A comparison of GRPO and PPO in Reinforcement Learning Environments

Alexander Cremer, Koen Ponse*, Thomas M. Moerland*
* = supervisor

Reinforcement Learning Group, Leiden University

1 Introduction

Recently, reinforcement learning (RL) has played a key role in fine-tuning Large Language Models (LLMs) through the use of RLHF [3]. Proximal Policy Optimization (PPO) [4], a well-established algorithm for RL, has often been used in this domain. As an alternative, Deepseek proposed Group Relative Policy Optimization (GRPO) [5], which avoids the critic model required in PPO. Instead, GRPO uses a group mechanism for the policy update. Although promising in the LLM setting, there is little research on GRPO in classic RL tasks, which for example contain non-terminal rewards. Therefore, this paper compares PPO and GRPO on various standard RL environments.

2 Methodology

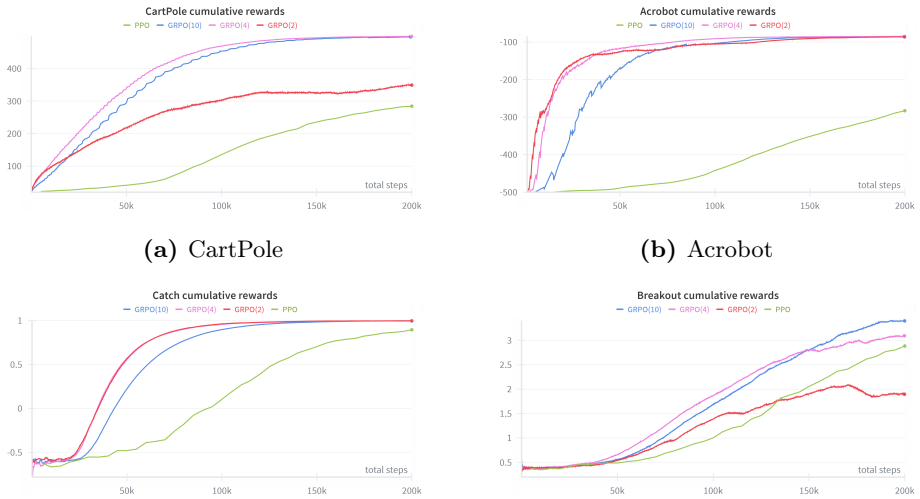
In our experiments, we have used the PPO implementation of the CleanRL library [1]. Our GRPO implementation is an adaptation of the same CleanRL PPO implementation. We have tested both implementations in CartPole, Acrobot [6], Catch [2] and MinAtar Breakout [7]. All experimentation code, including a GRPO implementation suited for general RL environments, is open-sourced on GitHub.[1]

GRPO mirrors PPO in its clipping objective, but drops the critic network entirely. Instead, to calculate advantages, GRPO compares the cumulative rewards acquired by each rollout to the average cumulative reward of all rollouts in the group. Therefore, at the start of each rollout, all traces need to be initialized to the same state. An illustration of the GRPO's rollouts is given in Appendix A. Importantly, after completion of the group of rollouts, *all* observed state-action pairs in a trace are updated based on the final calculated advantage of that whole episode (at the root state). After updating, we take one step in the environment and repeat the group rollout from the new root state (to avoid only sampling from the initial state). As common in GRPO, we have disabled the entropy coefficient in our implementation and added a KL-term to the policy loss[5]. A full list of PPO's and GRPO's most important hyperparameters and their respective values can be found in Appendix B.

[1] `https://github.com/AlexanderCremer/GRPO-vs-PPO-in-RL-env`

3 Results

Figure 1 compares GRPO against PPO in cumulative reward against environment steps. In general, GRPO shows faster learning than PPO, where the optimal group size varies over tasks. A likely explanation is that GRPO does not rely on (slow) value network training to obtain good advantage estimates. However, a confounding factor could be the update frequency. PPO runs 10 parallel environments that generate 5000 steps per learning cycle, resulting in only 40 learning cycles over 200,000 total steps. GRPO, on the other hand, updates more frequently as it updates after all episodes in the group have terminated (i.e., using fewer environment steps per learning cycle).



(a) CartPole **(b)** Acrobot

(c) Catch **(d)** Breakout (MinAtar)

Fig. 1: Performance for PPO (10 parallel envs) and GRPO, averaged over 10 runs. Generally, we observe GRPO performs better in terms of per-step performance, indicating a higher sample-efficiency in our experiments.

4 Discussion & Conclusion

The experiments show that GRPO can be a promising alternative to PPO in general RL tasks, especially when data efficiency is crucial. Future work could align the update frequency of both methods, to eliminate this possible confounder. Also, notice that GRPO is harder to parallelize efficiently, since each group of rollouts needs to start from the same state and rollouts are usually of different lengths. Data efficiency and wall-time speed therefore do not align one-on-one. Finally, we hypothesize that GRPO could suffer more in sparse-reward settings, where the variance of the sample-based value estimate may be higher.

References

1. Huang, S., Dossa, R.F.J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., Araújo, J.G.: Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. Journal of Machine Learning Research **23**(274), 1–18 (2022), http://jmlr.org/papers/v23/21-1342.html
2. Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., et al.: Behaviour suite for reinforcement learning. arXiv preprint arXiv:1908.03568 (2019)
3. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L.E., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.J.: Training language models to follow instructions with human feedback. ArXiv **abs/2203.02155** (2022)
4. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
5. Team, D.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. DeepSeek Research (2024), https://github.com/deepseek-ai/DeepSeek-R1/blob/main/DeepSeek_R1.pdf
6. Towers, M., Kwiatkowski, A., Terry, J., Balis, J.U., Cola, G.D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Tan, H., Younis, O.G.: Gymnasium: A standard interface for reinforcement learning environments (2024), https://arxiv.org/abs/2407.17032
7. Young, K., Tian, T.: Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments (2019), https://arxiv.org/abs/1903.03176

A GRPO Rollout Illustration

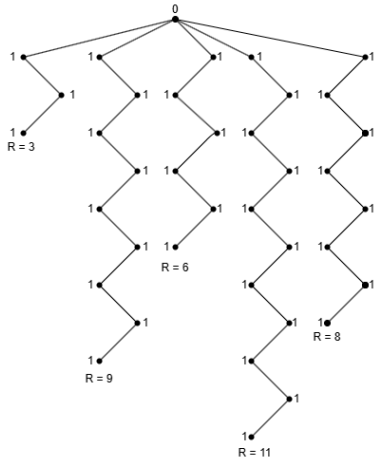


Fig. 2: Illustration of a GRPO group of rollouts on a CartPole-like environment (with a reward of 1 per step until termination). Here, states are represented as nodes and actions as edges with rewards assigned to states after taking an action. The mean return of the roll-outs (a sample-based value estimate at the root state) is computed as $(3 + 9 + 6 + 11 + 8)/5 = 7.4$.

B Hyperparameters

	PPO	GRPO
learning rate	2.5e-4	1e-3
gamma	0.99	N/A
gae lambda	0.95	N/A
number of minibatches	4	1
number of update epochs	4	4
clipping coefficient	0.2	0.2
entropy coefficient	0.01	N/A
value function coefficient	0.5	N/A
kl coefficient	N/A	0.2
max steps per rollout	500	500
number of parallel envs	10	Varying

Table 1: A list of the most important hyperparameters used in the experiments for both GRPO and PPO. Hyperparameters are the same for all environments.