced by the `print()` statement. This output answer is then compared to the generated solution for verification. If the solution matches the output answer, we consider the solution valid. After verifying all candidate plans and solutions, we conduct majority voting among the valid answers to determine the final answer. If no valid answers remain, we perform majority voting across all invalid answers instead.

C. Case Study on MATH Dataset

We an example where PROVE succeeds in answering the question correctly, unlike the baseline methods. In this example, the problem requires differentiation, but Phi-3-mini-4k-instruct performs the differentiation incorrectly, leading to an incorrect answer. In contrast, PROVE avoids such errors by offloading the differentiation calculation to the *sympy* library, allowing it to filter out flawed reasoning paths effectively.

Question from MATH dataset

Find the minimum value of

$$\frac{(x+5)(x+2)}{x+1}$$

for $x > 0$.

Generated Solution by Phi-3-mini-4k-instruct

We want to find the minimum value of the expression

$$\frac{(x+5)(x+2)}{x+1}$$

for $x > 0$.

One approach is to use calculus. We can find the critical points of the function by taking the derivative and setting it equal to zero.

The derivative of the function is:

$$\frac{d}{dx} \left(\frac{(x+5)(x+2)}{x+1} \right) = \frac{(2x+7)(x+1) - (x+5)(1)}{(x+1)^2} = \frac{x^2 + 8x + 7}{(x+1)^2}.$$

Setting the derivative equal to zero, we get:

$$x^2 + 8x + 7 = 0.$$

This quadratic equation does not have real solutions, which means there are no critical points.

Since the function is continuous and differentiable for $x > 0$, and there are no critical points, we can check the behavior of the function as x approaches infinity and zero.

As x approaches infinity, the function approaches the limit:

$$\lim_{x \rightarrow \infty} \frac{(x+5)(x+2)}{x+1} = \lim_{x \rightarrow \infty} \frac{x^2 + 7x + 10}{x+1} = \infty.$$

As x approaches zero from the right (since $x > 0$), the function approaches:

$$\lim_{x \rightarrow 0^+} \frac{(x+5)(x+2)}{x+1} = \lim_{x \rightarrow 0^+} \frac{x^2 + 7x + 10}{x+1} = 10.$$

Since the function approaches infinity as x approaches infinity and approaches 10 as x approaches zero from the right, we can conclude that the minimum value of the function is 10.

Translated Python Program by GPT-4o

Certainly! Below is the Python code that follows the plan and solution to find the minimum value of the given expression:

```
'''python
import sympy as sp

# Define the variable
x = sp.symbols('x', positive=True)

# Define the function
y = (x + 5) * (x + 2) / (x + 1)

# Simplify the function
y_simplified = sp.simplify(y)

# Find the derivative
y_prime = sp.diff(y_simplified, x)

# Solve for critical points
critical_points = sp.solve(y_prime, x)

# Check the second derivative to confirm minimum
y_double_prime = sp.diff(y_prime, x)

# Evaluate the second derivative at critical points
min_value = None
for point in critical_points:
    if y_double_prime.subs(x, point) > 0:
        min_value = y_simplified.subs(x, point)
        break

# Print the minimum value
print(f"The minimum value is: \boxed{{{min_value}}}")
'''
```

This code uses the 'sympy' library to perform symbolic mathematics, including differentiation and solving equations. It finds the critical points, checks the second derivative to confirm a minimum, and evaluates the function at the critical point to find the minimum value.



(b) Implementing Python Program from Plan and Solution

Figure 6: An actual example of prompts used for MATH dataset, along with the outputs generated by Phi3-mini-4k-instruct and GPT-4o to translate the plan and solution into Python code.