# REINFORCE++: A SIMPLE AND EFFICIENT APPROACH FOR ALIGNING LARGE LANGUAGE MODELS

**Jian Hu**
janhu9527@gmail.com

**ABSTRACT**

Reinforcement Learning from Human Feedback (RLHF) is rapidly evolving, with algorithms such as Proximal Policy Optimization (PPO), Direct Preference Optimization (DPO), REINFORCE Leave One-Out (RLOO), ReMax, and Group Relative Policy Optimization (GRPO) emerging in quick succession. By integrating various optimization techniques from PPO into the traditional REINFORCE algorithm, we propose REINFORCE++, which aims to enhance performance and stability in RLHF while reducing computational resource requirements by eliminating the need for a critic network. Our experimental results demonstrate that REINFORCE++ is more stable than GRPO and is also more computationally efficient than PPO. We have open-sourced the REINFORCE++ codes in https://github.com/OpenRLHF/OpenRLHF.

## 1 Introduction

Reinforcement Learning from Human Feedback (RLHF) has become a pivotal approach in training large language models to align with human preferences. The field has witnessed rapid evolution with the development of various algorithms, each attempting to address different aspects of the optimization challenge. Traditional algorithms like Proximal Policy Optimization (PPO) [5] have established foundational approaches, while newer methods such as Direct Preference Optimization (DPO) [4], REINFORCE Leave One-Out (RLOO) [7], ReMax [2], and Group Relative Policy Optimization (GRPO) [6] have emerged to tackle specific limitations in RLHF implementation. In this paper, we introduce REINFORCE++, an enhanced version of the REINFORCE algorithm that incorporates optimization techniques from PPO to achieve better performance and stability without the necessity of a critic network. Our approach focuses on streamlining the training process while maintaining robust performance characteristics. By eliminating the need for a critic network, REINFORCE++ significantly reduces computational resource requirements while maintaining training stability. Our experimental results demonstrate two key advantages of REINFORCE++. First, it exhibits superior stability compared to GRPO. Second, it achieves greater computational efficiency than PPO due to its simplified architecture that eliminates the critic network. These improvements make REINFORCE++ a promising alternative for practical RLHF implementations.

## 2 Background

### 2.1 REINFORCE Algorithm

REINFORCE is a fundamental policy gradient method in reinforcement learning designed to maximize expected cumulative rewards through direct policy optimization. The algorithm operates based on Monte Carlo methods, following these key steps:

- **Policy Sampling:** The agent interacts with the environment according to its current policy, generating a sequence of states, actions, and rewards (trajectories).
- **Return Calculation:** For each trajectory, returns are computed using discounted cumulative rewards:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t} r_k \tag{1}$$

where $\gamma$ is the discount factor, and $r_k$ is the immediate reward at time step $k$.

- **Gradient Estimation:** The policy gradient is calculated using Monte Carlo methods, updating the policy parameters $\theta$ as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[ G_t \nabla_\theta \log \pi_\theta(A_t|S_t) \right] \tag{2}$$

- **Policy Update:** The policy parameters are updated using gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) \tag{3}$$

where $\alpha$ is the learning rate.

### 2.2 Challenges in RLHF

RLHF implementations like PPO often rely on a critic network to estimate the value function, which can be computationally expensive and prone to instability during training [3].

## 3 REINFORCE++ Enhancements

To stabilize model training, several optimization techniques are integrated into REINFORCE++:

### 3.1 Token-Level KL Penalty

The Kullback-Leibler (KL) divergence between the response distributions of the reinforcement learning (RL) model and the supervised fine-tuning (SFT) model is calculated for each token. This divergence is then incorporated as a penalty term in the reward function during training. Specifically, the per-token reward is represented as follows:

$$r(s_t, a_t) = \mathbf{I}(s_t = [\text{EOS}]) r(x, y) - \beta \, \text{KL}(t) \tag{4}$$

$$\text{KL}(t) = \log \left( \frac{\pi_{\theta_{\text{old}}}^{\text{RL}}(a_t|s_t)}{\pi^{\text{SFT}}(a_t|s_t)} \right) \tag{5}$$

where $x$ is the prompt, $y$ is the response, and $\mathbf{I}(s_t = [\text{EOS}])$ is the indicator function that denotes whether $t$ is the last token. The advantage of this token-level KL penalty is that it seamlessly integrates with Process Reward Models (PRM) and facilitates credit assignment.

### 3.2 PPO-Clip

A critical technique from PPO that limits policy update magnitudes by introducing a clipping mechanism to ensure that updates do not exceed certain thresholds. This can be expressed as:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta) \hat{A} t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \tag{6}$$

where $r_t(\theta)$ represents the ratio of new to old policies, $\hat{A}_t$ is the advantage estimate, and $\epsilon$ is a small constant. Indeed, we are incorporating the PPO loss function within the REINFORCE framework.

### 3.3 Mini-Batch Updates

To enhance training efficiency and stability, REINFORCE++ employs mini-batch updates, allowing for more frequent policy updates compared to the traditional REINFORCE algorithm.

- Dividing training data into smaller batches rather than using the entire dataset for updates.
- Allowing multiple parameter updates per mini-batch, which accelerates convergence and reduces memory usage.
- Introducing randomness that helps avoid local optima and improves generalization.

### 3.4 Reward Normalization and Clipping

This approach addresses instability in reward signals through:

- **Reward Normalization**: Standardizing rewards (e.g., subtracting the mean and dividing by the standard deviation) to achieve smoother reward signals, enhancing training stability.
- **Reward Clipping**: Limiting reward values within a specific range to mitigate the impact of extreme rewards on model updates, thus preventing gradient explosion.

### 3.5 Advantage Normalization

Advantage normalization helps manage variance in the advantage function of REINFORCE++ estimates defined as:

$$A_t(s,a) = R(s,a) - \beta \cdot \sum_{i=t}^{T} \text{KL}(i) \tag{7}$$

where $R$ is the reward function and $KL$ is the per-token KL reward and $t$ is the token position. The normalization process includes:

- Calculating mean and variance for computed advantage values.
- Normalizing these values to improve numerical stability and enhance learning effectiveness.

## 4 Empirical Evaluation

In this section, we present the experimental setup and hyper-parameters utilized in our study of REINFORCE++, highlighting its advantages over alternative methods such as GRPO and RLOO.

### 4.1 Experimental Setup

The experiments were conducted using a controlled environment to assess the performance of REINFORCE++ in various scenarios. The key components of the experimental apparatus included a computational framework capable of executing reinforcement learning algorithms, along with a robust data processing pipeline to evaluate the results. We conducted distributed RLHF training experiments based on OpenRLHF [1].

#### 4.1.1 Hyper-parameters

We employed the following hyper-parameters in our experiments:

- **KL Penalty Coefficient**
  - General Scenario: $\beta = 0.01$
  - Mathematical Test: $\beta = 0.001$
- **Training Configuration**
  - Maximum Samples: $25{,}000$
  - Samples per Prompt: $4$
  - Rollout Batch Size: $256$
  - Training Batch Size: $128$
  - Actor Learning Rate: $5 \times 10^{-7}$
  - Critic Learning Rate: $9 \times 10^{-6}$
  - Discount $\gamma$: $1.0$

#### 4.1.2 Datasets

- **General Scenario**
  - Prompt Dataset: prompt-collection-v0.1
  - Reward Model Training: preference_700K

- **Mathematical Scenario**
  - Training Data: MetaMathQA (Mathematical Scenario 1)
  - Reward Model: Closed-source mathematical model (Mathematical Scenario 2)

#### 4.1.3 Base Models

- Llama3.1-8B-SFT
- Qwen2.5-7B-SFT

### 4.2 Experimental Results

Figure 1 demonstrates that GRPO is more susceptible to hacking rewards and output length compared to REINFORCE++, while REINFORCE++ and PPO exhibit similar performance. Figure 2 shows that GRPO and REINFORCE perform similarly under purely rule-based rewards. Figure 3 indicates that the reward improvement under GRPO's unit KL divergence consumption is not as significant as that of REINFORCE++. In summary, on average, REINFORCE++ is more stable than GRPO.

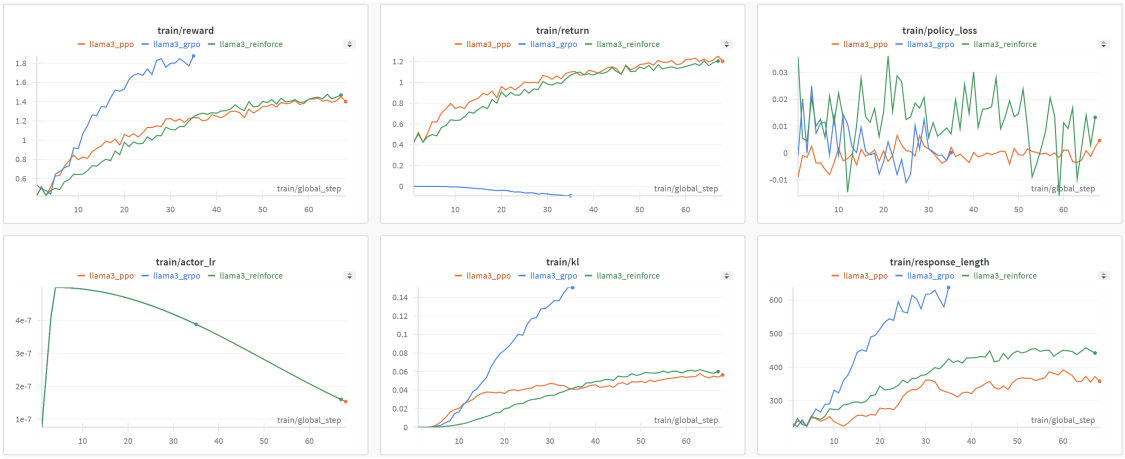


Figure 1: For Bradley-Terry Reward Models and general scenarios (PPO / REINFORCE++ is better than GRPO)

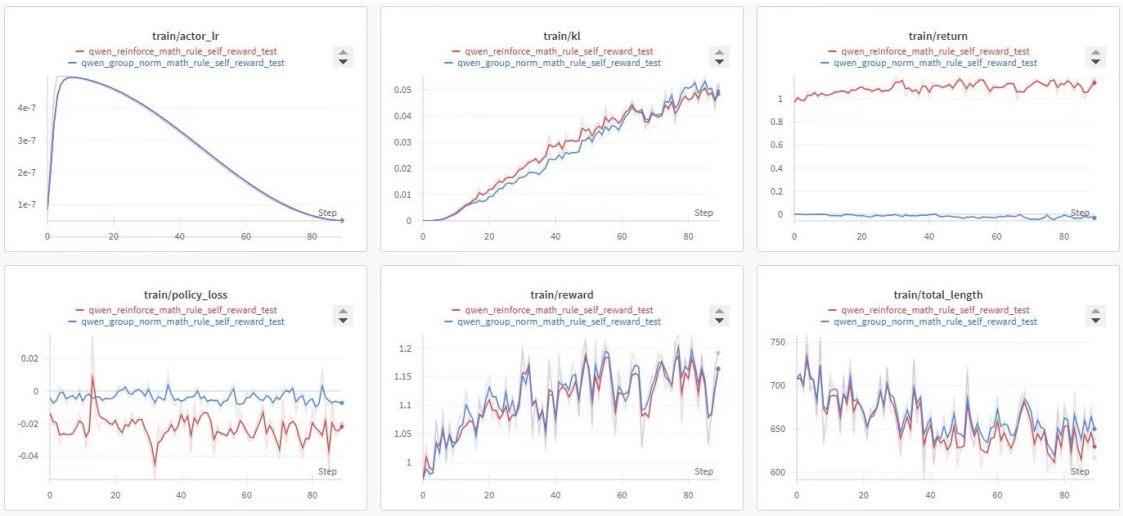


Figure 2: For rule-based reward and mathematical scenarios 1 (REINFORCE++ == GRPO)


Figure 3: For mathematical scenarios 2 ( REINFORCE++ / RLOO is better than GRPO)

## 5 Conclusion

REINFORCE++ presents a simple yet efficient approach for aligning large language models through reinforcement learning from human feedback. By incorporating optimization techniques from PPO into the traditional REINFORCE algorithm, REINFORCE++ achieves improved stability and performance without the need for a critic network, making it a promising method for future RLHF applications.

## References


[1] Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

[2] Ziniu Li, Tian Xu, Yushun Zhang, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

[3] 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. *arXiv preprint arXiv:2203.02155*, 2022.

[4] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1132–1145. PMLR, 2017.

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

[7] Yuan Wu et al. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.