Journal of Machine Learning Research 23 (2022) 1-10 Submitted 1/21; Revised 5/22; Published 9/22

A Survey of Optimization Techniques on PPO

Eric Wang E246WANG@UWATERLOO.CA
Department of Mathematics
University of Waterloo

Abstract

Upon an ablation study this work examines the contribution of different optimization techniques involved in deep policy gradient algorithms to PPO algorithmic performance. Specifically, the effects of trust region, different normalization methods, clipping methods, neural network initialization, activation functions, learning rates, and structural improvement of PPO on RL tasks are discussed. The analysis shows that these detailed optimization techniques are crucial to the convergence and stability of the PPO algorithm.

Introduction

In large scale Deep Reinforcement Learning (DRL) domain, model-free Policy Gradient based methods have long been the mainstream. However, the use of same set of policies (on-policy) for sampling and optimization causes the relatively low sampling efficiency of policy gradient methods. In addition, the training process of policy gradient methods is also known to be unstable and easy to crash. By introducing important sampling and gradient update constraints and optimization techniques, the improved method of policy gradient, Proximal Policy Optimization (PPO), not only has some advantages of Trust Region Policy Optimization (TRPO) but also brings better sampling complexity. Furthermore, in practical application, the performance of PPO can still be improved by optimization in terms of many aspects. In this work, I will investigate in detail some key techniques that significantly optimize the performance of the PPO algorithm. I will also discuss the experiment results in related work as baselines to justify and explore the quantitative and qualitative effects of different techniques on the performance of PPO. Some state of the art structural or technical optimizations made for PPO will also be introduced. As an insight into the field of deep reinforcement learning by intensively understanding PPO and possible potentials from the optimization perspectives of different technical parts, this work will be of great help to those who intent to progress towards more reliable deep reinforcement learning and policy gradient based methods.

Related Work and Background

Before Sutton et al. (1999) built the generic RL framework for policy gradient methods, Williams (1992) proposed the REINFORCE algorithm for the first time that hires gradient estimates to update the parameters of neural network based RL agents Engstrom et al.

(2020). However, policy gradient based algorithms are very sensitive to the step size and it is difficult to choose the appropriate step size. Meanwhile, a large variation between the new and old policies during the training process also harms the learning stability. These issues had not been alleviated effectively until TRPO Schulman et al. (2015) and PPO Schulman et al. (2017) emerged. The use of trust regions in TRPO and PPO dates back to the conservative policy update in Kakade (2001), and the use of natural gradient descent based greedy policy update is inspired by Peters et al. (2010). Specifically, PPO applies a new objective function that can update small batches within multiple training steps, which not only theoretically solves the problem that the step size is difficult to determine in policy gradient algorithms but also brings convincing performance in most complicated RL tasks. Build upon PPO, the reinforcement learning optimization techniques to be investigated in this work are subcategorized and discussed in order of importance. The use of some of these techniques are recommended in Engstrom et al. (2020) and Huang et al. (2022). Lizhi (2022) and Huang et al. (2022) provided appreciated PPO implementation and important relevant experiment results with these techniques. All of the aforementioned work have inspired this ablation analysis.

DRL trains an agent control policy to make a sequence of decisions by interacting with the environment. Usually this is well modelled using a Markov Decision Process (MDP). A MDP can be defined by a 5-tuple $\{\mathcal{S},\ \mathcal{A},\ \mathcal{P},\ R, \gamma\}$. Here, $\mathcal{S} \in \mathbb{R}^n$ denotes the state space and $\mathcal{A} \in \mathbb{R}^m$ denotes the action space. $\mathcal{P}$: $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ denotes a transition probability density function. Particularly, given an action a_t executed under state s_t at time t, $\mathcal{P}(s_{t+1}|s_t, a_t)$ denotes the probability density of s_{t+1}. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function which assigns an scalar reward value for each transition. Finally, $\gamma \in [0, 1]$ denotes a discount factor which controls how much the agent discounts future rewards when making a decision. The ultimate goal of reinforcement learning process is to obtain an optimal policy $\pi(a|s, \theta)$ that maximizes the expectation on the cumulative rewards $J_\pi(\theta)$ where θ denotes the parameters of the policy

$$J_\pi(\theta) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t,\theta), s_{t+1} \sim P, s_0 \sim p_0} \left[\sum_{t=0}^{T} \gamma^t R\left(s_t, a_t\right) \right]. \tag{1}$$

In the deep learning approach to this problem, the policy parameters θ typically represent the learned parameters of a neural network.

Optimization Techniques and Analysis

Trust Region

Many policy gradient algorithms need to ensure that the parameter update steps remain close to the current policy in order to achieve the predictability because the update steps computed based on a particular policy are only guaranteed to be predictable in the neighborhood around that parameter Engstrom et al. (2020). Schulman et al. (2015) proposed TRPO to constrain the locally varying parameter updates by solving a optimization problem of maximizing the corresponding objective function given a constraint on the KL divergence

between the successive policies on the optimization trajectory. Simply put, TRPO tries as much as possible to update the policy in a way that can improve the value of the state, which restricts the change of the entire parameter space to a small range by adding constraints between the old and new policies. This method avoids the collapse of value caused by bad decisions and keeps the rapid and monotonous improvement. Particularly, The training process of TRPO is actually based on policy gradient but applies Trust Region in Kakade (2001). However, when we use the neural network to approximate the policy where there are a large number of parameters, the computation of nonlinear conjugate gradients estimation in TRPO is also costly. Schulman et al. (2017) proposed PPO to solve this problem by replacing the KL-constrained objective in TRPO with clipping the objective function directly. Specifically, the key contribution of PPO is the proposed policy loss function

$$L^{Clip}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2}$$

where r_t and $\hat{A}_t$ denote the ratio between the new policy and old policy and the advantage function estimate computed by GAE respectively. Since the enforcement of trust region is essential to parameter update in policy gradient methods, the evaluation of KL divergence between successive policies in training process can reflect the performance of different policy gradient based methods. On the one hand, Engstrom et al. (2020) measures the mean KL divergence between successive policies in training run of TRPO and PPO and shows that both two methods, trained by optimal parameters and implemented with some of the following code-level optimization techniques, are able to maintain a KL-based trust region, which justifies that the trust region optimization method of clipping objective used in PPO is promising. On the other hand, Engstrom et al. (2020) also demonstrates that the convincing performance of PPO and the constrained degree of policy change during training are attributed to the code-level optimization techniques to a considerable extent, which makes the following investigation of optimization techniques more meaningful.

Entropy

In reinforcement learning tasks, we usually want the probability of the output of the policy network not to be concentrated on one certain action. In other words, at least some non-zero probability should be given to other actions to achieve the appropriate exploration. Therefore, we can naturally include a regular policy entropy term in the actor loss so as to optimize the loss and the entropy term in parallel. The policy entropy in RL can be represented as

$$\mathcal{H}(\pi(\cdot|s_t)) = -\Sigma_{a_t} \pi(a_t|s_t) \log(\pi(a_t|s_t)) = \mathbb{E}_{a_t \sim \pi} \left[-\log(\pi(a_t|s_t)) \right]. \tag{3}$$

In Lizhi (2022) with the implementation of PPO the ablation study on different continuous action space environments in *gym*, *BipedalWalker* and *HalfCheetah*, was performed as shown in Figure 1, where the term *max* with the red curve denotes the PPO algorithm including every optimization technique in this work, while the pink curve denotes the *max* PPO algorithm with the policy entropy turned off. The similar notation will be used in the following experiment figures. Figure 1 shows that the introduction of policy entropy significantly increases the model convergence and cumulative reward. A concern of this

technique is that the excessive exploration brought by the inclusion of policy entropy might lead the model to the sub-optimal solution.

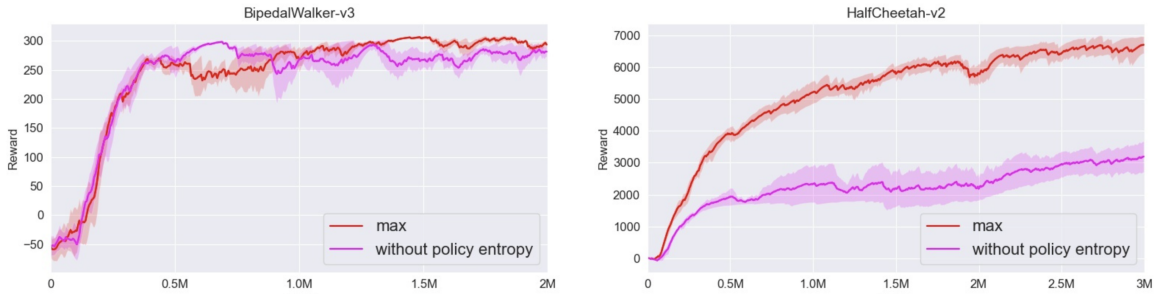

Figure 1: Policy Entropy

Normalization

The performance of RL algorithms is well known as quite sensitive to parameter variation, therefore, in training process of PPO, normalization operation could be applied to different terms involved in model parameters and objective function to help maintain the stability of parameter update. Firstly, rewards could be normalized before feeding into the value function to avoid huge fluctuations. A very common normalization method here is to divide each reward by a fixed value. Instead, Engstrom et al. (2020) proposed a discount-based scaling scheme, Reward Scaling, where the rewards are divided through by the standard deviation of a rolling discounted sum of the rewards (without subtracting and re-adding the mean). Reward Scaling considers rolling discounted sum of the rewards, dynamically computes its standard deviation and scales the current reward by the standard deviation. Since this method scales the rewards using the rolling discounted rewards, the scaling factor naturally includes the previous reward information. Lizhi (2022) does demonstrates that Reward Scaling, after replacing the common reward normalization method, sometimes helps model converge faster as shown in Figure 2, but an issue with normalizing rewards is that it might also disrupt training if the events with extremely high or low rewards occur rarely while most episodes only experience common events with moderate rewards.

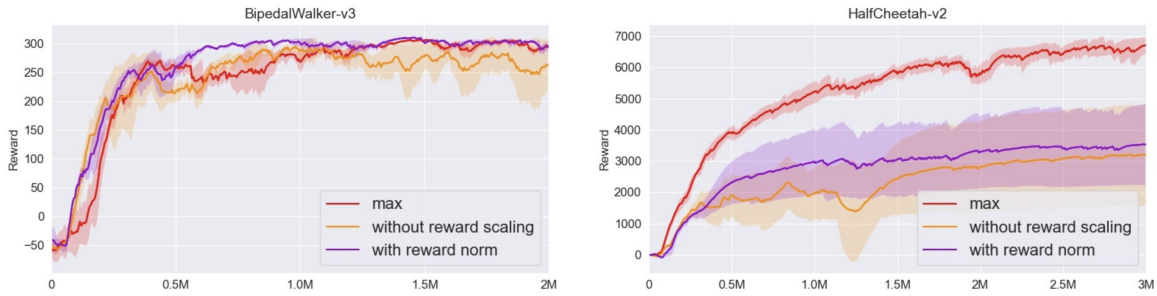

Figure 2: Reward Normalization

Secondly, note that in the loss function of policy gradient based algorithms that policy updates by, Advantage calculated by Generalized Advantage Estimation (GAE) can be

considered as a relative evaluation of a specific action. Simply put, it refers to how good the action is compared to other actions given a specific state. A critical problem of the policy gradient based methods is the bias of the data. Because the advantage estimate with bias is not completely accurate, then if the policy is updated too far at once, the next sampling will be quite deviated. Though PPO leverages importance sampling and clipped surrogate objective to solve this problem, the use of the original advantage might still make PPO training unstable. Tucker et al. (2018) shows that since we may assume that the distribution of advantage is relatively stable, the normalization of advantage, compared to rewards, can be more direct, i.e. operating on data batch wise (subtracting their mean and dividing them by their standard deviation) to improve the performance of algorithms. Furthermore, the normalization of advantage can be subcategorized into two ways according to minibatch level versus the whole batch level. Andrychowicz et al. (2020) and Lizhi (2022) found that advantage normalization has considerably important influence to model convergence and batch normalization outperforms minibatch normalization as shown in Figure 3. This result is reasonable because in minibatch normalization the mean values and standard deviations fluctuate substantially for each computation so as to hinder the model convergence.

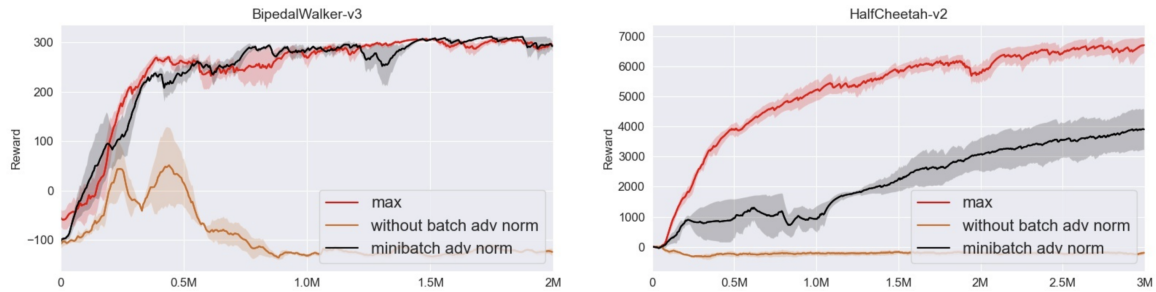

Figure 3: Advantage Normalization

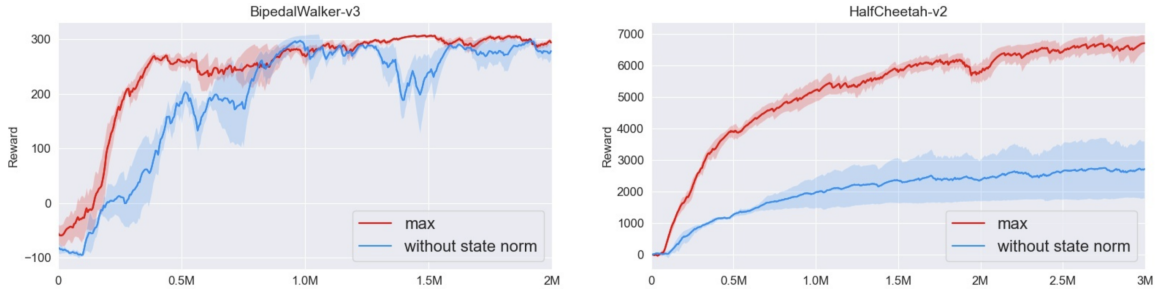

Figure 4: Observation Normalization

Engstrom et al. (2020) also proposed observation normalization in a similar manner to the rewards to stabilize the PPO training process. In particular, we can normalize the observations to zero-mean, unit-variance vectors. Like we do in reward normalization, specifically, a mean value and a standard deviation of observations can be dynamically maintained and used for normalizing the current observation. This method is rational

because stable observations as input are beneficial to the neural network training. Though Lizhi (2022) shows that observation normalization has a certain improvement on the model convergence and cumulative reward as shown in Figure 4, a potential problem with state normalization is that it might distort the true distribution of observations, resulting in model convergence to the sub-optimal solution.

Clipping

Recall that PPO leverages clipped objective to make the difference between the new policy obtained after importance sampling update and the previous policy as much close. Simply put, actually we ignore the change in probability ratio when the probability ratio $r_t(\theta)$ wants to make the objective function better, while take it into account when the ratio makes the objective worse. In other words, we cannot simply allow the objective function to become better without restraints, because this often makes the new policy far from the previous policy. Meanwhile, Schulman et al. (2017) actually performs a clipping operation on the loss of the value function similar to the loss of the policy, instead of direct regression to target values, in fitting the value network, in order to improve the training stability Engstrom et al. (2020). Furthermore, to improve the model scalability so that PPO can be applied in wider scenarios, reward clipping and observation clipping, following the corresponding normalization, are of second importance.

Except for value function clipping, reward clipping and observation clipping, another important clipping operation is gradient clipping which is generally used to avoid the gradient explosion problem that may occur in neural network training of DRL. Schulman et al. (2017) proposed a clipping-by-norm method such that the norm of the concatenated gradients of all parameters does not exceed a threshold so as to stabilize the training process. Though Lizhi (2022) shows that PPO might converge faster with gradient clipping method as shown in Figure 5, a concern about gradient clipping is that the appropriate threshold is not easy to determine. Overall, though Ilyas et al. (2018) states that the effect of value function clipping might not be significant enough to the model improvement, it is still safe to say that these clipping techniques have no harm in the algorithm performance.

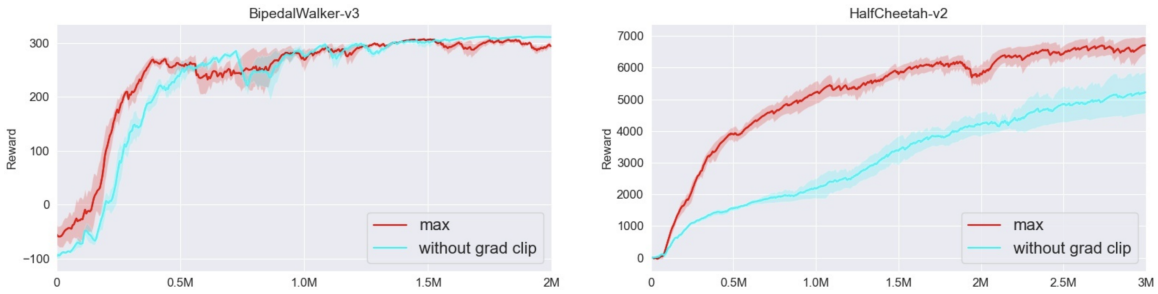

Figure 5: Gradient Clipping

Initialization, Activation and Learning Rate Decay

There are also many optimization techniques designed for the different stages of network training of PPO. Common neural network implementation (in TensorFlow and Torch) for policy and value networks applies Xavier initialization Glorot and Bengio (2010) by default. Instead in PPO, Schulman et al. (2017) proposed to use orthogonal initialization with scaling that varies from layer to layer Saxe et al. (2013). In PPO implementation, we can achieve orthogonal initialization by normalizing weights generated from independent normal distributions row wise or using singular value decomposition. Lizhi (2022) and Huang et al. (2022) show that the orthogonal network initialization also improves the model training performance as shown in Figure 6, which is reasonable because orthogonal initialization of neural network can effectively help prevent gradient explosion and vanishing gradient at the beginning of the training process.

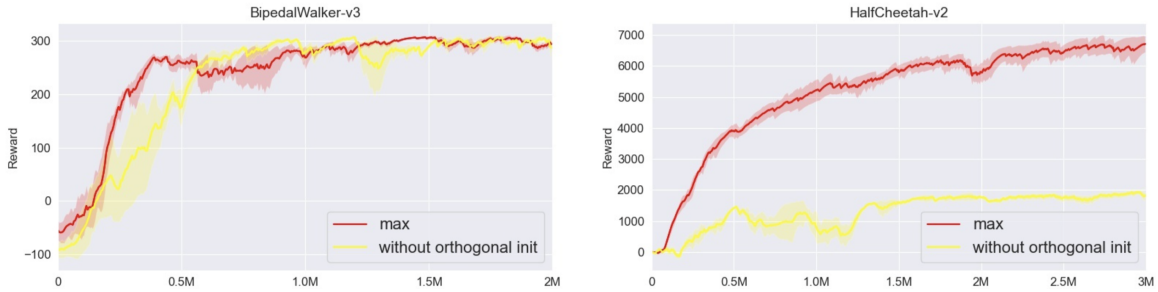

Figure 6: Network Initialization

The ReLU activation function is very common in general deep learning algorithms, but Engstrom et al. (2020) recommended the use of Tanh activation function in PPO. Lizhi (2022) demonstrates that Tanh is slightly better than ReLU in two different testing environments as shown in Figure 7, which might be because the larger value range of the derivative of the Tanh function can help alleviate the problem of vanishing gradient to some extent and the performance of Tanh is desirable when the features are significantly different, but it still may lead to saturation problem.

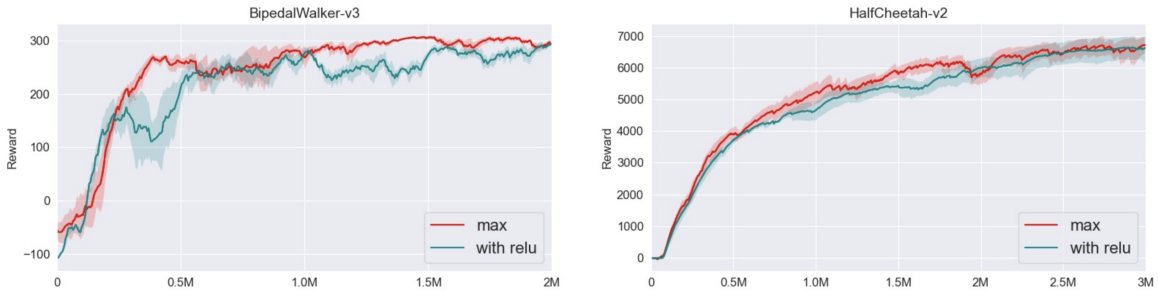

Figure 7: Activation Function

Using decaying learning rate Kingma and Ba (2014) at different learning epochs is also a common technique in deep learning. A simple linearly decaying learning rate is able to

improve the stability of training, which can be verified in Huang et al. (2022) and Lizhi (2022) as shown in Figure 8. However, though learning rate decay may make the model converge more stable, if a saddle point is encountered in the process of optimization the model may not be able to keep converging.

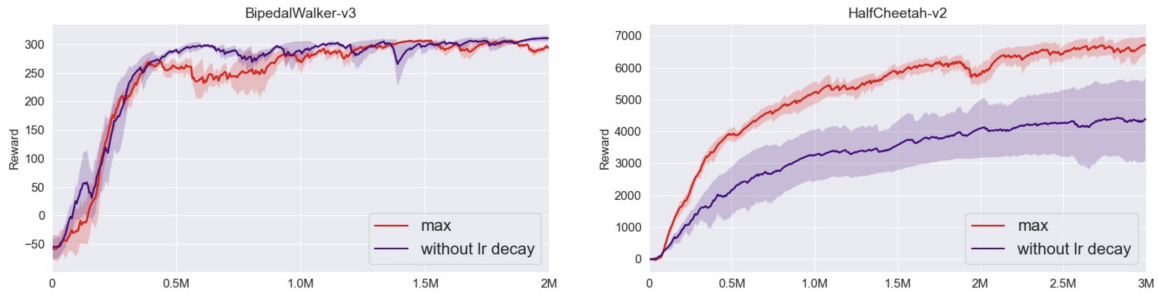

Figure 8: Learning Rate Decay

Phasic Policy Gradient

For PPO or other Actor Critic based RL algorithms, in the policy and value networks, the previous network parameters are shared or completely independent. To better optimize sample efficiency Cobbe et al. (2021) proposed Phasic Policy Gradient (PPG), where PPG uses independent value function network and policy network so that their training processes do not affect each other. Meanwhile, an additional auxiliary stage (knowledge distillation) is hired to transfer useful information from the value function network to the policy network, which not only avoids mutual interference in their training processes but also shares a part of the representation information. Experiments show that PPG significantly improves data utilization efficiency and achieves better performance than PPO. Given the limited space, more analysis and experiment results for PPG could not be discussed here.

Conclusion

Through an ablation study, in this work I examined the contribution of different optimization techniques involved in deep policy gradient based algorithms to PPO algorithmic performance. In conclusion, these detailed optimization techniques, including trust region, different normalization methods, clipping methods, neural network initialization, activation functions, learning rates, are all crucial and have different level of positive influence on the convergence and stability of PPO algorithm. In algorithm design or application, one should carefully consider the impact that each part of the algorithm may have on the performance of the model and employ targeted adjustments according to both experiment results and optimization techniques.

On the other hand, overthinking parameter tuning with these optimization techniques but ignoring the algorithmic adaptability is putting the cart before the horse. In this sense, I recommend that one also focuses more on structural improvements of PPO and other policy gradient based algorithms (i.e. PPG is an excellent starting point) to achieve greater sample efficiency so as to solve more complicated problems in reinforcement learning domain.

References

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International conference on learning representations*, 2020.

Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Are deep policy gradient algorithms truly policy gradient algorithms? 2018.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lizhi. Lizhi-sjtu/drl-code-pytorch: Concise pytorch implements of drl algorithms, including reinforce, a2c, dqn, ppo(discrete and continuous), ddpg, td3, sac., 2022. URL `https://github.com/Lizhi-sjtu/DRL-code-pytorch/tree/main/5.PPO-continuous`.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

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.

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

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *International conference on machine learning*, pages 5015–5024. PMLR, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.