Addressing Reward Engineering for Deep Reinforcement Learning on Multi-stage Task

Bin Chen[1,2] and Jianhua Su[1(✉)]

[1] The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
{chenbin2017,jianhua.su}@ia.ac.cn
[2] The school of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

Abstract. In the field of robotics, it is a challenge to deal with multi-stage tasks based on Deep reinforcement learning (Deep RL). Previous researches have shown manually shaping a reward function could easily result in sub-optimal performance, hence choosing a sparse reward is a natural and sensible decision in many cases. However, it is rare for the agent to explore a non-zero reward with the increase of the horizon under the sparse reward, which makes it difficult to learn an agent to deal with multi-stage task. In this paper, we aim to develop a Deep RL based policy through fully utilizing the demonstrations to address this problem. We use the learned policy to solve some difficult multi-stage tasks, such as picking-and-place, stacking blocks, and achieve good results. A video of our experiments can be found at: https://youtu.be/6BulNjqDg3I.

Keywords: Deep reinforcement learning · Robotic manipulation · Multi-stage task

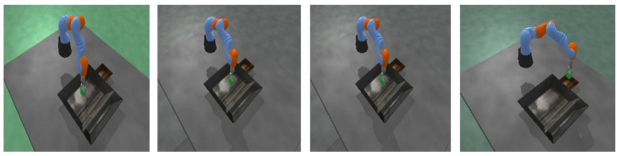
1 Introduction

In recent years, the research of robotic grasping strategies has attracted increasingly attention in the field of robotic manipulation. A key challenge is how to perform a successful grasping without a prior knowledge of the manipulated object. Analytic or model-based method can achieve excellent performance to situations that satisfy their assumption. However, the complexity and diversity of objects in a new environment have a tendency to confound these assumptions, hence learning-based methods have emerged as a powerful complement. Recently, using end-to-end approaches to handle robotic manipulation tasks has achieved great success both in the simulation environment [4] and in the real environment [9].

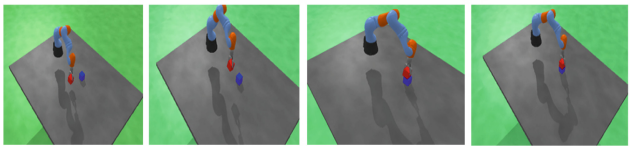
Using deep reinforcement learning to handle multi-stage tasks is often challenging due to the lack of professional knowledge of a special task. A way to deal

© Springer Nature Switzerland AG 2019
T. Gedeon et al. (Eds.): ICONIP 2019, CCIS 1143, pp. 309–317, 2019.
https://doi.org/10.1007/978-3-030-36802-9_33

with this situation is only to provide a sparse reward − +1 for success and 0 for failure. In many cases it is difficult for an agent to learn a feasible action at every moment in each episode for the delaying of the reward.



(a) Pick-and-place



(b) Stack blocks

Fig. 1. Frames from rollouts of the learned policy are shown above. (a) This is a picking-and-place task, which is a classic robotic manipulation task from industry to service robot. (b) This is a stacking blocks task, and robot must place one block to the top of another block one by one.

There are many works to deal with the grasping task with learning-based approaches, and these methods can be divided into two classes: reinforcement learning and imitation learning(similar to supervised learning or learning from demonstrations). For reinforcement learning, most of prior works [6,7] only train a Deep RL policy to grasp an object with 4-D actions (x, y, z, θ), where the last dimension specifies the anger of two gripper fingers. Hence, these policies often fail to work when grasping irregularly shaped objects. In contrast, imitation learning can learn an end-to-end policy to deal with this problem [8,11] if there is enough labeled-data to be provided for training. Although the results are impressive, it's restricted for the widespread using of imitation learning as the large amount of data are needed to be collected. In our work, we extend the learning action to the 5-D space (x, y, z, rz, θ) based on the Deep RL method, where the learned 4th dimension action is related to the posture of the object.

In reinforcement learning, learning a policy through sparse rewards has been popular with researchers for people do not need the special knowledge to design a reward function. However, learning an available policy is a huge challenge with the increase of the horizon under the sparse reward, which makes it difficult to deal with the long-horizon multi-stage task. There some related works [3,6,10] combine demonstrations with RL algorithm to deal with this problem. The work of [3,6] are essentially the same while they forcing the action generated by the policy the same as the action from the demonstration by adding an additional loss function. The method proposed in [10] is more reasonable while it gives

different sampling probabilities through the importance of different transitions, which increases the using-efficiency of transitions.

In this work, we use the improved Deep reinforcement learning algorithm Deep Deterministic Policy Gradient (DDPG), combined with the our proposed method to train an agent to complete the multi-stage task with high success rate and good learning efficience. The input of the policy is only RGB image and auxiliary low-dimensional information (joint angles, gripper state, etc). Figure 1 shows the images we put into the policy and the robot manipulation tasks we are going to complete. Further more, we extend the learning action to the 5-D spaces (x, y, z, rz, θ), which ensures the agent can adjust the angle of the z-axis according to the variety of the shape of different objects and greatly improves generalization ability of the policy. In summary, the main contributions of our work are as follows:

1. In order to make full use of the demonstration data to improve learning efficiency, we artificially make the network fully utilize the state-action pairs in demo until the performance of the actions generated by the network exceeds the actions in demo.
2. Most of the work focus on learning actions limited to 4-D spaces, while we extend the learning actions to 5-D spaces, which is proved to be a significant work for improving the generalization ability of the learned policy.

2 Background

2.1 Reinforcement Learning

We consider the continuous space Markov Decision Process which can be denoted as the tuple $(S, A, P, r, \gamma, S_{-})$, where S is a set of continuous states, A is a set of continuous actions, P is a state transition probability function, r is a reward function, γ is a discount factor and S_{-} is a set of initial continuous states. In reinforcement learning, the goal is to learn a policy $a_t = \pi(x_t)$ to maximize the expected return from the state t_i, where the return can be denoted by $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ and T is the horizon that the agent optimizes.

Various reinforcement learning algorithms have been proposed in order to solve the max expected return problem in RL. Most of the methods involve constructing an estimate of the expected return from a given state after taking an action:

$$Q_\pi(s_t, a_t) = E_{r_i, s_i, a_i}(\sum_{i=t}^{T} \gamma^{i-t} r_i | s_t, a_t) \tag{1}$$

$$= E_{r_t, s_{t+1}, a_i \ E}[r_t + \gamma E_{a_{t+1}}(Q_\pi(s_{t+1}, a_{t+1}))] \tag{2}$$

where the $Q_\pi(s_t, a_t)$ is called the action-value function. Equation 2 is a recursive version of Eq. 1, which is known as the Bellman equation.

In this work, we combine our method with a reinforcement learning algorithm: Deep Deterministic Policy Gradient (DDPG). DDPG is an actor-critic

method which bridges the gap between policy gradient methods and value approximation method for RL, hence it composes of actor network $\pi(s)$ and critic network $Q(s, a|\theta^Q)$. However, experiments have shown if we only use single 'Q-network', it will make the learning process unstable − the parameters of the critic network for calculating the Q function are employed to calculate the gradient of the actor network while performing frequent gradient updates. We can better understand this process with reference to Eqs. (3–5). In order to stabilize learning, DDPG creates two target networks for actor network and critic network, respectively. The update of the critic network is similar to the method of supervised learning. The formula of the loss is:

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))_2 \tag{3}$$

where y_i can be seen as a 'label':

$$y_i = r_i + \gamma Q^{'}(s_{i+1}, \mu^{'}(s_{i+1}|\theta^{\mu^i})|\theta^{Q^{'}}) \tag{4}$$

When updating the actor network by off-policy method, the policy gradient of the strategy is as follows:

$$\nabla_{\theta^\pi} J_\beta(\pi) = E_{s\rho^\beta}[\nabla_a Q(s, a|\theta^Q)|_{a=\pi(s)}.\nabla_{\theta^\pi}\pi(s|\theta^\pi)] \tag{5}$$

Note π is an actor policy $a_t = \pi(s_t|\theta)$, and β is a behaviour policy $a_t = \pi(s_t|\theta^\pi) + N_t$ used to explore potential better policies, where N_t is a Uhlenbeck-Ornstein (UO) stochastic process as a random noise.

2.2 DDPG from Domenstrations

In our method, we use sparse reward as a feedback to an agent, in which the agent would get +1 reward when success otherwise getting 0. However, combining sparse reward with raw DDPG algorithm to deal with the multi-stage task is impossible as sparse reward can not give enough information to update parameters in right way. This problem can be partly overcome with the demonstrations as the agent can learn a suitable policy for completing the task from the demonstrations. Moreover, prioritized experience replay [10] modifies the agent to sample more important transitions from its replay buffer more frequently. The probability that each particular transition in the replay buffer is extracted as $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$, where $p(i)$ represents the priority of each transition. In the experiment, $p_i = \delta_i^2 + \lambda|\nabla_a Q(s_i, a_i|\theta^Q)|^2 + \varepsilon$, where δ_i is the last TD error calculated for this transition, the second term is the loss applied to the actor, ε is a positive constant to ensure that all transitions have a certain probability of being sampled, and λ is used to weight the contributions. DDPGfD [10] merged the above two strategies and made further improvements. It collected the demonstration before pre-training, as well as initialized the replay buffer with the demo, and pre-trained the agent with the demo transitions. Besides, they set $p_i = \delta_i^2 + \lambda|\nabla_a Q(s_i, a_i|\theta^Q)|^2 + \varepsilon + \varepsilon_D$, where ε_D is a positive constant for demo transitions to increase their probability of being sampled.

3 Methods

3.1 Behaviour Clone Loss and Reward

[6] introduced a method of adding loss function to the network, which made full use of the positive effects of the demonstration on policy training. This loss is applied only when a demonstration is sampled from the replay buffer for training. When the value function $Q(a_d)$ generated by the action a_d in the demonstration is bigger than $Q(a_t)$, it means that a_d is closer to the optimal action than a_t. At this time, they would give the policy a penalty item L_{BC}, and encouraged the policy to propose the same action as the demonstration in the given state.

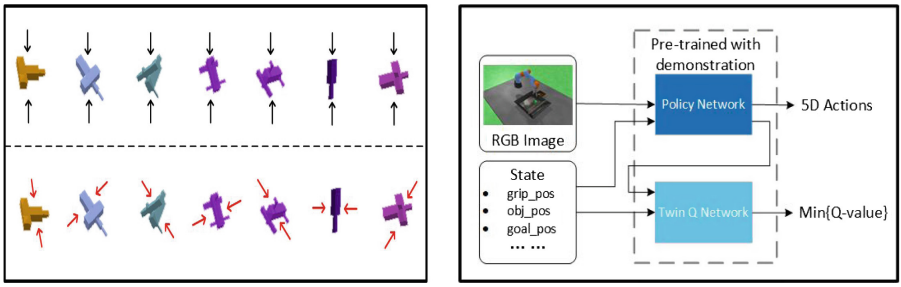
$$L_{BC} = \begin{cases} |\pi(o_d) - a_d|, & if \quad Q(s_d, a_d) > Q(s_d, \pi(o_d)) \\ 0, & otherwise \end{cases} \tag{6}$$

However, this way of adding loss to the network seems still does not fully utilize the information in the demonstration. When each transition is collected from the demonstration for the reason of initializing the replay buffer and pre-training the policy for multi-stage task, the previous work always sets the reward for each state transition to 0, while only the last state transition step is set to 1, as shown in formula 7. Note that T is the total step size for an episode. However, if we set the reward of the corresponding action in most transitions to 0, the agent may consider this action to be undesirable and reject this action in training (since reward is 0, the corresponding Q_target value is smaller) even though this action may benefit completing the task. Therefore, we set the reward of all the transitions to 1 for an episode if the task is completed successfully when collecting the transition. In the early stage of training, this method ensures the policy generate the action the same as the demonstrator in the given state. In the later stage of training, we use the Q_filter method proposed in [6] to ensure that the agent can use the Q-value to determine whether to use the action in demonstration or not. It should be noted that this method is compatible with the behavior clone Loss.

$$r_t = \begin{cases} 1, & if \quad t = T \\ 0, & otherwise \end{cases} \tag{7}$$

3.2 Extend the Learning Action to 5-D Spaces

For most of the Deep RL based policy only learning a 4-D action (x, y, z, θ) to deal with the grasping task, it's often fail to work if we are going to grasp irregularly shaped objects as be shown in Fig. 2(a). Hence, in our work we extend the learning action to the 5-D space (x, y, z, rz, θ), where the learned 4th dimension action is related to the posture of the object. Figure 2(b) shows the framework of our method. The input of the actor network includes RGB image and full-low dimensional state, while the twin critic networks only receive the full-low dimensional state as input. Besides, we add the detection loss to actor loss to help the policy recognize the essential scene features quickly. The output of the actor network is a 5-D actions.



(a) Predict the possible grasping points for a given observation

(b) Framework of the proposed Method

Fig. 2. We extend the learning action to 5-D spaces. (a) If we only learn 4-D actions, the predicting points is not always leading to successfully grasping (top), while the learned 5-D actions from our method could predict the better grasping points (bottom). (b) The framework of the proposed method.

4 Experimental Results

4.1 Environmental Setup

We evaluate our algorithm on several simulated Pybullet [2] environments. In simulation environment we use the Kuka robot and a robotic grippers which proposed by Pybullet. In the collection of demonstration's data, the observation is rendered by a virtual camera and the states of environment, such as robot arm state, gripper state, goal position can be obtained by corresponding api function. In our experiments, we will perform the experiments in a 5-D space as the extra dimension representing the angle of rotation around the z-axis.

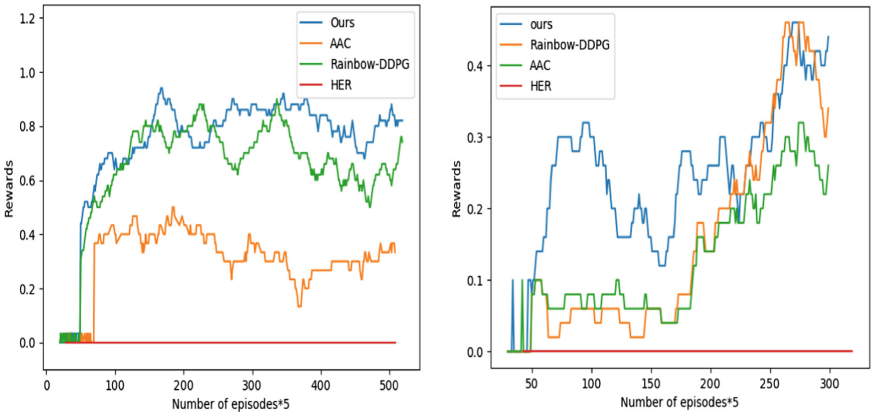
4.2 Evaluation

Comparison with Prior Work. In this section, we will demonstrate the superiority of our algorithm through comparing with several competitive methods by executing the picking-and-place and stocking blocks task. [1] proposed a 'HER' algorithm of storing experienced transitions with different goals in the replay buffer used in off-policy RL algorithms that allows to learn the policy more efficiently with sparse rewards. [5] proposed a method which incorporates several improvements to DDPG and can accomplish most of the simple tasks with excellent results, and we call this method 'Rainbow-DDPG'. [7] proposed an asymmetric actor critic algorithm for visual-based task in which the critic is trained on full states while the actor is trained on images and we call this algorithm 'AAC'.

In our experiments, we all use a fully sparse reward to train the policy, where the policy get a $+1$ reward if the object is at its goal position after rollouts ending.

$$r_t = \begin{cases} 1, & if \quad ||x_i - g_i|| < \delta \\ 0, & otherwise \end{cases} \tag{8}$$

where the threshold δ is 3 cm.



(a) Pick-and-place (b) Stock blocks

Fig. 3. We compare our method with some existing competitive methods and as can be seen our method outperforms these methods. (a) Evaluation experiments on picking-and-placing task. (b) Evaluation experiments on stacking blocks task.

Table 1. Comparison of our method with baselines

Task	Ours	Rainbow-DDPG	AAC	HER
Pick-and-place	95%	87%	50%	0%
Stack blocks	86%	72%	38%	0%

Table 1 reports the results of our evaluation experiments. The effect of our method and 'Rainbow-DDPG' are more outstanding compared with the 'AAC' and 'HER'. The main factor is that we utilize demonstration and take some strategies to make the action generated by the policy close to the demo, which greatly improves the success rate of learning. Figure 3 shows the variation of the reward value during training. We run the each experiment 5 times in each episode, and the corresponding reward value of each time step takes the average of the past 10 episodes experimental results, so the curves report the mean of past 50 experimental results. We adapt most of the improvements proposed in 'Rainbow-DDPG', as well as our own methods, and achieve competitive results.

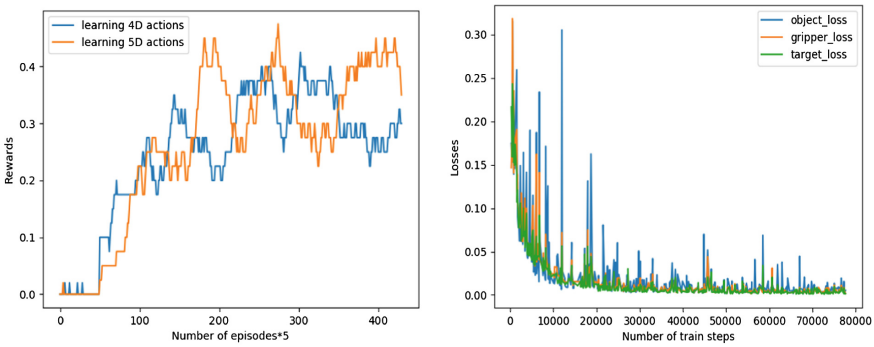
As can be seen from Fig. 3, HER is unable to complete this task. While Marcin Andrychowicz et al. explained in their paper that they could use HER to complete the pick-and-place task, which seems to be contrary to our experimental results. However, in their experiments, they assumed that the agent have got the knowledge of the first half rollouts action (the box is grasped) before training and the agent only needs to learn the action of rollouts in the latter half, which greatly reduces the difficulty of learning the task. In our experiments, if we don't give any extra information to the agent about the task, it's almost impossible to complete the pick-and-place task through HER.

Extend the Learning Action to 5-D Spaces. In this study, we only use the improved reinforcement learning algorithm proposed in this paper to learn a policy to grasp the irregularly shaped objects. We only compare the results of grasping objects through the learned 4-D and 5-D actions by method proposed in this paper due to the lack of existing methods for learning 5-D actions through reinforcement learning as baseline. Table 2 shows that for objects with irregular shapes, the learned 5-D actions can greatly improve the success rate of grasping. Although our results are not outstanding enough for they do not reach the level of supervised learning, the advantage of reinforcement learning does not require a large number of annotation data drives us to explore the potential of reinforcement learning algorithms to accomplish more difficult grasping tasks.

Table 2. The success rate comparison for learning different actions

Task	Learning 4-D action	Learning 5-D action
Pick-and-place for irregularly shaped objects	54 %	72 %

In order to test the ability of the agent to analyze the scene for a given image, we force the agent to predict the state of the key elements in the scene at each moment, such as object position, gripper position. In Fig. 4(b), we show the loss curves of the object-position, the robot gripper-position and the target-position, respectively. It can be seen that as the training going, the agent can analyze the location of each element accurately.



(a) (b)

Fig. 4. (a) Evaluation experiments on grasping the irregularly shaped objects while learning different actions. (b) The loss curve of the object-location, the robot gripper-location and the target-location. It can be seen that as the training going, the agent can analyze the location of every element in the scene accurately.

5 Conclusion and Future Work

In this paper, we improve the policy based on DDPG by incorporating several improvement methods and achieve impressive results on the multi-stage task. In the experiment, compared with some baselines, the performance of the policy proposed is of the great improved. At the same time, we extend the learning action to 5-D space, which greatly improves the generalization of the policy.

In future work, we will deploy our algorithm to real word robots. For the most domain adaption technologies are achieved by enriching the diversity of elements in the scene, hence the success of the method in the real environment is depend on the diversity of environment during training. The limitations of this approach have been reflected in many of the previous works. Therefore our further research will focus on exploring a domain adaption technique that allows the algorithm to maintain comparable performance in the real world as it does in the simulation environment.

Acknowledgements. This work was supported in part by NSFC under Grant No.91848109, supported by Beijing Natural Science Foundation under Grant No.4182068 and supported by Science and Technology on Space Intelligent Control Laboratory under No. HTKJ2019KL502013.

References

1. Andrychowicz, M., et al.: Hindsight experience replay. In: Advances in Neural Information Processing Systems, pp. 5048–5058 (2017)
2. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. GitHub repository (2016)
3. Hester, T., et al.: Learning from demonstrations for real world reinforcement learning. arXiv preprint arXiv:1704.03732 (2017)
4. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. **17**(1), 1334–1373 (2016)
5. Matas, J., James, S., Davison, A.J.: Sim-to-real reinforcement learning for deformable object manipulation. arXiv preprint arXiv:1806.07851 (2018)
6. Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Overcoming exploration in reinforcement learning with demonstrations. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6292–6299. IEEE (2018)
7. Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., Abbeel, P.: Asymmetric actor critic for image-based robot learning. arXiv preprint arXiv:1710.06542 (2017)
8. Pinto, L., Gupta, A.: Supersizing self-supervision: learning to grasp from 50k tries and 700 robot hours. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 3406–3413. IEEE (2016)
9. Popov, I., et al.: Data-efficient deep reinforcement learning for dexterous manipulation. arXiv preprint arXiv:1704.03073 (2017)
10. Večerík, M., et al.: Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv preprint arXiv:1707.08817 (2017)
11. Viereck, U., Pas, A.t., Saenko, K., Platt, R.: Learning a visuomotor controller for real world robotic grasping using simulated depth images. arXiv preprint arXiv:1706.04652 (2017)