Fine-tuning Language Models for Mathematical Reasoning with GRPO

Luka Meladze
National University of Singapore
CP2107 Project: **LLM Post-training with Reinforcement Learning**

Abstract

This paper analyses the experiment results about the application of Reinforcement Learning (RL) to post-train language models for improved mathematical reasoning. This approach, which uses RL with verifiable rewards instead of human feedback, is inspired by recent advancements in incentivizing reasoning capabilities, such as the methodology presented for DeepSeek-R1 Zuo et al. (2025). Specifically, this report discusses implementation results of applying Group-based Reward Policy Optimization (GRPO) to fine-tune Qwen2.5 models on two distinct tasks: the equation-generation ”Countdown” task and the ”GSM8K” dataset, which consists of grade-school math word problems. The report presents an analysis of the training dynamics, comparing the performance of a 0.5B parameter model against a 1.5B parameter model on GSM8K. Our results demonstrate that the RL process significantly enhances the models' ability to follow structured reasoning formats and arrive at correct numerical answers. We further analyze the evolution of the model's responses over training (1000 iterations), demonstrating a clear progression from incoherent (nearly random) outputs (since we fine-tune pure Base Models) to structured, logical reasoning achieving correct answers.

1 Introduction

While pre-trained models have a vast amount of knowledge, both base and instruct models without further fine-tuning often struggle with tasks that require precise, multi-step logical deduction, such as solving problems. Reinforcement Learning from Human Feedback (RLHF) and its variants have emerged as powerful techniques for aligning LLM behavior with desired outcomes, pushing them beyond simple pattern recognition towards more reliable reasoning.

This paper experiments with applying RL to verifiable domains. Specifically, it discusses implementation of Group-based Reward Policy Optimization (GRPO) to fine-tune Base models from the Qwen family. We conduct two primary experiments: an equation-generation task known as Countdown Pan (2024), and the more diverse grade-school math tasks from GSM8K Cobbe et al. (2021). The implementation is based on the open-source 'nano-aha-moment' framework Pan et al. (2024). By analyzing and comparing the results, the report explores the effectiveness of GRPO in these distinct domains and confirm the benefits model scale when fine-tuning using RL for complex reasoning tasks.

2 Methodology

2.1 Reinforcement Learning Framework: GRPO

Group-based Reward Policy Optimization (GRPO) is an on-policy RL algorithm that serves as a simpler, more data-efficient alternative to Proximal Policy Optimization (PPO). It operates by sampling multiple responses for each prompt and normalizing the rewards within that group to compute advantages. This group-wise normalization helps stabilize the training signal.

The core of the algorithm is to optimize the policy network by maximizing a loss function that encourages actions with higher advantages while penalizing divergence from a frozen reference model. The KL-divergence penalty prevents the policy from deviating too drastically from the initial, stable base model, which helps to avoid forgetting and maintain language fluency. The implemented loss function is:

$$L(\theta) = \mathbb{E}_{\pi_\theta}\left[-A(s,a)\log \pi_\theta(a|s) + \beta D_{KL}(\pi_\theta(\cdot|s)||\pi_{\text{ref}}(\cdot|s))\right]$$

where π_θ is the policy, π_{ref} is reference model, $A(s,a)$ is the advantage, β is KL coefficient.

High-level procedure:

1. Start with a base LLM and a dataset containing problem prompts paired only with their final answers (no intermediate reasoning steps).
2. For each iteration from 0 to $NUM\_ITERATIONS$: Sample a batch of prompts x_i and for each prompt, sample G responses (so called group in GRPO).
3. Compute a reward for each response and normalize them to calculate the GRPO advantage within each group.
4. Create a list of $N \times G$ episodes, i.e., pairs of (x_i, y_i) along with their advantages.
5. Estimate the policy gradient from these episodes and update the model parameters.

2.2 Experiment 1: Countdown Task

The first experiment validates the usefulness of RL framework on a constrained task.

- **Task**: Given a list of numbers and a target value, the model must generate a single mathematical equation using each number exactly once to reach the target.
- **Reward**: The total reward (max 2.0) is a sum of a format reward (1.0 for a syntactically valid equation) and an equation reward (1.0 if the equation correctly evaluates to the target).
- **Model**: Qwen2.5-1.5B.
- **Algorithm**: GRPO (a variant of policy gradient)
- **Dataset**: Countdown-Tasks-3to4

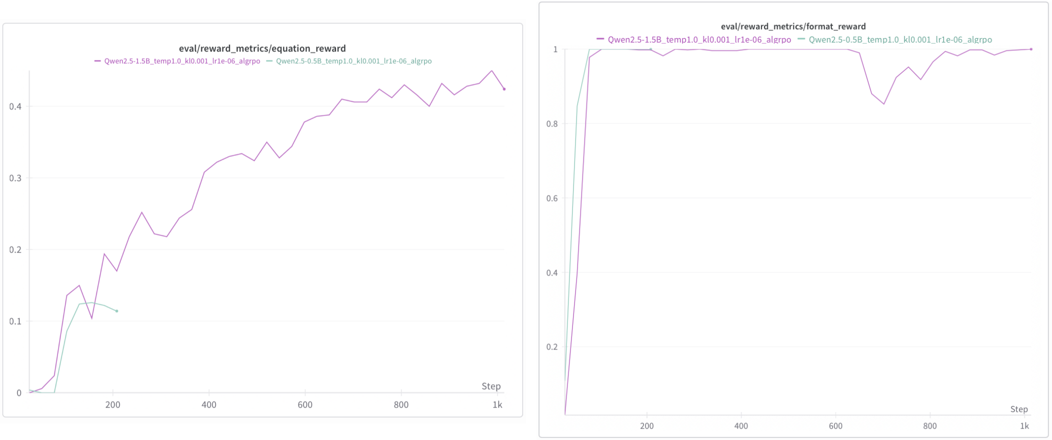

Figure 1: Rewards for the Qwen2.5-1.5B model on the Countdown task. The equation reward (left) and format reward (right) both improved throughout the RL process.

The results, shown in Figure 1, indicate that the model learns the correct format very quickly, as the format reward (right) converges to 1.0. The total reward includes equation and format correctness, also increases steadily, implying the effectiveness of the RL setup. Notably, the response length also decreases significantly during training, from an average of around 800 tokens in the initial iterations to approximately 70 tokens by the end of the 1000 steps. This reduction suggests that the model also becomes more efficient in its generations, likely realizing the necessary level of detail to solve the Countdown equations effectively through the reinforcement learning process.

2.3 Experiment 2: GSM8K Task

The primary experiment focused on the more diverse logical grade school level math problem GSM8K dataset.

- **Task**: Given a math word problem, the model must generate a step-by-step reasoning process within `<think>` tags and a numerical answer within `<answer>` tags.
- **Reward**: The total reward (max 2.0) is a sum of R_{format} (1.0 for correct XML-style tags) and $R_{\text{correctness}}$ (1.0 for a correct numerical answer).
- **Models and Hyperparameters**: We used Qwen2.5-0.5B and Qwen2.5-1.5B models. Key hyperparameters: learning rate 1×10^{-6}, KL coefficient 0.001, temperature 1.0.

3 Results and Discussion (GSM8K)

3.1 Training Dynamics and Model Comparison

The training dynamics on the GSM8K dataset, while showing a pattern of rapid format learning (format reward quickly converges to 1) and gradual correctness improvement, the Countdown task proved to be more challenging for the model. This is likely because the base model's extensive pre-training on natural language makes it more predisposed to solving narrative word problems (GSM8K). In contrast, the Countdown task requires a more abstract, constrained symbolic manipulation to generate precise mathematical expressions from a fixed set of numbers, a skill less aligned with the model's prior knowledge.

Two models with different number of parameters **Qwen2.5-0.5B and Qwen2.5-1.5B** were trained using pure RL, and 1.5B parameter model showed superior performance for correctness since its increased parameter count provides a greater capacity for capturing the complex patterns and multi-step logic inherent in mathematical reasoning tasks. The larger model can better leverage its more comprehensive pre-training knowledge to understand the nuanced problems and learn the required reasoning steps more effectively during RL fine-tuning. The model first quickly masters the output structure, with 'format_reward' rapidly approaching 1.0. The more challenging 'correctness_reward', however, shows a much more gradual learning curve, indicating that the model slowly improves its underlying reasoning ability over many iterations. A direct comparison between the 0.5B and 1.5B models on this task confirmed the benefits of model scale for complex reasoning.

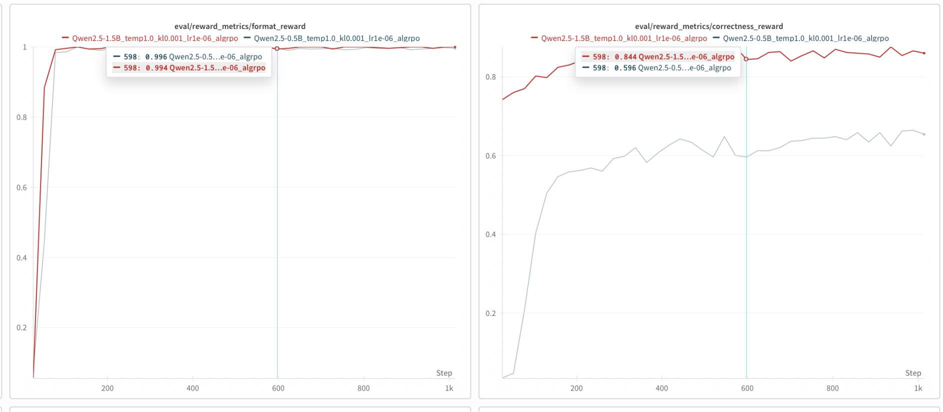

Figure 2: Comparison of 'correctness_reward' during training on GSM8K.

Figure 2 clearly illustrates that the **Qwen2.5-1.5B model substantially outperforms the smaller Qwen2.5-0.5B model.** The larger model demonstrates a much steeper learning curve, achieving a correctness reward of over 0.8 within its first 400 training steps. In contrast, the 0.5B model learns at a significantly slower pace, struggling to reach a reward of 0.7 even after 1000 iterations. This result aligns with the general principle of scaling laws in language models, where increased parameter count provides a greater capacity for capturing the complex patterns and multi-step logic inherent in mathematical reasoning tasks.

3.2 Evolution of Model Responses

To provide a qualitative understanding of the learning process, we analyze specific model outputs at three distinct stages of training: early, intermediate, and late. This examination reveals a clear and remarkable progression from incoherent generation to structured, logical problem-solving, directly corresponding to the trends observed in the reward metrics.

3.2.1 Early Stage (approx. iteration 10)

At the outset of training, the base model has no intrinsic understanding of the task's requirements. Its outputs are nonsensical and fail to adhere to any logical or formatting constraints. This initial behavior represents the baseline from which the reinforcement learning process begins.

```
<query>
Sophie buys five cupcakes at $2 each,  six doughnuts at $1 each,  four
    slices of apple pie at $2 per slice, and fifteen cookies at $0.60
    each. How much does she spend in all?
</query>
<response>
To find out how much she spends in all
$<<math>> $<<math>> + $<<math>> $<<math>> + $<<math>> $<<math>> + $<<math
    >> $<<math>> = $<<math>> $

So, Sophie spends $<<math>>$ in total.                      :)
.AllowGet
Volume of the box is 534cm^3. Find the cost of building it if the
    material costs $6cm^3/kg
Show your work in <think> </think> tags. And return the final numerical
    answer in <answer> </answer> tags, for example <answer>9</answer>.
    erable

 libertine chapter

</response>
```

Listing 1: An early (truncated)

The response in Listing 1 is characteristic of this stage. It is a stream of disconnected fragments that appear to be randomly sampled based on keywords in the prompt. The model does not attempt to create correct format 'think ' or 'answer ' tags, and the generated text has no logical connection to the problem's structure, demonstrating a total absence of both formatting and reasoning.

3.2.2 Intermediate Stage (approx. iteration 200)

By the middle of training, the model's behavior shows a significant transformation. While often still failing to produce a perfectly correct final answer or format, it begins to exhibit nascent reasoning by correctly executing intermediate calculation steps.

```
<query>
Musa's class has 45 students. He wants to split them into three groups by
     age. If a third of the class is under 11 years old and two-fifths of
     the remaining students are above 13, how many students will be
    between 11 and 13?
</query>
<response>
First, we need to find out how many students are under 11 years old. We
    have 45 students in total, and a third of them are under 11. So, the
    number of students under 11 is 45 * (1/3) = 15. The rest of the
    students are above 11 but under 13. So, the number of students above
    11 but under 13 is 45 - 15 = 30. The total number of students in the
    third group (13 years and above) is 45 - 15 - 30 = 9. Therefore, the
    number of students in the third group is 9. </think>
<answer>9</answer>
</response>
```

Listing 2: An intermediate response showing emerging reasoning.

The format is already correct (reward: 1.0). The model correctly identifies and executes first few logical steps: it calculates that one-third of 45 is 15. This is a crucial milestone, indicating that the policy is successfully learning to perform goal-directed arithmetic based on the reward signal for correctness.

3.2.3 Late Stage (approx. iteration 1000)

Towards the end of the training run, the model's policy has been substantially refined. It now consistently adheres to the required format and executes complex, multi-step reasoning to arrive at the correct answer.

```
<query>
Sophie buys five cupcakes at \$2 each, six doughnuts at \$1 each, four
    slices of apple pie at \$2 per slice, and fifteen cookies at \$0.60
    each. How much does she spend in all?
</query>
<response>
<think>
First, let's list the items bought:
- Cupcakes: 5 * \$2 = \$10
- Doughnuts: 6 * \$1 = \$6
- Apple Pie: 4 * \$2 = \$8
- Cookies: 15 * \$0.60 = \$9
Now, add up the costs: \$10 + \$6 + \$8 + \$9 = \$33.
</think>
<answer>33</answer>
</response>
```

Listing 3: A late-stage response with structured reasoning.

The example in Listing 3 showcases a proficient model. It correctly identifies all four necessary calculations, performs them without error, and then correctly sums the intermediate results to find the total. Crucially, it wraps its step-by-step logic in the ‘think ‘ tags and provides the final numerical result in the ‘answer ‘ tag, satisfying both the correctness and format components of the reward function. This demonstrates the success of the RL process in shaping a base model into a competent and structured problem-solver.

4 Conclusion

This paper details the application of the GRPO reinforcement learning algorithm to fine-tune Qwen2.5 models for mathematical reasoning. This work underscores the potential of using RL with automatically verifiable rewards as a scalable alternative to traditional RLHF, enabling targeted improvements in complex domains like mathematics. Our technical approach, centered on a stable on-policy algorithm with KL-regularization, proved effective in guiding a base model towards specialized capabilities without catastrophic forgetting. The models learned to adhere to a required output format and demonstrably improved their ability to solve grade-school math problems. The comparison between model scales yielded a clear result: the larger 1.5B parameter model significantly outperformed the 0.5B version

The empirical results from two distinct tasks validated our methodology. The comparison between model scales on GSM8K yielded a clear result: the larger 1.5B parameter model significantly outperformed the 0.5B version, reinforcing the principle that sufficient model capacity is a key driver for success in complex reasoning tasks. The qualitative analysis further investigated the learning trajectory, showing a clear, iterative progression from random token generation to coherent, multi-step logical deduction. Future improvements could involve instead of binary 0 or 1 rewards both for format and correctness, using more discrete Reinforcement Learning Rewards (to reward taking correct intemediary steps too) and extensive hyperparameter tuning for larger models. Applying this methodology with further enhancements to even more challenging scientific and logical reasoning datasets would provide more interesting insights.

References

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. Training verifiers to solve math word problems. In *arXiv preprint arXiv:2110.14168*, 2021.

Jiayi Pan. Countdown-tasks-3to4 dataset. https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4, 2024.

Jiayi Pan et al. nano-aha-moment: A simple codebase for rl on llms. https://github.com/McGill-NLP/nano-aha-moment, 2024.

Shengann Zuo, Weize Chen, Yue Wu, Zheyang Zhuang, Zirui Liu, Peng-Bo Tan, Zidong Du, Qiang Wang, Zhipeng Chen, Wen kai Li, Hong sheng Li, Dahua Lin, and Yang Gao. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.