WizardMath: Empowering Mathematical Reasoning for Large Language Models via *Reinforced Evol-Instruct*

Haipeng Luo[2][*]   **Qingfeng Sun**[1][*]   **Can Xu**[1][†]   **Pu Zhao**[1]   **Jianguang Lou**[1]
Chongyang Tao[1]   **Xiubo Geng**[1]   **Qingwei Lin**[1]   **Shifeng Chen**[2][†]   **Dongmei Zhang**[1]

[1]Microsoft
[2]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
{caxu,qins,puzhao,jlou,chotao,xigeng,qlin,dongmeiz}@microsoft.com
{hp.luo,shifeng.chen}@siat.ac.cn

Abstract

Large language models (LLMs), such as GPT-4, have shown remarkable performance in natural language processing (NLP) tasks, including challenging mathematical reasoning. However, most existing open-source models are only pre-trained on large-scale internet data and without math-related optimization. In this paper, we present *WizardMath*, which enhances the mathematical reasoning abilities of Llama-2, by applying our proposed *Reinforcement Learning from Evol-Instruct Feedback* (**RLEIF**) method to the domain of math. Through extensive experiments on two mathematical reasoning benchmarks, namely GSM8k and MATH, we reveal the extraordinary capabilities of our model. *WizardMath* surpasses all other open-source LLMs by a substantial margin. Furthermore, our model even outperforms ChatGPT-3.5, Claude Instant-1, PaLM-2 and Minerva on GSM8k, simultaneously surpasses Text-davinci-002, PaLM-1 and GPT-3 on MATH. More details and model weights are public at `https://github.com/nlpxucan/WizardLM` [3] and `https://huggingface.co/WizardLM`.

1 Introduction

Recently, Large-scale language models (LLMs) have garnered significant attention and become the go-to approach for numerous natural language processing (NLP) tasks, including open domain conversation [1–4], coding [5–13] and math [14–19]. A conspicuous example is ChatGPT, developed by OpenAI. This model uses extensive pre-training on large-scale internet data and further fine-tuning with specific instruction data and methods. As a result, it achieves state-of-the-art zero-shot performance on various benchmarks. Subsequently, Anthropic, Google, and Meta also launched their competitive products one after another. Notably, Meta's series of Llama [4, 20] models have sparked an open-source revolution and quickly narrowed the gap with those closed-source LLMs. This trend also gradually stimulates the releases of MPT[8], Falcon [21], StarCoder [12], Alpaca [22], Vicuna [23], and WizardLM [24], etc. However, these open models still struggles with the scenarios which require complex multi-step quantitative reasoning, such as solving mathematical and science challenges [25–35].

---

[*]   Equal contribution. Work done during the internship of Luo at Microsoft Research.
[†]   Corresponding author: caxu@microsoft.com and shifeng.chen@siat.ac.cn
[3]   We are working with our legal team to review and publicly release the code and data in accordance with our policy.
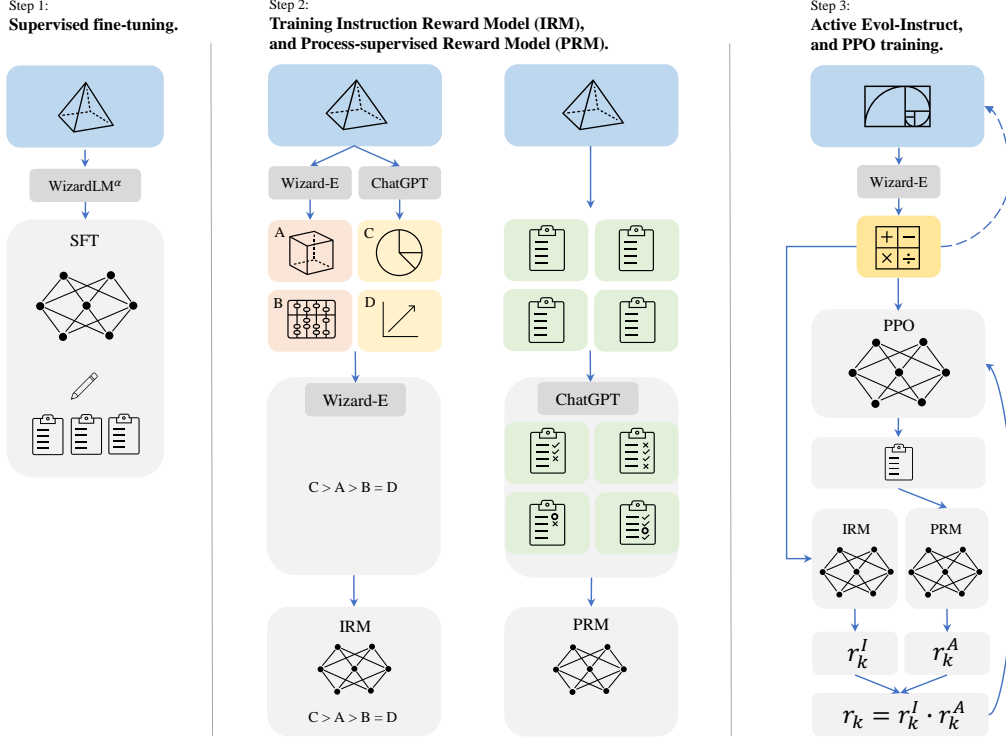
Figure 1: A diagram illustrating the three steps of our *Reinforcement Learning from Evol-Instruct Feedback* (**RLEIF**): (1) supervised fine-tuning (SFT), (2) Instruction Reward Model (IRM) training and Process-supervised Reward Model (PRM) training, and (3) Active Evol-Instruct and reinforcement learning via proximal policy optimization (PPO).

Chain-of-thought (CoT) [31] proposes to design better prompts to generate step-by-step solutions, which can lead to improved performance. Self-Consistency [34] also achieves remarkable performance on many reasoning benchmarks, which generates several possible answers from the model and selects the correct one based on majority vote [35]. In recent, [36] finds that process supervision with reinforcement learning significantly outperforms outcome supervision for solving challenging MATH problems.

Inspired by *Evol-Instruct* and Process-supervised Reinforcement Learning, this work aims to enhance the mathematical reasoning abilities of the SOTA open-source LLM, Llama-2 [20]. As shown in the Figure 1, we propose a new method named *Reinforcement Learning from Evol-Instruct Feedback* (**RLEIF**), which could firstly generate diverse math instructions data by math-specific *Evol-Instruct*, then we train an instruction reward model (IRM) and a process-supervised reward model (PRM) [16, 36–41], the former indicates the quality of the evolved instruction and the later receives feedback for each step in the solution. The brand-new *Evol-Instruct* method includes two downward evolution and upward evolution progress to produce the grade school math and challenging math respectively. Initially, we re-generate, filter and finetune the original math instruction data from GSM8k [42] and MATH [43]. Immediately, we train the Llama-2 models to obtain the reward models and our *WizardMath*.

We perform experiments on two mathematical reasoning benchmarks, namely GSM8k [42] and MATH [43], the results demonstrate that our *WizardMath* outperforms all other open-source LLMs, achieving state-of-the-art performance. Specifically, *WizardMath* observe a substantial improvement in pass@1 with an increase of +24.8 (81.6. vs. 56.8) on GSM8k, and +9.2 (22.7 vs. 13.5) on MATH. Notably, our model even also significantly surpasses OpenAI's ChatGPT-3.5[5], Anthropic's Claude Instant-1 [39], and Google's PaLM-2 [44] in terms of pass@1 on GSM8k.

The main contributions of this work are as following:

- We introduce *WizardMath* model, which enhances the mathematical reasoning abilities for open-source pretrained large language model Llama-2 [20].

- We propose a new method, *Reinforcement Learning from Evol-Instruct Feedback* (**RLEIF**), alongside *Evol-Instruct* and Reinforcement Learning, for improving LLM reasoning performance.

- *WizardMath* surpasses all other open-source LLMs by a substantial margin in terms of mathematical reasoning, including Llama-2 70B [20], Llama-1 65B [4], Falcon-40B [21], MPT-30B[8], Baichuan-13B Chat[9] and ChatGLM2 12B [45] on both GSM8k [42] and MATH [43].

- *WizardMath* significantly outperforms various main closed-source LLMs, such as ChatGPT[5], GPT-3.5, Claude Instant [39], PaLM-2 [44], PaLM-1 [7] and Minerva[15] on GSM8k.

2   Method

In this section, we elaborate on the details of our *WizardMath*. Following WizardLM and PRMs[36], we propose *Reinforcement Learning from Evol-Instruct Feedback* (**RLEIF**), which integrates the *Evol-Instruct* and reinforced process supervision method to evolve GSM8k and MATH, and fine-tune the pre-trained Llama-2 with the evolved data and reward models.

As shown in the Figure 1, our methods apply three steps:

1. Supervised fine-tuning.

2. Training instruction reward model, and process-supervised reward model.

3. Active Evol-Instruct, and PPO training.

2.1   Supervised fine-tuning

Following InstructGPT[2], we also firstly fine tune the base with supervised instruction-response pairs, which contains:

1. To make the parsing of each step easier, we few-shot re-generate 15k answers for GSM8k and MATH with an Alpha version of WizardLM 70B model to produce solutions in a step-by-step format, then find out those with a correct answer, and use this data to finetune base Llama model.

2. To enhance the model's ability to adhere to the neural and diverse instructions, we also sample 1.5k open-domain conversations from WizardLM's training data, then merge it with above math corpus as the final SFT training data.

2.2   *Evol-Instruct* principles for math

Motivated by the Evol-Instruct [24] method proposed by WiazrdLM and its effective application on WizardCoder [13], this work attempts to make math instructions with various complexities and diversity to enhance the pre-trained LLMs. Specifically, we adapt Evol-Instruct to a new paradigm including two evolution lines:

1. Downward evolution: It enhances instructions by making the questions easier. For example i): revising high difficulty questions to lower difficulty, or ii) producing a new and easier question with another different topic.

2. Upward evolution: Derived from original Evol-Instruct method, it deepens and generates new and harder questions by i) adding more constraints, ii) concretizing, iii) increasing reasoning.

2.3   *Reinforcement Learning from Evol-Instruct Feedback* (*RLEIF*)

Inspired by InstructGPT[2] and PRMs[36], we train two reward models to predict the quality of the instructions and the correctness of each step in the answer respectively:
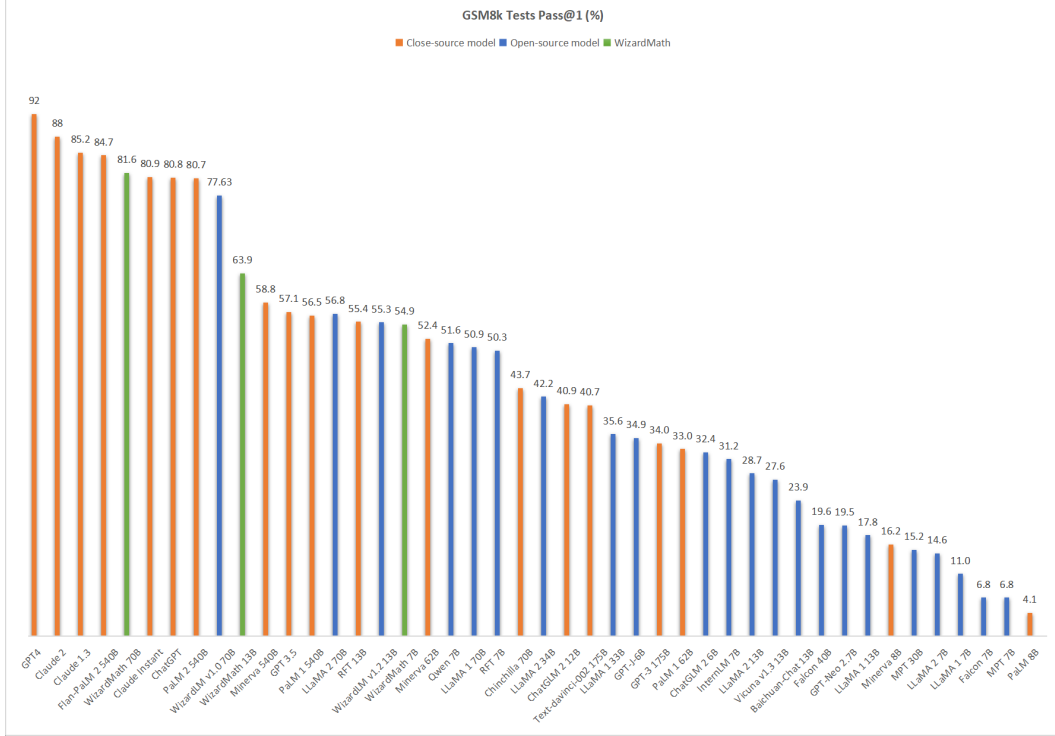
Figure 2: The pass@1 performance of main LLM models on the GSM8k benchmark, our model is currently ranked in the top five, slightly outperforming some close-source models such as ChatGPT-3.5[5], Claude Instant-1[6], PaLM 2 [44], and substantially surpassing all open-source models.

1. Instruction Reward Model (IRM): This model aims to judge the quality of the evolved instructions on three aspects: i) Definition, ii) Precision, and iii) Integrity. To produce the ranking list training data of IRM, for each instruction, we firstly use ChatGPT and Wizard-E [4] to generate 2~4 evolved instructions respectively. Then we leverage Wizard-E to rank the quality of those 4~8 instructions.

2. Process-supervised Reward Model (PRM): As there is no powerful open-source math reasoning LLMs before this work, there is no simple way to support highly precise process supervision without professional human-labelers and close-source ChatGPT. Therefore, we depend on ChatGPT to provide process supervision, and ask it to assess the correctness of each step in the solutions generated by our model.

3. PPO training. We evolve the original math (GSM8k + MATH) instructions by 8 turns, increasing the data size from 15k to 96k. We use IRM and PRM to generate the instruction reward (r^I) and the answer reward (r^A). Then apply a product as the final reward $r = r^I \cdot r^A$.

3 Experiment

This section provides a comprehensive overview of the baseline models in our experiments. Subsequently, we mainly elucidate the performance metrics of our models on two prevalent mathematical benchmarks: GSM8k [42] and MATH [43].

3.1 Baselines

Close-Source Models. Numerous technology companies have effectively created exceptionally proficient Large Language Models (LLMs) [3, 4, 7, 20, 44, 45, 47, 51–53], but have opted against

---

[4] Wizard-E named Wizard-Evol-Generator, which is an Alpha version fine-tuned Llama model specifically used to execute *Evol-Instruct* without APIs.

Table 1: Results of pass@1 (%) on GSM8k and MATH. In this study, to ensure equitable and cohesive evaluations, we report the socres of all models within the settings of greedy decoding and CoT [31]. We report the improvement between WizardMath and baseline model with similar parameter size.

Model	Params	GSM8k	MATH
Closed-source models			
GPT-4 [3]	-	92.0	42.5
Claude 2[7]	-	88.0	-
Claude 1.3[7]	-	85.2	-
Flan-PaLM 2 [44]	540B	84.7	33.2
Claude Instant[7]	-	80.9	-
ChatGPT [46]	-	80.8	34.1
PaLM 2 [44]	540B	80.7	34.3
	8B	16.2	14.1
Minerva [15]	62B	52.4	27.6
	540B	58.8	33.6
GPT-3.5 [3]	-	57.1	-
	8B	4.1	1.5
PaLM [7]	62B	33.0	4.4
	540B	56.5	8.8
RFT-13B [16]	13B	55.4	-
Chinchilla [47]	70B	43.7	-
ChatGLM 2 [45]	12B	40.9	-
Text-davinci-002 [15]	175B	40.7	19.1
GPT-3 [1]	175B	34.0	5.2
GPT-2 [43]	1.5B	-	6.9
Open-source models			
GAL [14]	30B	-	12.7
	120B	-	20.4
	7B	14.6	2.5
LLaMA 2 [20]	13B	<u>28.7</u>	<u>3.9</u>
	34B	42.2	6.24
	70B	<u>56.8</u>	<u>13.5</u>
Qwen [10]	7B	<u>51.6</u>	-
	7B	11.0	2.9
LLaMA 1 [4]	13B	17.8	3.9
	33B	35.6	7.1
	65B	50.9	10.6
RFT-7B [16]	7B	50.3	-
GPT-J-6B [48]	6B	34.9	-
ChatGLM 2 [45]	6B	32.4	-
InternLM-7B [49]	7B	31.2	-
Vicuna v1.3 [23]	13B	27.6	-
Baichuan-chat [9]	13B	23.9	-
Falcon [21]	7B	6.8	2.3
	40B	19.6	2.5
GPT-Neo-2.7B [50]	2.7B	19.5	-
MPT[8]	7B	6.8	<u>3.0</u>
	30B	15.2	3.1
WizardMath	7B	**54.9** (+3.3)	**10.7** (+7.7)
WizardMath	13B	**63.9** (+35.2)	**14.0** (+10.1)
WizardMath	70B	**81.6** (+24.8)	**22.7** (+9.2)

Table 2: Results of pass@1 (%) on MATH Subtopics with WizardMath 70B model.

MATH subtopics	WizardMath 70B
Intermediate Algebra	7.1
Precalculus	12.6
Geometry	15.7
Number Theory	16.3
Counting & Probability	17.3
Prealgebra	41.7
Algebra	33.3
Overall	**22.7**

making them publicly available, so they are referred to as close-source models. In our research, we extensively integrate a significant number of close-source models as the foundational benchmarks. Specifically, our baselines encompass the following models: (i) OpenAI's GPT-3 [51], GPT-3.5, ChatGPT[5], GPT-4 [3]; (ii) Google's PaLM 2 [44], PaLM [7], and Minerva [15]; (iii) Anthropic's Claude Instant [39], Claude 1.3[6], Claude 2[7], DeepMind's Chinchilla [47].

Open-Source Models. Massive open-source LLMs [4, 20–23, 45, 52, 53] have been accessible to the AI community. Nonetheless, their performance consistently tends to significantly lag behind the close-source models. As part of our research, we incorporate a significant number of these open-source models as our baselines, which mainly contain the following: Llama 1 [4] & Llama 2 [20], GAL [14], GPT-J [48], GPT-Neo [50], Vicuna [23], MPT[8], Falcon[21], Baichuan[9], ChatGLM [45], Qwen[10] and RFT [16].

3.2 Evaluate Benchmarks

We mainly evaluate WizardMath on two benchmarks (GSM8k [42] and MATH [43]). The GSM8k [42] dataset contains approximately 7500 training data and 1319 test data, mainly on grade school level math problems, each of which consists of basic arithmetic operations (addition, subtraction, multiplication, and division), and generally requires 2 to 8 steps to solve. The MATH [43] dataset collects math problems from prestigious math competitions such as AMC 10, AMC 12, and AIME. It contains 7500 training data and 5,000 challenging test data in seven academic areas: Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus. Furthermore, these problems are divided into five levels of difficulty, with '1' denoting the relatively lower difficulty level and '5' indicating the highest level.

3.3 Train and Evaluation prompt

The Llama 2 [20] base serves as our foundation model.

We undertake the training of our *WizardMath* by employing the prompt from Alpaca [22]:

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

We evaluate GSM8k [42] and MATH benchmarks [43] by employing the following CoT [31] prompt:

```
Below is an instruction that describes a task.  Write a
response that appropriately completes the request.\n\n###
Instruction:\n{instruction}\n\n### Response:  Let's think step by step.
```

3.4 Evaluation on GSM8k and MATH

Notably, in the Figure 2 and Table 1, we cite the metrics of GPT-4 and GPT-3.5 from [3]. The evaluation of the ChatGPT model's scores are from [46]. For the assessment of Claude Instant, Claude 1.3, and Claude 2, the scores are extracted from [7]. The scores of PaLM 1, PaLM 2, and Minerva are garnered from [7, 15, 44]. Finally, the scores associated with Text-davinci-002, GPT-3 and GPT-2 are garnered from [15, 43]. On the open-source models, most scores are retrieved from the paper of Llama 2 [20] or their self-reports. Additionally, we evaluate the Baichuan-chat, Vicuna v1.3 by ourselves. In the Table 2, we show the detailed results of MATH subtopics with our WizardMath 70B model.

Comparing with the Close-Source Models.   In Table 1, our *WizardMath 70B* slightly outperforms some close-source LLMs on GSM8k, including ChatGPT, Claude Instant and PaLM 2 540B. And as shown in Figure 2, our model is currently ranked in the top five on all models. Simultaneously,*WizardMath 70B* also surpasses the Text-davinci-002 on MATH. The detailed results are as follows:

1. *WizardMath 13B* outperforms PaLM 1 540B (63.9 vs 56.5), Minerva 540B (63.9 vs 58.8), and GPT-3.5 (63.9 vs 57.1) on GSM8k. Meanwhile,it surpasses PaLM 1 540B (14.0 vs. 8.8), GPT-3 175B (14.0 vs. 5.2) on MATH.

2. *WizardMath 70B*, our largest model, achieves the superior or comparable performance with Claude Instant (81.6 vs 80.9), ChatGPT (81.6 vs 80.8) and PaLM 2 (81.6 vs 80.7) on GSM8k. Concurrently, WizardMath 70B also exceeds Text-davinci-002 (22.7 vs. 19.1) by a margin of 3.6% on the MATH benchmarks.

Comparing with the Open-Source Models.   The findings illustrated in the table 1 explicitly demonstrate that our *WizardMath 70B*, distinctly manifest a substantial performance advantage over all the open-source models across both the GSM8k and MATH benchmarks. The detailed results are as follows:

1. *WizardMath 7B* surpasses most open-source models with parameter counts ranging approximately from 7B to 40B, including MPT, Falcon, Baichuan-chat, Vicuna v1.3, ChatGLM 2, Qwen, Llama 1 and Llama 2 on the GSM8k and MATH benchmarks. Even though its parameter counts are significantly lower.

2. *WizardMath 13B* is significantly superior to Llama 1 65B (63.9 vs. 50.9) and Llama 2 70B (63.9 vs. 56.8) on GSM8k. Additionly, it substantially outperforms both Llama 1 65B (14.0 vs. 10.6) and Llama 2 70B (14.0 vs. 13.5) on MATH.

3. *WizardMath 70B*, our most extensive model, exemplifies a substantial advancement in performance, surpassing Llama 2 70B (81.6 vs. 56.8) by a significant margin of 24.8% on GSM8k. Concurrently, it also outperforms Llama 2 70B (22.7 vs. 13.5) by a margin of 9.2% on MATH.

3.5 Case Study

Appendix A shows some examples generated by our *WizardMath*. The examples demonstrate that our model consistently generates accurate response answers accompanied by clear explanations.

4 Related Work

Large Language Models.   LLMs have achieved substantial advancements within the realm of Natural Language Processing (NLP), providing a valuable and task-agnostic foundation for widespread applications. These models typically encompass parameter counts reaching into the hundreds of billions, which are trained on extensive large-scale corpuses of textual data. The prominent instances

entail OpenAI's GPT3&4 [3, 51], Anthropic's Claude[7], Google's PaLM [7, 44], Bard[11], DeepMind's Chinchilla [47], and Gopher [52]. However none of them have been open-sourced so far, and some of them can only be exclusively accessible through APIs.

Recently, the AI landscape has borne witness to the emergence of numerous open-source LLMs, characterized by publicly accessible model codes and weight parameters. EleutherAI has contributed GPT-NeoX-20B [54] and GPT-J-6B [48]. BigScience has introduced BLOOM [55]. Similarly, Meta has made strides by releasing OPT [53], Llama 1 [4], Llama 2 [20], and GAL [14]. Tsinghua University has unveiled GLM-130B and ChatGLM [45]. TII has facilitated the release of Falcon [21]. Additionally, LLMs such as Baichuan[9] and Qwen[10] have also surfaced. Presently, Llama assumes a pivotal role as the foundational model for supervised fine-tuning, ushering in the emergence of several extremely remarkable models, including Alpaca [22], Vicuna [23], Guanaco [56], WizardLM [24], and Orca [57], RFT [16] etc.

Large Language Models For Mathematical reasoning. It's well known that complex reasoning problems are challenging for NLP models, which include mathematical reasoning [25–30], commonsense reasoning [58, 59], and logical reasoning [31]. A substantial body of current research is centered around the intricate task reasoning of the Mathematical Word Problems(MWP) [30, 42, 60–64], which requires the ability to understand mathematical concepts, computation and multi-step reasoning [16–19, 36, 40, 46]. Addtitionly, models are evaluated across different levels of MWP benchmarks on some mathematical reasoning datasets such as AddSub [65], MultiArith [66], SingleEQ [67], SVAMP [60], GSM8K [42], AQuA [29] and MATH [43].

To enhance the reasoning ability of LLMs, [31] proposed Chain-of-Thought Prompting, which attaches multiple reasoning steps before obtaining the answer for a question. By employing the simple few-shot reasoning strategy, LLMs are able to perform better in complex reasoning problems. Least-to-Most [68] prompting decomposes the problem into sub-problems that are then solved incrementally. Additionally each step has a more detailed reasoning process. Similarly, the Complex CoT [35] underscores the pivotal role of prompt complexity by strategically choosing the most intricate problems and their corresponding solutions to function as prompts. To alleviate the burden of manual efforts, [33] introduced Auto-CoT, an approach that automates the process of acquiring k samples through the application of clustering techniques on a provided dataset. With the objective of mitigating manual intervention, [32] proposed Zero-shot-CoT, which entails the straightforward practice of appending the phrase "Let's think step by step" to each answer, eliciting the inference steps without examples. Moreover, [34] expanded upon this notion by suggesting the exploration of diverse inference paths throughout the reasoning process. Consequently, the ultimate outcome is determined through either the aggregation of answers using majority voting or by leveraging a validation mechanism, as posited by [69]. [16] employs a straightforward approach for generating augmented samples, focusing on probing the correlation between LLMs and math reasoning ability.

Large Language Models For Reinforcement Learning. Nevertheless, even state-of-the-art models frequently manifest logical errors and a range of illusions [70, 71]. These anomalies become especially challenging within domains necessitating multi-step reasoning, where a singular logical misstep maybe precipitate the unraveling of an entire solution. An effective strategy involves the training of reward models aimed at discriminating between favorable and unfavorable outputs [36]. Early outcome-based approaches were mainly performed on algorithmic tasks [72–75]. [42] demonstrated the significant benefits of reward models or validators, and [76] proposed a heuristic-based step-size-aware RM. [2, 77–79] proposed the use of reward models for a reinforcement learning pipeline. [20, 37–39, 42, 80–82] employed rejection sampling for searching to achieve alignment of LLMs with human preferences.

The differences between outcome-based and process-based reward modelling are further discussed by [40]. Outcome-supervised reward models (ORMs) undergo training exclusively utilizing the ultimate outcomes derived from the model's chain-of-thought process. Conversely, process-supervised reward models (PRMs) are designed to solicit feedback for each individual step within the chain-of-thought progression. In the domain of logical reasoning, ORMs frequently employ incorrect reasoning pathways yet yield the correct final answer [41, 83]. Notably, PRMs has been demonstrated to effectively alleviate this phenomenon of inconsistent behavior [40]. [36, 84, 85] amassed an expansive corpus of process-based supervised signals through meticulous manual annotation, which

---

[11] https://bard.google.com/

verified that PRMs and supervision with manual annotation yielded more pronounced advantages for LLMs as compared to ORMs.

Large Language Models For Instruction Fine-Tuning. The initial endeavors in instruction-following training work primarily focused on enhancing the language model's capacity for generalization across diverse tasks. This often involves the process of fine-tuning across substantially available Natural Language Processing datasets, and evaluates on the different NLP tasks. T5 [86] undertake the earliest attempts to train a range of NLP tasks, including Question and Answer, Document Summarization, and Sentiment Classification, by employing a consistent prompt format across all the data. Subsequently, instruction fine-tuning work such as FLAN [87], ExT5 [88], T0 [89], UnifiedQA [90], ZeroPrompt [91], and FLAN-T5 [92] emerged to adapt for a large number of downstream tasks. To address the challenge of misalignment between model outputs and human requirements, OpenAI manually annotates the instruction library to construct a diverse range of tasks. Simultaneously, Reinforcement Learning from Human Feedback technology is employed, which facilitate the rapid development of LLMs such as InstructGPT [2], ChatGPT[5], GPT-4 [3]. To reduce manual involvement, self-instruct [93] improves instruction-following through self-generated instructions. Alpaca [22] used a dataset of 50k instructions generated from a limited (e.g., 175 samples) seed set of manually-written instructions. Vicuna [23] used 70k user-shared conversations with ChatGPT collected from ShareGPT.com. Meanwhile, WizardLM [24] introduces the evol-instruct approach, which seeks to refine the existing instruction data by enhancing both its complexity and diversity.

5   Conclusion and Future Work

This paper introduces *WizardMath*, a mathematics model fine-tuned with ***RLEIF***. The experimental results demonstrate that *WizardMath* achieves SOTA performance surpassing all existing open-source LLMs on two widely recognized mathematical reasoning benchmarks: GSM8k and MATH. Furthermore, *WizardMath* exhibits superior performance compared to some of the largest close-source LLMs, including ChatGPT, GPT-3.5, Claude Instant, PaLM-2, PaLM-1 and Minerva on the GSM8k benchmark.

Future Work. Although our *WizardMath* achieves impressive mathematics performance, as depicted in Figure 2, our model still falls significantly behind the SOTA LLM, GPT-4 and Claude-2. Therefore, future work will prioritize the enhancement of the ***RLEIF*** or better method to further augment the performance of our model.

Broader Impact. Similar to the other LLMs, our *WizardMath* could also generate unethical, harmful, or misleading information sometimes. Therefore, future research to address the ethical and societal implications is needed.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[3] OpenAI. Gpt-4 technical report, 2023.

[4] 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 arXiv:2302.13971*, 2023.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[6] Microsoft. Azure openai service models. https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/models, 2023.

[7] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[8] 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 *The Eleventh International Conference on Learning Representations*, 2023.

[9] Yue Wang, Weishi Wang, Shafiq R. Joty, and Steven C. H. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 8696–8708. Association for Computational Linguistics, 2021.

[10] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. Codet5+: Open code large language models for code understanding and generation, 2023.

[11] Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x, 2023.

[12] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

[13] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023.

[14] 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 arXiv:2211.09085*, 2022.

[15] 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. *arXiv preprint arXiv:2206.14858*, 2022.

[16] 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 arXiv:2308.01825*, 2023.

[17] Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*, 2023.

[18] Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*, 2023.

[19] 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 arXiv:2305.04091*, 2023.

[20] 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 arXiv:2307.09288*, 2023.

[21] 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 arXiv:2306.01116*, 2023.

[22] 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.

[23] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[24] 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 arXiv:2304.12244*, 2023.

[25] Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. A survey of deep learning for mathematical reasoning. *arXiv preprint arXiv:2212.10535*, 2022.

[26] Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. Mathematical capabilities of chatgpt. *arXiv preprint arXiv:2301.13867*, 2023.

[27] Arindam Bhattacharya. A survey of question answering for math and science problem. *arXiv preprint arXiv:1705.04530*, 2017.

[28] Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[29] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *ACL*, 2017.

[30] 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*, pages 1152–1157, San Diego, California, June 2016. Association for Computational Linguistics.

[31] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[32] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

[33] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

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

[35] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

[36] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[37] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

[38] Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

[39] 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 arXiv:2212.08073*, 2022.

[40] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

[41] Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

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

[43] 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. *arXiv preprint arXiv:2103.03874*, 2021.

[44] 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 arXiv:2305.10403*, 2023.

[45] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

[46] Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Qizhe Xie. Automatic model selection with large language models for reasoning. *arXiv preprint arXiv:2305.14333*, 2023.

[47] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

[48] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

[49] InternLM Team. Internlm: A multilingual language model with progressively enhanced capabilities. `https://github.com/InternLM/InternLM`, 2023.

[50] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Rose Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. 2021.

[51] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[52] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[53] 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 arXiv:2205.01068*, 2022.

[54] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

[55] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[56] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[57] 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 arXiv:2306.02707*, 2023.

[58] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[59] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

[60] 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*, pages 2080–2094, 2021.

[61] Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. Mwptoolkit: an open-source framework for deep learning-based math word problem solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 13188–13190, 2022.

[62] Zhanming Jie, Jierui Li, and Wei Lu. Learning to reason deductively: Math word problem solving as complex relation extraction. *arXiv preprint arXiv:2203.10316*, 2022.

[63] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*, 2023.

[64] Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *arXiv preprint arXiv:2305.17306*, 2023.

[65] 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)*, pages 523–533, Doha, Qatar, October 2014. Association for Computational Linguistics.

[66] Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[67] 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.

[68] Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Huai hsin Chi. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625, 2022.

[69] 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)*, pages 5315–5333, Toronto, Canada, July 2023. Association for Computational Linguistics.

[70] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[71] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*, 2020.

[72] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[73] Scott Reed and Nando De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

[74] Chengtao Li, Daniel Tarlow, Alexander L. Gaunt, Marc Brockschmidt, and Nate Kushman. Neural program lattices. In *International Conference on Learning Representations*, 2016.

[75] Jonathon Cai, Richard Shin, and Dawn Song. Making neural programming architectures generalize via recursion. *arXiv preprint arXiv:1704.06611*, 2017.

[76] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*, 2022.

[77] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[78] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[79] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[80] Eric Nichols, Leo Gao, and Randy Gomez. Collaborative storytelling with large-scale neural language models. In *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–10, 2020.

[81] Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. Generate & rank: A multi-task framework for math word problems. *arXiv preprint arXiv:2109.03034*, 2021.

[82] Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. Preference ranking optimization for human alignment. *arXiv preprint arXiv:2306.17492*, 2023.

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

[84] Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2023.

[85] Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Alex Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. Learning math reasoning from self-sampled correct and partially-correct solutions. In *The Eleventh International Conference on Learning Representations*, 2022.

[86] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[87] 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. *arXiv preprint arXiv:2301.13688*, 2023.

[88] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Prakash Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards extreme multi-task scaling for transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[89] 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*. OpenReview.net, 2022.

[90] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single QA system. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1896–1907. Association for Computational Linguistics, 2020.

[91] Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. Zeroprompt: Scaling prompt-based pretraining to 1, 000 tasks improves zero-shot generalization. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4235–4252. Association for Computational Linguistics, 2022.

[92] 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 arXiv:2210.11416*, 2022.

[93] 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. *arXiv preprint arXiv:2212.10560*, 2022.

A Appendix

A.1 GSM8k Case Study

Table 3: A comparison case on different scale size models

Instruction: Meredith is a freelance blogger who writes about health topics and submits to clients each day as her permanent job. A blog article takes an average of 4 hours to research and write about. Last week, she wrote 5 articles on Monday and 2/5 times more articles on Tuesday than on Monday. On Wednesday, she wrote twice the number of articles she wrote on Tuesday. Calculate the total number of hours she spent writing articles in the three days.	
Ground Truth Answer: 104	
Note: WizardMath 7B, 13B and 70B answer correctly.	
WizardMath-7B	Step 1: Calculate the number of articles Meredith wrote on Monday. Meredith wrote 5 articles on Monday. Step 2: Calculate the number of articles Meredith wrote on Tuesday. Meredith wrote 2/5 times more articles on Tuesday than on Monday. So, she wrote 2/5 * 5 = 2 articles more on Tuesday than on Monday. Therefore, she wrote 5 + 2 = 7 articles on Tuesday. Step 3: Calculate the total number of articles Meredith wrote in the two days. In the two days, she wrote 5 + 7 = 12 articles. Step 4: Calculate the number of articles Meredith wrote on Wednesday. Meredith wrote twice the number of articles she wrote on Tuesday on Wednesday. So, she wrote 2 * 7 = 14 articles on Wednesday. Step 5: Calculate the total number of articles Meredith wrote in the three days. In the three days, she wrote 12 + 14 = 26 articles. Step 6: Calculate the total number of hours Meredith spent writing articles in the three days. Each article takes 4 hours to research and write. So, she spent 4 hours * 26 = 104 hours writing articles in the three days. Therefore, Meredith spent 104 hours writing articles in the three days. The answer is: 104.
WizardMath-13B	Step 1: Find the number of articles Meredith wrote on Tuesday. We know that Meredith wrote 5 articles on Monday and 2/5 times more articles on Tuesday than on Monday. To find the number of articles she wrote on Tuesday, we can multiply the number of articles she wrote on Monday by 2/5: 5 articles * 2/5 = 2 articles So, Meredith wrote 2 more articles on Tuesday than on Monday. To find the total number of articles she wrote on Tuesday, we can add the 2 articles to the 5 articles she wrote on Monday: 5 articles + 2 articles = 7 articles Step 2: Find the number of articles Meredith wrote on Wednesday. We know that Meredith wrote twice the number of articles she wrote on Tuesday on Wednesday. So, we can multiply the number of articles she wrote on Tuesday by 2: 7 articles * 2 = 14 articles Step 3: Find the total number of articles Meredith wrote in the three days. Now, we can add the number of articles she wrote on each day to find the total number of articles: 5 articles (Monday) + 7 articles (Tuesday) + 14 articles (Wednesday) = 26 articles Step 4: Calculate the total number of hours Meredith spent writing articles in the three days. We know that each article takes an average of 4 hours to research and write about. So, to find the total number of hours Meredith spent writing articles in the three days, we can multiply the total number of articles by the number of hours per article: 26 articles * 4 hours/article = 104 hours

Therefore, Meredith spent a total of 104 hours writing articles in the three days. The answer is: 104.

WizardMath-70B	1. On Monday, Meredith wrote 5 articles. 2. On Tuesday, she wrote 2/5 times more articles than on Monday. To find out how many articles she wrote on Tuesday, we need to multiply the number of articles she wrote on Monday by 2/5. So, 5 articles * 2/5 = 2 articles. Therefore, she wrote 7 articles on Tuesday (5 articles + 2 articles). 3. On Wednesday, she wrote twice the number of articles she wrote on Tuesday. To find out how many articles she wrote on Wednesday, we need to multiply the number of articles she wrote on Tuesday by 2. So, 7 articles * 2 = 14 articles. Now, let's calculate the total number of articles she wrote in the three days: Total articles = Monday's articles + Tuesday's articles + Wednesday's articles Total articles = 5 articles + 7 articles + 14 articles = 26 articles Since each article takes an average of 4 hours to research and write about, we can calculate the total number of hours she spent writing articles in the three days: Total hours = Total articles * Time per article Total hours = 26 articles * 4 hours/article = 104 hours So, Meredith spent a total of 104 hours writing articles in the three days. The answer is: 104.

Table 4: A comparison case on different scale size models

Instruction: A mechanic charges different rates to repair the tires of trucks and cars. For each truck tire that is repaired, the mechanic will charge $60 and for each car tire that is repaired, the mechanic will charge $40. On Thursday, the mechanic repairs 6 truck tires and 4 car tires. On Friday, the mechanic repairs 12 car tries and doesn't repair any truck tires. How much more revenue did the mechanic earn on the day with higher revenue?	

Ground Truth Answer: 40

Note: WizardMath 7B answers incorrectly, while 13B and 70B answer correctly.

WizardMath-7B	
	On Thursday, the mechanic repaired 6 truck tires and 4 car tires. Therefore, the revenue from repairing truck tires on Thursday was 6 * $60 = $360, and the revenue from repairing car tires on Thursday was 4 * $40 = $160. On Friday, the mechanic repaired 12 car tires, but did not repair any truck tires. Therefore, the revenue from repairing car tires on Friday was 12 * $40 = $480.

To calculate the total revenue from repairing truck tires and car tires on Thursday and Friday, we need to add up the revenue from each day. Therefore, the total revenue from repairing truck tires and car tires on Thursday and Friday was $360 + $160 + $480 = $900.

To calculate the revenue from repairing truck tires on Friday, we need to know how many truck tires the mechanic repaired on Friday. However, this information is not given in the question. Therefore, we cannot calculate the revenue from repairing truck tires on Friday.

To calculate the revenue from repairing car tires on Thursday and Friday, we need to add up the revenue from each day. Therefore, the total revenue from repairing car tires on Thursday and Friday was $160 + $480 = $640.

To calculate the difference between the revenue from repairing truck tires and car tires on Thursday and Friday, we need to subtract the revenue from repairing car tires on Thursday and Friday from the total revenue from repairing truck tires and car tires on Thursday and Friday. Therefore, the difference between the revenue from repairing truck tires and car tires on Thursday and Friday was $900 - $640 = $260.

Therefore, the mechanic earned $260 more on the day with higher revenue. The answer is: 260.
The answer is: 260.

WizardMath-13B

Step 1: Calculate the revenue earned on Thursday.
On Thursday, the mechanic repaired 6 truck tires and 4 car tires. The mechanic charges $60 for each truck tire and $40 for each car tire. Therefore, the revenue earned on Thursday can be calculated as follows:
Revenue on Thursday = (Number of truck tires) * (Price per truck tire) + (Number of car tires) * (Price per car tire)
Revenue on Thursday = 6 * $60 + 4 * $40
Revenue on Thursday = $360 + $160
Revenue on Thursday = $520
Step 2: Calculate the revenue earned on Friday.
On Friday, the mechanic repaired 12 car tires and didn't repair any truck tires. Therefore, the revenue earned on Friday can be calculated as follows:
Revenue on Friday = (Number of car tires) * (Price per car tire)
Revenue on Friday = 12 * $40
Revenue on Friday = $480
Step 3: Compare the revenue earned on both days.
Now, we need to compare the revenue earned on Thursday and Friday to find out which day earned more revenue.
Difference in revenue = Revenue on Thursday - Revenue on Friday
Difference in revenue = $520 - $480
Difference in revenue = $40

Therefore, the mechanic earned $40 more in revenue on the day with higher revenue, which is Thursday. The final answer is $40.
The answer is: 40.

WizardMath-70B	Step 1: Calculate the revenue from repairing truck tires on Thursday. The mechanic repaired 6 truck tires on Thursday, and each truck tire repair costs $60. So, the revenue from repairing truck tires on Thursday is 6 * $60 = $360. Step 2: Calculate the revenue from repairing car tires on Thursday. The mechanic repaired 4 car tires on Thursday, and each car tire repair costs $40. So, the revenue from repairing car tires on Thursday is 4 * $40 = $160. Step 3: Calculate the total revenue on Thursday. The total revenue on Thursday is the sum of the revenue from repairing truck tires and car tires. So, the total revenue on Thursday is $360 + $160 = $520. Step 4: Calculate the revenue from repairing car tires on Friday. The mechanic repaired 12 car tires on Friday, and each car tire repair costs $40. So, the revenue from repairing car tires on Friday is 12 * $40 = $480. Step 5: Calculate the total revenue on Friday. The mechanic didn't repair any truck tires on Friday, so the total revenue on Friday is the revenue from repairing car tires. So, the total revenue on Friday is $480. Step 6: Calculate the difference in revenue between the two days. The difference in revenue is the difference between the total revenue on the day with higher revenue and the total revenue on the day with lower revenue. So, the difference in revenue is $520 - $480 = $40. Therefore, the mechanic earned $40 more revenue on the day with higher revenue. The final answer is $40. The answer is: 40.

Instruction: Adrien's total salary was 30 percent higher than Lylah's. Four years later, his salary had increased, and he was earning 40% more than what he was making four years ago. If Adrien's and Lylah's salary increased simultaneously, and Adrien earned $40000 four years ago, calculate the total salary the two were receiving four years later?

Ground Truth Answer: 95200

Note: WizardMath 7B and 13B answer incorrectly, while 70B answers correctly.

WizardMath-7B	
	Step 1: Calculate Adrien's salary four years ago.
	Since Adrien earned $40000 four years ago, his total salary four years ago was $40000.
	Step 2: Calculate the increase in Adrien's salary four years later.
	Four years later, Adrien's salary had increased by 40% of his salary four years ago. Therefore, the increase in Adrien's salary was 40/100*$40000 = $16000.
	Step 3: Calculate Adrien's total salary four years later.
	Adrien's total salary four years later was the sum of his salary four years ago and the increase in his salary. Therefore, Adrien's total salary four years later was $40000+$16000 = $56000.
	Step 4: Calculate the increase in Lylah's salary four years later.
	Since Adrien's total salary was 30% higher than Lylah's, the increase in Lylah's salary four years later was 30/100*$56000 = $16800.
	Step 5: Calculate Lylah's total salary four years later.
	Lylah's total salary four years later was the sum of her salary four years ago and the increase in her salary. Therefore, Lylah's total salary four years later was $56000+$16800 = $72800.
	Step 6: Calculate the total salary the two were receiving four years later.
	The total salary the two were receiving four years later was the sum of Adrien's and Lylah's total salary four years later. Therefore, the total salary the two were receiving four years later was $72800+$56000 = $128800.
	Therefore, the total salary the two were receiving four years later was $128800.
	The answer is: 128800.
WizardMath-13B	
	Step 1: Calculate Adrien's total salary four years ago.
	We know that Adrien's total salary was 30 percent higher than Lylah's. If Adrien was earning $40000 four years ago, then Lylah was earning $40000/30/100 = $12000.
	Step 2: Calculate Adrien's total salary four years later.
	Four years later, Adrien's salary had increased, and he was earning 40% more than what he was making four years ago. Therefore, Adrien's total salary four years later is $40000 + $40000*40/100 = $56000.
	Step 3: Calculate Lylah's total salary four years later.
	If Adrien's salary increased by 40%, then Lylah's salary also increased by 40%. Therefore, Lylah's total salary four years later is $12000 + $12000*40/100 = $16800.
	Step 4: Calculate the total salary the two were receiving four years later.
	The total salary the two were receiving four years later is the sum of Adrien's and Lylah's total salaries. Therefore, the total salary the two were receiving four years later is $56000 + $16800 = $72800.
	Therefore, the total salary the two were receiving four years later is $72800.
	The answer is: 72800.
WizardMath-70B	
	Step 1: Determine Lylah's salary four years ago.
	Since Adrien's total salary was 30 percent higher than Lylah's, and Adrien earned $40000 four years ago, we can find Lylah's salary by subtracting 30% of $40000 from Adrien's salary.
	Lylah's salary = $40000 - (30/100)*$40000 = $40000 - $12000 = $28000
	Step 2: Calculate Adrien's salary four years later.

Four years later, Adrien's salary had increased by 40% of what he was making four years ago. So, we need to add 40% of $40000 to his salary from four years ago.

Adrien's salary four years later = $40000 + (40/100)*$40000 = $40000 + $16000 = $56000

Step 3: Calculate Lylah's salary four years later.

Since their salaries increased simultaneously, Lylah's salary also increased by 40% of what she was making four years ago. So, we need to add 40% of $28000 to her salary from four years ago.

Lylah's salary four years later = $28000 + (40/100)*$28000 = $28000 + $11200 = $39200

Step 4: Calculate the total salary the two were receiving four years later.

Now, we just need to add Adrien's and Lylah's salaries from four years later to find their total salary.

Total salary = Adrien's salary four years later + Lylah's salary four years later = $56000 + $39200 = $95200

So, the total salary the two were receiving four years later was $95200.

The answer is: 95200.

A.2 MATH Case Study

Table 6: A comparison case on different scale size models

Instruction: The midpoint of the line segment between (x, y) and $(-9, 1)$ is $(3, -5)$. Find (x, y).	

Ground Truth Answer: (15,-11).

Note: WizardMath 7B, 13B and 70B answer correctly.

WizardMath-7B

Step 1: Understand the problem
We are given the midpoint of the line segment between (x, y) and $(-9, 1)$ as $(3, -5)$. We need to find the coordinates of (x, y).
Step 2: Recall the formula for the midpoint
The midpoint of a line segment with endpoints (x_1, y_1) and (x_2, y_2) can be found using the midpoint formula:

$$\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$$

Step 3: Apply the midpoint formula
In this case, the endpoints are (x, y) and $(-9, 1)$. So, we can plug these values into the midpoint formula:

$$\left(\frac{x + (-9)}{2}, \frac{y + 1}{2} \right)$$

Step 4: Simplify the expression
Now, we can simplify the expression by performing the arithmetic operations:

$$\left(\frac{x - 9}{2}, \frac{y + 1}{2} \right)$$

Step 5: Compare the coordinates
We are given that the midpoint is $(3, -5)$. So, we can compare the coordinates of the simplified expression with the given midpoint:

$$\frac{x - 9}{2} = 3$$

$$\frac{y + 1}{2} = -5$$

Step 6: Solve for the coordinates
Now, we can solve for the coordinates of (x, y) by multiplying both sides of the equations by 2:

$$x - 9 = 6$$

$$y + 1 = -10$$

Step 7: Write the final answer
Now, we can write the final answer by adding the results of the equations:

$$x = 6 + 9$$

22

$$y = -10 - 1$$

$$x = 15$$

$$y = -11$$

So, the coordinates of (x, y) are $\boxed{(15, -11)}$.
The answer is: (15,-11).

WizardMath-13B	

Step 1: Understand the problem
We are given that the midpoint of the line segment between (x, y) and $(-9, 1)$ is $(3, -5)$. We need to find the coordinates of the point (x, y).
Step 2: Recall the midpoint formula
The midpoint of a line segment with endpoints (x_1, y_1) and (x_2, y_2) can be found using the midpoint formula:
Midpoint = $\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$
Step 3: Apply the midpoint formula
In our case, the endpoints are (x, y) and $(-9, 1)$. So, we can plug these values into the midpoint formula:
Midpoint = $\left(\frac{x + (-9)}{2}, \frac{y + 1}{2} \right)$
Step 4: Simplify the midpoint
Now, we can simplify the midpoint by performing the arithmetic operations:
Midpoint = $\left(\frac{x - 9}{2}, \frac{y + 1}{2} \right)$
Step 5: Set up an equation
Since the midpoint is given as $(3, -5)$, we can set up an equation using the simplified midpoint formula:
$\frac{x-9}{2} = 3$ and $\frac{y+1}{2} = -5$
Step 6: Solve the equations
Now, we need to solve these two equations to find the values of x and y. Let's start with the first equation:
$\frac{x-9}{2} = 3$
Multiply both sides by 2 to get rid of the fraction:
$x - 9 = 6$
Now, add 9 to both sides:
$x = 15$
Next, let's solve the second equation:
$\frac{y+1}{2} = -5$
Multiply both sides by 2 to get rid of the fraction:
$y + 1 = -10$
Now, subtract 1 from both sides:
$y = -11$
Step 7: Write the final answer
We have found the values of x and y. So, the coordinates of the point (x, y) are $(15, -11)$. Therefore, the final answer is:
$(x, y) = \boxed{(15, -11)}$
The answer is: (15,-11).

WizardMath-70B	

Step 1: Understand the problem
We are given the midpoint of a line segment and the coordinates of one of its endpoints. We need to find the coordinates of the other endpoint, which is (x, y).

Step 2: Recall the midpoint formula

The midpoint of a line segment with endpoints (x_1, y_1) and (x_2, y_2) is given by the formula:

Midpoint = $\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$

Step 3: Plug in the given information

We know the midpoint is $(3, -5)$ and one endpoint is $(-9, 1)$. So, we can plug these values into the midpoint formula:

$(3, -5) = \left(\frac{-9 + x}{2}, \frac{1 + y}{2} \right)$

Step 4: Solve for x and y

Now, we need to solve the system of equations:

1) $\frac{-9 + x}{2} = 3$

2) $\frac{1 + y}{2} = -5$

Step 5: Solve equation 1 for x

Multiply both sides by 2:

$-9 + x = 6$

Add 9 to both sides:

$x = 15$

Step 6: Solve equation 2 for y

Multiply both sides by 2:

$1 + y = -10$

Subtract 1 from both sides:

$y = -11$

Step 7: Write the final answer

Now that we have found the values of x and y, we can write the final answer as:

$(x, y) = (15, -11)$

So, the coordinates of the other endpoint are $(15, -11)$.

The answer is: (15,-11).

Table 7: A comparison case on different scale size models

Instruction: Evaluate $\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80°$.	

Ground Truth Answer: 1

Note: WizardMath 7B answers incorrectly, while 13B and 70B answer correctly.

WizardMath-7B

First, we can rewrite the given expression using the identity $\tan(90° - x) = \frac{1}{\tan x}$:

$$\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80° = \frac{1}{\tan 80°}$$

Now, we can use the double-angle formula for tangent, which is $\tan 2x = \frac{2\tan x}{1-\tan^2 x}$:

$$\frac{1}{\tan 80°} = \frac{2}{\tan 160°} = \frac{2}{\frac{1-\tan^2 80°}{2\tan 80°}} = \frac{1-\tan^2 80°}{2\tan 80°}$$

Next, we can use the double-angle formula for tangent again:

$$\frac{1-\tan^2 80°}{2\tan 80°} = \frac{1 - \frac{1}{2}\tan^2 80°}{2\tan 80°} = \frac{1 - \frac{1}{2}\tan^2 80°}{2\tan 80°} \cdot \frac{2\tan 80°}{1 - \tan^2 80°}$$

Now, we can multiply the fractions:

$$\frac{1 - \frac{1}{2}\tan^2 80°}{2\tan 80°} \cdot \frac{2\tan 80°}{1 - \tan^2 80°} = \frac{1 - \frac{1}{2}\tan^2 80°}{1 - \tan^2 80°}$$

Finally, we can simplify the expression:

$$\frac{1 - \frac{1}{2}\tan^2 80°}{1 - \tan^2 80°} = 1 - \frac{1}{2}\tan^2 80°$$

So, the rewritten answer is:

$$\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80° = \boxed{1 - \frac{1}{2}\tan^2 80°}$$

The answer is: $1 - \frac{1}{2}\tan^2 80°$.

WizardMath-13B

Step 1: Recall the angle addition formula for tangent:
$\tan(A + B) = \frac{\tan A + \tan B}{1 - \tan A \tan B}$
Step 2: Use the formula to find $\tan 110°$:
$\tan 110° = \tan(10° + 100°) = \frac{\tan 10° + \tan 100°}{1 - \tan 10° \tan 100°} = \frac{1+0}{1-1\cdot 0} = 1$
Step 3: Observe that $\tan 110°$ is the only term in the given expression that is not equal to 0.
Step 4: Multiply both sides of the equation by $\tan 110°$:
$\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80° \tan 110° = 1 \cdot \tan 110°$
Step 5: Simplify the equation:
$\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80° = \tan 110°$
Step 6: Since $\tan 110° = 1$, the final answer is:
$\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80° = \boxed{1}$.
The answer is: 1.

WizardMath-70B

Step 1: Recall the definitions of tangent and cotangent.

We know that $\tan x = \frac{\sin x}{\cos x}$ and $\cot x = \frac{\cos x}{\sin x}$.

Step 2: Find the tangent and cotangent of each angle.

We need to find the tangent and cotangent of each angle in the given expression:

$\tan 10° = \frac{\sin 10°}{\cos 10°}$

$\tan 20° = \frac{\sin 20°}{\cos 20°}$

$\tan 30° = \frac{\sin 30°}{\cos 30°}$

$\tan 40° = \frac{\sin 40°}{\cos 40°}$

$\tan 50° = \frac{\sin 50°}{\cos 50°}$

$\tan 60° = \frac{\sin 60°}{\cos 60°}$

$\tan 70° = \frac{\sin 70°}{\cos 70°}$

$\tan 80° = \frac{\sin 80°}{\cos 80°}$

Step 3: Find the cotangent of each angle.

Now, we need to find the cotangent of each angle:

$\cot 10° = \frac{\cos 10°}{\sin 10°}$

$\cot 20° = \frac{\cos 20°}{\sin 20°}$

$\cot 30° = \frac{\cos 30°}{\sin 30°}$

$\cot 40° = \frac{\cos 40°}{\sin 40°}$

$\cot 50° = \frac{\cos 50°}{\sin 50°}$

$\cot 60° = \frac{\cos 60°}{\sin 60°}$

$\cot 70° = \frac{\cos 70°}{\sin 70°}$

$\cot 80° = \frac{\cos 80°}{\sin 80°}$

Step 4: Use the identity $\tan x \cot x = 1$.

We know that $\tan x \cot x = 1$. So, we can rewrite the given expression as:

$\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80° = \frac{\sin 10°}{\cos 10°} \cdot \frac{\sin 20°}{\cos 20°} \cdot$
$\frac{\sin 30°}{\cos 30°} \cdot \frac{\sin 40°}{\cos 40°} \cdot \frac{\sin 50°}{\cos 50°} \cdot \frac{\sin 60°}{\cos 60°} \cdot \frac{\sin 70°}{\cos 70°} \cdot \frac{\sin 80°}{\cos 80°}$

Step 5: Simplify the expression.

Now, we can simplify the expression by canceling out the terms:

$\frac{\sin 10° \sin 20° \sin 30° \sin 40° \sin 50° \sin 60° \sin 70° \sin 80°}{\cos 10° \cos 20° \cos 30° \cos 40° \cos 50° \cos 60° \cos 70° \cos 80°}$

Step 6: Calculate the final answer.

Now, we can calculate the final answer:

$\tan 10° \tan 20° \tan 30° \tan 40° \tan 50° \tan 60° \tan 70° \tan 80° = \boxed{1}$

So, the final answer is $\boxed{1}$.

The answer is: 1.

Instruction: Let $f(x)$ be an odd function, and let $g(x)$ be an even function. Is $f(f(g(f(g(f(x))))))$ even, odd, or neither?

Enter "odd", "even", or "neither".

Ground Truth Answer: even

Note: WizardMath 7B and 13B answer incorrectly, while 70B answers correctly.

WizardMath-7B	
	1. We know that $f(x)$ is an odd function, which means that $f(-x) = -f(x)$ for all x in its domain.

1. We know that $f(x)$ is an odd function, which means that $f(-x) = -f(x)$ for all x in its domain.
2. We also know that $g(x)$ is an even function, which means that $g(-x) = g(x)$ for all x in its domain.
3. Now, let's find the value of $f(g(-x))$ and $f(-g(x))$. Since $g(-x) = g(x)$ and $f(-x) = -f(x)$, we can conclude that:
$f(g(-x)) = f(-g(x))$
4. Next, let's find the value of $f(f(g(-x)))$ and $f(f(-g(x)))$. Since $f(-x) = -f(x)$ and $f(g(-x)) = f(g(x))$, we can conclude that:
$f(f(g(-x))) = f(f(-g(x)))$
5. Now, let's find the value of $f(f(f(g(-x))))$ and $f(f(f(-g(x))))$. Since $f(f(-x)) = -f(f(x))$ and $f(f(g(-x))) = f(f(-g(x)))$, we can conclude that:
$f(f(f(g(-x)))) = -f(f(f(-g(x))))$
6. Finally, let's find the value of $f(f(f(f(g(-x)))))$ and $f(f(f(f(-g(x)))))$. Since $f(f(-x)) = -f(f(x))$ and $f(f(f(g(-x)))) = -f(f(f(-g(x))))$, we can conclude that:
$f(f(f(f(g(-x))))) = -f(f(f(f(-g(x)))))$
Therefore, the final answer is $\boxed{\text{odd}}$.
The answer is: odd.

WizardMath-13B	

1. We know that $f(x)$ is an odd function, which means that $f(-x) = -f(x)$.
2. We also know that $g(x)$ is an even function, which means that $g(-x) = g(x)$.
Now, let's analyze the given expression: $f(f(g(f(g(f(x))))))$.
3. Replace x with $-x$ in the expression: $f(f(g(f(g(f(-x))))))$. Since f is odd, we have $f(-x) = -f(x)$, so the expression becomes $-f(f(g(f(g(-x)))))$.
4. Since g is even, we have $g(-x) = g(x)$. So, we can rewrite the expression as $-f(f(g(f(g(x)))))$.
5. Now, we have another instance of f and g. Since f is odd and g is even, their product is an odd function. So, we can rewrite the expression as $-[f(g(f(x)))]$.
6. We have another instance of f and g. Since f is odd and g is even, their product is an odd function. So, we can rewrite the expression as $-f(g(f(x)))$.
7. Now, we have another instance of f. Since f is odd, we have $f(-x) = -f(x)$. So, we can rewrite the expression as $-f(f(x))$.
8. Finally, we have another instance of f. Since f is odd, we have $f(-x) = -f(x)$. So, we can rewrite the expression as $-f(f(x))$.
Now, we can see that the expression is equal to $-f(f(x))$, which is an odd function.
Therefore, the answer is $\boxed{\text{odd}}$.
The answer is: odd.

WizardMath-70B	

1. We are given that $f(x)$ is an odd function and $g(x)$ is an even function.
2. Recall the definitions of odd and even functions:
- A function $f(x)$ is odd if $f(-x) = -f(x)$ for all x in its domain.
- A function $g(x)$ is even if $g(-x) = g(x)$ for all x in its domain.
3. Now, let's analyze the given function: $f(f(g(f(g(f(x))))))$.
4. We can rewrite this function as: $f(f(g(f(g(f(x)))))) = h(x)$, where $h(x) = f(f(g(f(g(f(x))))))$.
5. Now, let's find $h(-x)$:

- $h(-x) = f(f(g(f(g(f(-x))))))$.
6. Since $g(x)$ is even, we have $g(-x) = g(x)$. So, we can rewrite the expression as:
- $h(-x) = f(f(g(f(g(f(x))))))$.
7. Now, we can see that $h(-x) = h(x)$. This means that the function $h(x)$ satisfies the definition of an even function.
8. Therefore, the given function $f(f(g(f(g(f(x))))))$ is $\boxed{\text{even}}$.
The answer is: even.