PPO-UE: Proximal Policy Optimization via Uncertainty-Aware Exploration

Qisheng Zhang[1], Zhen Guo[1], Audun Jøsang[3], Lance M. Kaplan[4], Feng Chen[5], Dong H. Jeong[6], Jin-Hee Cho[1]

[1]Department of Computer Science, Virginia Tech, VA, USA; [2]2810 Jackson Avenue, NY, USA;
[3]University of Oslo, Oslo, Norway; [4]DEVCOM Army Research Laboratory, MD, USA;
[5]University of Texas at Dallas, Richardson TX, USA; [6]University of the District of Columbia, DC, USA
qishengz19@vt.edu, zguo@vt.edu, audun.josang@mn.uio.no, lance.m.kaplan.civ@army.mil, feng.chen@utdallas.edu, djeong@udc.edu, jicho@vt.edu

Abstract

Proximal Policy Optimization (PPO) is a highly popular policy-based deep reinforcement learning (DRL) approach. However, we observe that the homogeneous exploration process in PPO could cause an unexpected stability issue in the training phase. To address this issue, we propose PPO-UE, a PPO variant equipped with self-adaptive uncertainty-aware explorations (UEs) based on a *ratio uncertainty level*. The proposed PPO-UE is designed to improve convergence speed and performance with an optimized *ratio uncertainty level*. Through extensive sensitivity analysis by varying the *ratio uncertainty level*, our proposed PPO-UE considerably outperforms the baseline PPO in Roboschool continuous control tasks.

Introduction

Deep reinforcement learning (DRL) is a set of reinforcement learning algorithms equipped with deep neural networks (DNNs). DRL algorithms can be categorized into two: *value-based* and *policy-based*. Value-based approaches include Deep Q-network (Mnih et al. 2013), Double Deep Q-Networks (Van Hasselt, Guez, and Silver 2016), and Dueling Deep Q-Networks (Wang et al. 2016). These approaches separate the exploration and exploitation processes with additional rule-based methods. Policy-based approaches include Advantage Actor Critic (A2C) (Mnih et al. 2016), Policy Gradient (Peters and Schaal 2006), Trust Region Policy Optimization (TRPO) (Schulman et al. 2015), and Proximal Policy Optimization (PPO) (Schulman et al. 2017). These policy-based DRL approaches learn policy functions that directly map states into actions. This allows the exploration and exploitation processes to be combined by sampling actions based on policies.

In continuous control problems, value-based approaches cannot effectively learn the optimal action from a value function due to the continuous action space. Thus, policy-based approaches have been widely used to solve these problems. However, the performance of policy-based approaches largely depends on the sampling process. For instance, PPO's Gaussian action exploration mechanism has a stability issue (Ciosek et al. 2019). In this work, we aim to refine the exploration mechanism in PPO to balance data exploration and exploitation better. Specifically, we propose a variant of PPO, called *PPO-UE*, to leverage the uncertainty information in learned policies and improve the overall performance of the original PPO algorithm.

Via *PPO-UE*, we make the following **key contributions**:

1. We give a rigorous theoretical analysis of the sampling techniques used in general policy gradient methods. The analysis shows the stability issue in these techniques.
2. We propose an algorithm called *PPO-UE*, which enables uncertainty-aware explorations in the training phase. The uncertainty-aware exploration can intelligently adapt to different policy statuses and provide adaptive, state-dependent exploration strategies.
3. We conduct extensive experiments to investigate the impact of the uncertainty metric in our proposed PPO-UE on learning performance in Roboschool continuous control tasks. The experiment results show that PPO-UE can achieve faster convergence and better performance than the original PPO baseline.

Related Work

Recently, much work has been done to refine the PPO algorithm further. Xiao et al. (2020) focused on the policy iteration process and subtracted a baseline term from the advantage function in the loss function to further improve the learning efficiency of PPO. In addition, more work has been done in terms of the exploration process in PPO. Zhang et al. (2022) improved the exploration efficiency of PPO using a revised reward function with an additional term called *uncertain reward*. The uncertain reward aims to give incentives for more explorations. Khoi et al. (2021) took a similar approach to leverage a technique called *Curiosity Driven Exploration*. Similarly, an additional term, called *intrinsic reward signal*, is added to the original reward function to improve the exploration efficiency. Following a similar line of ideas, Liu and Su (2022) used an additional reward term called *internal reward* to encourage more balanced explorations. However, instead of improving reward functions, Hämäläinen et al. (2020) proposed an algorithm called *PPO-CMA* to periodically update the covariance matrix of sample (CMA) distribution along with the policy iterations.

Unlike the works above, we consider self-adaptive, uncertainty-aware explorations without changing the sampling distribution and reward function in PPO. Therefore, our approach can provide easy implementability and parameter-tuning capability during the training phase.

Preliminaries

In this work, we consider the on-policy DRL algorithms. Given a policy π_θ parameterized by θ and a state s_t at time t, the agent takes an action $a_t \sim \pi_\theta(a_t|s_t) = \pi_\theta(p_e(\mu))$. Here the policy π_θ includes two components: one is μ output by the actor neural network, the other is the exploration distribution p_e as a function of μ. In the following text, we would refer to a_t as the action output by π_θ and μ as the policy mean output by π_θ. After sampling the action a'_t, the new state s_{t+1}, the reward, r_t, will be given by the environment. The agent's goal is to find the optimal θ and corresponding π_θ to maximize the accumulated expected reward $E(\sum_{t=0}^{\infty} \gamma^t r_t)$ where γ is a decay factor.

We consider a continuous environment, which is simulated until a predefined terminal state or a maximum episode length T_e is reached. Then, the agent will update θ and π_θ periodically with a constant update interval T_u.

Policy Gradient

The policy gradient method (Sutton et al. 1999) updates the policy gradient with two steps. First, the gradient is estimated by differentiating the following loss function:

$$L^{PG}(\theta) = E_t(\log \pi_\theta(a_t|s_t) A_t). \tag{1}$$

Here, A_t is an advantage function used to estimate the benefit of taking a_t, given state s_t. Second, this process can lead to forming the following gradient:

$$g = E_t(\nabla \log \pi_\theta(a_t|s_t) A_t). \tag{2}$$

Proximal Policy Optimization

To improve the training efficiency and stability of the policy gradient method, Trust Region Methods (TRPO) (Schulman et al. 2015) was proposed with the surrogate objective function under constraints on the policy update size. However, the surrogate objective function of TRPO could not accurately estimate the policy performance. Thus, to refine the objective function and reduce the computational overhead, Schulman et al. (2017) proposed Proximal Policy Optimization (PPO) to perform multiple minibatch gradient steps in one policy update iteration. Specifically, PPO is refined to improve the original TRPO in two aspects. First, the clipped surrogate objective is used as the substitution for the original surrogate objective. Second, the adaptive Kullback–Leibler (KL)-divergence penalty coefficient is used rather than a constant penalty. Schulman et al. (2017) claim that the clipped surrogate objective outperforms the original surrogate objective. Hence, in this work, we use it in all PPO vs. PPO-UE performance comparative evaluations.

Continuous Action Spaces

In continuous control problems, the policy gradient algorithms, including PPO, output a desired action rather than an action distribution. Thus, we form the action distribution based on a predefined probability distribution model. A *multivariate normal distribution*, a.k.a. *multivariate Gaussian distribution*, is commonly used to sample the new actions based on the action output by the policy. The multivariate Gaussian distribution enables the policy to sample the action $a \sim \mathcal{N}(\mu, \Sigma)$, where μ, also known as the policy mean, is the action output by the actor neural network of policy π_θ and Σ is the covariance matrix. In this work, we follow the original PPO algorithm (Schulman et al. 2017) and employ a diagonal covariance matrix Σ. Other than using the Gaussian distribution to sample actions for all states, we adjust the sampling process to be well applicable in the Roboschool problem based on the given state.

Proposed PPO-UE Algorithm

Problem Analysis

As mentioned earlier, PPO uses a global Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ to sample actions for all states. The predefined covariance matrix Σ is independent of the learned policies. Since we have the PDF of $\mathcal{N}(\mu, \Sigma)$ with

$$f(a) = \frac{1}{\sqrt{(2\pi)^d \|\Sigma\|}} \exp(-\frac{1}{2}(a-\mu)^T \Sigma^{-1} (a-\mu)), \tag{3}$$

we have

$$\log(f(a)) = -\frac{1}{2}(d \log(2\pi) + \log(\|\Sigma\|) + d_M^2(a, \mathcal{N}(\mu, \Sigma))). \tag{4}$$

Here d_M refers to the Mahalanobis distance of action a from $\mathcal{N}(\mu, \Sigma)$. Suppose $a \in \mathbb{R}^d$, $\Sigma = \text{diag}(\vec{\sigma})$, Eq. (4) can be further simplified to

$$\log(f(a)) = -\frac{1}{2}\Big(d \log(2\pi) + \sum_i^d \Big(\log(\sigma_i) + \frac{(a_i - \mu_i)^2}{\sigma_i}\Big)\Big). \tag{5}$$

Therefore, minimizing the loss function Eq. (1) is equivalent to minimizing the following,

$$L^{PG}(\theta) = E_t((\sum_i^d (\log(\sigma_i) + \frac{(a_i - \mu_i)^2}{\sigma_i}))A_t). \tag{6}$$

This means the gradient of Eq. (6) points to the direction where the policy fits positive-advantage actions other than negative-advantage actions. However, sampling with a global Gaussian distribution cannot distinguish positive-advantage actions from negative-advantage actions. This destabilizes the update process of policy iterations.

Uncertainty-Aware Exploration

To mitigate the stability issue caused by the negative-advantage actions, we introduce *Uncertainty-Aware Exploration* (UE) to balance the exploration and exploitation further. For a given state s, assume the policy π_{θ_t} outputs the optimal action a, then the policy $\pi_{\theta_{t+1}}$ should be trained to output the same optimal action a, as the policy mean μ is trained towards positive-advantage actions. This means we should keep exploiting the action if the action taken is good enough. However, in practice, we cannot evaluate the action optimality from the policy. Thus, we need to estimate it from policy iterations.

Action Distance Ratio We aim to approximate the action optimality from two consecutive policies. For a given state s_t at time step t with two policies $\pi_{\theta_{t-1}}$ and π_{θ_t}, we denote the actions output by $\pi_{\theta_{t-1}}$ and π_{θ_t} as a_{t-1} and a_t, respectively. Then we define the *action distance* $d(s_t) = \|a_t - a_{t-1}\|$. Furthermore, we can derive the *action distance ratio* as $r(s_t) = \frac{\|a_t - a_{t-1}\|}{\|a_{t-1}\|}$. Note that $r(s_t)$ measures the degree of the policy update with respect to s_{t-1}. Thus, $r(s_t) = 0$ is the necessary condition for policy convergence in s_t. The smaller $r(s_t)$ indicates higher optimality in local actions which are restricted by $\pi_{\theta_{t-1}}$ and π_{θ_t}.

Exploration Threshold and Ratio Uncertainty Since the action distance ratio can estimate the optimality of an action taken, we only enable exploration when the ratio is sufficiently high. We need a global view of these ratios to select an appropriate action distance ratio as the exploration threshold. To this end, we rank all ratios $r(s)$ between the k-th and $(k+1)$-th updates in ascending order and select the exploration threshold at the desired ranking. Specifically, we set an *ratio uncertainty level* $U_k \in [0, 1]$ and define the *exploration threshold* τ_k as the ratio with the ranking $\lfloor (1 - U_k)L_k \rfloor$. Here, L_k is the ranking length. This means the policy will exploit an action as the policy mean μ only when its corresponding ratio is smaller than τ_k. Otherwise, the agent will choose to sample a new action. As a special case, the original PPO algorithm (Schulman et al. 2017) has a ratio uncertainty level $U_k = 1$ and $\tau_k = 0$ for any k.

Algorithm Summary

To apply UE to the PPO algorithm, we maintain a copy of the old policy when each policy is updated. We cannot calculate the exploration threshold based on the ongoing policy since the sampling process depends on the exploration threshold. Hence, we calculate the exploration threshold from the previous update and use it for the ongoing sampling process. We describe the details of the proposed PPO-UE in Algorithm 1.

Table 1: Algorithm Hyper-parameter Setting

Param.	Meaning	Value
T	Total training steps	1×10^6
T_e	Maximum episode length for training	512
T'_e	Maximum episode length for testing	2,048
T_u	Policy update interval	2,048
U_0	Initial ratio uncertainty level	1
τ_0	Initial exploration threshold	0
ϵ	Clipping parameter in PPO	0.2
K	Number of epochs in PPO	80
$\log(\sqrt{\sigma_i})$	Log standard deviation of action distribution in PPO	LinearAnneal $(-0.1, -1.6)$

Experiment Setup

To demonstrate the outperformance of our proposed PPO-UE algorithm in terms of high-dimensional continuous control problems, we trained and tested PPO-UE and the baseline PPO on the OpenAI Gym RoboschoolWalker2d (Brockman et al. 2016). In this environment, a 3D humanoid must balance multiple factors to walk optimally. The robot is trained to meet the following three goals: (1) moving as fast as possible; (2) finding the least number of actions to perform a move; (3) maintaining a healthy status. We performed 10 training runs with different random seeds for a given environment and setting. The only difference of PPO-UE from PPO is the exploration phase. We use the same hyper-parameters for other components in PPO. We also use a longer horizon T'_e in the testing phase to show the generalizability of trained policies. Table 1 describes the details of the hyper-parameter setting used in this study.

Algorithm 1: **PPO-UE**

```
$T \leftarrow$ total training steps
$T_u \leftarrow$ policy update interval
$k \leftarrow$ policy update iterations
$T_e \leftarrow$ maximum episode length
$U_0 \leftarrow$ initial ratio uncertainty level
$\tau_0 \leftarrow$ initial exploration threshold
$\pi_{\theta_0} \leftarrow$ initial policy
$t_1, k = 0$
while $t_1 < T$ do
    $t_2 = 0$
    while $t_2 < T_e$ do
        if $r(s_{t_2}) > \tau_k$ then
            Sample a new action with $\mathcal{N}(\mu, \Sigma)$
        else
            Sample an action output by $\pi_{\theta_{t_1}}$
        end if
        if $t_1 \equiv 0 \pmod{T_u}$ then
            $k = \frac{t_1}{T_u}$
            $\tau_k = \tau_k(U_k, L_k)$
            Update policy $\pi_{\theta_{t_1}}$
        end if
    end while
end while
```

Comparing Schemes

To simplify the parameter tuning process, we use a fixed ratio uncertainty level $U = U_k$ for all iteration k. Under this setting, we aim to find an optimal ratio uncertainty level U for maximizing the testing reward. We also conduct a sensitivity analysis of U which is ranged in [0.8, 0.9, 0.96, 0.98, 0.99]. To simplify the notations, we rename the corresponding schemes as PPO-UE$_{0.8}$, PPO-UE$_{0.9}$, PPO-UE$_{0.96}$, PPO-UE$_{0.98}$, and PPO-UE$_{0.99}$. The baseline PPO is equivalent to PPO-UE with $U = 1$. We have six schemes parameterized by U and conduct their comparative performance analysis.

Metrics

We use three metrics for our experiments: (1) Training reward R_{train} with horizon T_e; (2) Testing reward R_{test} with horizon T'_e; (3) Posterior ratio uncertainty level PU given by actual action distance ratio rankings. Specifically, after sampling with exploration threshold τ_k, we obtain the action distance ratio ranking for all samples. Denote the number of samples with an action distance ratio below τ_k as L_{low}, we have $PU = 1 - \frac{L_{low}}{L_k}$, where L_k is the ranking length. We propose PU to check the consistency of ratio uncertainty level U. We discuss more details in the following section.

Experiment Results & Analysis

We trained PPO-UE and PPO baseline for 10 simulation runs. Each simulation run includes a total of 1 million in training steps. In the testing phase, the agent chooses the policy mean μ as the next action instead of random sampling. We evaluate each scheme 100 times to obtain the testing reward.

Training Reward

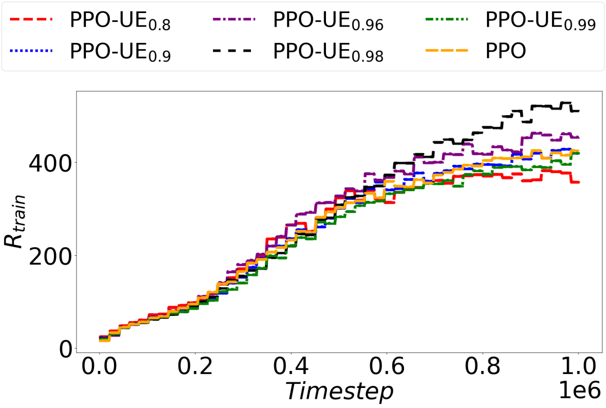

Figure 1: Training rewards with respect to training time steps.

Figure 1 shows the learning curve of the six schemes with respect to training time steps. The overall performance order is: PPO-UE$_{0.98}$ $\geq$ PPO-UE$_{0.96}$ $\geq$ PPO-UE$_{0.9}$ $\approx$ PPO $\geq$ PPO-UE$_{0.99}$ $\geq$ PPO-UE$_{0.8}$. It is clear that PPO-UE$_{0.98}$ performs the best among all schemes after 1 million training time steps. This implies that there exists an optimal ratio uncertainty level U to maximize the training reward. Furthermore, the convergence speed is also affected by U. PPO-UE$_{0.8}$ has the greatest convergence speed but worst performance overall. This is because U is too low to ensure adequate exploration during the training process. Hence, the agent chooses actions based on exploitation excessively and the corresponding policy quickly converges to a sub-optimal solution. In general, a smaller uncertainty level, U, indicates a faster convergence.

Testing Reward

Figure 2 shows the overall performance of the six schemes evaluated by a testing reward. Again, PPO is equivalent to PPO-UE when $U = 1$. Thus, we can rank the aforementioned six schemes with respect to different ratio uncertainty levels U. The overall performance order is: PPO-UE$_{0.96}$ $\geq$ PPO-UE$_{0.98}$ $\geq$ PPO-UE$_{0.9}$ $\geq$ PPO-UE$_{0.8}$ $\approx$ PPO-UE$_{0.99}$ $\geq$ PPO. It is noticeable that PPO performs the worst in the testing phase, which deviates from the performance order in the training phase. This is because PPO is trained with the largest ratio uncertainty level U, which destabilizes the training results. Due to the destabilized training, the learned policy is not stable enough in a generalized testing environment. PPO-UE$_{0.98}$ and PPO-UE$_{0.99}$ also have the same issue as their performance rankings drop from the training phase to the testing phase. Overall, the testing reward increases to its optimal at $U = 0.96$, and after then drops. This is because of the trade-off between exploration and exploitation. In general, the schemes with low ratio uncertainty levels cannot explore well. They converge to sub-optimal policies while schemes with high ratio uncertainty levels cannot exploit well. They also perform well with high variance.

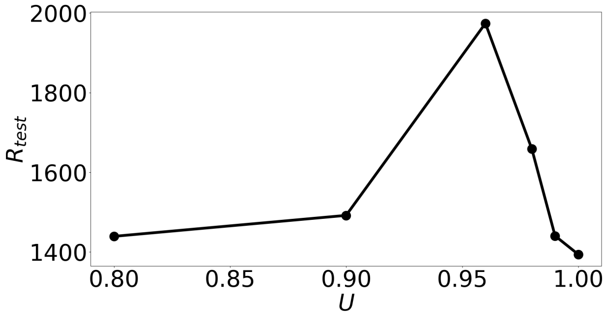

Figure 2: Testing rewards with respect to ratio uncertainty levels.

Posterior Ratio Uncertainty Level

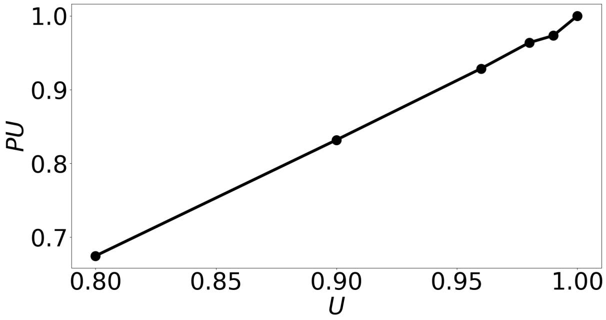

Figure 3: Posterior ratio uncertainty levels with respect to ratio uncertainty levels.

Figure 3 shows the posterior ratio uncertainty levels with respect to varying ratio uncertainty levels. We observe that the posterior ratio uncertainty level is positively related to the ratio uncertainty level. Furthermore, they are linearly related. This means we can effectively control the sampling process using predefined ratio uncertainty levels.

Conclusion & Future Work

In this work, we proposed PPO-UE, a PPO variant with enhanced sampling technique based on a well-defined uncertainty metric, i.e., ratio uncertainty level. The ratio uncertainty level provides a simple but efficient approach to balance exploration and exploitation in the policy training process. By incorporating this technique, PPO-UE can achieve faster convergence and better performance than the PPO baseline with an adaptive uncertainty level. However, this result is only limited to the given environment considered in this work. As our future work, we will delve into how to generalize our approach to multiple agent cases under more complex environments.

Acknowledgment

This work is partly supported by the Army Research Office under Grant Contract Number W91NF-20-2-0140 and NSF under Grant Numbers 2107449, 2107450, and 2107451.

References

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Ciosek, K.; Vuong, Q.; Loftin, R.; and Hofmann, K. 2019. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32.

Hämäläinen, P.; Babadi, A.; Ma, X.; and Lehtinen, J. 2020. PPO-CMA: Proximal policy optimization with covariance matrix adaptation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6. IEEE.

Khoi, N. D. H.; Van, C. P.; Tran, H. V.; and Truong, C. D. 2021. Multi-Objective Exploration for Proximal Policy Optimization. In *2020 Applying New Technology in Green Buildings (ATiGB)*, 105–109. IEEE.

Liu, Y.; and Su, X. 2022. Capacity Control and Simulation in Multi-Level Fare Class Based on Enhanced Exploration PPO Algorithm. In *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 10, 268–274. IEEE.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 1928–1937. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Peters, J.; and Schaal, S. 2006. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2219–2225. IEEE.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International Conference on Machine Learning*, 1889–1897. PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, 1995–2003. PMLR.

Xiao, Z.; Xie, N.; Yang, G.; and Du, Z. 2020. Fast-PPO: proximal policy optimization with optimal baseline method. In *2020 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 22–29. IEEE.

Zhang, J.; Zhang, Z.; Han, S.; and Lü, S. 2022. Proximal policy optimization via enhanced exploration efficiency. *Information Sciences*, 609: 750–765.