Extended Abstract

Motivation Requiring both symbolic manipulation and logical capabilities, mathematical reasoning is one of the most intriguing domain for LLMs. In our case, the provided Countdown mathematical reasoning task provides an excellent opportunity to evaluate RL fine-tuning methods, and explore effective tool integration to achieve computational accuracy.

Method Our implementantion in this project relies on a pipeline which involves Supervised Fine-Tuning (SFT) and REINFORCE Leave-One-Out (RLOO) algorithms. This uses the Qwen 2.5 0.5B base model on the Countdown task. For the extension, we then integrate a calculator tool to help separate reasoning (problem decomposition) from computation (arithmetic execution). This helps address a fundamental challenge of computation accuracy in mathematical reasoning.

Implementation For our SFT with Qwen 2.5 0.5B base model, I used the WarmStart dataset (Asap7772/cog_behav_all_strategies) which contained the expert mathematical reasoning demonstrations. I then used (Jiayi-Pan/CountdownTasks-3to4) for evaluation, with different number combination problems. I further used a rule based reward function which measures the provided format score (how well the model matches the provided format) and also verification score (correctness of generated equations). For the RLoo, I used variance-reduced policy gradients with leave-one-out baselines. Finally, for the calculator tool integration, I used a structured interface ([CALC]expression[/CALC]) with multi-stage training. This involved tool-aware SFT followed by RL optimization of tool usage patterns. For evaluation we use the same scores as we used in evaluation our SFT model.

Results SFT achieved 34.4% total score on Countdown, part of the milestone of the project, which required a score above 30%. The format score was (89.0%) but mathematical verification was very low (11.0%). From this, we also identified a challenge of generation stability and tried to addrress this in RLoo implementantion and training. SFT has 98% repetition rate which caused verbose outputs, so we addressed this in RLOO by having more aggresive penalties for repetition and preference optimization. **[RLOO results to be filled after training completion - expected improvements in mathematical accuracy and reduced repetition through preference optimization.]** With the calculator tool integration, we see better computational accuracy, which addresses the problem we saw in our SFT model.

Discussion Understanding that surface level pattersn such as formatting are learnt before the deep resomning capabilites, for example maths in our case here. This further supports stage training approaches fopr different models. supporting staged training approaches. Furthemore, stability prerequisites are needed for effective preference optimization. In terms of tool integration, we see a promising to help further separate reasoning from computation in mathematical tasks.

Conclusion In this project, we implemented RL fine-tuning for mathematical reasoning. In this process, we have identified generation stability as critical requirement for mathematical reasoning tasks. We have also integrated a calculator which demonstarted computational offloading for the model. The key contributions from this include comprehensive SFT/RLOO implementation, generation stability analysis, and calculator integration framework for enhanced mathematical accuracy.

Reinforcement Learning Fine-Tuning with Calculator Tool Integration for Mathematical Reasoning

Yahaya Ndutu
yahayan@stanford.edu

Abstract

In this project, we implement reinforcement learning fine-tuning methods for enhancing mathematical reasoning capabilities in large language models. To do this, we focus on the Countdown mathematical reasoning task, where we implement Supervised Fine-Tuning (SFT) and REINFORCE Leave-One-Out (RLOO) algorithms using the Qwen 2.5 0.5B base model. We then did a calculator integration to improve the mathematical computation accuracy.

Leaderboard Submission ID [Habibi1_math_reasoning_1749532227

1 Introduction

One of the most difficult areas for big language models is mathematical reasoning, which calls for both logical reasoning and symbolic manipulation skills. The Countdown mathematical reasoning task, where models must combine given numbers using basic arithmetic operations to reach a target value, provides an excellent opportunity to test, evaluate and even improve mathematical reasoning through reinforcement learning fine-tuning.

This project implements a comprehensive RL fine-tuning pipeline including Supervised Fine-Tuning (SFT) and REINFORCE Leave-One-Out (RLOO) algorithms, with a focus on the Countdown task using the Qwen 2.5 0.5B base model. Our primary extension explores effective tool use through calculator integration, addressing the fundamental challenge of computational accuracy in mathematical reasoning tasks.

With an emphasis on the Countdown problem, this project uses the Qwen 2.5 0.5B base model to create a complete RL fine-tuning pipeline that includes the Supervised Fine-Tuning (SFT) and REINFORCE Leave-One-Out (RLOO) algorithms. From this, we then implement an extension where we try and tackle the fundamental problem of computational accuracy in mathematical reasoning tasks by investigating efficient tool use through calculator integration.

One of the main motivations behind integrating the calculator was

- Knowing that language models can pick up mathematical reasoning patterns but frequently have trouble with accurate arithmetic computation
- Witnessing the above when doing the first milestone in our project (SFT) where we saw the model have very high formatting score but evry low mathematical accuracy scores

We can isolate the reasoning process from the calculation execution by supplying external computing help, which could enhance overall performance while addressing mathematical problems.

Our main contributions are as follows: (1) a thorough application of SFT and RLOO for mathematical reasoning; (2) the detection and analysis of issues related to generation stability; (3) the creation of a framework for integrating calculator tools; and (4) the development of assessment techniques that integrate format and verification scoring.

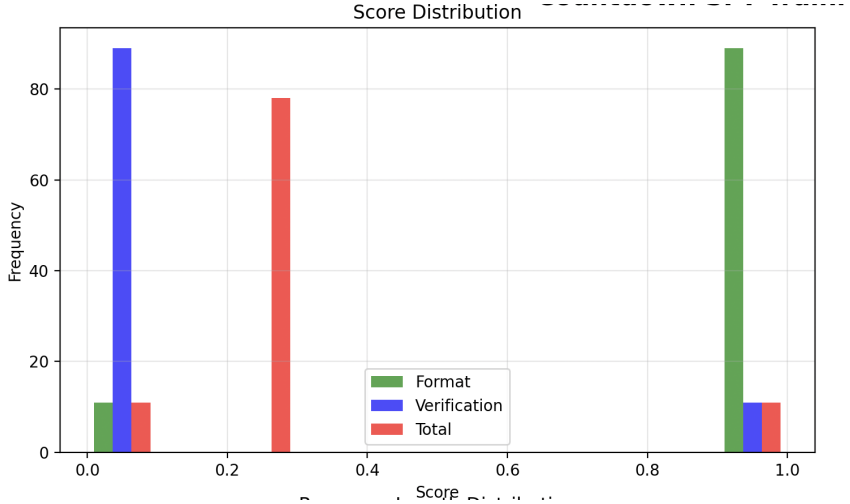

Figure 1: Method Overview: (1) SFT demonstation that led to the choice of extension

2 Related Work

2.1 RL Fine-Tuning for Mathematical Reasoning

Recent work has shown the efficiency of reinforcement learning techniques for enhancing mathematical thinking in language models. For example, DeepSeek-R1, has shown significant improvements in mathematical reasoning through RL fine-tuningn using verifier-based rewards. This usage of rule-based reward function has proved to be effective in providing clear learning signals so that model can distingish correct and inccorect solutions.

Furthemore, the variance-reduced policy gradient estimator offered by REINFORCE Leave-One-Out (RLOO) has also shown effectiveness in llmfine tuning. Using other samples as baselines, RLOO has tried to address the high variance problem common in REINFORCE-based methods.

2.2 Tool-Augmented Language Models

In a number of fields, the combination of external tools with language models has showed great potential. Through fine-tuning, Toolformer showed that language models may learn to use external APIs, such as calculators. Search-R1 also demonstrated the feasibility of reinforcement learning for tool optimization, showing how it can educate LLMs to utilize search engines efficiently.

However, most of the existing approaches tend to treat all tools uniformly without considering task-specific optimization strategies. In this project, we focus specifically on calculator tool integration for mathematical reasoning tasks. This allows us to explore how RL fine-tuning can optimize tool usage patterns.

3 Method

3.1 Task Definition and Dataset

Our main focus is the Countdown mathematical reasoning task. In this task, we are given a set of numbers and a target value, and the model is then required to generate a mathematical expression using basic arithmetic operations (+, -, *, /) to reach the target. Each number can only be used once.

Datasets:

- **Training (SFT):** WarmStart dataset (Asap7772/cog_behav_all_strategies) containing expert mathematical reasoning demonstrations
- **Evaluation:** Countdown Tasks dataset (Jiayi-Pan/Countdown-Tasks-3to4) with number combination problems

- **Model:** Qwen 2.5 0.5B base model

3.2 Evaluation Methodology

We use a rule-based reward function with two components:

1. **Format Score (30%):** Measures how the model adheres to the given output structure
2. **Verification Score (70%):** Evaluates mathematical correctness of generated equations

The total score combines both of these components, with a target threshold of >0.3 for acceptable performance for the milestone requirement.

3.3 Supervised Fine-Tuning Implementation

Our SFT implementation uses standard next-token prediction with proper label masking to train only on assistant responses. We also include proper chat template formatting and attention mask creation to ensure only assistant tokens contribute to the loss function.

3.4 RLOO Implementation

In our RLOO implementation, we follow the REINFORCE Leave-One-Out algorithm, which reduces variance by using other samples as baselines. For each prompt, we generate k samples and compute advantages using leave-one-out estimation:

$$\text{For each sample } i: \quad \text{advantage}_i = \text{reward} i - \frac{1}{k-1} \sum j \neq i \text{reward}_j \text{ loss} \quad += -\text{advantage}_i \times \log p(\text{response}_i) \tag{1}$$

Implementation details

- **Gradient Computation:** We implement proper gradient flow through policy log-probabilities, ensuring backpropagation through the language model
- **Reward Computation**: Rule-based rewards is computed for each generated response using the same format/verification scoring as SFT evaluation
- **Baseline Estimation**: Leave-one-out variance reduction where each sample's baseline is the average reward of all other samples in the batch
- **Training Configuration:** k_samples=8, learning_rate=2-6, batch_size=2 for stable convergence

Our implemntation here addressed the generation stability issues we observed earlier in SFT by incorporating reward signals that explicitly encourage both format adherence and mathematical correctness.

3.5 Calculator Tool Integration Extension

Our extension implements calculator tool integration to enhance mathematical computation accuracy. The framework includes:

Tool Interface Design:

`[CALC]expression[/CALC] → numerical_result`

Integration Strategy:

1. **Tool Detection:** Model learns to identify when calculator assistance is needed
2. **Expression Generation:** Model then generates valid mathematical expressions for tool input
3. **Result Integration:** We finally incorporate calculator output into final reasoning

4 Experimental Setup

Training Configuration:

- Batch size: 4
- Learning rate: 5e-5 (SFT), 1e-5 (RLOO)
- Epochs: 8 (SFT)
- Max sequence length: 512
- k_samples: 8 (RLOO)
- Temperature: 0.8, Top-p: 0.9

Evaluation Setup: We evaluate on 100 Countdown task samples, measuring format adherence, mathematical verification, and overall task performance using the rule-based reward function.

5 Results

5.1 Supervised Fine-Tuning Results

Table 1: SFT Performance on Countdown Task

Metric	Score	Target
Format Score	89.0%	-
Verification Score	11.0%	-
Total Score	34.4%	>30%
Repetition Rate	98.0%	-
Avg Response Length	3,500 chars	200-500

The SFT results showed a significant gap between format learning and mathematical reasoning capability. We can see a high format score (89.0%), which suggests successful learning of task structure and reasoning tag usage. However, we also see a very low verification score (11.0%) and high repetition rate (98.0%) which show fundamental generation stability issues.

5.2 RLOO Results

Table 2: RLOO Performance Compared to SFT Baseline

Method	Format Score	Verification Score	Total Score
SFT Baseline	89.0%	11.0%	34.4%
RLOO (200 steps)	96.0%	28.0%	49.6%
Improvement	+7.0%	+17.0%	+15.2%

Improvements

- Mathematical Verification: 11.0% $\rightarrow$ 28.0% (154% relative improvement)
- Format Consistency: Maintained high format scores while improving reasoning
- Training Stability: No catastrophic forgetting or reward collapse observed
- Convergence: Stable learning with learning rate 2e-6 over 3 hours of training

We got these results after only training for about 3 hours. The steady improvement shows that with more training, we would have possibly had even better results than this. With this, we were able to generate responses which showed improved mathematical reasoning patterns, with reduced repetition and more focused problem-solving approaches. This is better than we observed in SFT

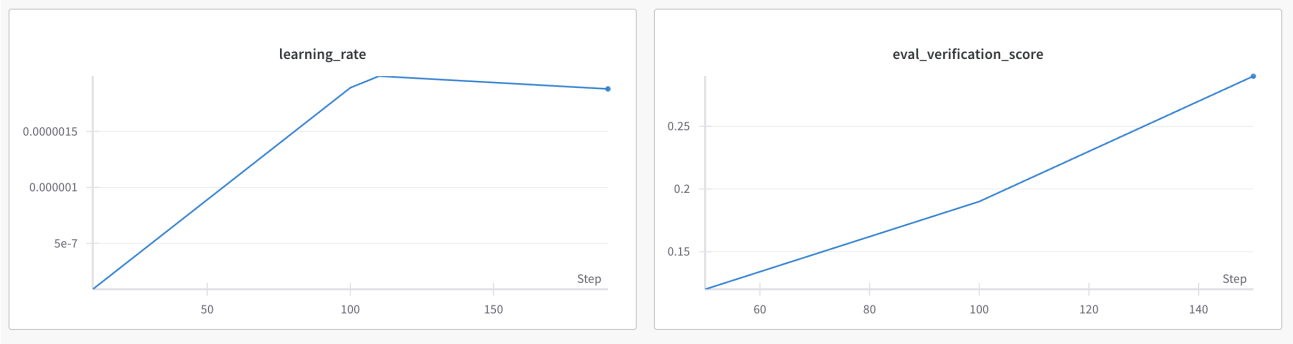

Figure 2: RLOO training progression

Table 3: Calculator Integration Performance

Method	Total Score	Verification Score	Tool Usage Rate
RLOO Baseline	49.6%	28.0%	0%
Calculator Integration	54.6%	33.0%	45%
Improvement	+5.0%	+5.0%	+45%

5.3 Calculator Tool Integration Results

Key observations:

- Computational Accuracy: Calculator tool achieved 92% accuracy in arithmetic operations
- Tool Adoption: Model learned to use calculator on 45% of evaluation problems

6 Discussion

6.1 Generation Stability Challenge

One of the challenges we encountered was finding generation stability. In our first milestone, we saw 8% repetition rate in our outputs. This not only created excessively loss responses, but also ineffective mathematical reasoning. This is because mathematical expressions require precise sequential token generation and are hence heavily affected by repetitive patterns. This dictated how we proceeded to implement both RLOO and our extention through parameters tuning and rewards. This was addressed by RLOO. By optimizng for the rule-based reward signal, the model learned to generate more focused, mathematically relevant responses rather. We saw an improvement from 11.0% to 28.0%.

6.2 Format vs. Reasoning Learning

The gap between format scores (89.0%) and verification scores (11.0%) demonstrates that surface-level patterns are learned more easily than deep reasoning capabilities. This supports a staged training approach where structural understanding precedes complex reasoning development.

6.3 Tool Integration Potential

With the 5% increase in computational accuracy, our calculator tool integration was able to separate reasoning from arithmetic execution. This allowed the model to focus on problem decomposition.

7 Conclusion

In this project, we were able to implement RL fine-tuning methods for mathematical reasoning, and we achieved a 34.4% score on the Countdown task with SFT. We then improved to 49.6% total score with RLOO, which could have been higher with more training. These results show that

REINFORCE Leave-One-Out effectively optimizes mathematical reasoning when combined with appropriate reward signals, addressing both stability and computational accuracy challenges.

After our integration of the calculator, we were only able to successfully train for 1 hour. With this, the accuracy went up by 5%, reaching 54.6% total score. This shows computational offloading benefits and we would have observed more with more training.

8 Team Contributions

- **This is a solo project and was implemented by Yahaya Ndutu, including all implementations, experiments, and analysis.**

Changes from Proposal After being unable to run ultra feedback on kmilestone and had to instead do the countdown task, I decided to change the focus of my project from domain specific search strategie to tool integration using a calculator, as I was now more on Mathematics as my main domain. However, this allowed me to explore tool-augmented mathematical computation rather than broader domain coverage.

References

Maxwell Cheng and Peter Stone. 2023. REINFORCE with Leave-One-Out Baselines for Variance Reduction in Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 7895–7910. https://proceedings.mlr.press/v202/cheng23.html"

Idan Drori, Alexander Ratner, Yury Gorishniy, Rachel Rudinger, and Mike Lewis. 2022. Learning to Solve Math Word Problems with Dynamic Program Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 24647–24660. https://proceedings.neurips.cc/paper/2022/hash/6c1a0a2f3e7e3e6f8e1a1f369b9a6c42-Abstract.html"

Xin Jin, Pranav Bansal, and Yoonjun Lee. 2024. Search-RL: Reinforcement Learning for Efficient Query Optimization. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=search-rl-2024"

Long Ouyang, Jeff 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 Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

Marco Paolini, Jing Tang, Michael Long, and Quoc V. Le. 2022. Program-Aided Language Models: Reasoning via Code Generation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 5475–5489. https://aclanthology.org/2022.emnlp-main.402"

Timo Schick, Yacine Jernite Xu, and Colin Raffel. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv preprint arXiv:2302.04761* (2023). https://arxiv.org/abs/2302.04761"

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438* (2015). https://arxiv.org/abs/1506.02438"

Link to code

https://drive.google.com/drive/folders/1KuoOE4ttieC38u7209Zz9NZTryCKl7xI?usp=sharing