

Mathematical Reasoning Through LLM Finetuning

Stanford CS224N Custom Project

Yash Mehta

Department of Computer Science
Stanford University
yashm@stanford.edu

Karthik Seetharaman

Department of Computer Science
Stanford University
kvseet17@stanford.edu

Abstract

Using LLMs to generate step-by-step solutions to math problems can be extremely difficult due to the logical reasoning required. In this paper, we explore several different methods to finetune LLMs for this task. We first implement Multi-Task Sequential Fine-Tuning (MTSFT) from Liu et al. (2023), which works off the observation that training language models sequentially as solution evaluators then as solution generators improves their performance. We then implement our own novel method, Logic-Enhanced Sequential Fine-Tuning (LESFT), which combines MTSFT with additional fine-tuning on the LogiQA dataset of multiple-choice deductive reasoning questions to improve the model's general logical reasoning abilities. We use the open-source Mistral-7B model and measure Pass@1 and Maj1@64 accuracy on the MATH dataset. When compared to a baseline model with no fine-tuning, we notice accuracy improvements of roughly 10% in Pass@1 accuracy and roughly 15% in Maj1@64 accuracy for both MTSFT and LESFT trained models. Upon further analysis of specific problems, we notice that fine-tuning helps the model correctly execute the logic for problems it could not initially. However, the task still remains quite difficult - when using an LESFT trained model and generating 64 solutions for each problem, upwards of 50% of problems have less than 5% of generated solutions correct.

1 Key Information to include

- Mentor: Tathagat Verma
- We do not have any external collaborators.
- We are not sharing this project with another class.
- Collaboration: Yash Mehta worked on implementing LESFT and helped write the paper. Karthik Seetharaman worked on implementing MTSFT and helped write the paper.

2 Introduction

Solving mathematical problems requires creativity, reasoning, and calculation; for this reason, developing statistical models with this ability is a worthy goal. Currently, LLMs are very inaccurate at solving challenging math problems, such as those found in high school math competitions like the AMC and AIME. In particular, if the model only attempts a highly complex, multi-step problem once, there is a high probability of error since an error in any one step may lead to an incorrect answer. As Liu et al. (2023) aptly notes, the pre-trained PaLM 2-L can achieve only around 33.4% accuracy on the MATH dataset with greedy decoding, yet achieves 79.4% accuracy when sampling 64 solutions (measured as the probability that at least one of the 64 solutions is correct).

Math reasoning is a weakness for LLMs on multiple fronts. For example, they deliver different solutions when presented with the same problem written differently, suggesting an overreliance

on statistical probabilities. They also commonly provide different final answers on the same exact problem over multiple trials Ahn et al. (2024), making consistency an issue.

In this paper, we aim to explore different finetuning methods for LLMs to improve their ability to solve complex math problems. Thus far, Liu et al. (2023) has provided a promising finetuning framework called Multi-Task Sequential Fine-Tuning (MTSFT) for the MATH dataset involving training the LLM to both evaluate and generate solutions, although there is no public implementation. Furthermore, as stated earlier, logical reasoning is paramount in challenging math problems and current models are lacking in this area. We also use the insight that past works like those of Mishra et al. (2022) and Yue et al. (2023) demonstrate that certain models trained on external dataset(s) surprisingly fare better in-domain. We thus have two goals here: first, to implement the MTSFT method and second, to implement our own method that attempts to improve a language model’s capacity for logic.

In pursuit of the second goal, we first conduct fine-tuning on the multiple choice logic dataset LogiQA in a similar fashion to Supervised Solution Fine-Tuning (SSFT) from Liu et al. (2023). We then implement and execute MTSFT, which is comprised of two methods: SSFT on the MATH dataset and Solution-Cluster Re-Ranking (SCR), which trains the model to better evaluate the correctness of solutions. We call this entire pipeline Logic-Enhanced Sequential Fine-Tuning, or LESFT. We compare all these fine-tuning methods to a baseline pre-trained model by measuring performance on the MATH dataset, a dataset of complex high-school math competition problems.

3 Related Work

Crux of the Problem Since large language models became mainstream, there has been a large line of work on using them to solve mathematics problems, which is a particularly difficult task. In particular, Zhang et al. (2022) found that understanding how to reason logically is a weakness for data-driven language models, which is essential to solving math problems. That said, Wei et al. (2022) found that prompting GPT-3 to generate a chain of thought, a series of intermediate reasoning steps, significantly improved its ability for complex reasoning. By introducing the MATH dataset of 12,500 challenging competition-level problems, Hendrycks et al. (2021) facilitated research into more complex mathematical problems. At the time, even large Transformer models fared poorly on this dataset. This launched an ongoing stream of research into trying to improve the mathematical accuracy of these models, and a host of methods to do this have already been explored.

Factors Affecting Arithmetic A comprehensive evaluation by Yuan et al. (2023) shows multiple factors that impact the arithmetic skills of LLMs. Specialized tokenization, for example, appears to improve arithmetic proficiency. It shows that including LaTeX or code in pre-training data is beneficial, at least for Galactica Taylor et al. (2022), which is an LLM specializing in reasoning about scientific knowledge. The model Galactica performs better at arithmetic than OpenAI’s GPT-3 Code-DaVinci, which beats the former at reasoning, suggesting a distinction in arithmetic and reasoning skills. Additionally, model size is also linked with better arithmetic capability, but only to a point: Ahn et al. (2024) points out that Galactica achieves similar performance with both 30B and 120B parameters.

Fine-Tuning On MATH Lightman et al. (2023) compares process-supervised reward models (PRMs) and outcome-supervised reward models (ORMs). Reward Model training is an advanced Reinforcement Learning technique, and PRMs reward each step of the process of a reasoning solution while ORM only reward based on outcome. Testing PRMs and ORM with a majority voting baseline on the aforementioned MATH dataset, they conclude that PRMs outperform both and provide a PRM800K dataset consisting of step-level correctness labels for model-generated solutions to problems from the MATH dataset. Liu et al. (2023) notably uses a finetuning pipeline consisting of supervised step-by-step solution finetuning (SSFT) and solution cluster reranking (SCR). This pipeline, called Multi-Task Sequential Fine-Tuning, is built on the insight that LLMs are poor at generating the correct solution in one attempt but relatively strong at evaluating solutions. Researchers in Liu et al. (2023) also find that using PRM800K dataset for SSFT improves performance on MATH compared to simply using MATH.

Fine-Tuning On Related Math Datasets Cross-dataset generalization is a major point brought up thanks to Mishra et al. (2022) and Yue et al. (2023), who introduced composites of existing math datasets LILA and MathInstruct, respectively. LILA is a unified mathematical reasoning benchmark consisting of 23 diverse tasks, where most problems are accompanied by explainable solutions in the form of Python programs, and MathInstruct is an instruction tuning dataset compiled from 13 math datasets with intermediate rationales. Mishra et al. (2022) finds that multi-tasking models tend to have a better in-domain and out-of-domain performance than single-task models. Yue et al. (2023) introduces a series of MAMmoTH models trained on MathInstruct, each of whose performance surpasses that of any other open-source model of corresponding size on the MATH dataset, especially MAMmoTH-7B, which reaches 33% on MATH compared to the next best WizardMath’s 10%. This suggests that training on related datasets benefits performance both on generalized tasks and on the original dataset.

Fine-Tuning On Variations of the Same Dataset To develop the intricate reasoning required for multi-step math problems, Yu et al. (2024) developed a method termed “question bootstrapping,” involving rewriting the question in multiple ways, resulting in a new dataset, MetaMathQA, which contains questions rewritten in many different fashions. They finetune LLaMA-2 models on MetaMathQA and find that they outperform current standards on datasets like MATH. This improvement may result from decreasing the model’s overfitting to the specific phrasing and placement of words in the question. Indeed, Chen et al. (2023) finds that using parallel corpora to employ Supervised Fine-Tuning on math across multiple languages not only improves model performance multilingually but also monolingually. Introducing the first multilingual math reasoning dataset MGSM8KInstruct, its corresponding model MathOctopus-13B marginally outperforms ChatGPT in few-shot scenarios.

4 Approach

Our two chief aims are to replicate the methods from Liu et al. (2023), for which there is no public implementation, and to introduce our own novel method which prepends fine-tuning specifically on deductive logical reasoning problems to *Multi-Task Sequential Fine-Tuning* (MTSFT) from Liu et al. (2023). For a baseline, we use a 4-shot prompt which provides 4 problems with example correct solutions and then asks the model to generate a solution to a new problem in the same style. This same prompt is used for testing all fine-tuning methods (see Appendix for full 4-shot prompt). We compare the performance of our combined method, *Logic-Enhanced Sequential Fine-Tuning* (LESFT), to the baseline, testing on three subjects within the MATH dataset.

4.1 Multi-Task Sequential Fine-Tuning (MTSFT)

MTSFT combines two fine-tuning methods described below, SSFT and SCR. In particular, we first fine-tune the model as a solution generator using SSFT, then as a solution evaluator using SCR, then again as a solution generator using SSFT.

Supervised Solution Fine-Tuning (SSFT) involves finetuning a language model M to, given a problem P , generate a step-by-step solution S and an answer A . To do this, we create a dataset where S and A are concatenated into a single text sequence X , and we use standard cross-entropy loss to finetune the model. In particular, if

$$p_M(X|P) = \prod_i p_M(x_i|X_{0,\dots,i-1}, P)$$

is the probability distribution given by M (where x_i is the i -th token of X), then we fit to the loss

$$L_{\text{mle}} = -\log p_M(X|P).$$

Note that ground-truth solutions are provided by the MATH dataset Hendrycks et al. (2021).

Solution-Cluster Re-Ranking (SCR) aims to finetune the model as an evaluator. To do this, given a problem P and candidate solution \tilde{X} (which consists of both a solution and answer), define $T(P, \tilde{X})$ as the prompt "Here is a math problem: [P]. Here is a candidate solution: [\tilde{X}]. The above candidate solution is:" and define

$$p_{\text{cls}}(\text{right}|\tilde{X}, P) = p_M(\text{right}|T(P, \tilde{X}))$$

$$p_{\text{cls}}(\text{wrong}|\tilde{X}, P) = p_M(\text{wrong}|T(P, \tilde{X}))$$

Essentially, the model’s probability of predicting the next token as "right" is interpreted as the probability the solution is right. Then, the solution score is the normalized probability

$$S_{\text{cls}}(\tilde{X}|P) = \frac{p_{\text{cls}}(\text{right}|\tilde{X}, P)}{p_{\text{cls}}(\text{right}|\tilde{X}, P) + p_{\text{cls}}(\text{wrong}|\tilde{X}, P)}.$$

The loss for this method is standard binary cross-entropy loss:

$$L_{\text{cls-xent}} = -\mathbb{1}_{\tilde{X} \text{ is right}}(\tilde{X}) \log p_{\text{cls}}(\text{right}|\tilde{X}, P) + \mathbb{1}_{\{\tilde{X} \text{ is wrong}\}}(\tilde{X}) \log p_{\text{cls}}(\text{wrong}|\tilde{X}, P).$$

(Note that, in Liu et al. (2023), the authors use the tokens "correct" and "incorrect" instead of "right" and "wrong." As we are using a different LLM than them, we found, through experimentation, that using the tokens "right" and "wrong" resulted in more sensible probabilities while still retaining the same fundamental idea. Hence, we proceed with this slight reformulation of SCR.)

4.2 Logic-Enhanced Sequential Fine-Tuning (LESFT)

We capitalize on the insight that one of the chief difficulties of solving math problems is attempting to perform logical reasoning. We develop and test a new method called *Logic-Enhanced Sequential Fine-Tuning* (LESFT), which involves supervised finetuning on a dataset of logical reasoning questions followed by MTSFT from Liu et al. (2023).

The dataset we choose to use is LogiQA Liu et al. (2020). Section 5.1 further elaborates on the structure of this dataset, but note for now that the dataset consists of multiple-choice questions with four choices labeled A through D, each of which is a sentence. Similar to SCR, we prompt the language model M with the question Q and interpret the model-predicted probability of producing the text sequence in choice A as the probability answer A is correct. More formally,

$$p_{\text{logic}}(A|Q) = p_M(\tilde{A}|Q),$$

where \tilde{A} is the text associated with answer choice A. We define similar expressions for choices B, C, and D. The score of choice A is the normalized probability

$$S_{\text{logic}}(A|Q) = \frac{p_{\text{logic}}(A|Q)}{p_{\text{logic}}(A|Q) + p_{\text{logic}}(B|Q) + p_{\text{logic}}(C|Q) + p_{\text{logic}}(D|Q)},$$

and similarly for choices B, C, and D. Then, we fit to the standard cross-entropy loss

$$L_{\text{logic-xent}} = - \sum_{i \in \{A, B, C, D\}} p_i \log p_{\text{logic}}(i|Q),$$

where p_i is the one-hot vector corresponding to the correct answer.

LESFT proceeds by first fine-tuning the language model based on the above paradigm and then performing MTSFT.

4.3 Determining Maj1@64 Accuracy

As further described in the evaluation section, in addition to measuring the Pass@1 accuracy of the model (simply the accuracy of the model’s generated solution using greedy decoding), we measure the Maj1@64 accuracy, which is the accuracy of the most common solution when 64 solutions are generated by the language model and clustered by similarity. To calculate the similarity score between two candidate solutions S_1 and S_2 , we calculate the Levenshtein distance between S_1 and S_2 and divide by the length of the longer string to give a score in $[0, 1]$ such that a score of 0 indicates two identical strings. When clustering potential solutions, two solutions are placed in the same cluster if their similarity score is at most 5% and they achieve the same answer; this hyperparameter was chosen through experimentation.

5 Experiments

5.1 Data

5.1.1 MATH

We use two datasets in this project. The first is the MATH dataset, from Hendrycks et al. (2021). This dataset consists of 12,500 high-school competition-level problems in many different areas of math, such as algebra, number theory, and combinatorics. Each dataset entry consists of a problem and a correct solution; a few examples are shown in Figure 1. This dataset, which comes with premade train-test splits, is used for all evaluation.

MATH Dataset (Ours)

Problem: Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?

Solution: There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.

Problem: If $\sum_{n=0}^{\infty} \cos^{2n} \theta = 5$, what is $\cos 2\theta$?

Solution: This geometric series is $1 + \cos^2 \theta + \cos^4 \theta + \dots = \frac{1}{1 - \cos^2 \theta} = 5$. Hence, $\cos^2 \theta = \frac{4}{5}$. Then $\cos 2\theta = 2 \cos^2 \theta - 1 = \boxed{\frac{3}{5}}$.

Problem: The equation $x^2 + 2x = i$ has two complex solutions. Determine the product of their real parts.

Solution: Complete the square by adding 1 to each side. Then $(x + 1)^2 = 1 + i = e^{\frac{\pi}{4}} \sqrt{2}$, so $x + 1 = \pm e^{\frac{\pi}{8}} \sqrt{2}$. The desired product is then $(-1 + \cos(\frac{\pi}{8}) \sqrt{2})(-1 - \cos(\frac{\pi}{8}) \sqrt{2}) = 1 - \cos^2(\frac{\pi}{8}) \sqrt{2} = 1 - \frac{(1 + \cos(\frac{\pi}{4}))}{2} \sqrt{2} = \boxed{\frac{1 - \sqrt{2}}{2}}$.

Figure 1: Examples From The MATH Dataset.

The size of each subsection of the dataset is given in Table 1.

MATH Dataset	Train	Test
Algebra	1,744	1,187
Counting & Probability	771	474
Geometry	870	479
Intermediate Algebra	1,295	903
Number Theory	869	540
Prealgebra	1,205	871
Precalculus	746	546

Table 1: MATH Dataset Splits.

5.1.2 LogiQA

The second dataset, which is used for finetuning in LESFT, is the LogiQA dataset from Liu et al. (2020). This dataset consists of 8,678 question-and-answer instances of multiple types of deductive reasoning. There are 7,376 instances in the train set and 651 in each of the eval and test sets. Figure 2 shows examples of the LogiQA dataset, which are multiple choice logical reasoning questions.

5.2 Evaluation Method

In general, we evaluate the correctness of a generated solution by just examining if the generated answer is correct. To do this, we implemented a custom parser that can extract answers from the model-generated solution in LaTeX. This parser uses regular expressions to search for certain commands in the LaTeX-generated solution under the assumption the answer is boxed ($\boxed{\quad}$), which is always the case due to the structure of the 4-shot prompt used for testing.

Although correctness of answer does not necessarily correspond to correctness of solution, given the relatively high complexity of problems in the MATH dataset, the probability of getting the correct

<p>P1: David, Jack and Mark are colleagues in a company. David supervises Jack, and Jack supervises Mark. David gets more salary than Jack.</p> <p>Q: What can be inferred from the above statements?</p> <p>A. Jack gets more salary than Mark. B. David gets the same salary as Mark. C. One employee supervises another who gets more salary than himself. ✓ D. One employee supervises another who gets less salary than himself.</p> <p>P2: Our factory has multiple dormitory areas and workshops. None of the employees who live in dormitory area A are textile workers. We conclude that some employees working in workshop B do not live in dormitory area A.</p> <p>Q: What may be the missing premise of the above argument?</p> <p>A. Some textile workers do not work in workshop B. B. Some employees working in workshop B are not textile workers. ✓ C. Some textile workers work in workshop B. D. Some employees living in dormitory area A work in the workshop B.</p>
--

Figure 2: Examples From The LogiQA Dataset.

answer with wrong reasoning is sufficiently low such that this is a good proxy. Other authors like Lightman et al. (2023) use novel methods to determine the mathematical equivalence of two solutions, but these methods are not public and redefining them is outside the scope of this project.

For each fine-tuning method, we report the Pass@1 and Maj1@64 accuracy on the algebra, combinatorics, and number theory portions of the MATH dataset. We separate by subject during training as different subjects each have their own distinct methods of solving problems, so it does not make sense to finetune on all subjects together. The Pass@1 accuracy is defined as the accuracy of the first solution generated by the language model, while the Maj1@64 accuracy is defined as the accuracy of the solution in the solution cluster of highest size when 64 solutions are generated by the language model.

5.3 Experimental Details

We use the open-source Mistral-7B LLM Jiang et al. (2023) as our base model before finetuning - the pretrained model and tokenizer are available on HuggingFace. All of our finetuning loops use a learning rate of $5 \cdot 10^{-5}$, 20 epochs, and a batch size of 5. As is standard, we pad all tensors in finetuning with a special padding token. We use an AdamW optimizer.

5.4 Results

We report the Pass@1 and Maj1@64 accuracy of each fine-tuning method on each of the algebra, number theory, and combinatorics sections of the MATH dataset in Table 2.

Method	Algebra		Number Theory		Combinatorics	
	Pass@1	Maj1@64	Pass@1	Maj1@64	Pass@1	Maj1@64
Baseline	22.1%	31.6%	23.7%	32.4%	20.5%	31.7%
SSFT	26.5%	32.3%	27.4%	34.2%	24.8%	33.9%
SCR	27.4%	33.6%	26.7%	36.8%	28.0%	35.2%
MTSFT	31.5%	44.2%	33.6%	44.3%	31.9%	43.4%
LESFT	33.6%	45.3%	34.5%	44.7%	35.0%	47.6%

Table 2: Accuracies of the different fine-tuning methods.

6 Analysis

6.1 General Observations

Examining Table 2, we see that, in all cases, the Maj1@64 accuracy surpasses the Pass@1 accuracy, which makes sense: if 64 solutions are generated and the majority solution is chosen, there is more room to correct for error than if just one solution is generated.

We see that SSFT and SCR both have incremental accuracy improvements over the baseline model with no fine-tuning. However, combining both methods as in MTSFT leads to a sizeable performance improvement, mirroring the results in Liu et al. (2023). The exact accuracies do not match, but this is to be expected due to different experimental parameters as well as the fact that we used the Mistral-7B model as opposed to PaLM-2. We also only compared answers as opposed to the full solutions - this could potentially inflate accuracies

When adding finetuning with the LogiQA dataset, as predicted, we see performance improvements, although marginal. The biggest improvement is in the combinatorics subportion of the dataset - we suspect that this is because combinatorics problems contain reasoning most similar to the reasoning found in the LogiQA dataset, as well as the least amount of esoteric theory, making improvements in logical reasoning the most notable here.

6.2 Closer Look At One Problem

We observe that, in general, fine-tuning the model leads to less logical errors in mathematics. In some cases, the baseline model had the correct approach to a problem, but was unable to execute it properly, leading to an incorrect answer. For example, consider the following problem from the algebra dataset and the responses from both the baseline model and finetuned model:

Problem: Find the fraction that equals $0.72\bar{6}$.

Baseline Response: We can write $0.72\bar{6}$ as the sum of an infinite geometric series:

$$0.72\bar{6} = 0.72 + 0.006 + 0.0006 + \dots$$

The first term is 0.72, and the common ratio is 0.001. We can find the sum of this infinite geometric series using the formula:

$$S = \frac{a}{1 - r}$$

where a is the first term and r is the common ratio. Plugging in the values, we get:

$$S = \frac{0.72}{1 - 0.001} = \frac{0.72}{0.999} \approx 0.720721$$

Therefore, the fraction that equals $0.72\bar{6}$ is approximately $\frac{720721}{1000000}$.

Finetuned Response:

We can write $0.72\bar{6}$ as the sum of an infinite geometric series:

$$0.72\bar{6} = 0.72 + 0.006 + 0.0006 + \dots$$

After the first term, which is 0.72, this is an infinite geometric series where the first term is 0.006 and the common ratio is 0.1. We can find the sum of this infinite geometric series using the formula:

$$S = \frac{a}{1 - r}$$

where a is the first term and r is the common ratio. Plugging in the values and adding the first term, we get:

$$S = 0.72 + \frac{0.006}{1 - 0.1} = 0.72 + \frac{0.006}{0.9} = \frac{72}{100} + \frac{6}{900} = \frac{648}{900} + \frac{6}{900} = \frac{654}{900} = \frac{109}{150}$$

Therefore, the fraction that equals $0.72\bar{6}$ is $\frac{109}{150}$.

We see that the baseline model has the right general idea for the problem but does not execute the details properly, as it misidentifies the first term of 0.72 as part of the geometric series, and therefore misidentifies the common ratio. With these incorrect values, the model still plugs them into the geometric series formula correctly, however.

In contrast, the fine-tuned model is able to correctly identify that the first term is not part of the geometric series, likely due to seeing another problem involving geometric series during training.

After identifying that the series starts from the second term, it correctly states that the first term of the geometric series portion is 0.006 and the common ratio is 0.1, after which the model computes the answer. This suggests that fine-tuning improved the model’s logical reasoning abilities, as it was able to notice a subtlety in this problem that the baseline model was not.

6.3 Generating Multiple Solutions

Given that Maj1@64 accuracy was higher than Pass@1 accuracy by a somewhat noticeable margin in all cases, a natural question to ask is how solutions are clustered when multiple solutions are generated from the LLM. In particular, we can analyze what percentage of solutions are correct when 64 solutions are generated. Using the LESFT finetuned model, we generate the plot in Figure 3. We see that upwards of 50% of problems have less than 5% of generated solutions correct. Agreeing with the numerical results, this plot suggests that, while generating many solutions does improve accuracy, a very large percentage of solutions are still wrong, making the correct solution difficult to recover from clustering and re-ranking.

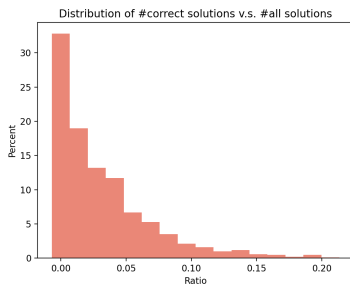


Figure 3: Distribution of Number of Correct Solutions vs. All Solutions during Maj1@64 sampling.

7 Conclusion

In this work, we implemented and combined key insights from multiple papers to develop and implement Logic-Enhanced Sequential Fine-Tuning (LESFT), which is a combination of supervised logic fine-tuning on the LogiQA dataset of deductive reasoning questions and Multi-Task Sequential Fine-Tuning (MTSFT) from Liu et al. (2023). We find that using MTSFT on the open-source Mistral-7B model surpasses the baseline method, as expected, and further adding finetuning on the LogiQA dataset results in another modest improvement over MTSFT.

Our novel result and contribution is that math reasoning ability responds well to training in general logical reasoning in addition to specialized fine-tuning on math problems. This broadens the scope and promise of finetuning methods with respect to mathematical ability in LLMs, showing that methods beyond just fine-tuning on math problems can result in accuracy improvements.

In the future, we would like to repeat this work with LLMs that are much larger in scope, as scale tends to correspond to higher performance improvements. We are interested to see if the observations from this paper replicate in larger models. Further research is also required to see how large language models perform on higher-level math problems (say, undergraduate level), which require more theory - it is unclear whether finetuning on general logic would be as effective on these problems, where more esoteric definitions and knowledge is required. In the future, it would be beneficial to test a more robust implementation of LESFT on MATH in which the LogiQA dataset is expanded using “question bootstrapping” as in Yu et al. (2024) or multilingually as in Chen et al. (2023). Perhaps, an avenue of improving mathematical reasoning lies in making logical reasoning itself less dependent on statistical word probabilities and more representative of its idealistic symbolic roots.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

- Nuo Chen, Zinan Zheng, Ning Wu, Ming Gong, Yangqiu Song, Dongmei Zhang, and Jia Li. 2023. Breaking language barriers in multilingual mathematical reasoning: Insights and observations. *arXiv preprint arXiv:2310.20246*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.
- Yixin Liu, Avi Singh, C Daniel Freeman, John D Co-Reyes, and Peter J Liu. 2023. Improving large language model fine-tuning for solving math problems. *arXiv preprint arXiv:2310.10047*.
- Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. 2022. Lila: A unified benchmark for mathematical reasoning. *arXiv preprint arXiv:2210.17517*.
- Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.
- Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*.
- Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.
- Honghua Zhang, Liunian Harold Li, TAO Meng, Kai-Wei Chang, and Guy Van den Broeck. 2022. On the paradox of learning to reason from data. *arXiv preprint arXiv:2205.11502*.

A Appendix

The full 4-shot prompt used for the baseline and testing is provided here:

Prompt begins here:

Please examine the following four examples:

Problem:

Find the domain of the expression $\frac{\sqrt{x-2}}{\sqrt{5-x}}$.

Solution: The expressions inside each square root must be non-negative. Therefore, $x - 2 \geq 0$, so $x \geq 2$, and $5 - x \geq 0$, so $x \leq 5$. Also, the denominator cannot be equal to zero, so $5 - x > 0$, which

gives $x < 5$. Therefore, the domain of the expression is $\boxed{[2, 5)}$. Final Answer: The final answer is $[2, 5)$. I hope it is correct.

Problem: If $\det \mathbf{A} = 2$ and $\det \mathbf{B} = 12$, then find $\det(\mathbf{AB})$.

Solution: We have that $\det(\mathbf{AB}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = \boxed{24}$. Final Answer: The final answer is 24. I hope it is correct.

Problem: Terrell usually lifts two 20-pound weights 12 times. If he uses two 15-pound weights instead, how many times must Terrell lift them in order to lift the same total weight?

Solution: If Terrell lifts two 20-pound weights 12 times, he lifts a total of $2 \cdot 12 \cdot 20 = 480$ pounds of weight. If he lifts two 15-pound weights instead for n times, he will lift a total of $2 \cdot 15 \cdot n = 30n$ pounds of weight. Equating this to 480 pounds, we can solve for n :

$$\begin{aligned} 30n &= 480 \\ \Rightarrow n &= 480/30 = \boxed{16} \end{aligned}$$

Final Answer: The final answer is 16. I hope it is correct.

Problem: If the system of equations

$$\begin{aligned} 6x - 4y &= a, \\ 6y - 9x &= b. \end{aligned}$$

has a solution (x, y) where x and y are both nonzero, find $\frac{a}{b}$, assuming b is nonzero.

Solution: If we multiply the first equation by $-\frac{3}{2}$, we obtain

$$6y - 9x = -\frac{3}{2}a.$$

Since we also know that $6y - 9x = b$, we have

$$-\frac{3}{2}a = b \Rightarrow \frac{a}{b} = \boxed{-\frac{2}{3}}.$$

Final Answer: The final answer is $-\frac{2}{3}$. I hope it is correct.

Now, please provide a solution to the following problem in the exact same format:

Problem: