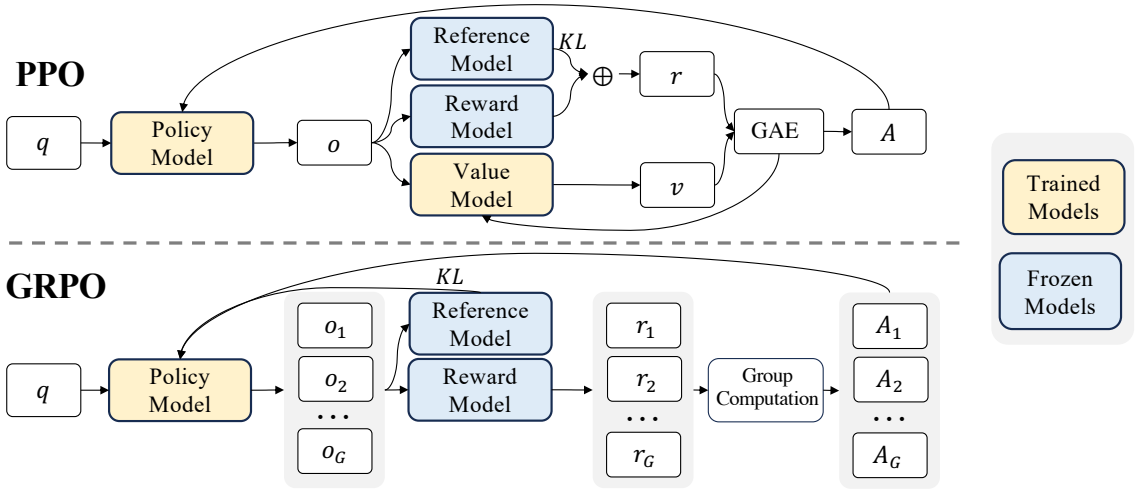

Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

on the rewards $\{r_{\geq t}\}$ and a learned value function V_ψ. Thus, in PPO, a value function needs to be trained alongside the policy model and to mitigate over-optimization of the reward model, the standard approach is to add a per-token KL penalty from a reference model in the reward at each token (Ouyang et al., 2022), i.e.,

$$r_t = r_\varphi(q, o_{\leq t}) - \beta \log \frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{ref}(o_t|q, o_{<t})}, \tag{2}$$

where r_φ is the reward model, π_{ref} is the reference model, which is usually the initial SFT model, and β is the coefficient of the KL penalty.

As the value function employed in PPO is typically another model of comparable size as the policy model, it brings a substantial memory and computational burden. Additionally, during RL training, the value function is treated as a baseline in the calculation of the advantage for variance reduction. While in the LLM context, usually only the last token is assigned a reward score by the reward model, which may complicate the training of a value function that is accurate at each token. To address this, as shown in Figure 4, we propose Group Relative Policy Optimization (GRPO), which obviates the need for additional value function approximation as in PPO, and instead uses the average reward of multiple sampled outputs, produced in response to the same question, as the baseline. More specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model by maximizing the following objective:

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

where ε and β are hyper-parameters, and $\hat{A}_{i,t}$ is the advantage calculated based on relative rewards of the outputs inside each group only, which will be detailed in the following subsections. The group relative way that GRPO leverages to calculate the advantages, aligns well with the comparative nature of rewards models, as reward models are typically trained on datasets of comparisons between outputs on the same question. Also note that, instead of adding KL penalty in the reward, GRPO regularizes by directly adding the KL divergence between the trained policy and the reference policy to the loss, avoiding complicating the calculation of $\hat{A}_{i,t}$.

Algorithm 1 Iterative Group Relative Policy Optimization

Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, ..., I **do**
  reference model $\pi_{ref} \leftarrow \pi_\theta$
  for step = 1, ..., M **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
    Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
    Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
    Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running r_φ
    Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
    for GRPO iteration = 1, ..., μ **do**
      Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
  Update r_φ through continuous training using a replay mechanism.

Output π_θ

And different from the KL penalty term used in (2), we estimate the KL divergence with the following unbiased estimator (Schulman, 2020):

$$\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right] = \frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - \log\frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - 1, \tag{4}$$

which is guaranteed to be positive.

4.1.2. Outcome Supervision RL with GRPO

Formally, for each question q, a group of outputs $\{o_1, o_2, \cdots, o_G\}$ are sampled from the old policy model $\pi_{\theta_{old}}$. A reward model is then used to score the outputs, yielding G rewards $\mathbf{r} = \{r_1, r_2, \cdots, r_G\}$ correspondingly. Subsequently, these rewards are normalized by subtracting the group average and dividing by the group standard deviation. Outcome supervision provides the normalized reward at the end of each output o_i and sets the advantages $\hat{A}_{i,t}$ of all tokens in the output as the normalized reward, i.e., $\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$, and then optimizes the policy by maximizing the objective defined in equation (3).

4.1.3. Process Supervision RL with GRPO

Outcome supervision only provides a reward at the end of each output, which may not be sufficient and efficient to supervise the policy in complex mathematical tasks. Following Wang et al. (2023b), we also explore process supervision, which provides a reward at the end of each reasoning step. Formally, given the question q and G sampled outputs $\{o_1, o_2, \cdots, o_G\}$, a process reward model is used to score each step of the outputs, yielding corresponding rewards: $\mathbf{R} = \{\{r_1^{index(1)}, \cdots, r_1^{index(K_1)}\}, \cdots, \{r_G^{index(1)}, \cdots, r_G^{index(K_G)}\}\}$, where $index(j)$ is the end token index of the j-th step, and K_i is the total number of steps in the i-th output. We also normalize these rewards with the average and the standard deviation, i.e., $\widetilde{r}_i^{index(j)} = \frac{r_i^{index(j)} - \text{mean}(\mathbf{R})}{\text{std}(\mathbf{R})}$. Subsequently, the process supervision calculates the advantage of each token as the sum of the normalized rewards from the following steps, i.e., $\hat{A}_{i,t} = \sum_{index(j)\geq t} \widetilde{r}_i^{index(j)}$, and then optimizes the policy by maximizing the objective defined in equation (3).

4.1.4. Iterative RL with GRPO

As the reinforcement learning training process progresses, the old reward model may not be sufficient to supervise the current policy model. Therefore, we also explore the iterative RL with GRPO. As shown in Algorithm 1, in iterative GRPO, we generate new training sets for the reward model based on the sampling results from the policy model and continually train the old reward model using a replay mechanism that incorporates 10% of historical data. Then, we set the reference model as the policy model, and continually train the policy model with the new reward model.

4.2. Training and Evaluating DeepSeekMath-RL

We conduct RL based on DeepSeekMath-Instruct 7B. The training data of RL are chain-of-thought-format questions related to GSM8K and MATH from the SFT data, which consists of around 144K questions. We exclude other SFT questions to investigate the impact of RL on benchmarks that lack data throughout the RL phase. We construct the training set of reward models following (Wang et al., 2023b). We train our initial reward model based on the DeepSeekMath-Base 7B with a learning rate of 2e-5. For GRPO, we set the learning rate of the policy model as 1e-6. The KL coefficient is 0.04. For each question, we sample 64 outputs. The max length is set to 1024, and the training batch size is 1024. The policy model only has a single update following each exploration stage. We evaluate DeepSeekMath-RL 7B on benchmarks following DeepSeekMath-Instruct 7B. For DeepSeekMath-RL 7B, GSM8K and MATH with chain-of-thought reasoning can be regarded as in-domain tasks and all the other benchmarks can be regarded as out-of-domain tasks.

Table 5 demonstrates the performance of open- and closed-source models with both chain-of-thought and tool-integrated reasoning on English and Chinese benchmarks. We find that: 1) DeepSeekMath-RL 7B attains accuracies of 88.2% and 51.7% on GSM8K and MATH, respectively, utilizing chain-of-thought reasoning. This performance surpasses that of all open-source models in the 7B to 70B range, as well as the majority of closed-source models. 2) Crucially, DeepSeekMath-RL 7B is only trained on chain-of-thought-format instruction tuning data of GSM8K and MATH, starting from DeepSeekMath-Instruct 7B. Despite the constrained scope of its training data, it outperforms DeepSeekMath-Instruct 7B across all evaluation metrics, showcasing the effectiveness of reinforcement learning.

5. Discussion

In this section, we will share our findings in pre-training and RL experiments.

5.1. Lessons Learnt in Pre-Training

We first share our experience in pre-training. Unless otherwise specified, we will adhere to the training settings outlined in Section 2.2.1. It is worth noting that, when referring to the DeepSeekMath Corpus in this section, we use an 89B-token dataset from the second iteration of the data collection process.

5.1.1. Code Training Benefits Mathematical Reasoning

A popular yet unverified hypothesis suggests that code training improves reasoning. We attempt to offer a partial response to this, particularly within the mathematical domain: code training