



NO SUPERVISION, NO PROBLEM: PURE REINFORCEMENT LEARNING IMPROVES MATHEMATICAL REASONING IN SMALL LANGUAGE MODELS

Bachelor's Project Thesis

Quinten Steringa, s5098327, q.e.steringa@student.rug.nl,
Supervisor: Rafael F. Cunha

Abstract: This study explores whether pure reinforcement learning (RL), without supervised fine-tuning (SFT), can improve the mathematical reasoning ability of small language models. Using Group Relative Policy Optimization (GRPO), four pre-trained Qwen model variants were post-trained using only RL on a subset of the GSM8K dataset. Models specialized in mathematical reasoning, such as Qwen2.5-Math-1.5B and Qwen2-Math-1.5B, achieved significant improvements in pass@1 accuracy compared to their baselines. General-purpose models showed modest improvements, while a smaller 0.5B model suffered a performance drop, revealing capacity limitations when optimizing multiple objectives. Notably, a direct comparison showed that pure RL outperformed the conventional SFT-to-RL approach in both accuracy and training efficiency under a fixed maximum output token limit. The experimental results demonstrate that pure RL can effectively improve reasoning ability when sufficient domain specialization and model capacity are present, potentially eliminating the need for costly SFT in resource-limited settings.

1 Introduction

Large language models (LLMs) have become part and parcel of contemporary life, demonstrating extraordinary capabilities in recent years across a variety of tasks (Zhong et al., 2024; Ahn et al., 2022; Zhao et al., 2023). Training these LLMs can be divided into two main stages: pre-training and post-training. During pre-training, models are trained on large amounts of text to learn common patterns. Post-training refines a pre-trained language model to better align it with user intent (Tie et al., 2025).

A common post-training method is Reinforcement Learning from Human Feedback (RLHF), which enhances alignment by using human input. Typically, RLHF consists of three steps: supervised fine-tuning (SFT) using human demonstrations, creating a reward model from human preferences, and optimizing the model using reinforcement learning (RL) (Ouyang et al., 2022). While effective, RLHF is both computationally expensive and time-consuming.

DeepSeek-R1-Zero (Guo et al., 2025) has demonstrated that applying a pure RL approach for post-

training, without SFT, can lead to remarkable reasoning abilities. This is achieved by using Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to optimize the LLM, and not Proximal Policy Optimization (PPO) (Schulman et al., 2017) as is done in RLHF. However, DeepSeek-R1-Zero had issues with readability and fluency. To address these drawbacks, DeepSeek-R1 was introduced, which incorporates a small amount of cold-start SFT before further RL training. Subsequently, this model was used to generate reasoning samples to train smaller models via SFT. In their paper, the DeepSeek team noted that they did not apply RL for these distilled models, leaving this for the wider research community.

The success of DeepSeek's approach demonstrates the promise of RL and the importance of effective reward mechanisms, especially for complex reasoning tasks. Mathematical reasoning has proven to be an ideal domain for studying process-based learning approaches because such tasks inherently require stepwise solutions. Uesato et al. (2022) presented two methods for training reward

models. First, Outcome-supervised Reward Models (ORMs), which are trained by evaluating only the final response. And second, Process Reward Models (PRMs), where the model is trained by providing feedback for intermediate steps. OpenAI’s research (Lightman et al., 2023) confirmed that process supervision performs significantly better than outcome supervision for training models to solve challenging math problems, establishing mathematical reasoning as a key benchmark for process-based paradigms.

Several studies have focused on implementing PRMs for mathematical problem-solving (Xu et al., 2025). MATH-SHEPHERD (Wang et al., 2023) leverages process-reward paradigms using automatically constructed process-wise supervision data, demonstrating improved performance on benchmarks like GMSK8 (Cobbe et al., 2021) when applied to models such as Mistral-7B (Jiang et al., 2023). Similarly, DeepSeekMath (Shao et al., 2024) integrates PRM within the GRPO algorithm, achieving strong results on mathematical benchmarks by evaluating intermediate steps.

Recent advances in exploring RL-based fine-tuning inspired by DeepSeek-R1 (Luo et al., 2025; RUCAIBox STILL Team, 2025) and its GRPO framework still rely on expansive datasets or require substantial computational resources. To address this, Dang & Ngo (2025) used RL on a model that had been previously distilled (SFT-trained). They showed that, with a carefully curated dataset, RL fine-tuning can yield significant improvements in smaller language models with few computational resources. However, their study does not isolate the specific effect of RL because they applied RL to a model that had already undergone SFT.

This study addresses a critical gap in the current research by applying pure RL using GRPO to unmodified pre-trained models in resource-limited settings. The aim of this research is to evaluate whether RL alone can improve the mathematical reasoning abilities of such models, without the confounding influence of prior SFT.

2 Methods

This section outlines the experimental methodology adopted in this study. It details the RL algorithm selected for post-training, describes the

model and dataset configurations, and explains the reward structure, prompting strategy, evaluation metric, and experimental setup.

2.1 Algorithm Choice

The same RL algorithm that was used to successfully train DeepSeek-R1-Zero (Guo et al., 2025) was also applied here. This algorithm is GRPO (Shao et al., 2024), which does not require a separate critic network. For each question q , GRPO generates G outputs $\{o_1, o_2, \dots, o_G\}$ using the old policy $\pi_{\theta_{\text{old}}}$, where the old policy refers to the model’s parameters from the previous iteration. Here, $o_i \in O$ represents the i -th sampled output for a given question, where O denotes the space of possible model outputs, and G is the number of sampled outputs per question. The new policy π_{θ} is updated by maximizing the objective:

$$\begin{aligned} \mathcal{J}_{GRPO}(\theta) = \mathbb{E} & \left[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(O|q) \right] \\ & \frac{1}{G} \sum_{i=1}^G \left(\min(r_i A_i, \text{clip}(r_i, 1 - \epsilon, 1 + \epsilon) A_i) \right. \\ & \left. - \beta \mathbb{D}_{KL}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \end{aligned}$$

Where $P(Q)$ is the distribution of training questions, ϵ is a hyperparameter that limits policy updates, β is a hyperparameter that regulates policy divergence and r_i is the probability ratio of the new and old policies:

$$r_i = \frac{\pi_{\theta}(o_i|q)}{\pi_{\text{old}}(o_i|q)}$$

The term $\mathbb{D}_{KL}(\pi_{\theta} \parallel \pi_{\text{ref}})$ acts as a regularization penalty where π_{ref} is the frozen initial reference policy. It is computed as:

$$\mathbb{D}_{KL}(\pi_{\theta} \parallel \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_{\theta}(o_i|q)} - \log \frac{\pi_{\text{ref}}(o_i|q)}{\pi_{\theta}(o_i|q)} - 1$$

The advantage A_i measures how good an output o_i is relative to the group. It is computed by standardizing the reward R_i (obtained for output o_i) within the set of all G rewards $\{R_1, R_2, \dots, R_G\}$ sampled for the given question, as:

$$A_i = \frac{R_i - \text{mean}(\{R_1, R_2, \dots, R_G\})}{\text{std}(\{R_1, R_2, \dots, R_G\})}$$

2.2 Model and Dataset

This study examines the impact of GRPO on four different open-source language models that vary in version, parameter scale and domain specialization. The models investigated include Qwen2.5-Math-1.5B, Qwen2-Math-1.5B, Qwen2.5-1.5B, and Qwen2.5-0.5B.

Qwen2.5-Math-1.5B (Yang, Zhang, et al., 2024) is a math-specific pre-trained large language model with 1.5 billion parameters. Qwen2-Math-1.5B is an earlier generation specialized mathematical reasoning model (Yang, Yang, Zhang, et al., 2024). In addition, Qwen2.5-1.5B and Qwen2.5-0.5B are included, which are general-purpose language models with 1.5 billion and 0.5 billion parameters, respectively, without specific optimization for mathematical reasoning (Yang, Yang, Hui, et al., 2024).

All the GRPO-trained models are compared with the original pre-trained models without RL. The only difference between the GRPO-trained and baseline models is the presence or absence of RL fine-tuning. This isolates the impact of RL on mathematical reasoning performance.

To directly address the central research question of whether SFT is required prior to RL, a comparison is included between the Qwen2.5-Math-1.5B model trained purely with GRPO and the supervised fine-tuned DeepSeek-R1-Distill-Qwen-1.5B model (Guo et al., 2025), henceforth referred to as SFT-DeepSeek. This controlled comparison isolates the specific contribution of the SFT step in the post-training pipeline, as the two models differ only in the presence or absence of SFT prior to RL.

For post-training and evaluation, all models were trained and tested on the GSM8K dataset (Cobbe et al., 2021). This is a dataset of 8.5K linguistically diverse mathematical word problems in primary schools that require reasoning in multiple steps, making it ideal for assessing mathematical problem-solving skills. Hugging Face’s built-in train/test division is used to split the GSM8K dataset, with a training set of 7,473 problems and a test set of 1,319 problems.

2.3 Reward Structure

All models were trained using the same rule-based reward structure used by Guo et al. (2025) to train DeepSeek-R1-Zero, consisting of two types of rewards. The first kind of reward is a format reward, which ensures that the model structures its response correctly. The thinking process must be enclosed within “<reasoning>” tags, and the final answer must be enclosed within “<answer>” tags. If the model manages to do this correctly, the model is given a reward of 0.4. The second type of reward is an accuracy reward. All numeric tokens are extracted from the answer to ensure that the correct value is not overlooked. If a number from the answer tags matches the correct answer from the dataset, a reward of 0.6 is assigned. However, the answer given by the model is only checked if the formatting is done properly. In this way, the model first learns the correct formatting before focusing on accuracy. The total reward is calculated by the sum of the format reward and the accuracy reward. While not novel in itself, this reward structure provides clear signals suitable for assessing RL’s impact within the comparative experimental setup.

2.4 Prompting Strategy

Each model was prompted using a reasoning-first template, in a format similar to chain-of-thought prompting (CoT) (Wei, Wang, et al., 2022). The template instructs the model to first make its step-by-step reasoning process clear, respectively within tags to then provide the final answer, again within tags.

The template used differs slightly from CoT prompting in that the latter is used during inference time, while the template is integrated within the training process. Instead of a particular reasoning style being presented in the prompt itself, this will be done through RL, which will ensure that the model’s reasoning process is optimized through rewards. This approach is the same as the one used in DeepSeek-R1-Zero.

Importantly, only the GRPO-trained models were exposed to this instructional format during training and evaluation. Baseline models were evaluated with only the raw questions from the dataset. This setup reflects the fact that baseline models were not trained to follow reasoning templates and

were evaluated purely for comparison.

2.5 Evaluation Metric

To evaluate all GRPO-trained models, the pass@1 metric was used, following Guo et al. (2025). This metric is derived from the more general pass@k evaluation framework introduced by Chen et al. (2021), which provides a robust measure of model performance when sampling is used. Instead of relying on deterministic greedy decoding, where correctness is evaluated based solely on the first generated answer, n outputs per question are now sampled using a temperature of $T = 0.6$ and top- p sampling with $p = 0.95$. The pass@1 accuracy is then calculated as:

$$\text{pass@1} = \frac{1}{n} \sum_{i=1}^n p_i$$

where p_i denotes the correctness (0 or 1) of the i -th response.

2.6 Experimental Setup

All experiments were conducted on a high-performance computing (HPC) cluster. Jobs were executed on a single GPU node with 8 CPU cores and 1 NVIDIA A100 GPU (40GB). The implementation was built with the Hugging Face libraries: Datasets, Transformers and TRL. Source code is available at: <https://github.com/Dichotoom/Bachelor-Project>.

A shuffled subset of the GSM8K dataset was used, consisting of 240 training examples and 60 testing examples. This smaller subset was chosen to accommodate the multiple experimental runs within the available computational budget on the HPC cluster, aiming to demonstrate the core effects of pure RL in a resource-constrained scenario. For training, a batch per device of 1 was used, with gradient accumulation over 8 steps, giving an effective batch size of 8. The learning rate was set at 2×10^{-5} , and models were trained for 2 epochs. For each training example, 6 generations were produced, and the maximum completion length was set to 300 tokens. Full hyperparameter details are provided in Section B.

During testing, the last two numbers of the baseline answer were extracted to avoid numerical ambiguity. For example, models may output both “132 ingredients are needed to make 16 muffins” and “The number of ingredients to make 16 muffins is 132.” Extracting the final two numbers ensures that the correct answers are captured consistently regardless of phrasing. The answers of the GRPO post-trained model were extracted in the same way as during training, that is, all numbers in the answer tags. A maximum completion length of 2048 tokens was used during testing to align with the study’s focus on performance under resource-constrained conditions.

To ensure statistical reliability, all experiments were repeated in 5 independent runs. During the training phase, a different random seed was assigned for each run, which controlled the composition of the training subset. After training for one run, the model was evaluated on a fixed test set. The “Total Training Time (h)” reported in Table 3.2 represents the complete wall-clock duration for each GRPO job, encompassing both the iterative model training steps and these periodic evaluations on the test set. After the 5 runs were complete, the mean Pass@1 performance was calculated to obtain a more reliable estimate of the generalization ability of the model. Similarly, the baseline model was independently evaluated 5 times on the same test set to allow a fair comparison.

3 Results

This study trained several Qwen model variants with the GRPO algorithm. As shown in Table 3.1, the application of GRPO provided notable performance improvements for the majority of models tested. The most significant improvements were observed in the models specialized in mathematical domains, with Qwen2.5-Math-1.5B showing a 9.6% improvement and Qwen2-Math-1.5B showing a 13.1% improvement. The general-purpose Qwen2.5-1.5B model showed a modest improvement of 4.4%. In contrast, the smaller Qwen2.5-0.5B model experienced a substantial 74.2% decrease in performance.

Figure 3.1 shows training progression across 3,000 steps, organized in three comparative analyses. Four key metrics were tracked: Total Reward

Table 3.1: Pass@1 accuracy before and after Group Relative Policy Optimization (GRPO) for four Qwen model variants. Scores are mean \pm standard deviation over five runs; all evaluations used a 2048-token output limit. Domain-specific models (Qwen2-Math-1.5B and Qwen2.5-Math-1.5B) achieve the greatest performance increase after training. In contrast, the Qwen2.5-0.5B model deteriorates in performance, highlighting model capacity as a limiting problem.

Model	Baseline Pass@1	Post-GRPO Pass@1	Relative Improvement
Qwen2.5-Math-1.5B	0.7183 ± 0.0103	0.7872 ± 0.0129	9.6%
Qwen2-Math-1.5B	0.6800 ± 0.0114	0.7689 ± 0.0275	13.1%
Qwen2.5-1.5B	0.6533 ± 0.0166	0.6822 ± 0.0301	4.4%
Qwen2.5-0.5B	0.4472 ± 0.0114	0.1156 ± 0.0946	-74.2%

(maximum 1.0), Format Reward (maximum 0.4), Correct Answer Reward (maximum 0.6), and Completion Length.

The first row compares two models specialized in the mathematical domain: Qwen2.5-Math-1.5B and Qwen2-Math-1.5B. Both models showed similar performance trajectories in each subplot. Format rewards for both models converged within the 0.36–0.38 range, approaching the maximum possible value of 0.4. Qwen2.5-Math-1.5B consistently outperformed Qwen2-Math-1.5B in the Correct Answer Reward subplot. The length of completion for both models converged to 90–100 tokens after an initial drop from the higher initial values.

The second row examines domain specialization and parameter size effects across Qwen2.5 variants: Qwen2.5-Math-1.5B, Qwen2.5-1.5B, and Qwen2.5-0.5B. Both the general Qwen2.5-1.5B and the domain-specialized Qwen2.5-Math-1.5B showed similar format reward convergence trajectories, stabilizing between 0.35 and 0.38 after about 800 training steps. However, Qwen2.5-Math-1.5B consistently outperformed the general-purpose model in the Correct Answer Reward subplot, with values near 0.44 versus 0.39 for the general model.

The smaller Qwen2.5-0.5B model exhibited markedly different behavior across all metrics. The correct answer reward remained much lower and stabilized at about 0.06, although it reached format reward parity with the larger models. This resulted in a lower overall reward of about 0.45 compared

to the larger models, which approached 0.8. In addition, this model generated completions of about 30 tokens after convergence, which is significantly shorter than the 90–100 tokens generated by the 1.5B parameter models.

The third row of Figure 3.1 illustrates the impact of SFT training by comparing Qwen2.5-Math-1.5B with SFT-DeepSeek. The SFT-DeepSeek model showed similar training dynamics to the non-SFT Qwen2.5-Math-1.5B model, but with slightly lower reward convergence across all metrics, particularly in the Correct Answer Reward where it stabilized around 0.34 compared to 0.39 for the non-SFT version. Notably, the SFT-DeepSeek model also produced substantially longer completions, converging at around 170 tokens.

As shown in Table 3.2, the Qwen2.5-Math-1.5B model without prior SFT achieved a higher final post-GRPO Pass@1 score compared to the SFT-DeepSeek model. Moreover, the non-SFT model completed training in approximately 4.9 hours, whereas the SFT model completed training in approximately 14.4 hours. This indicates a remarkable difference in training efficiency and performance between the two models.

Table 3.2: Pass@1 accuracy before and after Group Relative Policy Optimization (GRPO). Compares the base Qwen2.5-Math-1.5B model with its SFT-trained counterpart, DeepSeek-R1-Distill-Qwen-1.5B (SFT-DeepSeek). Scores are mean \pm standard deviation over five runs; all evaluations used a 2048-token output limit. The base model trained directly with GRPO achieved higher final accuracy with significantly less total training time compared to the SFT model fine-tuned with GRPO.

Model	Baseline Pass@1	Post-GRPO Pass@1	Total Training Time (h)
Qwen2.5-Math-1.5B	0.7183 ± 0.0103	0.7872 ± 0.0129	4.9
SFT-DeepSeek	$0.4672 \pm 0.0191^*$	0.5911 ± 0.0409	14.4

*Increasing the output token limit to 4096 improved the SFT-DeepSeek model’s Pass@1 accuracy from 0.4672 ± 0.0191 to 0.5489 ± 0.0048 , highlighting its sensitivity to the maximum number of tokens it is allowed to generate. The 2048-token output limit used in this study reflects a deliberate focus on resource-constrained performance.

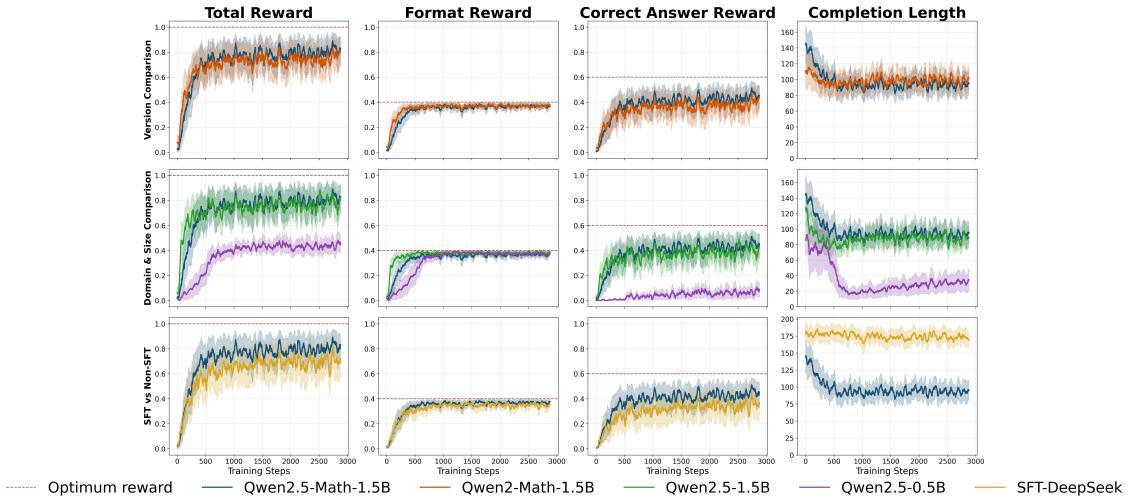


Figure 3.1: Comparison of training dynamics across different model variants over 3,000 Group Relative Policy Optimization (GRPO) optimization steps. The figure is organized in three rows of comparative analyses: (1) version comparison between Qwen2.5-Math-1.5B and Qwen2-Math-1.5B; (2) domain specialization and parameter scale effects across Qwen2.5 variants (Math-1.5B, 1.5B, and 0.5B); and (3) the impact of SFT comparing Qwen2.5-Math-1.5B with DeepSeek-R1-Distill-Qwen-1.5B (SFT-DeepSeek). Each column shows a different metric: Total Reward (maximum 1.0), Format Reward (maximum 0.4), Correct Answer Reward (maximum 0.6), and Completion Length. The mean of five separate runs, smoothed over 41 steps, is shown by solid lines; the standard error is shown by shaded areas. Results show that GRPO improves reasoning performance consistently across model versions (row 1), that smaller models face capacity constraints (row 2), and that prior supervised fine-tuning (SFT) may marginally reduce reward optimization during GRPO training (row 3).

4 Discussion

The significant improvements in Qwen2.5-Math-1.5B (9.6% increase) and Qwen2-Math-1.5B (13.1% increase) show that domain specialization plays a crucial role in the effectiveness of GRPO post-training. The slight improvement in the general model Qwen2.5-1.5B (a 4.4% increase) further supports this interpretation. Although this model is capable of general reasoning, it lacks the specific mathematical structures needed to maximize the effectiveness of GRPO post-training.

Despite the specific architectural differences between the model versions, the similar performance trajectories of Qwen2.5-Math-1.5B and Qwen2-Math-1.5B imply that GRPO improves mathematical reasoning ability. This result suggests that the method improves domain-specific knowledge representations rather than general problem-solving skills.

The performance drop of the Qwen2.5-0.5B model (74.2% decrease) highlights how crucial model capacity is when using GRPO for post-training. This smaller model faced competing demands of mathematical accuracy and format rewards, which created competing pressures that exceeded the model’s representational capabilities. Such limitations are consistent with research showing that complex reasoning abilities often emerge at larger model scales (Wei, Tay, et al., 2022), making smaller models more susceptible to interference when multiple objectives are pursued simultaneously during fine-tuning (Touvron et al., 2023). The model preferred the simpler task of format following to mathematical reasoning, as evidenced by the successful convergence of format and poor mathematical accuracy.

The capacity limitation was also evident in the completion length metric. The 0.5B model generated answers that were roughly one-third as long

as those of its larger counterparts. Although the shorter answers followed the format, they lacked the necessary intermediate steps in reasoning to solve the mathematical problems.

The most significant finding of this study challenges the conventional SFT-to-RL pipeline. In small-resource settings, pure RL outperformed SFT models that are followed by RL fine-tuning. Specifically, the base model, Qwen2.5-Math-1.5B, which was directly trained with GRPO achieved higher accuracy while also requiring less total training time by producing shorter responses compared to the SFT-DeepSeek model.

The efficiency gap is mainly due to the verbosity of SFT models. The SFT-DeepSeek model consistently produces longer and more repetitive outputs. This is consistent with Zhang et al. (2025), who demonstrated that SFT models fail to find the optimal stopping point. Further tests with a higher maximum completion length confirmed the lack of an optimal stopping point, with Pass@1 accuracy increasing from 0.4672 ± 0.0191 to 0.5489 ± 0.0048 when the output token limit was increased from 2048 to 4096 tokens. This suggests that SFT models require larger output budgets to reach peak performance, which compounds their inefficiency in constrained environments.

Beyond just output length, empirical evidence from this study further highlights the baseline SFT-DeepSeek’s strong prompt sensitivity. To characterize the inherent behavior of the SFT model before GRPO, this study evaluated this particular baseline variant with different prompt styles. The model achieved 0.6706 ± 0.0170 accuracy when prompted using the structured format described in Section 2.4, but dropped to 0.4672 ± 0.0191 accuracy when given only the raw question. This prompt dependence reveals another limitation of SFT models in achieving consistent reasoning performance, offering additional support for the findings on SFT inefficiencies.

When GRPO was applied to the SFT-DeepSeek model, performance improved to 0.5911 ± 0.0409 compared to its SFT baseline, but remained substantially below the performance of the base Qwen2.5-Math-1.5B model trained directly with GRPO. This counterintuitive limitation is consistent with Zhang et al. (2025), who found that GRPO without length-aware reward functions inherits the verbosity and inefficiencies of SFT mod-

els, and can reinforce them without actually reducing them.

Unlike previous work (Dang & Ngo, 2025), which applied RL to models that had already undergone SFT through distillation and therefore could not isolate the effects of RL alone, the findings of the current study demonstrate that GRPO can effectively optimize reasoning directly from a pre-trained checkpoint. This validates a more streamlined and cost-effective training pipeline, that eliminates the SFT bottleneck.

5 Limitations and Future Work

During the experiments, several hyperparameters were left unchanged. For instance, the format reward was always set to 0.4, although this may not be the optimal value. Similarly, all model variations used the same learning rate and other optimization parameters derived from previous research. Further performance improvements might be achievable through more hyperparameter tuning, especially for the smaller model, where competing optimization objectives presented significant challenges.

Since training was conducted on a HPC cluster, where only a few GPUs were available, resource constraints further limited the research. These constraints also necessitated the use of a smaller subset of the GSM8K dataset to manage the experimental load. Consequently, these computational limitations made it infeasible to extend the approach to larger models (7B+ parameters) or the full dataset at this stage, and occasionally caused long queuing times, reducing the number of experiments that could be performed.

Despite these limitations, several promising directions for future research have been identified:

Research on larger models. Applying similar GRPO post-training approaches to larger models (7B, 72B parameters) to determine whether performance improvements scale with model capacity.

Cross-domain applicability. Applying the GRPO framework to non-mathematical reasoning tasks to determine its generalizability to other specialized domains. One such domain is coding, which, like mathematics, provides structured prob-

lems with verifiable rewards.

Managing objective conflicts in smaller models. Exploring techniques for balancing competing optimization objectives in smaller models without performance degradation.

6 Conclusion

This research has shown that GRPO post-training can significantly improve mathematical reasoning capacity in domain-specific models, with improvements ranging from 9.6 to 13.1%. Two critical elements have been identified that must be present for GRPO post-training to be effective. First, sufficient model capacity is essential, as evidenced by the poor performance of the 0.5B parameter model. Second, domain specialization plays a crucial role, as evidenced by the finding that mathematically oriented models perform more than twice as well as general models.

A key finding of this study challenges conventional training pipelines: pure RL training outperforms the standard SFT-to-RL approach in small-resource settings. Results showed that a purely RL-trained model performed better in accuracy while requiring less total training time due to generating more efficient responses. In contrast, a model distilled with SFT, inherently produced verbose and often repetitive outputs, which led to greater computational overhead.

Overall, these results indicate that pure RL can be an efficient approach for improving mathematical reasoning ability when applied to pre-trained models with appropriate capacity and specialization, potentially without the need for costly SFT in resource-limited settings.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., ... others (2022). Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... others (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., ... others (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dang, Q.-A., & Ngo, C. (2025). Reinforcement learning for reasoning in small llms: What works and what doesn't. *arXiv preprint arXiv:2503.16219*.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., ... others (2025). Deepseekr1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., ... Sayed, W. E. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., ... Cobbe, K. (2023). Let's verify step by step. In *The twelfth international conference on learning representations*.

Luo, M., Tan, S., Wong, J., Shi, X., Tang, W. Y., Roongta, M., ... Stoica, I. (2025). Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. <https://github.com/agentica-project/deepscaler>.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... others (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35, 27730–27744.

RUCAIBOX STILL Team. (2025). Still-3-1.5b-preview: Enhancing slow thinking abilities of small models through reinforcement learning. <https://github.com/RUCAIBOX/SlowThinkingwithLLMs>.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., ... others (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Tie, G., Zhao, Z., Song, D., Wei, F., Zhou, R., Dai, Y., ... others (2025). A survey on post-training of large language models. *arXiv preprint arXiv:2503.06072*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... others (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., ... Higgins, I. (2022). Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., ... Sui, Z. (2023). Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... others (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... others (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35, 24824–24837.

Xu, F., Hao, Q., Zong, Z., Wang, J., Zhang, Y., Wang, J., ... others (2025). Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., ... others (2024). Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., ... others (2024). Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., ... others (2024). Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Zhang, X., Huang, Z., Ni, C., Xiong, Z., Chen, J., & Oymak, S. (2025). *Making small language models efficient reasoners: Intervention, supervision, reinforcement*. Retrieved from <https://arxiv.org/abs/2505.07961>

Zhao, L., Zhang, L., Wu, Z., Chen, Y., Dai, H., Yu, X., ... others (2023). When brain-inspired ai meets agi. *Meta-Radiology*, 100005.

Zhong, T., Liu, Z., Pan, Y., Zhang, Y., Zhou, Y., Liang, S., ... others (2024). Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*.

A Comparative Analysis of Response Quality

The following example from GSM8K compares the output of the GRPO-trained Qwen2.5-Math-1.5B model to its baseline variant. The GRPO model shows a well-structured way of reasoning, clearly addressing each part of the problem. In contrast, the base model frames its answer as a Python code fragment, which is unnecessarily complex for the given task and less easy to interpret in a reasoning context. Moreover, the base model introduces several computational errors, including misinterpreting the number of carrot bags and incorrectly applying multipliers to both carrot and sugar cube weights. These errors offset each other, resulting in an accidentally correct final answer.

A.1 GSM8K Problem:

A farmer is buying feed for his horses. He buys a variety of hay, oats, carrots and sugar cubes. Since sugar cubes are a rare treat, he only buys two 1-pound boxes of them for the whole stable. He only wants enough carrots to feed the horses while the vegetables are fresh, so he buys four 12-pound bags. Hay is the main diet of his horses, so he buys forty-two 75-pound bales. Oats are a staple to supplement the hay, so he buys twenty 65-pound sacks. If his farm truck can carry 2250 pounds at a time, how many trips does the farmer need to transport all the feed?

A.2 Qwen2.5-Math-1.5B (GRPO-trained):

```
<reasoning>
1. The farmer buys 2 bags of sugar cubes, so that's  $2 \times 1$  pound = 2 pounds.
2. He buys 4 bags of carrots, so that's  $4 \times 12$  pounds = 48 pounds.
3. He buys 42 bales of hay, so that's  $42 \times 75$  pounds = 3150 pounds.
4. He buys 20 sacks of oats, so that's  $20 \times 65$  pounds = 1300 pounds.
5. The total weight of the feed is  $2 + 48 + 3150 + 1300 = 4500$  pounds.
6. The truck can carry 2250 pounds at a time, so the farmer needs to make  $4500 / 2250 = 2$ 
trips. </reasoning>
<answer>2 trips</answer>
```

A.3 Qwen2.5-Math-1.5B (Baseline):

To determine the number of trips the farmer needs to transport all the feed, we need to calculate the total weight of the feed and then divide this by the capacity of the truck. Here's the step-by-step approach:

1. Calculate the total weight of the carrots.
2. Calculate the total weight of the hay.
3. Calculate the total weight of the oats.
4. Calculate the total weight of the sugar cubes.
5. Sum up the weights from steps 1 to 4 to get the total weight of the feed.
6. Divide the total weight of the feed by the truck's capacity and round up to the nearest whole number to get the number of trips needed.

Let's do this step-by-step in Python.

```
import math

# Given data
carrots_boxes = 2
carrots_per_box = 12 * 12 # 12 pounds per box
```

```

hay_bales = 42
hay_per_bale = 75 * 1 # 75 pounds per bale
oats_sacks = 20
oats_per_sack = 65 * 1 # 65 pounds per sack
sugar_cubes = 2
sugar_cubes_per_box = 1 * 12 # 12 pounds per box

# Calculate total weight of each type of feed
total_carrots = carrots_boxes * carrots_per_box
total_hay = hay_bales * hay_per_bale
total_oats = oats_sacks * oats_per_sack
total_sugar_cubes = sugar_cubes * sugar_cubes_per_box

# Calculate the total weight of the feed
total_weight = total_carrots + total_hay + total_oats + total_sugar_cubes

# Truck capacity
truck_capacity = 2250

# Calculate the number of trips needed
trips_needed = math.ceil(total_weight / truck_capacity)

print((total_weight, trips_needed))
(3398, 2)

```

Therefore, the number of trips the farmer needs to transport all the feed is 2.

B Hyperparameter Setup

Table B.1 shows the parameters that were used in the training phase.

Table B.1: GRPO Training Hyperparameters

Hyperparameter	Value
output_dir	"./grpo_math_finetune"
logging_steps	1
per_device_train_batch_size	1
gradient_accumulation_steps	8
bf16	True
num_generations	6
max_grad_norm	0.1
learning_rate	2×10^{-5}
num_train_epochs	2
max_prompt_length	256
max_completion_length	300