Reinforcement Learning Experiment Report

Mengqi Liao

February 2, 2025

1 Experimental Content

1.1 Overview

After fine-tuning, we obtained a large model capable of solving the 24-point game. However, the inference efficiency of this model remains low, as it frequently engages in a substantial amount of ineffective exploration. Even when the reasoning length was constrained to no more than 1024 tokens during fine-tuning and the generation length was allowed to extend to 4096 tokens during the inference stage, approximately 20% of the test set instances still failed to complete reasoning within the specified generation length.

Recently, the remarkable performance of DeepSeek-R1[1] has demonstrated the immense potential of reinforcement learning in improving the reasoning capabilities of large models. To address the issue of inference efficiency, we attempted to implement the reinforcement learning process of DeepSeek-R1 to further enhance the reasoning ability of the large model.

1.2 GRPO

Large models are typically aligned with human preferences through Human Feedback Reinforcement Learning (HFRL), with common reinforcement learning algorithms such as PPO. However, PPO requires loading four models during online learning: the actor model, the reference model, the reward model, and the critic model. This results in significant computational overhead and a more complex training process. In contrast, DeepSeek-R1 utilizes the GRPO algorithm[1] for reinforcement learning, which only requires loading the actor model and the reference model, thereby significantly reducing the complexity of reinforcement learning. The optimization objective of GRPO is to maximize the following formula:

[1] https://github.com/deepseek-ai/DeepSeek-R1

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \min\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})} \hat{A}_{i,t}, \right.\right.$$
$$\left. \text{clip}\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_{i,t} \right)$$
$$\left. - \beta D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}] \right]. \tag{1}$$

The optimization objective of GRPO is highly similar to that of PPO, but the key difference lies in the way the advantage value $\hat{A}_{i,t}$ is calculated. GRPO does not rely on a critic model to compute the advantage value. Instead, it uses the normalized mean reward from the intra-group sampling outputs as the baseline. The advantage value is calculated as:

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}, \tag{2}$$

where $\mathbf{r} = \{r_1, \ldots, r_G\}$ represents the rewards corresponding to each group's sampling results $\{o_i\}_{i=1}^{G}$. The term $D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]$ in the formula represents the KL divergence constraint, which limits the optimization direction of the model to prevent it from deviating excessively from the reference model.

In standard GRPO, the advantage calculation relies on the normalization of intra-group rewards. However, this method depends solely on the mean and standard deviation of the current group's data, which may result in excessive sensitivity to individual sampling results, thereby affecting the stability of the model's training process. To address this issue, we adopt a moving average and standard deviation to smooth the reward normalization process. Specifically, the current reward's mean and standard deviation are not directly used for computation but are updated through momentum-based historical averages as follows:

$$\mu_t = (1-\alpha)\cdot\mu_{t-1} + \alpha\cdot\text{mean}(\mathbf{r}_t), \tag{3}$$
$$\sigma_t = (1-\alpha)\cdot\sigma_{t-1} + \alpha\cdot\text{std}(\mathbf{r}_t), \tag{4}$$

where μ_t and σ_t represent the smoothed mean and standard deviation at time t, α is the momentum coefficient, and $\mathbf{r}_t$ represents the rewards of the current group. This smoothing mechanism reduces sensitivity to individual sampling variations and enhances the robustness and stability of the training process.

$$\text{mean} = \text{mean}\times\text{momentum} + \text{mean}(\mathbf{r})\times(1-\text{momentum}) \tag{5}$$

$$\text{std} = \text{std}\times\text{momentum} + \text{std}(\mathbf{r})\times(1-\text{momentum}) \tag{6}$$

$$\hat{A}_{i,t} = \tilde{r}i = \frac{r_i - \text{mean}}{\text{std}} \tag{7}$$

This approach ensures continuity of the normalized mean and standard deviation across different batches, reducing the impact of outliers in a single sampling batch on the normalization process. Additionally, the momentum starts from a relatively low initial value and increases with each update but does not exceed a predefined maximum value. This dynamic adjustment allows the model to quickly adapt to changes in the reward distribution during the early stages of training while maintaining stability with higher momentum in the later stages.

1.3 Rule-Based Reward Model

Another significant modification in the reinforcement learning pipeline of DeepSeek-R1 is the shift from a trained reward model to a rule-based reward model. This is a critical design choice aimed at addressing the reward hacking problem that can occur when training reward models. Reward hacking refers to situations where the agent exploits loopholes in the reward function to maximize its reward but engages in behaviors that do not align with the intended objectives. In such cases, a trained reward model might lead the agent to deviate from its expected goals or even learn completely meaningless strategies.

Our rule-based reward model divides the reward into three components: output correctness reward r_{oc}, process correctness reward r_{pr}, and reasoning length reward r_{l}. The model receives the output correctness reward when it generates the final expression. If the expression is correct, $r_{\text{oc}} = 1$; otherwise, $r_{\text{oc}} = 0.35$ (to encourage the model to output a final answer). The process correctness reward $r_{\text{pr}} \in [0, 1]$ measures the accuracy of all reasoning steps, and the length reward optimizes the search efficiency of the reasoning path. The reward function is defined as follows:

$$r = w_{\text{oc}} \cdot r_{\text{oc}} + w_{\text{pr}} \cdot r_{\text{pr}} + w_{\text{l}} \cdot r_{\text{l}}, \tag{8}$$

where w_{oc}, w_{pr}, and w_{l} are the weights assigned to the output correctness reward, process correctness reward, and reasoning length reward, respectively. This reward design effectively balances the correctness of the final output, the accuracy of intermediate reasoning steps, and the efficiency of the reasoning path, guiding the model toward achieving the desired behavior.

$$r_{\text{l}}(l) = \begin{cases} 1 & \text{when } l \leq t \\ \dfrac{1}{1 + e^{k(l-t-m)}} & \text{when } l > t \end{cases} \tag{9}$$

In this design, $t = 4$ represents the threshold step length (each action counts as one step) at which reward decay is triggered. The parameter $m = 20$ controls the offset of the midpoint in the Sigmoid curve, and $k = 0.2$ adjusts the steepness of the decay rate. This reward function is characterized by a dual adjustment mechanism:

- (1) Full rewards are granted for efficient step lengths ($l \leq 4$).

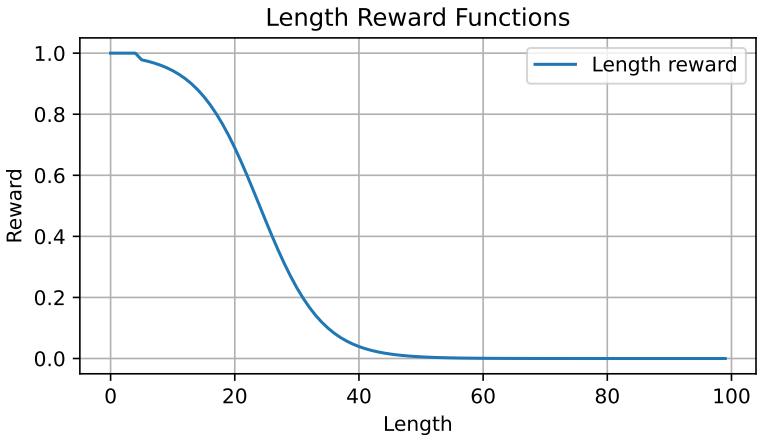

Figure 1: Relationship between length reward values and reasoning step length.

- (2) When the step length exceeds the threshold, a penalization is progressively applied using a shifted Sigmoid function (with its midpoint at $t + m = 24$).

This design encourages the model to explore efficient paths early while avoiding premature harsh penalization for intermediate reasoning processes of reasonable length.

The final reward is computed using the following formula:

$$r = \lambda_{\text{oc}} \cdot r_{\text{oc}} + \lambda_{\text{pr}} \cdot r_{\text{pr}} + \lambda_{\text{l}} \cdot r_{\text{l}}, \tag{10}$$

where λ_{oc}, λ_{pr}, and λ_{l} are the weights for the output correctness reward, process correctness reward, and length reward, respectively. This formulation balances the incentives for output correctness, intermediate reasoning accuracy, and reasoning efficiency, guiding the model toward optimal performance.

2 Experiment

2.1 Experimental Setup

We initiated reinforcement learning using a base model fine-tuned with Format v3 mixed data. During training, the learning rate was fixed at 2×10^{-6}, and β was set to 0.01. The maximum momentum was set to 0.2. The batch size was 12, and the group size G was set to 8. The coefficients of the reward model, λ_{oc}, λ_{pr}, and λ_{l}, were assigned values of 0.55, 0.3, and 0.15, respectively. For sampling, the temperature was set to 0.7, and nucleus sampling was adopted with a top-p parameter of 0.9. Both during training sampling and evaluation, the maximum generation length was set to 1024.

2.2 Experimental Results

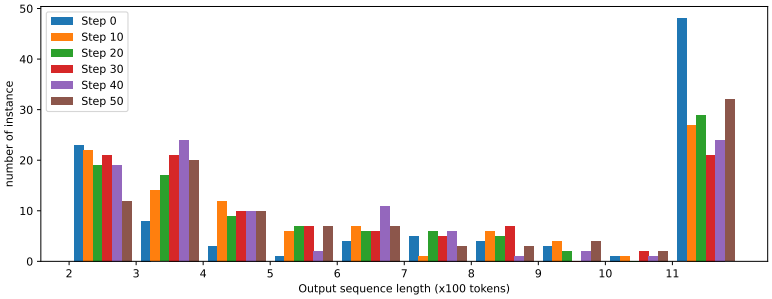

Figure 2: Distribution of token lengths in output sequences generated by models at different training steps.

Figure 2 illustrates the distribution of token lengths in the output sequences generated by the model at different training steps (Step 0 to Step 50). From the figure, it is evident that after reinforcement learning, the number of samples with generation lengths between 300 and 600 increased significantly, particularly those within the range of 300 to 400, which exhibit a clear upward trend as training progresses. Concurrently, the number of samples exceeding the window size (the far-right side of the figure) decreased markedly. This demonstrates that reinforcement learning effectively enhanced the reasoning efficiency of the model, reducing excessive and redundant generation.

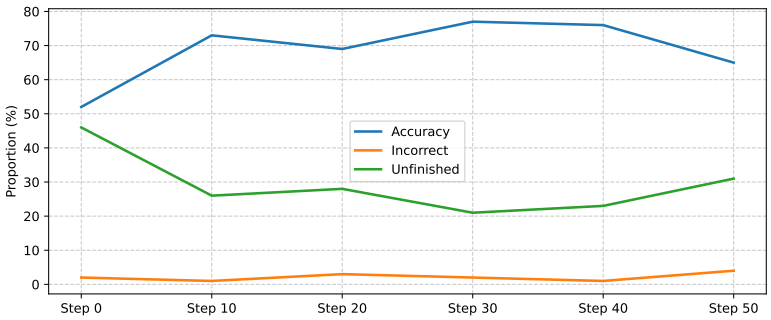

Figure 3: Changes in accuracy, incompletion rate, and error rate during training.

Figure 3 depicts the trends in the model's accuracy, incompletion rate, and error rate as training progresses. As shown in the figure, accuracy and completion rate demonstrated consistent improvement during the first 30 training steps, reaching their optimal levels at approximately Step 30, with accuracy nearing 80%. However, as training continued beyond this point, accuracy began to decline gradually, while both the incompletion rate and error rate exhibited a rebound. This phenomenon might indicate that the model overfits to

the reward signal as training steps increase, thereby degrading the quality of the generated outputs. Employing more fine-grained rewards, such as assigning different reward values to individual tokens within a sequence, could potentially yield better results.

References

[1] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.