OCCAM'S RAZOR FOR RL FINE-TUNING: THE SIMPLE DESIGN IS EFFICIENT AND SUFFICIENT

Kehan Lan* **Yutong Huang***

ABSTRACT

Recent studies have shown that Reinforcement Learning (RL) fine-tuning on Large Language Models (LLMs) can endow models with strong reasoning abilities for complex mathematical tasks. With the recent scaling of model parameters, efficient fine-tuning methods such as LoRA have demonstrated performance comparable to full-parameter fine-tuning when applied to post-training on small datasets. However, several questions remain: (1) What LoRA capacity is the best? (2) Do various RL techniques still work? (3) Does reward design still matter? In this work, we conduct an empirical study on these three questions via experiments on the MATH dataset using Qwen3-1.7B with GRPO. We find that (1) rank-1 LoRA achieves performance comparable to that of higher-rank configurations. (2) Standard deviation normalization and a simple outcome reward consistently improve performance, whereas other techniques such as KL penalties, clipping, DAPO-style token-level loss tend to degrade it. (3) Process reward models and overlong punishment may induce reward hacking. These results suggest that in a minimal setting, introducing additional complexity is often unnecessary for achieving strong performance. And the simple design is efficient and sufficient. Our code is available at https://github.com/wishhyt/PRML_pj.

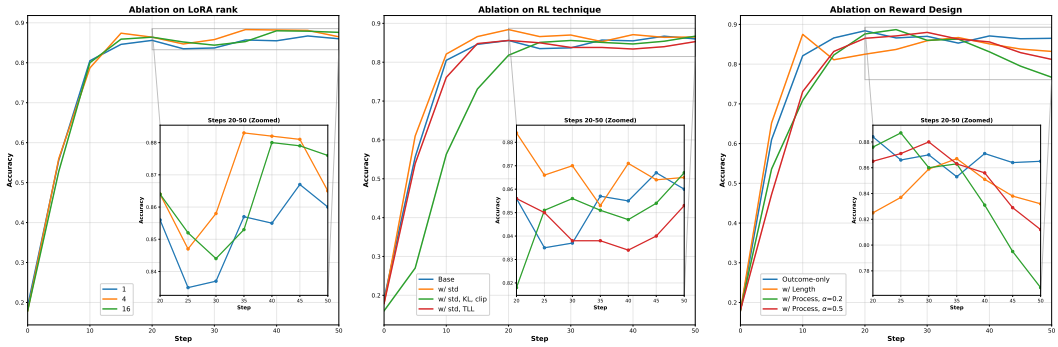

Figure 1: **Overview of experiments.** Validation accuracy versus training steps for (left) different LoRA ranks, (middle) different RL stabilization techniques, and (right) different reward designs. Each panel includes a zoom-in over steps 20–50 to highlight late-stage differences. Results show that the simple design (rank-1 LoRA, std norm and outcome reward only) is efficient and sufficient.

1 INTRODUCTION

"Entities are not to be multiplied without necessity" — *Occam's razor*

Reinforcement Learning (RL) fine-tuning (Guo et al., 2025; Yu et al., 2025; Jaech et al., 2024) has recently become a standard approach for improving the reasoning abilities of Large Language Models (LLMs), particularly on tasks such as mathematics (Balunović et al., 2025; Li et al., 2024; Hendrycks et al., 2021) and code generation (Jimenez et al., 2023). Different RL algorithms and reward shaping techniques (Yu et al., 2025; Liu et al., 2026; Shao et al., 2024; Zheng et al., 2025) have shown up, introducing much complexity. At the same time, parameter-efficient fine-tuning methods such as Low-Rank Adaptation (LoRA; Hu et al. 2022) have made it possible to adapt

*Course Project

large models using only a small number of trainable parameters, enabling RL fine-tuning to be applied even in resource-constrained settings. However, when performing RL post-training on small datasets and models with LoRA, we wonder that how much LoRA capacity is needed, and whether the complexity is necessary or it should be eliminated.

In this work, we study RL fine-tuning in a minimal setting: applying Group Relative Group Optimization (Shao et al., 2024) to Qwen3-1.7B (Yang et al., 2025) on the MATH dataset (Hendrycks et al., 2021). Within this setting, we focus on the following three questions.

How much LoRA capacity is actually needed? Low-Rank Adaptation (LoRA) enables parameter-efficient fine-tuning by introducing trainable low-rank updates to frozen model weights. While prior work has shown that LoRA can match full fine-tuning (Schulman & Lab, 2025), it remains unclear how much capacity is actually necessary in reinforcement learning settings, especially when both the model and the dataset are relatively small. In particular, it is unknown whether higher LoRA ranks provide meaningful benefits, or whether very low ranks are already sufficient to capture the policy updates induced by RL.

Do standard RL techniques still help in this setting? Modern RL pipelines for LLMs often include various stabilization and regularization techniques, such as KL penalties, clipping, and token-level policy gradient losses (Shao et al., 2024; Yu et al., 2025). While these techniques have been shown to improve stability and performance in large-scale settings, it is not clear whether they remain beneficial when the policy is parameterized by low-rank adapters and trained on limited data. It is therefore important to empirically examine whether these techniques still help, or whether they introduce unnecessary constraints that hinder learning.

What reward design fits most in this setting? Reward design plays a central role in RL fine-tuning for reasoning. Recent work has emphasized the use of verifiable rewards (RLVR) (Guo et al., 2025; Lambert et al., 2024), where rewards are computed by deterministic checkers such as programmatic verifiers or unit tests. In contrast, Reinforcement Learning with Human Feedback (RLHF) (Ouyang et al., 2022; Ziegler et al., 2019) relies on learned reward models trained from human or synthetic preference data. In our setting, we explore that whether the widely-used reward shaping designs like process rewards and length penalty are really necessary or a simple outcome reward is enough.

We address these questions through a series of controlled experiments. We systematically vary LoRA rank, training techniques, and reward designs, and evaluate their impact on reasoning performance. Our results show that **rank-1 LoRA achieves comparable results** to that of higher-rank configurations, and **only simple standard deviation normalization and an outcome reward** improve the performance, while other methods such as KL penalties, clipping (Schulman et al., 2017; Shao et al., 2024), DAPO-style (Yu et al., 2025) token-level loss, length penalties (Yu et al., 2025) and process reward models (Zhang et al., 2025; He et al., 2024) degrade it or lead to reward hacking.

2 Related Work

In this section, we focus on related work in reward design. Other experimental preliminaries are described in section 3.

Reinforcement Learning with Human Feedback (RLHF) RLHF aims to align language models with human intentions and preferences. The standard RLHF pipeline first trains a reward model to capture human preferences and then optimizes the policy model using reinforcement learning algorithms such as PPO (Schulman et al., 2017) and its variants. Subsequent studies have demonstrated the effectiveness of RLHF in improving LLMs' reasoning and instruction-following abilities (Havrilla et al., 2024; Ramamurthy et al., 2022; Glaese et al., 2022). However, traditional RLHF requires training multiple models and incurs substantial computational cost. Recent approaches, such as GRPO combined with RLVR, alleviate these issues by removing the need for separate value networks and leveraging verifiable rewards for more efficient and stable optimization.

Reinforcement Learning with Verifiable Rewards (RLVR) RLVR is an emerging paradigm in which a large language model acts as a policy, generating a chain of thought (CoT) (Wei et al., 2022; Yao et al., 2022) as a sequence of actions and receiving feedback on answer correctness from deterministic verifiers. Following the release of Deepseek-R1, RLVR has inspired a growing body of work on objective design, data curation, hyperparameter tuning, and other aspects (Liu et al., 2025; He et al., 2025; Xie et al., 2025; Chen et al., 2025; Bhaskar et al., 2025). In this work, we

explore a hybrid reward formulation that combines RLHF-style model-based rewards with RLVR-style rule-based rewards, aiming to outperform purely outcome-based RLVR.

Task-Specific Rewards Deepseek-R1 (Guo et al., 2025) has motivated a variety of downstream applications that train LLMs or multimodal LLMs (MLLMs) for diverse tasks. For example, VLM-R1 (Shen et al., 2025) and SAM-R1 (Huang et al., 2025) apply R1-style RL training to improve visual perception and grounding abilities in MLLMs. Time-R1 (Wang et al., 2025) focuses on post-training MLLMs for temporal video grounding, while Search-R1 (Jin et al., 2025) trains models to autonomously generate search queries during step-by-step reasoning with real-time retrieval. These works highlight the generality of RL across domains and often combine format-based or process-based rewards with task-specific signals such as IoU-based rewards. In our work, we focus on the MATH dataset and adopt a simple outcome-based reward in the RLVR component.

3 Methods for Comparison

In this section, we describe the theoretical foundations of the methods that will be compared in section 4.

LoRA LoRA is a parameter-efficient adaptation method that imbues a frozen pretrained model with adaptability using only a small number of trainable parameters. It freezes a pretrained weight matrix $W_0 \in \mathbb{R}^{D\times K}$ and introduces a trainable low-rank update $\Delta W = BA$, where $B \in \mathbb{R}^{D\times R}$ and $A \in \mathbb{R}^{R\times K}$, with rank $R \ll \min(D, K)$. The adapted forward pass becomes

$$h = W_0 x + BAx, \tag{1}$$

where $x \in \mathbb{R}^K$ is the input and $h \in \mathbb{R}^D$ is the output. The rank R controls the capacity of the adaptation: a larger R provides greater flexibility but increases the number of trainable parameters, while a smaller R keeps the adaptation lightweight but may limit expressiveness.

GRPO GRPO estimates advantages in a group-relative manner. For a given question–answer pair (q, a), the behavior policy $\pi_{\theta_{\text{old}}}$ samples a group of G responses $\{o_i\}_{i=1}^G$. The advantage of the i-th response is computed by normalizing the group-level rewards $\{R_i\}_{i=1}^G$:

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}. \tag{2}$$

Standard GRPO adopts a clipped objective together with an explicit KL penalty:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\Big(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\Big)\hat{A}_{i,t}\right) - \beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right)\right], \tag{3}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}. \tag{4}$$

To validate their effectiveness, we remove the standard deviation normalization and regularization terms (KL penalty and clipping). The resulting baseline objective is

$$\mathcal{J}_{\text{BASE}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(r_{i,t}(\theta)\big(R_i - \text{mean}(\{R_i\}_{i=1}^G)\big)\Big)\right]. \tag{5}$$

We find that standard deviation normalization is beneficial, whereas regularization is not.

Token-Level Policy Gradient Loss The original GRPO algorithm employs a sample-level loss, in which token-wise losses are averaged within each response and then aggregated across samples. Since all samples contribute equally, tokens in longer responses receive a smaller effective weight, which may lead to adverse effects.

DAPO addresses this issue by introducing a token-level policy gradient loss in long-CoT scenarios. Following this idea, we apply the same modification to our baseline after observing the effectiveness of standard deviation normalization:

$$\mathcal{J}_{\text{TLL}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\left(r_{i,t}(\theta)\frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}\right)\right]. \tag{6}$$

In this formulation, longer sequences have greater influence on the overall gradient update. However, in our experiments, this modification does not improve performance.

Reward Design Following standard reward design practices, we implement a verifiable outcome reward based on the boxed answer and use rule-based verifiers to determine equivalence with the ground truth:

$$R_{\text{outcome}} = \begin{cases} 1, & \texttt{is\_equivalent}(\hat{y}, y), \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

Here y denotes the ground-truth answer and $\hat{y}$ the predicted answer.

DAPO also introduces a soft overlong punishment to discourage excessively long responses. When the response length exceeds a predefined maximum, a penalty is applied according to

$$R_{\text{length}} = \begin{cases} 0, & |y| \le L_{\max} - L_{\text{cache}}, \\ \frac{(L_{\max}-L_{\text{cache}})-|y|}{L_{\text{cache}}}, & L_{\max} - L_{\text{cache}} < |y| \le L_{\max}, \\ -1, & L_{\max} < |y|, \end{cases} \quad R^{\text{length}}_{\text{combined}} = R_{\text{length}} + R_{\text{outcome}}. \tag{8}$$

Here $L_{\max}$ is the maximum length and L_{cache} is a buffer range. We find that this penalty degrades exploration.

Finally, we combine the RLVR-style outcome reward with an RLHF-style process reward to guide exploration and optimization. The process reward is defined as

$$R_{\text{process}} = \texttt{reward\_model}(x, \hat{y}), \quad R^{\text{process}}_{\text{combined}} = \alpha R_{\text{outcome}} + (1-\alpha)R_{\text{process}}, \tag{9}$$

where x denotes the input question and the reward is computed by an off-the-shelf reward model. We find that introducing process reward model might lead to reward-hacking.

The Best Design Based on our empirical findings, only standard deviation normalization and simple outcome reward are retained. The final training objective is

$$\mathcal{J}_{\text{BEST}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(r_{i,t}(\theta)\frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}\right)\right], \tag{10}$$

with the best reward

$$R_{\text{best}} = R_{\text{outcome}}. \tag{11}$$

4 Experiments

4.1 Implementation Details

We conduct all experiments using the Qwen3-1.7B (Yang et al., 2025) model on the MATH dataset. We use all 7,500 training samples and take the first 1,000 test samples as the evaluation set. We apply LoRA adapters to all projection layers, with the scaling factor $\alpha = 32$. The learning rate is fixed to 9×10^{-5} and the Adam optimizer (Kingma, 2014) is used for training. At each training step, we sample 32 problems and generate 8 responses for each problem. Notably, we use vLLM (Kwon et al., 2023) for efficient response generation, which supports loading LoRA adapters. We use Skywork-o1-Open-PRM-Qwen-2.5-1.5B (He et al., 2024) as the process reward model. Due to limited resources, we run 50 GRPO training steps and perform evaluation every 5 steps. All experiments are conducted on three NVIDIA RTX A6000 GPUs with 48 GB memory each.

Table 1: **Study on LoRA.** Evaluation accuracy (%) on the MATH validation set. Values in **bold** indicate the highest in the column, while values underlined indicate the second highest.

LoRA rank	GRPO Steps									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
1	55.6	**80.5**	84.6	85.6	83.5	83.7	85.7	85.5	86.7	86.0
4	**56.0**	78.7	**87.4**	**86.4**	84.7	**85.8**	**88.3**	**88.2**	**88.1**	86.5
16	52.8	80.0	85.9	**86.4**	**85.2**	84.4	85.3	88.0	87.9	**87.6**

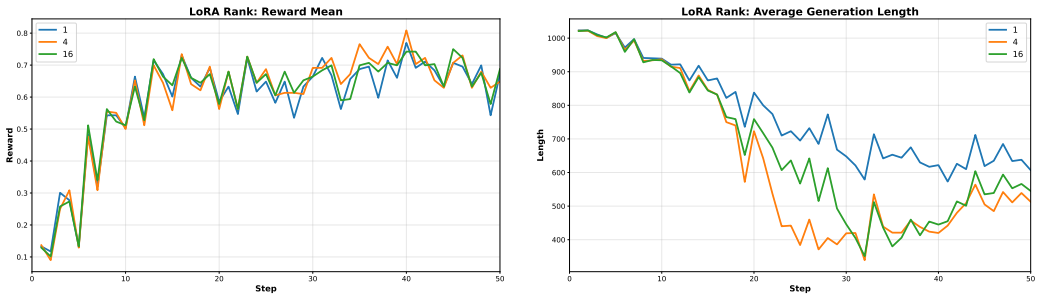

Figure 2: **Effect of LoRA rank on training dynamics.** Mean group reward (left) and average generation length (right) across GRPO steps for LoRA ranks 1, 4, 16.

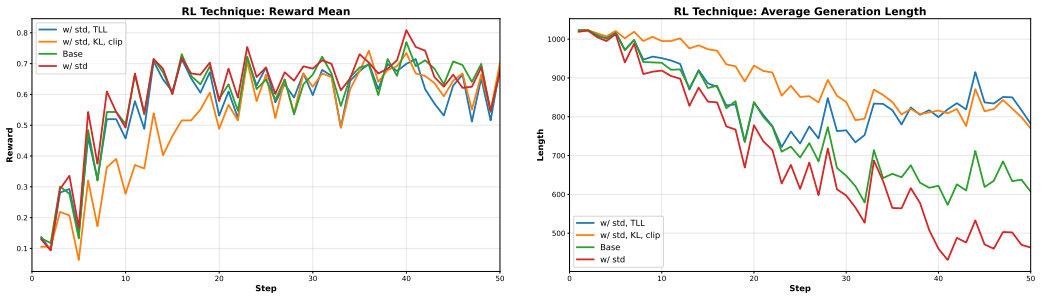

Figure 3: **Effect of RL techniques on training dynamics.** Mean group reward (left) and average generation length (right) over GRPO steps for baseline (mean-subtracted), adding standard-deviation normalization, adding KL + clipping, and token-level loss (TLL).

4.2 LoRA CAPACITY

We set the LoRA rank to 1, 4, and 16 to investigate whether higher ranks provide meaningful benefits. For a fair comparison, we used the baseline algorithm described in equation 5. As shown in Table 1, the rank-4 configuration achieves the highest accuracy across most GRPO training steps, reaching up to **88.3%** at 35 steps, outperforming both rank-1 and rank-16. Rank-1 performs slightly worse than rank-4, with its peak accuracy at **86.7%** at 45 steps, but the difference is relatively small (around 1–2%). The trajectory of mean rewards and response length shown in Figure 2 further prove that. Meanwhile, rank-1 requires fewer trainable parameters, which makes it more efficient in terms of both memory and computation. Considering this trade-off between performance and efficiency, we adopt rank-1 in the subsequent experiments.

4.3 RL TECHNIQUES

We compared four different configurations using rank-1 LoRA for a fair comparison: (1) the baseline algorithm, which applies only mean normalization as described in equation 5; (2) adding standard deviation normalization (std norm) as in equation 10; (3) incorporating KL divergence and advantage clipping on top of std norm, as in equation 3; and (4) adding a DAPO-style token-level loss (TLL) to std norm, as in equation 6.

Table 2: **Study on RL techiques.** Evaluation accuracy (%) on the MATH validation set. Values in **bold** indicate the highest in the column, while values underlined indicate the second highest.

RL technique	GRPO Steps									
	5	10	15	20	25	30	35	40	45	50
Base	55.6	80.5	84.6	85.6	83.5	83.7	**85.7**	85.5	**86.7**	86.0
w/ std	**61.0**	**82.1**	**86.6**	**88.4**	**86.6**	**87.0**	85.3	**87.1**	86.4	86.5
w/ std, KL, clip	27.0	56.3	73.1	81.8	85.1	85.6	85.1	84.7	85.4	**86.7**
w/ std, TLL	53.9	76.1	84.8	85.6	85.0	83.8	83.8	83.4	84.0	85.3

Table 3: **Study on Reward Design.** Evaluation accuracy (%) on the MATH validation set. Values in **bold** indicate the highest in the column, while values underlined indicate the second highest.

Reward Design	GRPO Steps									
	5	10	15	20	25	30	35	40	45	50
Outcome-only	61.0	82.1	**86.6**	**88.4**	86.6	87.0	85.3	**87.1**	**86.4**	**86.5**
w/ Length	**65.2**	**87.5**	81.1	82.5	83.7	85.9	**86.7**	85.1	83.8	83.2
w/ Process, $\alpha = 0.2$	53.6	70.9	82.3	87.6	**88.7**	86.0	86.3	83.1	79.5	76.7
w/ Process, $\alpha = 0.5$	47.2	73.1	83.2	86.5	87.1	**88.0**	86.3	85.6	82.9	81.2

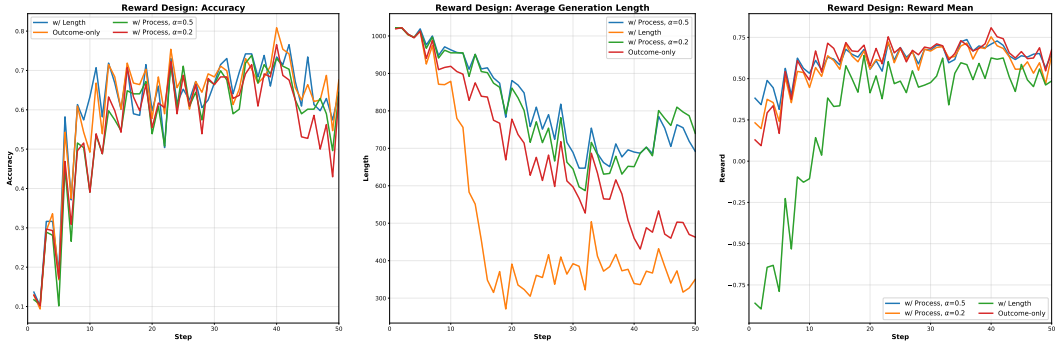

Figure 4: **Effect of reward design on training dynamics.** Training accuracy (left), average generation length (middle), and mean group reward (right) versus GRPO steps for outcome-only reward, adding overlong length penalty, and combining outcome reward with process reward (α = 0.2 / 0.5).

The baseline algorithm converges moderately, achieving evaluation accuracies up to 86.7% (shown in Table 2 and Figure 3). Introducing std norm accelerates convergence and consistently improves performance, reaching the highest evaluation accuracies across most GRPO steps, with a peak of **88.4%** at 20 steps. Adding KL divergence and advantage clipping, however, substantially slows convergence, especially in the early stages, although final evaluation accuracy eventually reaches competitive levels (e.g., 86.7% at 50 steps). Incorporating TLL slightly slows convergence compared to std norm alone, and the average generation length remains almost constant in the later stages. This suggests that simply encouraging longer outputs does not effectively improve accuracy.

4.4 Reward Design

Based on the finding that standard deviation normalization yields the best performance, we investigated different reward designs: (1) outcome-only reward as described in equation 11; (2) incorporating an overlong length penalty (with $L_{\text{max}} = 1024$ and $L_{\text{cache}} = 512$) as in equation 8; (3) process reward with $\alpha = 0.2$ as in equation 9; and (4) process reward with $\alpha = 0.5$.

As shown in Table 3 and Figure 4, adding the length penalty accelerates convergence in the early stages, but the evaluation accuracy gradually declines in the later stages. Incorporating process reward increases the peak evaluation accuracy (e.g., up to **88.7%** for $\alpha = 0.2$), yet the final performance drops significantly. The larger decrease in training accuracy compared to the mean reward indicates that the model attempts to learn the preference of the reward model instead of improving problem solving abilities, which indicates reward hacking. This suggests that although process rewards can encourage higher peak performance, they may also introduce instability and unintended optimization behaviors in later training stages.

5 CONCLUSION

In this work, we conduct an empirical study on Reinforcement Learning (RL) fine-tuning in a minimal setting, utilizing the Qwen-3 1.7B model and the GRPO algorithm on the MATH dataset. We aimed to answer three critical questions regarding the necessity of model capacity and training complexity: the optimal LoRA rank, the effectiveness of various RL stabilization techniques, and the impact of different reward designs. Our experiments yield the following key findings:

- LoRA Capacity: We found that a minimal LoRA rank is sufficient for RL fine-tuning. Specifically, rank-1 LoRA achieves performance comparable to higher-rank configurations (such as rank-4 and rank-16) while remaining computationally efficient.
- RL Techniques: In this resource-constrained setting, complex stabilization techniques proved unnecessary or detrimental. While standard deviation normalization consistently improves performance and convergence speed, other common techniques (KL penalties, advantage clipping, and DAPO-style token-level loss) tended to degrade performance or slow down the learning process.
- Reward Design: A simple outcome-based reward is sufficient. Introducing additional complexity, such as process reward models or length penalties, frequently led to reward hacking or performance degradation in later training stages, rather than improving actual reasoning capabilities.

These results support the principle of Occam's Razor in RL fine-tuning. We conclude that in minimal settings, introducing additional complexity is often unnecessary. A simple design consisting of rank-1 LoRA, standard deviation normalization, and outcome-only rewards is both efficient and sufficient for achieving strong reasoning performance. However, due to limited resources, our experiments have not been conducted with more general settings, which can be explored in future studies.

REFERENCES

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

Adithya Bhaskar, Xi Ye, and Danqi Chen. Language models that think, chat better. *arXiv preprint arXiv:2509.20357*, 2025.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025.

Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. Skywork-o1 open series, November 2024. URL https://doi.org/10.5281/zenodo.16998085.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Jiaqi Huang, Zunnan Xu, Jun Zhou, Ting Liu, Yicheng Xiao, Mingwen Ou, Bowen Ji, Xiu Li, and Kehong Yuan. Sam-r1: Leveraging sam for reward feedback in multimodal segmentation via reinforcement learning. *arXiv preprint arXiv:2505.22596*, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026. URL https://arxiv.org/abs/2601.05242.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*, 2022.

John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

Ye Wang, Ziheng Wang, Boshen Xu, Yang Du, Kejun Lin, Zihan Xiao, Zihao Yue, Jianzhong Ju, Liang Zhang, Dingyi Yang, et al. Time-r1: Post-training large vision language model for temporal video grounding. *arXiv preprint arXiv:2503.13377*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.