Dynamic Entropy Oscillation for Enhanced Proximal Policy Optimization Algorithm

Nguyen Anh Khoa
Faculty of Social Sciences
Radboud University
anhkhoa.nguyen@ru.nl

Abstract

Proximal Policy Optimization (PPO) is a widely used reinforcement learning technique recognized for effectiveness. However, its dependence on a fixed entropy coefficient regularization constrains the adaptability in situations that require dynamic exploration. Thus, we propose a small improvement by introducing a sinusoidal oscillation to adjust this coefficient dynamically during training. Experiments across 5 classic Gymnasium environments demonstrate that DE-PPO outperforms PPO and exhibits higher stability than PPO. This suggests a simple yet effective mechanism that can help to boost the performance of the PPO algorithm.

1 Introduction

Reinforcement learning has made significant advancements in recent years, addressing complex decision-making challenges in fields such as robotics [1], autonomous cars [2], and gaming [3]. This involves the developments of state-of-the-art algorithms such as Deep Q-Networks (DQN) [4], Soft Actor-Critic (SAC) [5], Trust Region Policy Optimization [6], etc. Among these, the Proximal Policy Optimization algorithm (PPO) [7] has become a popular choice due to its balance of simplicity, stability, and high performance. Nonetheless, PPO depends on a fixed entropy regularization term to facilitate exploration, which can inadequately regulate exploration and exploitation in dynamic training contexts.

In this paper, we propose DE-PPO (Dynamic Entropy Proximal Policy Optimization), an extension of PPO that incorporates dynamic entropy oscillation. Inspired by the principles of dynamic regularization, DE-PPO varies the entropy coefficient through sinusoidal oscillations with exponential decay. This approach allows DE-PPO to flexibly prioritize exploration or exploitation throughout the training process. We then compare our approach with the standard PPO-clipped algorithm described in the original paper [7] in some classic Gymnasium environments which includes both discrete and continuous control tasks. By doing this, we aim to study to what extent our DE-PPO can produce better results than PPO as well as its ability to escape the local optima.

2 Method

In this section, we are going to describe the clipped PPO algorithm [7] in Actor-Critic style, and our method to improve this algorithm. Finally, the experimental setup is then described at the end of this section.

2.1 Trust Region Policy Optimization

Since PPO is a simplification of the TRPO algorithm, it is crucial to revisit the details of this algorithm. TRPO is designed to optimize policies that guarantee monotonic performance improvement. The main idea is to limit the magnitude of policy updates by introducing a trust region constraint, measured using the Kullback-Leibler (KL) divergence. This leads to the optimization problem [6]:

$$\max_{\theta} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \tag{1}$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t \left[\text{KL} \left[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t) \right] \right] \leq \delta. \tag{2}$$

where $\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ and $\pi_\theta(a_t \mid s_t)$ are old and new policies, respectively. δ is a hyper-parameter that is small enough to denote the trust region threshold.

2.2 Proximal Policy Optimization

Obviously, the training of TRPO requires the computation of KL divergence, which massively increases the computational cost of this algorithm. Thus, in order to overcome such a limitation, [7] proposed PPO which can effectively penalize large changes in the policy by using a clipped surrogate objective, which is used to calculate our actor loss:

$$\mathcal{L}_t^{actor}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \tag{3}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ denotes the ratio between new and old policy, and ϵ is a hyper-parameter for policy update constraint. In addition, to encourage exploration and avoid premature convergence to deterministic policies, an entropy regularization is also incorporated in the actor loss in many actor-critic style policy optimization methods [8] [9]. For PPO, we do exactly the same:

- For an environment with discrete action spaces:

$$H(\pi_\theta) = -\sum_a \pi_\theta(a \mid s_t) \log \pi_\theta(a \mid s_t), \tag{4}$$

- For an environment with continuous action spaces:

$$H(\pi_\theta) = \frac{1}{2} \log(2\pi\sigma^2), \tag{5}$$

The critic network in PPO learns to approximate the state-value function $V(s_t)$, which is used to evaluate the current state. In order to do this, a simple mean-squared error loss is used to compute the difference between the estimated value $V(s_t)$ and the target return R_t:

$$\mathcal{L}_t^{critic} = \hat{\mathbb{E}}_t \left[(V(s_t) - R_t)^2 \right], \tag{6}$$

where

$$R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}, \tag{7}$$

Furthermore, in our PPO version, we also make use of the generalized advantage estimate (GAE) [10] method to compute the advantage function for the actor-network, which is also described in PPO introduction paper [7], and modern RL platforms such as Stable-Baselines3 [11]:

$$A_t = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_{t+l} \tag{8}$$

$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{9}$$

By combining the loss described in equation 3 and 6 with the entropy regularization 4 and 5, we have the following optimization problem:

$$\max_{\theta} \quad \mathcal{L}_t(\theta) = \hat{\mathbb{E}}_t \left[\mathcal{L}_t^{actor}(\theta) - \alpha \mathcal{L}_t^{critic} + \beta H(\pi_\theta) \right] \tag{10}$$

where α and β are coefficients.

2.3 Dynamic Entropy Oscillation

In equation 10, the entropy coefficient β is always kept constant during the PPO training. This may limit the performance of the algorithm, especially in tasks that require a dynamic exploration-exploitation. Thus, to overcome this drawback, we introduce Dynamic Entropy Oscillation which is able to dynamically adjust the entropy coefficient β over time using a sinusoidal oscillation with exponential decay:

$$\beta_t = \beta_0 + A \cdot \sin\left(2\pi f \frac{t}{T} + \phi_t\right) \tag{11}$$

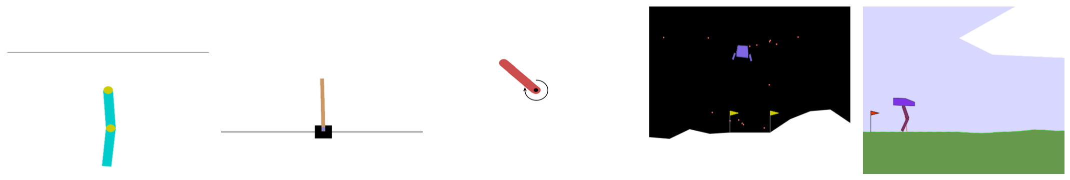
$$\text{where} \quad A = A_0 \cdot e^{\left(-\lambda \frac{t}{T}\right)} \quad \text{and} \quad \phi_t = \phi_0 + k\frac{t}{T} \tag{12}$$

where β_0 is the baseline entropy coefficient, A_0 is the baseline amplitude, λ is the decay rate, f is the frequency of oscillation, ϕ_0 is the baseline phase shift, k is the phase slope, and T is the total training time.

In essence, the sine wave introduces periodic changes to the entropy coefficient β_t, which encourages exploration that helps the agent to avoid local minima. However, since we do not want our agent to focus too much on exploration, the exponential decay $e^{-\lambda \frac{t}{T}}$ is also applied to make sure that it also leverages exploitation. The phase shift also evolves with time, facilitating non-stationary oscillations that adjust to the progress of training. Thus, this incorporation allows the agent to explore more during the early stages of the training, and gradually reduce exploration to prioritize exploitation in the later stages for convergence.

2.4 Experimental setup

In our experiments, we compare the performance of the original PPO [7] and our suggested DE-PPO. For the actor network, we used a Multi-Layer Perceptron (MLP) with 2 hidden layers, each consisting of 64 hidden units, and ReLU activation functions. This network outputs a softmax probability distribution for discrete actions or the mean of a Gaussian distribution for continuous actions. The critic network has the same architecture that outputs a single scalar value estimating the state value. Both algorithms are trained on 5 different Gymnasium environments [12] with 5 random seeds each and ADAM optimizer [13]. The 5 environments are classic control problems (Acrobot-v1, CartPole-v1, and Pendulum-v1) and Box2D (LunarLander-v2 and BipedalWalker-v3). All hyper-parameters for the additional Dynamic Entropy Oscillation module of DE-PPO are manually fine-tuned. The values of hyper-parameters for PPO and DE-PPO are presented in Table 2 and 3



(a) Acrobot (b) CartPole (c) Pendulum (d) LunarLander (e) BipedalWalker

Figure 1: Example frames from 5 Gymnasium environments: Acrobot and CartPole (discrete action spaces) and Pendulum, LunarLander, and BipedalWalker (continuous action spaces).

3 Results

For the baseline PPO, we set all hyperparameters to the identical settings specified in the original publication [7]. The metrics reported in Table 1 include the mean and standard deviation of the obtained rewards for all 5 environments with 5 different training seeds. As shown, our proposed DE-PPO outperforms the baseline PPO in all tasks, demonstrating the effectiveness of dynamic entropy oscillation in various environments. In addition, the incorporation of entropy oscillation also helps to reduce the variance of the training in most of the tasks, except for the complex BipedalWalker-v3 environment in which the standard deviation of DE-PPO is slightly higher than PPO (59.44 vs

Task	PPO	DE-PPO
Acrobot-v1	-82.34 ± 11.47	$\mathbf{-81.88 \pm 11.95}$
Cartpole-v1	481.05 ± 44.74	$\mathbf{496.37 \pm 23.63}$
Pendulum-v1	-168.08 ± 42.76	$\mathbf{-152.09 \pm 28.67}$
Lunarlander-v2	213.43 ± 51.68	$\mathbf{217.77 \pm 37.85}$
Bipedalwalker-v3	163.16 ± 53.36	$\mathbf{169.05 \pm 59.44}$

Table 1: Rewards achieved by PPO and DE-PPO averaged over last 50 iterations

53.66). Furthermore, to have further insight into the training process of both algorithms, Figure 2 demonstrates the evolution of rewards obtained. Overall, DE-PPO converges faster than PPO in all tasks, except Cartpole-v1, which shows the ability of DE-PPO to adapt to the environments and handle tasks with challenging dynamics. In tasks like Pendulum-v1 and Lunarlander-v2, DE-PPO demonstrates a clear advantage by achieving higher rewards in fewer timesteps, reflecting its superior exploration capabilities through dynamic entropy oscillation. Figure 2 also shows a high stability of DE-PPO compared to PPO. Altogether, these findings confirm the beneficial effects of the proposed DE-PPO algorithm in improving performance across various control tasks.

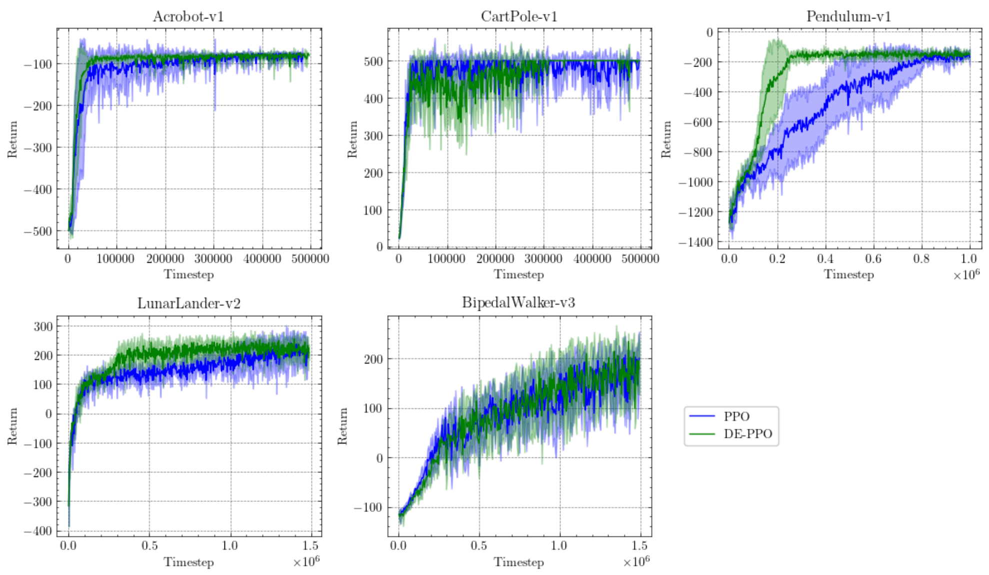


Figure 2: Performance comparison between PPO and DE-PPO across 5 Gymnasium environments.

4 Conclusion

In this paper, we introduced the Dynamic Entropy Oscillation Proximal Policy Optimization (DE-PPO) that utilizes a sinusoidal oscillation to dynamically adjust the entropy to improve exploration and exploitation. Our experiments also prove the superior capability of this improvement in increasing performance and achieving high stability in the training of PPO agents. However, there are some limitations in this work. The introduction of some additional hyper-parameters makes the algorithm more sensitive to tuning, which potentially results in suboptimal performance if not properly adjusted. Our study did not propose any method to fine-tune these parameters. In addition, due to limited computational resources, our proposed extension was only tested on simple classic control and Box2D tasks. Future works could further experiment and investigate our method on more complex and physics-based environments such as Mujuco [14] engine. Alternatively, our method can be further combined with the adaptive entropy approach presented in [15] to form a better and more comprehensive improvement for the current PPO algorithm.

References

[1] Eduardo F Morales, Rafael Murrieta-Cid, Israel Becerra, and Marco A Esquivel-Basaldua. A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning. *Intelligent Service Robotics*, 14(5):773–805, 2021.

[2] Peide Cai, Hengli Wang, Huaiyang Huang, Yuxuan Liu, and Ming Liu. Vision-based autonomous car racing using deep imitative reinforcement learning. *IEEE Robotics and Automation Letters*, 6(4):7262–7269, 2021.

[3] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*, 2019.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[6] John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[8] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, pages 151–160. PMLR, 2019.

[9] Haotian Xu, Junyu Xuan, Guangquan Zhang, and Jie Lu. Trust region policy optimization via entropy regularization for kullback–leibler divergence constraint. *Neurocomputing*, 589:127716, 2024.

[10] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[11] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

[12] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[13] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[15] Andrei Lixandru. Proximal policy optimization with adaptive exploration. *arXiv preprint arXiv:2405.04664*, 2024.

A Appendix

This appendix provides a detailed overview of the hyperparameters used for the PPO and DE-PPO algorithms across various Gymnasium environments in this paper.

Table 2: Hyperparameters for each Environment, which are used for both PPO and DE-PPO algorithms

Hyperparameter	**Acrobot-v1**	**Cartpole-v1**	**Pendulum-v1**	**LunarLander-v2**	**BipedalWalker-v3**
Steps per update (n_{steps})	1024	1024	2048	2048	4096
Max timesteps per episode	500	500	200	1000	1600
Numbers of update epochs (n_{epochs})	10	20	10	10	10
Learning rate (η)	3×10^{-4}	2×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Total simulation timesteps	500,000	500,000	1,000,000	1,500,000	1,500,000
GAE discount factor (γ)	0.99	0.99	0.99	0.99	0.99
PPO clipping parameter (ϵ)	0.2	0.2	0.2	0.2	0.2
Value loss coefficient (α)	0.5	0.5	0.5	0.5	0.5
Entropy coefficient (β)	0.0	0.0	0.0	0.01	0.0
Target KL Divergence	0.01	0.01	0.01	0.01	0.01

Table 3: Hyperparameters for sinusoidal oscillation in DE-PPO.

Parameter	**Acrobot-v1**	**Cartpole-v1**	**Pendulum-v1**	**LunarLander-v2**	**BipedalWalker-v3**
Base phase (ϕ_0)	0.0	0.0	$\pi/2$	π	0.0
Phase slope (k)	1×10^{-4}	0.3	1.0	1.5	0.1
Base amplitude (A_0)	1×10^{-4}	0.05	0.1	0.3	0.005
Base entropy coefficient (β_0)	0.0	0.01	0.01	0.05	0.001
Sinusoidal frequency (f)	2.0	1.0	1.0	0.5	0.05
Lambda decay (λ)	0.2	0.01	0.01	0.1	0.005