Reinforcement Learning Evokes Reasoning in Small Language Models

Abhay Nandiraju[1], Matthew Hernandez[2], Ayesha Khatun[1], Maksim Kulik[2]

[1] College of Information Science,

[2] Department of Linguistics

Abstract. Though mathematical reasoning is always a potential area to solve, however, Large Language Models (LLMs) face challenges in solving arithmetic tasks due to semantic misunderstanding. The main motivation of our work is to enhance the LLM's ability to solve mathematical problems and try to find out the actual reason for the low performance of different kinds of language models, from small to large. To do that, in our experiment, we used Group Relative Policy Optimization (GRPO) to fine-tune the Gemma-3-1B model to improve its mathematical reasoning performance on the GSM8K dataset. Our study assesses the accuracy improvements resulting from GRPO fine-tuning and compares the performance of fine-tuned models to other pre-trained language models like GPT-4o-mini, Falcon, etc. We have seen that fine-tuned GRPO performs 6% better than the pre-trained Gemma-3-1B model because of fewer mistakes in intermediate steps and other types of error. We also tested different reward mechanisms for further enhancing reasoning accuracy. A comprehensive error analysis and human-based evaluation, we present insights into the challenges and opportunities in improving LLM performance in mathematical reasoning tasks and lay the groundwork for future advancements in this area.

Keywords: Large Language Models, Group Relative Policy Optimization, GSM8K, Mathematical Reasoning

1. Introduction

Mathematical reasoning is an active research area where large language models (LLMs) have shown potential, especially when solving word problems that require multi-step thinking and the application of mathematical concepts. Despite their remarkable success in areas like question answering and text generation, LLMs have limited performance when solving general arithmetic problems. The challenges for mathematical reasonings are twofold: problems typically have a single ground truth label, making multiple generations necessary; and LLMs demonstrate sensitivity by propagating mistakes during generation, resulting in cascading errors that the system has no mechanism for handling. The models' basic weaknesses in handling multi-step reasoning have been shown by these experiments, especially quantitative reasoning (Hendricks et al., 2021; Minerva et al., 2022).

The main drawbacks of LLMs when it comes to answering arithmetic word problems, even though semantic misunderstanding mistakes severely impair LLMs' ability to solve math tasks, have received less attention in prior research. This has motivated us to investigate a technique that improves the model's comprehension of the problem's semantics to overcome these flaws more successfully. The goal of this study is to use a technique known as Group Relative Policy Optimization (GRPO) fine-tuning to create a solution that enhances LLMs' comprehension and reasoning skills while solving mathematical issues. The GRPO technique has shown efficacy, especially on benchmarks like the MATH dataset (Shao et al., 2024) previously. On the other hand, in quantitative reasoning, the Gemma-3-1B model is a cutting-edge method for resolving challenging issues, so we decided to fine-tune the Gemma-3-1B model to see how the performance turned out.

To evaluate mathematical reasoning, we have used the GSM8K dataset, a common benchmark used for assessing how well LLMs perform on math word problems. A wide range of problems, from basic algebra to more complex problem-solving exercises, are included in the collection. By using a dual method that combines GRPO fine-tuning with a trained LLM, our study aims to compare and figure out LLM's reasoning skills and better recognize and address the typical mistakes made while solving math problems.

The objective of this project is to develop a robust methodology for solving math word problems using LLMs, with a focus on addressing semantic misunderstanding errors, calculation errors, and step-missing errors. The project will explore the fine-tuned and pre-trained models for solving math word problems. We will compare the performance of the trained Gemma-3-1B-it-bnb-4bit model and the fine-tuned version of the model using the openai/gsm8k dataset. We will evaluate the accuracy of both the trained and fine-tuned models on math problem datasets and compare the results to assess the impact of the fine-tuning process. This will involve testing various reward mechanisms to find the best configuration for improving reasoning accuracy. We will conduct a human-based error analysis to understand why certain errors occur and provide insights into how the model's reasoning can be improved further.

Research Questions for this Project,

A. In what ways does the LLM's fine-tuning process enhance their ability to reason while solving math word problems?
B. Whether the GRPO approach reduces calculation mistakes, step-missing errors, and semantic misunderstanding errors while also giving the LLM a more accurate understanding of the math issue?
C. What is the accuracy difference between the GRPO fine-tuned Gemma-3-1B-it-bnb-4bit model and other pre-trained models?

2. Methodology

In this section, we describe the methodology and the data we used for the basic setup of the experiments. In Figure 1, we can see the architecture of reinforcement learning evokes reasoning in small language models.

2.1. Fine-Tuned LLMs (Gemma-3-1B)

The Gemma-3-1B model is one of the most capable smaller language models, balancing computational efficiency with impressive reasoning capabilities. With just 1 billion parameters, it manages to deliver strong performance across various natural language processing tasks, as well as mathematical reasoning. Among other reasons for selecting this model for our work was its accessible size, which allows deployment and tuning using widely available resources while still maintaining much of the reasoning power found in larger models.

When fine-tuned with techniques like GRPO (Group Relative Policy Optimization), Gemma-3-1B shows remarkable improvements in handling complex math word problems. What makes this model particularly interesting is its ability to respond well to optimization strategies without requiring massive computational resources for training or inference. In the current research landscape where balancing model size with performance is crucial, Gemma-3-1B seems a practical option for exploring reasoning capabilities in more compact language models.

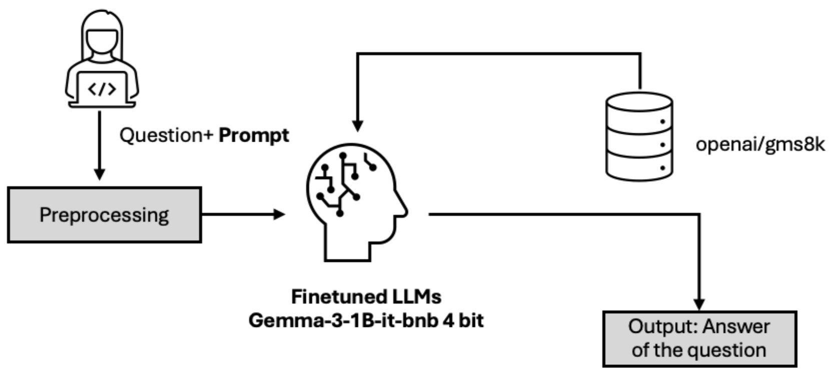

Figure 1. A diagram of the training pipeline used to evoke reasoning in Gemma 3 1B using GRPO.

2.2. Group Relative Policy Optimization (GRPO)

GRPO is an extension of the traditional reinforcement learning approach that addresses fairness and performance across different groups within a learning environment. The core concept is to learn by comparing a group of responses generated by the model itself instead of learning from a gold response. GRPO mainly has four components:

Group Generation: For a given prompt, the policy or the LLM(π_θ) generates a group of K candidate responses. In our code notebook, this is represented by the parameter num_generations, which is set to 2.

Reward Evaluation: Each of these K candidate responses is then evaluated using a set of predefined reward functions that assign a scalar reward score to each completion based on required criteria such as correctness of the answer, format of the answer, and other quality metrics.

Relative Comparison and Learning: The main idea of GRPO is to use these rewards to learn relative liking among the generated responses within the group. The model is then updated to increase the likelihood of generating responses that are preferred over less preferred responses in the group. This method is quite different from methods that compare the model's output to a single gold output.

Policy Update: In the context of reinforcement learning(RL), the LLM is called a policy. The typical objective is to update the parameters of the policy/LLM so as to maximize the expected rewards for the generated responses. The objective/loss function includes a KL divergence term to regularize the policy and prevent it from deviating too much from the reference policy/model, which is the model before GRPO fine-tuning. The objective function for GRPO is:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}$$

Figure 2. Equation showing the objective function used for fine-tuning with GRPO.

where G is the group of outputs/candidate responses from the old policy/model, ε and β are hyper-parameters, $\widehat{A}_{i,t}$ is the advantage computed using a group of rewards $\{r_1, r_2\}$ corresponding to the outputs $\{o_1, o_2\}$ within each group:

$$\widehat{A}_{i,t} = \frac{r_i - mean(\{r_1, r_2\})}{std(\{r_1, r_2\})}$$

In addition to this, instead of adding KL penalty in the reward, GRPO regularizes this by adding KL divergence between the trained policy and the reference policy to the loss, thus avoiding complicating the calculation of $\widehat{A}_{i,t}$. The KL divergence between trained policy/model and the reference model is calculated as:

$$\mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right] = \frac{\pi_{ref}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - \log\frac{\pi_{ref}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - 1,$$

Figure 3. The equation used for calculating KL divergence in GRPO.

where π_θ is the trained policy/model and π_{ref} is the reference policy/model.

2.3. Reward Functions:

We have defined several reward functions to assess specific aspects of the model's generated responses, particularly for mathematical reasoning tasks that require a structured output. These reward functions collectively guide the GRPO training process and pushes the model to generate responses that re well-formatted, contain the required thinking and solution sections, and provide numerically correct answers.The expected output format includes a thinking section and a final solution, marked by special tags such as: <start_working_out>, <end_working_out>, <SOLUTION>, and </SOLUTION>. The reward functions defined are as follows:

match_format_exactly(): This function rewards completions that exactly match the desired output format structure. It checks if a response contains the thinking part followed by the solution part, enclosed in their respective tags. After checking, the function assigns a score of +3.0 if the completion exactly matches this format else assigns a score of 0.0.

match_format_approximately(): This function provides a partial reward based on the presence and count of the individual formatting tags. It's useful when the policy/model doesn't produce the exact format but parts of it are correct. It counts the occurrences of each of the flour tags and scores them as: 0.5 if the tag appears exactly once and -0.5 if the tag appears zero times or more than once. A perfect partial match with one of each tag would yield a total reward of +2.0.

check_answer(): This function checks the correctness of the numerical answer extracted from the solution tags. Firstly, it tries to extract the content within the solution tags and then scores 0 if the extraction fails. Secondly, if the extracted guess matches the true answer it assigns a score of +3.0. If the extracted guess matches after stripping the whitespace it assigns +1.5, else it performs a numerical comparison converting both guess and true answer to floats, and assigns a score based on the tolerance value of the ratio of guess to the true answer. If the tolerance is less than 10%, the function rewards +0.5, else if it is less than 20%, +0.25 is assigned, and -1.0, if it is too far from the true answer. If the true answer and the guess are both equal to zero, the function awards +1.5 but 0 if only the true answer is zero. Finally, the function awards -0.5 if the float conversion fails. In this way, the function aims to reward numerically close answers along with exact string matches.

check_numbers(): This function is more focused towards numerical checking and rewards the model if the extracted number from the solution tags match numerically with the true answer. Initially, the function finds the first sequence of digits and periods within the solution tags and assigns a score 0.0 if no number is extracted and +1.5 if the extracted guess and true answer are equal.

2.4. Reasoning Dataset

We focus on a single dataset, GSM8K (Grade School Math 8K), a dataset of 8.5K high-quality, linguistically diverse grade school math word problems. The dataset is challenging for zero-shot learning and provides a solid criterion for evaluation. The main dataset configuration is used for training. GSM8K (Cobbe et al., 2021) was designed to support the task of question answering (QA) task on basic word problems that require multi-step mathematical reasoning. The questions were by human problem writers and segmented into training and test problems. Table 1 shows the number of word problems in training/dev sets for all configurations.

Configuration	Train	Test
main	7473	1319
socratic	7473	1319

Table 1: Number of word problems in each configuration of the GSM8K dataset. The two configurations each contain a string and its corresponding answer with multiple reasoning steps. The socratic configuration includes sub-questions in the answer written in philosophical prose.

The dataset was intended to be useful for probing the informal reasoning process of LLMs and is challenging for even the largest transformer models to achieve high performance on the test set. Each question takes between 2 and 8 steps to solve and consists of simple arithmetic (i.e., addition, subtraction, multiplication, and division) to reach the final answer. The word problems do not require concepts beyond early Algebra and are rated to be accessible to bright middle schoolers.

Problem: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
Solution: Natalia sold 48/2 = <<48/2=24>>24 clips in May. Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May.
Answer: 72

Problem: Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?
Solution: Weng earns 12/60 = $<<12/60=0.2>>0.2 per minute.Working 50 minutes, she earned 0.2 x 50 = $<<0.2*50=10>>10.
Answer: 10

Figure 4. Two example problems from the main configuration of GSM8K, calculations are written in red.

The questions themselves are written in natural language, as opposed to expressions which are suggested to be a more sensible input for language models. Example problems are shown in Figure 4.

3. Experiment Results

In this section, the experiment result actually has been shown in three different ways. First, we try to analyze the result quantitatively. Secondly, we did an error analysis of humans so that we could find out the actual reason behind the performance loss. Finally, we did the operational analysis, and all those results are described below.

3.1. Quantitative Analysis

To evaluate the effectiveness of the GRPO approach in improving the mathematical reasoning capabilities of small language models, we conducted extensive experiments on the GSM8K dataset. The final version of our experiment was run on an A100 GPU for approximately 6 hours, consuming 60 compute units at a total cost of about $10. Our evaluation methodology focused on comparing the pre-trained Gemma-3-1B model to our fine-tuned version using the GRPO technique. For inference, we employed a majority voting approach (maj@k) where k=2, meaning we generated two completions for each problem and selected the most consistent answer. All evaluations were conducted in a zero-shot setting, without providing examples in the prompt.

Model	Prompting	Parameter Size	Accuracy
Falcon RW (pre-trained)	few-shot	1B	2.63%
GPT-4o-mini (pre-trained)	few-shot	undisclosed	62.80%
Gemma-3 (pre-trained)	zero-shot	1 B (4 bit)	22.97%
Gemma-3 (fine-tuned)	zero-shot	1 B (4 bit)	**28.66%**

Table 2: Accuracy values of our Gemma models vs. another small model & big commercial LLM

The results demonstrate that our fine-tuned Gemma-3 model achieved a significant improvement of 5.69 percentage points over the pre-trained version, reaching an accuracy of 28.66% on the GSM8K test set. While this performance still lags behind larger commercial models like GPT-4o-mini (62.80%), it represents a substantial improvement for a small language model with only 1 billion parameters operating in a quantized 4-bit format.

3.2. Error Analysis

In our analysis, we identified 7 types of errors in the model's reasoning: calculation errors, semantic misunderstandings, missing steps, unit misinterpretations, contextual misunderstandings, potential patterns in the data, and answer formatting issues, which are represented in Figure 2.

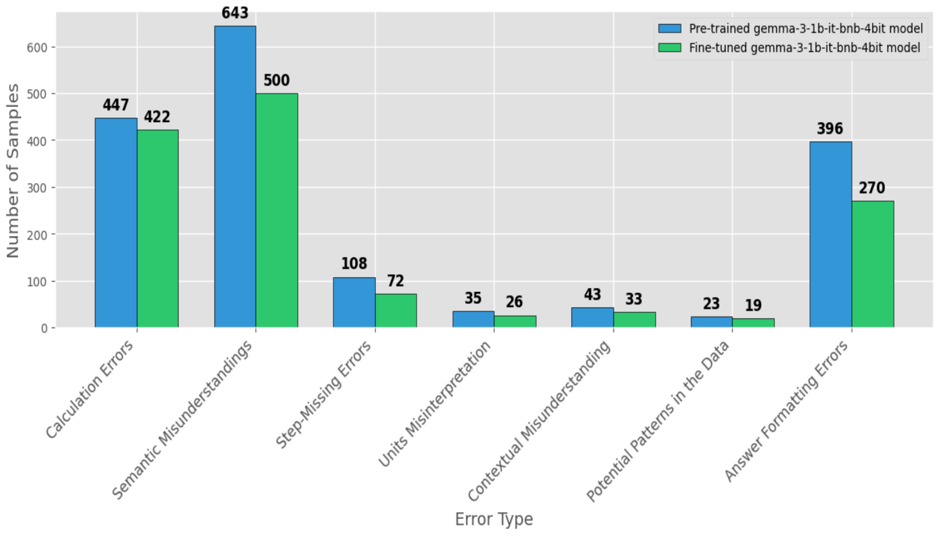

Figure 5. Error analysis of pre-trained and fine-tuned gemma-3-1b-it-bnb-4bit model for generating mathematical reasoning.

The most common errors are semantic misunderstandings, calculation errors, and answer formatting. Semantic misunderstandings occur when the model misinterprets the problem or includes unnecessary factors, often due to difficulty understanding the context or meaning. Calculation errors happen when the model miscalculates the answer, especially with complex math. Misinterpretation of units, like adding 10 chickens and 2 weeks, can also lead to calculation mistakes. After fine-tuning the Gemma-3-1B model, we observed a reduction in all types of errors. However, many errors still remain, which affects the model's overall accuracy.

3.3. Operational Analysis

Large language models require nuanced performance metrics, outside the model's predictions, during inference time. Inference is the process of entering a query (i.e., a prompt) and generating a response from the LLM. During inference time, operational metrics are used to monitor the speed and response times of a model or a system. While the training of Gemma 1B used a compressed model for experimental reasons, quantized models are often practical during the deployment stage. We focus on measuring latency with two operational metrics to simulate a situation where deploying our model requires monitoring the system performance. These metrics are not exclusive to NLP but are adapted to token/completion context. The first metric we use is called time to first token (TTFT), which is the length of time it takes for a user to start receiving a response from a model after entering their query. The second metric is the total generation time (i.e., end-to-end latency), which measures the length of time from querying to completion.

	Average time to first token (per milliseconds)	Average end-to-end (per seconds)	Trials
Human Baseline	180-200	—	—
Baseline Prompt	120	3.331 seconds	30

Table 3. The baseline metrics for the operations analysis, where the baseline prompt is "Hello, world" with a chat template. The experiment is run on a single T4 GPU and the prompt is averaged from 30 trials.

The human visual reaction time falls between 180 and 200 ms, and getting the first token to appear below this threshold makes chat applications appear responsive. CITATION We compare this baseline with a simple "Hello, world." prompt with a chat template in Table 3. Each experiment undergoes 30 trials on a single T4 GPU provided by Google Colab and is processed with the "warmup" configuration to better simulate downstream uses; warmup has been shown to improve latency. In our experiment, we first tokenize the training set and find the number of tokens to reflect the prompt length. We sort 9 intervals between 44 and 245 tokens, inclusively, with a step size of 15. Next, a random prompt is sampled from each interval. There will be 30 trials for the TTFT and E2E experiments with a running time of 42 seconds and 1 hour and 52 minutes, respectively.

The TTFT experiment in Figure 6 achieves strong performance that is nearly below the human baseline for 7/9 prompts of different lengths, where most prompts are near the performance of the baseline prompt. The combination of a quantized model that can achieve good performance on the GSM8K dataset and has low latency, demonstrates a suitable system for downstream uses. However, we must look at the E2E performance for another perspective on the actual task itself. In the succeeding experiment, we measure the E2E performance, shown in Figure 6, and show prompts of different lengths to take on average 25 seconds or less to complete and generate a response. The increase from approximately 3 seconds for the baseline prompt to around 25 seconds appears realistic; moreover, the prompt task is fairly complex as the model outputs the steps required to solve the word problem.

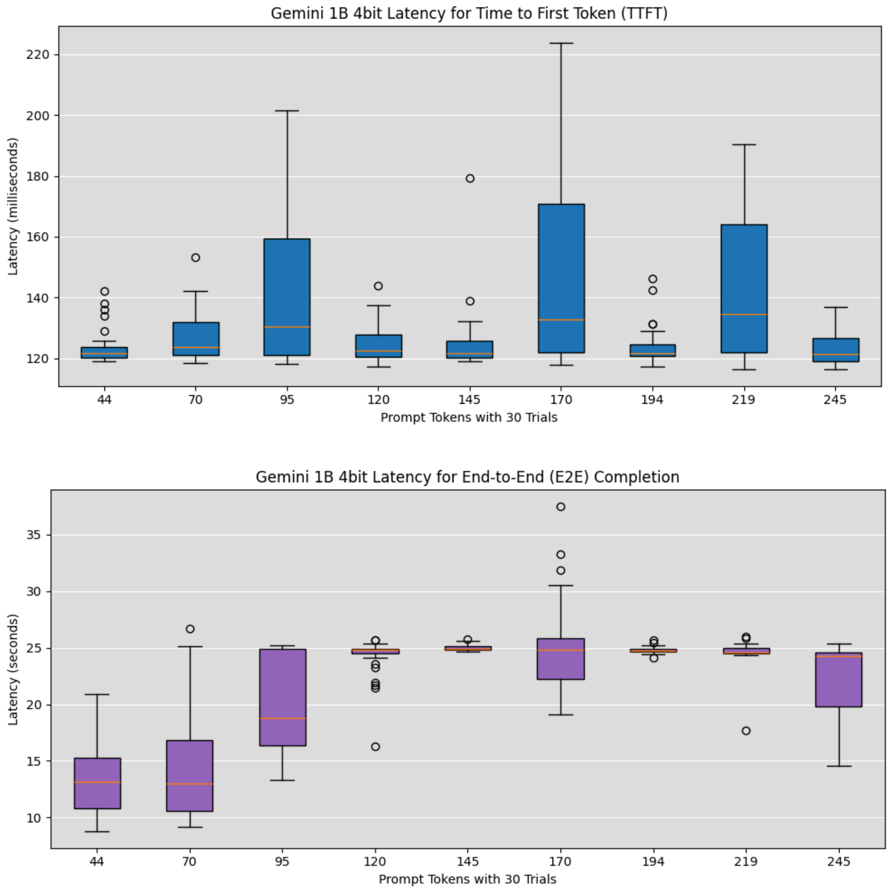

Figure 6. Results for the time to first token and end-to-end experiment with 30 trials on a single T4 GPU. The training set is tokenized and their prompt length is sorted from 44 tokens to 245 tokens with a step size of 15 tokens. Then 9 examples are sampled from each interval to run the TTFT and E2E experiments.

While we focus exclusively on performing latency experiments, the exercise further supports the benefits of using the quantized Gemma-3-1B-it 4-bit model for reasoning tasks, given the model is trained for enough epochs. Throughput is another valuable operational metric that is used to measure the number of requests processed per second, but we ignore this because the potential downstream application of our model isn't fully specified. Also, our project is largely experimental, and throughput is often used in conjunction with batch processing to improve system performance, which isn't immediately beneficial for our use case.

4. Conclusion and Future Work

Our study demonstrates that Group Relative Policy Optimization (GRPO) can significantly enhance the mathematical reasoning capabilities of small language models. The fine-tuned Gemma-3-1B model achieved a 5.69 percentage point increase in accuracy on the GSM8K dataset (from 22.97% to 28.66%). Put in other words, we get about 25% better performance compared to our baseline, the pre-trained Gemma-3-1B. While this performance improvement is promising, our error analysis reveals that semantic misunderstandings and calculation errors remain challenging areas. The operational analysis confirmed that the quantized model maintains acceptable inference speeds post-fine-tuning, with time-to-first-token responses often below human reaction thresholds.

For future work, we propose exploring (1) multi-modal reinforcement learning to address semantic misunderstandings, (2) hierarchical reasoning approaches to reduce calculation errors, (3) more sophisticated reward mechanisms, (4) investigating the cross-domain transfer of mathematical reasoning skills, and (5) integrating external verification tools to validate intermediate calculations. These directions could further bridge the gap between computational efficiency and advanced reasoning capabilities in small language models, making sophisticated AI assistance more accessible across various resource constraints.

References

James F. Allen, Nathanael Chambers, George Ferguson, and Lucian Galescu. 2007. Plow: a collaborative task learning agent.

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

Hendrycks, D., et al. (2021). "Measuring mathematical problem solving with the MATH dataset." Proceedings of the International Conference on Learning Representations (ICLR).

Minerva, A., et al. (2022). "Solving Quantitative Reasoning Problems with Language Models." Proceedings of the NeurIPS.

Wei, J., et al. (2022b). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." Proceedings of the 2022 Conference on Neural Information Processing Systems (NeurIPS).

Kojima, H., et al. (2022). "Chain-of-Thought Prompting Enables Large Language Models to Solve Arithmetic Problems." Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Shao, Z., et al. (2024). "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models." Proceedings of the 2024 Conference on Neural Information Processing Systems (NeurIPS).

Cobbe, K., et al. (2021). "GSM8K: A Benchmark for Generalized Arithmetic Problem Solving." Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Thompson, P.D., et al. (1992). 'Voluntary stimulus-sensitive jerks and jumps mimicking myoclonus or pathological startle syndromes.' *Movement Disorders*, 7(3), 257–262. doi:10.1002/mds.870070312."

Workload Split:

Name	Work
Ayesha Khatun	Data Preprocessing, Testing with GPT-4o-mini, falcon-rw-1b and Plot Generation, Abstract, Introduction, Error Analysis in paper
Maksim Kulik	Detailed Analysis of Gemma-3-1B, Methodology, Discussion, Conclusion and Future Work, Quantitative Analysis, Team Manager
Matthew Hernandez	Latency experimentation, Repository maintainer, Inference Notebook Design, GSM8K dataset, and Operational Analysis/Metrics in paper
Abhay Kumara Sri Krishna Nandiraju	Gemma-3-1b fine-tuning with GRPO, Pretrained Model and Fine-tuned Model accuracy evaluation, Reward Function Definition, GRPO and Reward function sections in the paper