Improvements to Proximal Policy Optimization through efficient mixture of on/off policy gradient methods

Zichao Yan
School of Computer Science
McGill University
zichao.yan@mail.mcgill.ca

Jianing Sun
Department of Electrical and Computer Engineering
McGill University
jianing.sun@mail.mcgill.ca

Abstract

Reinforcement learning algorithms are usually hard to train because of the high variance and, when talking about on-policy methods specifically, the inefficient use of available data. To reduce variance, many measures have been proposed to keep it in check, which include the General Advantage Estimation, a series of conservative policy optimization methods. For data efficiency, there are effective tricks such as experience replay and especially, the Hindsight Experience Replay which can even make the training agent learn from bad samples. In this project, we are interested in unifying on- and off-policy training methods, namely the Q-PROP algorithm and its newer and slightly different adaptation, the Interpolated Policy Gradient method. These are newly proposed methods, and we will demonstrate some augmentation to boost their performance empirically. Experiments are conducted in the Mujoco robotic continuous control tasks from the openAI gym environment.

1 Introduction

Model free reinforcement learning have been enjoying wild success at solving various control tasks, and some of the recent advancements in policy gradient methods have successfully solved a range of different but nonetheless difficult problems, which includes playing 2D computer games[9], and continuous robotic control tasks [14, 4, 3, 7]. Deep neural networks are greatly favored for the parameterization of policy and value functions, which allows end-to-end policy learning directly from high dimensional inputs via automatic feature mapping from images via Convolutional Neural Network (CNN). Direct numerical inputs of contrived state observations are also possible and only requires simpler Multi-layer Perceptrons (MLP). For on-policy gradient methods, there have already been nice theoretical guarantees of convergence if each layer is linear.

These model free reinforcement learning methods mainly consists of on-policy methods and off-policy methods. On-policy methods can be summarized as likelihood ratio methods because they all involve a probability of a trajectory happening under current policy. Likelihood ratio methods consist of REINFORCE , natural policy gradient methods [6] and their more recent adaptations, the Trust Region Policy Gradient method (TRPO) and Proximal Policy Gradient methods (PPO) [12, 14], all of which use on-policy samples for Monte Carlo returns to offer unbiased estimation of the gradients, hence resulting in high sample complexity in training these models. More specifically, NPO, TRPO and PPO are conservative methods in the sense that they constrain the updates made to current policy, by l2 distance, hard kl divergence constrains and soft kl divergence penalty respectively for each.

On the other hand, off-policy methods, such as Q-learning and off-policy actor-critic methods namely rhe DDPG[17, 7, 9], with their explicit use of experience replay, can make full use of all training samples. But these off-policy methods are biased after all, and their convergence in deep neural networks with non-linearity settings is not guaranteed. However, they do tend to have lower variance, due to a high data efficiency and the use of a deterministic policy instead of a stochastic one.

The most recent attempt to achieve a more robust and a unified view of on- and off-policy methods, would be the Q-PROP algorithm [4] and the Interpolated Policy Gradient method [3]. These methods combine likelihood ratio gradient methods with actor-critic methods, using on-policy data to train a policy and off-policy data to fit a state-action value critic. Originally these methods use TRPO as part of on-policy gradient estimation, and DDPG as a control variate in the objective, so that each time an update is made to the policy, it's both on- and off-policy.

Q-PROP and IPG have been shown to have greater improvement compared to mere conservative policy gradient methods and the DDPG. In this paper, we will further adapt these mixture methods with PPO instead, and apply Hindsight Experience Replay [1] (HER) to further improve their training speed.

2 Backgrounds

A reinforcement learning algorithm, at time step t as the learning agent emits an action a_t at state s_t with regard to a stochastic policy $\pi(a_t|s_t)$, takes in a quadruple in the form of $(s_t, a_t, r_{t+1}, s_{t+1})$. The agent receives a reward r_{t+1} as determined by a scalar reward function $r(s_t, a_t)$, and the environment transits to the next state s_{t+1} according to the state-transition dynamics $p(s_{t+1}|s_t, a_t)$. Then the goal of reinforcement learning is to maximize the expected total discounted sum of rewards along a trajectory from the initial states, given by $\mathbb{E}_{\pi_\theta}[R_0]$, where $R_0 = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ and the stochastic policy is parameterized by θ.

Before going into the mixed on-/off-policy methods Q-PROP and IPG, we will cover the basics of each of their integral part, the likelihood ratio methods including REINFORCE, TRPO and PPO, and the deterministic policy gradient method DDPG which make use of off-policy replay buffers.

2.1 Likelihood Ratio Policy Gradient Methods

The standard policy gradient, known as the REINFORCE algorithm, maximizes the objective function 1 via gradient ascent.

$$\nabla J(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \nabla_\theta ln\pi_\theta(a_t|s_t)\gamma^t R_t] = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta ln\pi_\theta(a_t|s_t)(R_t - b(s_t))]. \quad (1)$$

The choice of baseline $b(s_t)$ is usually a differentiable state-value function parameterization $v_\pi(s_t)$, to alleviate the high variance introduced by the Monte Carlo return R_t. An alternative to the full Monte Carlo REINFORCE is to do a value backup, and use advantage estimation shown in 2 in place for MC returns:

$$\begin{aligned} A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) &= r(s_t, a_t) + \gamma\, \mathbb{E}_{s\sim p(\cdot|s_t,a_t)}[V_\pi(s)] - V_\pi(s_t) \\ &= r(s_t, a_t) + \gamma V_\pi(s_{t+1}) - V_\pi(s_t) \end{aligned} \quad (2)$$

However, this advantage estimate $A_\pi(s_t, a_t)$ is only accurate when the baseline value function $V_\pi(s_t)$ is accurate, otherwise it would yield biased policy gradient estimates.

It's easy to see that the advantage formula in refadvantage is a "one step" estimate bootstraped from the baseline, and its generalization to multi-step advantage estimation is straightforward.

$$\begin{aligned} A_\pi^{(1)}(s_t, a_t) &= -V_\pi(s_t) + r(s_t, a_t) + \gamma V_\pi(s_{t+1}) \\ A_\pi^{(2)}(s_t, a_t) &= -V_\pi(s_t) + r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 V_\pi(s_{t+2}) \\ \\ A_\pi^{(k)}(s_t, a_t) &= -V_\pi(s_t) + r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + ... + \gamma^k V_\pi(s_{t+k}) \end{aligned} \quad (3)$$

And the Generalized Advantage Estimation [13] is defined analogously to TD(λ).

$$\begin{aligned} A_{\pi}^{GAE}(s_t, a_t) &= (1-\lambda)(A_{\pi}^{(1)}(s_t, a_t) + \gamma A_{\pi}^{(2)}(s_t, a_t) + \gamma^2 A_{\pi}^{(3)}(s_t, a_t) + ...) \\ &= \sum_{l=0}^{\infty} (\lambda\gamma)^l \delta(s_{t+l}, a_{t+l}) \end{aligned} \tag{4}$$

where $\delta(s_{t+l})$ is the one-step Temporal Difference (TD) residual at time step $t + l$, and by its definition, $\delta(s_t, a_t) = r(s_t, a_t) + \gamma V_{\pi}(s_{t+1}) - V_{\pi}(s_t)$.

This Generalized Advantage Estimation (GAE) is unbiased when λ is set to 1 but with prohibiting variance; otherwise for λ smaller than 1 it induces bias but shrinks variance. Hence λ is a bias-variance trade-off term and is often chosen close to 1 to maintain a proper balance. In the algorithms we considered later on, A^{GAE} is always used in place of advantage estimation.

2.2 Proximal Policy Optimization Methods

Proximal Policy Optimization methods (PPO) is very similar to likelihood ratio methods, and in fact it is a simplified version for Trust Region Policy Optimization(TRPO), which sets a hard constraints of policy updates using KL divergence.

TRPO has a linear approximation to the objective function 5 and a quadratic approximation to the KL constraints $D_{KL}(\theta_{old}, \theta) \approx \frac{1}{2}(\theta_{old}, \theta)^T F(\theta_{old}, \theta)$ where F is the Fisher Information Matrix (FIM). In every iteration TRPO computes a search direction by solving the linear system $Fx = g$, where g is the gradient to the objective function 5, and F doesn't need to be computed analytically with conjugate gradient descent and Hessian Vector Product, see appendix C in [12].

PPO is much simpler with nonetheless comparable performance to TRPO, which only needs stochastic gradient descent on a surrogate objective:

$$\mathbb{E}_t[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}} A(s_t, a_t) - \beta KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_{\theta}(\cdot|s_t)]] \tag{5}$$

where β is adjusted on the fly, and we refer the readers to [5] for detailed algorithm description. We use the exact implementation as theirs.

The on-policy part of subsequent Q-PROP and IPG would be implemented using this PPO formula.

2.3 Deterministic Policy Gradient Methods

Deep Deterministic Policy Gradient (DDPG) [8] uses neural network function approximators to learn in large state and action spaces online. Compared with directly implementing Q learning with neural networks proved to be unstable in many cases, DDPG modified the actor-critic by using "soft" target updates, rather than directly copying the weights.It creates a copy of the actor and critic networks, $Q'(s, a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$ respectively, that are used for calculating the target values. The weights of these target networks are then updated by having them slowly track the learned networks.DDPG optimizes a continuous deterministic policy by the following update equations, where $\theta^{Q'}$ and $\theta^{\mu'}$ denotes the weights for target network Q' and μ':

$$\begin{aligned} \theta^{Q\prime} &\to \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu\prime} &\to \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{6}$$

This provides the following deterministic policy gradient through the critic:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s_i} \tag{7}$$

This policy gradient is generally biased due to the imperfect estimator and off-policy state sampling. Deterministic Policy Gradient algorithms therefore allow training the policy on off-policy samples, at the cost of introducing potentially unbounded bias into the gradient estimate. This usually makes off-policy algorithms less stable during learning, compared to on-policy algorithms using a large batch size for each update.

3 Methods

Complementing the unbiased but high variance PPO method with a deterministic but biased DDPG, we would then come to the formulation of Q-PROP [4] and IPG [3]. We first present our adaptation to the Q-PROP algorithm with PPO in Section 3.1, delineating the necessary formulation. Then we present our augmentation to IPG with HER in Section 3.2.

3.1 Q-PROP

The whole method consists of a critic network Q_w, a stochastic policy π_θ and a baseline value function V_ϕ. The Q_w is trained exactly as in the DDPG algorithm [7], from off-policy samples using a experience replay buffer, and together with the determined part of the stochastic policy the mean of a parameterized gaussian they complete the specification of a DDPG.

On the other hand, the baseline V_ϕ is fitted with current trajectory samples hence is on-policy, as in the PPO algorithm[4], and with the original stochastic policy, they form a PPO.

As for the objective function, the first part of it is a trajectory likelihood ratio multiplies an on-policy estimation from GAE using the baseline V_ϕ, component-wise with regard to time steps, as in the PPO method. The second part is an off-policy estimation from the Q critic, derived as a control variate and luckily to share the same formula as in DDPG.

We omit the derivation of how the use control variates by Q_w turns out to be the DDPG policy gradient estimation. The Q-PROP policy gradient estimator is still straight-forward, and is shown in 8 as in [4].

$$\nabla_\theta J(\theta) = \mathbb{E}_{\rho_\pi,\pi}[\nabla_\theta log\pi_\theta(a_t, s_t)(\hat{A}(s_t|a_t) - \bar{A}_w(s_t, a_t))] + \mathbb{E}_{\rho_\pi}[\nabla_a Q_w(s_t, a)|_{a=\mu_\theta(s_t)} \nabla_\theta \mu_\theta(s_t)] \tag{8}$$

$$\bar{A}_w(s_t, a_t) = \nabla_a Q_w(s_t, a)|_{a=\mu_\theta(s_t)}(a_t - \mu_\theta(s_t)) \tag{9}$$

This formula consists of a residual REINFORCE gradient in the first expectation bracket, and can be readily substituted for PPO with soft penalty on the kl divergence between old policy and new policy. $\bar{A}_w(s_t, a_t)$ in 9 and the second expectation in 8 is the control variate, and in practice a weighting variable is used, to account for situations where $\hat{A}$ and $\bar{A}$ are negatively correlated. In these situations, the weighting variables would eliminate the control variates, which is also the off-policy policy gradients, from the update formula.

3.2 IPG

In Q-PROP the objective function is evaluated on-policy, that is using on-policy samples fitting both the PPO and the DDPG; the replay buffer and off-policy updates is used for training the value function of Critic only. While Q-prop does not introduce additional bias to the gradient estimator, the policy updates do not use off-policy data. Interpolated Policy Gradient (IPG) [3] merges on-policy and off-policy learning by making use of the off-policy data from replay buffer, which generates the possibility of combing Hindsight Experience Replay (HER) with state-of-art on-policy algorithms for multi-goal based reinforcement learning tasks. IPG directly interpolated on- and off-policy gradient objective function:

$$\nabla_\theta J(\theta) \approx (1 - v)\mathbb{E}_{\rho^\pi,\pi}\left[\nabla_\theta log\pi_\theta(a_t|s_t)\hat{A}(s_t, a_t)\right] + v\mathbb{E}_{\rho^\beta}\left[\nabla_\theta \overline{Q}^\pi_w(s_t)\right] \tag{10}$$

In particular, IPG generalized the deterministic policy gradient through the critic as $\nabla_\theta \overline{Q}_w(s_t) = \nabla_\theta \mathbb{E}_\pi\left[Q^\pi_w(s_t, \cdot)\right]$, which is an off-policy actor-critic algorithm and is closely connected to DDPG. Except that is does not use target policy network and its use of a stochastic policy enables on-policy exploration, proximal policy optimization updates, and no heuristic additive exploration noise.

3.3 On-Policy Hindsight Experience Replay in Multi-goal RL Environment

3.3.1 Multi-goal RL

We are interested in training agents which learn to achieve multiple different goals. We train policies and value functions which take as input not only a state $s \in S$ but also a goal $g \in G$. OpenAI came up with a new reinforcement learning environment [11] which extended from Gym and MuJoCo [15] - Robotics. It is a suite of challenging continuous control tasks based on currently existing robotics hardware. The tasks include pushing, sliding and pick & place with a Fetch robotic arm aw well aw in-hand object manipulation with a Shadow Dexterous Hand. All tasks have sparse binary rewards and follow a Multi-Goal Reinforcement Learning framework in which an agent is told what to do using an additional input.

The Fetch Environments are based on the 7-DoF Fetch robotics arm, which has a two-fingered parallel gripper. In all Fetch tasks, the goal is 3-dimensional and describes the desired position of the object (or the end-effector for reaching). Rewards are binary: The agent obtains a reward of 0 if the object is at the target location (within a tolerance of 5 cm) and -1 otherwise. Actions are 4-dimensional: 3 dimensions specify the desired gripper movement in Cartesian coordinates and the last dimension controls opening and closing of the gripper. Observations include the Cartesian position of the gripper, its linear velocity as well as the position and linear velocity of the robot's gripper. If an object is present, we also include the object's Cartesian position.

For this project, experiments are mainly based on FetchReach-v0 environment. Future work can be done in the other three Fetch environments. However, based on the experiments from the original HER paper by OpenAI, experiments on these environments apparently have a high requirement for the hardware. That is also the reason why we only chose FetchReach as the simulation environment for our project.

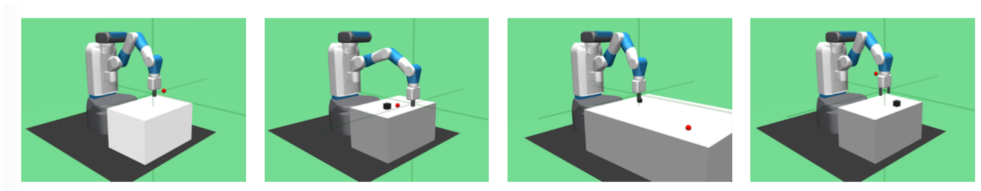

Figure 1: The four proposed Fetch environments: FetchReach, FetchPush, FetchSlide, and Fetch-PickAndPlace.

3.3.2 On-Policy HER with IPG

The idea behind Hindsight Experience Replay (HER) [**?**] is simple: after experiencing some episode $s_0, s_1, ..., s_T$ we store in the replay buffer every transition $s_t \rightarrow s_{t+1}$ not only with the original goal used for this episode but also with a sublet of other goals. HER may be seen as a form of implicit curriculum as the goals used for replay naturally shift from ones which are simple to achieve even by a random agent to more difficult ones. However, in contrast to explicit curriculum, HER does not require having any control over the distribution of initial environment states. In short, HER is an extension for normal experience replay, which mainly focuses on Multi-goal based RL tasks. Figure 2 illustrates the description of HER with standard experience replay.

However, initial HER was purposed with a obvious limitation: it can only be applied on off-policy algorithms, such as DDPG. The question we ask ourselves is: how to combine HER with state-of-the-art on-policy RL algorithms like PPO? Some preliminary results with vanilla Policy Gradients were presented by Rauber et al. (2017), but this approach needs to be tested on more challenging environments. One possible option would also be to use techniques similar to the ones employed in IPG, which we mentioned in section 3.2. Therefore, we applied HER on the off-policy term of the objective function of IPG. As IPG is basically a merge of both on- and off-policy gradient methods, we eventually achieved on-policy Hindsight Experience Replay and tested it in the FetchReach environment.

```
**for** $t = 0, T-1$ **do**
    $r_t := r(s_t, a_t, g)$
    Store the transition $(s_t||g,\ a_t,\ r_t,\ s_{t+1}||g)$ in $R$          ▷ standard experience replay
    Sample a set of additional goals for replay $G := \mathbb{S}($**current episode**$)$
    **for** $g' \in G$ **do**
        $r' := r(s_t, a_t, g')$
        Store the transition $(s_t||g',\ a_t,\ r',\ s_{t+1}||g')$ in $R$          ▷ HER
    **end for**
**end for**
```

Figure 2: Hindsight Experience Replay (HER)

4 Results

We demonstrate the performance of the modified algorithms, namely the Q-PROP and IPG upgraded with PPO and HER, with the baseline PPO. Experiments are confined to the Hopper and FetchReach environment only, due to limitation of our resources and scope of this report; also we believe these two environments are sufficient to show the difference between the algorithms.

4.1 Improvements from PPO to Q-PROP

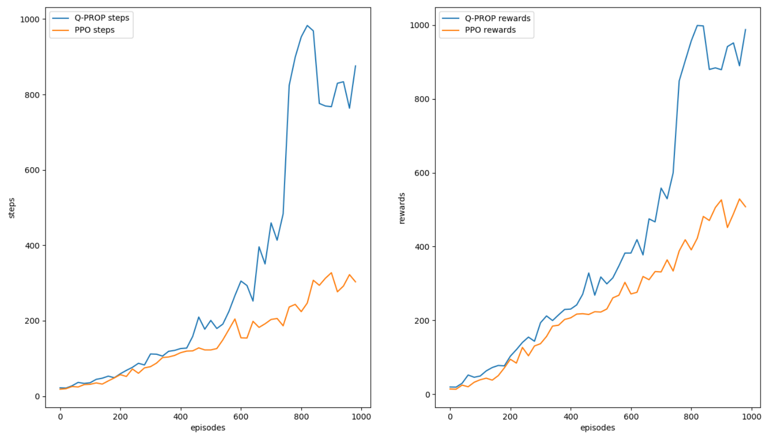

Figure 3: Experiments in the Hopper-v2 environment with Q-PROP and baseline PPO. Figure on the left shows average time steps per episode, and figure on the right shows average rewards per episode.

Hopper-v2 environment doesn't have goals in the observations, hence IPG+HER is not suited for this task. We present the comparison between Q-PROP and PPO instead in Figure 3, to demonstrate that mixing off-policy targets with traditional on-policy likelihood ratio objectives can yield substantial improvements.

4.2 Comparison between IPG+HER, Q-PROP and PPO

The experiments shown in 4 are evaluated in the FetchReach-v0 environment. This environment limits per episode time steps to 50 maximum, so that data efficiency plays a very important roles, and

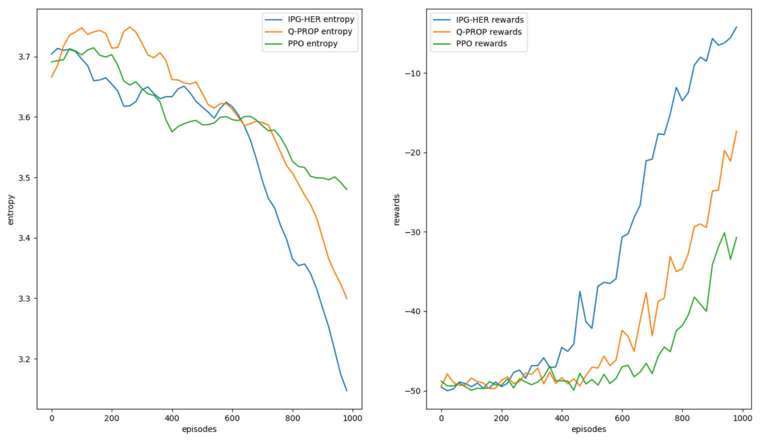

Figure 4: Compare the three algorithms in the FetchReach-v0 environment. Figure on the left shows policy entropy, which represents the information carried by the probability distribution of the stochastic policy. Hence the lower the entropy is, the more certain the policy is and the lower variance there is. Figure on the right shows averaged episodic rewards. It's straightforward to see that IPG+HER outperforms Q-PROP which outperforms PPO per se in both metrics.

with experience replay and off-policy gradient updates, learning speed is greatly improved compared to baseline on-policy PPO method, as reflected in the picture.

Further we can contrast the improvement from Q-PROP to IPG+HER in this environment. Since Q-PROP and IPG shares the same philosophy hence quite similar to each other, their difference can be marginal in simpler environments e.g. Hopper-v2.

After all in this FetchReach-v0 environment we are able to demonstrate the superiority of IPG+HER to Q-PROP to PPO.

4.3 Effect of HER

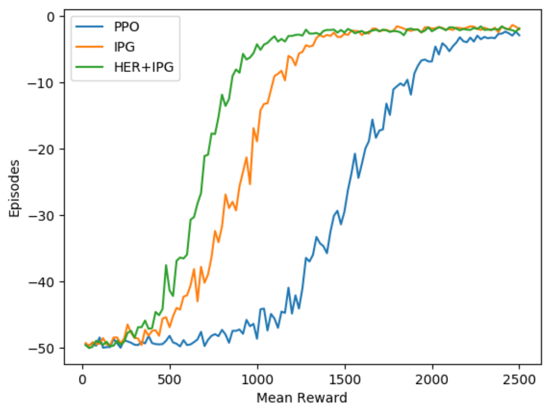

Figure 5: The combination of HER has substantial improvements compared with original IPG algorithm, and has much better performance than the baseline algorithm, PPO.

In this section we evaluate whether on-policy HER, IPG improves performance in the case where there is only one goal we care about, compared with the baseline experiment with PPO algorithm.

We did a series of experiments step by step from initial PPO algorithm to IPG with same environment seed, and extended to HER with IPG to evaluate the performance of these algorithms in FetchReach environment. From Figure 5 it is clear that practical results match with theoretical analyzes. In the end HER+IPG algorithm can achieve 50% success rate after 600 episodes (each episode has 50 timesteps), compared with only 20% and almost 0% success rate of IPG and PPO respectively.It confirms that HER is a crucial element which makes learning from binary rewards possible.

5 Conclusion

The mixture policy gradient methods that jointly train an on-policy target and an off-policy target, namely the Q-PROP and IPG algorithm, and our improvements to them using PPO and HER, are proved to have state-of-the-art learning performance in terms of higher sample efficiency and lower variance.

This claim is empirically verified in the FetchReach environment. Future experiments can be carried out in more difficult environments.

It has to noted that our adaptation of the algorithms: Q-PROP and IPG+HER, still exhibits a certain degree of variance. But we could confirm their overall superiority over mere on-policy PPO method.

Acknowledgments

Zichao Yan implements the Q-PROP, and Jianing Sun implements IPG plus HER. Experiments are carried out separately based on different interests, and results are gathered for final presentation and reports. This report is written by the two authors both.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017.

[3] Shixiang Gu, Tim Lillicrap, Richard E Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3849–3858, 2017.

[4] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

[5] Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[6] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[7] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[8] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[9] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[10] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[11] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018.

[12] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[13] 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.

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

[15] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012.

[16] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[17] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.