Robotic Arm Control Using Deep Reinforcement Learning with Gaussian Extrinsic Rewards

Wentai Zhang, Bolun Dai, Zhangsihao Yang
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{wentaiz, bdai, zhangsiy}@andrew.cmu.edu

1 Introduction

In this project, we aim to produce a control policy for a Kuka robotic arm in achieving grasping randomly placed objects with various shapes. For the baseline, we implemented a Double Deep Q-network (Double DQN) [1] in discrete action space. Subsequently, we implemented a Double Deep Q-network with prioritized experience replay (Double DQN + PER) which surpasses the performance of the baseline in discrete action space. We then moved our focus to continuous action space, where we explored several algorithms including Deep Deterministic Policy Gradients (DDPG), Deep Deterministic Policy Gradients with Prioritized Experience Replay (DDPG + PER) and Deep Deterministic Policy Gradients with Hindsight Experience Replay (DDPG + HER). The performance of these aforementioned methods all exceeds the baseline.

Finally, we introduce a new method using extrinsic rewards generated by Gaussian state similarity. Combined with DDPG + PER, our proposed method outperforms all the above mentioned methods we implemented, which achieves an average reward/episode of over 0.9 after training for 50,000 episodes.

2 Background

In this section, the implementation details for the baseline are given along with the results. The training curves of our implementation of Double DQN are shown in Fig.1(c) and Fig.1(d). We can see that the training process can be roughly divided into two phases (note that the average reward for each unit can be acquired from the slope of the training curve). For the first 25k training episodes, the average reward is roughly between 0.03 and 0.06. During this phase, we observe that the agent attempts to learn how to approach the target object. In the subsequent phase, a sharp change in the slope occurs, and the reward fluctuates around 0.45. For this phase, we observed the agent is learning how to rotate the gripper. As we see from our baseline model the robotic arm manages to learn how to approach the target object. However, it fails to rotate the gripper into a proper angle in adapting for different objects and the training progress stagnates. This can be explained as for approaching the target object we have explicit values in the feature vector to guide the agent, but for the gripper rotation there is no such indicator.

From Fig.1(a) and Fig.1(b), we can observe that the variance of average rewards is low for the first phase and relatively high for the last 10000 training episodes. One interpretation for this phenomenon is the optimal grasping angle varies hugely among different objects, behaviors the agent learned on one object may not be transferable to another.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

3 Related work

Deep Deterministic Policy Gradients (DDPG) [2] is a model-free reinforcement learning algorithm for continuous action space. In DDPG we have two networks an actor network which produces the control policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, and a critic network which gives the estimation of the current actor's action value $Q^{\pi} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The episodes are generated using a noisy version of the control policy generated by the actor $\pi_b(s) = \pi(s) + \mathcal{N}(0, 1)$. We train the critic similar as we train a DQN, the actor is trained using a replay buffer, and the gradients are computed by backpropagation through the entire actor-critic network.

Prioritized Experience Replay(PER) [3] is a framework for prioritizing experience, so as to replay important transitions more frequently, and therefore learn more efficiently. In particular, transitions with high expected learning progress are more frequently replayed, as measured by the magnitude of their temporal-difference (TD) error. This prioritization can lead to a loss of diversity, which then get alleviated with stochastic prioritization,

Hindsight Experience Replay (HER) [4] is a technique that is used to solve the binary and sparse reward situation. In HER we record all of the trajectories and for each trajectory, we generate fake goals using the states that trajectory actually visits and assign rewards to these states. After such goal generation, we save the state-reward tuples into the replay buffer and use any off-policy RL algorithm to replay using these memory transitions.

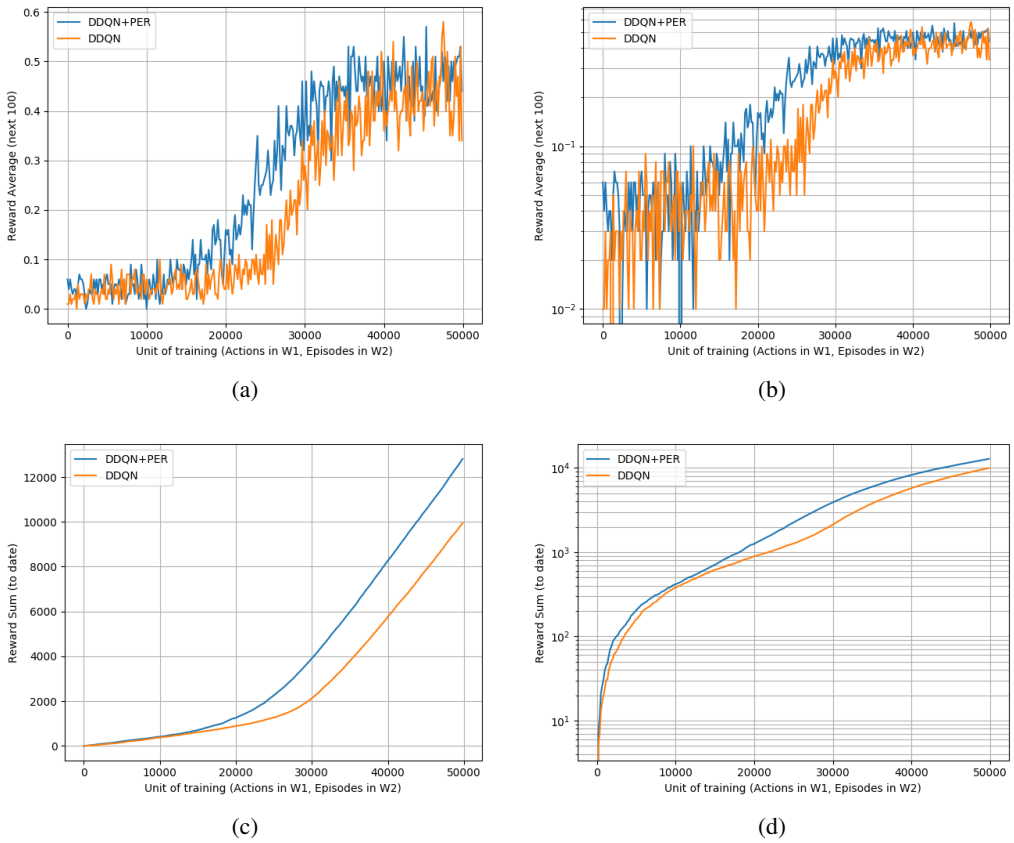


Figure 1: Training curves for DDQN(baseline) and DDQN+PER. (a) average reward during training in linear space; (b) average reward during training in log space; (c) cumulative reward during training in linear space; and, (d) cumulative reward during training in log space.

4 Methods

4.1 DDPG implementation

Our implementation of DDPG is similar to what is described in [2]. For both the critic and the actor we used a two layer neural network with the size of the layers being $[400, 300]$. We optimize the critic network $Q(s, a|\theta^Q)$ using

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

where y_i is the estimated discounted future reward. And we optimize the actor network $\mu(s|\theta^\mu)$ using

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

after each update of the evaluation network of both the critic and the actor we perform a Polyak averaging to the target network. To obtain the result shown in Fig.3 we trained the network for 50000 episodes using an ADAM optimizer.

4.2 PER implementation

During deployment, for each new transition recorded in the replay buffer, a corresponding priority value $pr \in [0, 1]$ is also added to a priority list. The priority value is equal to the reward of the transition if it is for burn-in memory. In this way, successful transitions, where the reward is 1.0, have higher probabilities to be sampled. During training, the priority value for new transitions is 1.0 to secure that the model is always curious for the latest transitions.

When generating batches from the replay buffer, we calculate a distribution(p) for sampling based on all the priority values:

$$p_i = \frac{pr_i^\alpha}{\sum_i pr_i^\alpha} \in p$$

where $\alpha = 0.4$ is a factor to control the relative difference. After back-propagation with a sampled batch, the priority values for all the transitions in this batch need to be updated with their TD-error:

$$pr = \min(\text{Abs}(\text{TD\_error}) + \epsilon, \text{error\_bond})$$

Here, $\epsilon = 0.01$ is a small value to avoid zero priority, which means the sample will never be sampled. $error\_bond = 1.0$ is utilized for truncation to alleviate the dominant role of transitions with huge TD errors.

4.3 HER implementation

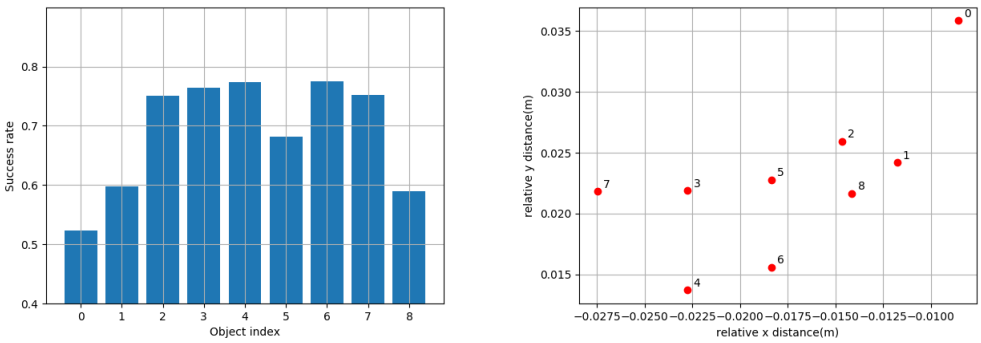
For HER we tried a similar approach described in [4] with a small modification. For episodes that got a non-zero reward we skip the generation of fake goals and fake rewards, we only use them when we obtained a zero reward for the episode. Also as for selecting states to act as the fake goal we used a strategy called final, where only use the terminal state as the fake goal, such choice is an adaptation for the environment.

4.4 Success burn-in

We use the already obtained agent trained using DDPG which has an approximate 70% success rate to burn 10000 tuples of state, reward, next state, terminal flag, action. We store these tuples into the replay buffer and use this buffer to train a random initialized agent.

4.5 Gaussian Extrinsic Reward(GER)

Based on the experiments using DDPG, we found out that the success rate is rapidly converged to around 65% only after 15000 episode of training. After rendering some test episodes using this model, we observe that the agent sometimes fails to approach the object when the grip is already



(a) (b)

Figure 2: (a)Success rate for each object. (b) mean relative x, y distance at the last state among successful transitions.

touching the gound(height-hack is on). In other failure cases, the agent touches the object but loses the object afterwards when grasping or lifting. To explore the main cause behind these cases, we calculate the success rate separately for each object and demonstrate the bar chart results in Fig.2(a). In the meantime, the mean relative x, y distance at the last state of grasping is illustrated individually for each object in Fig.2(b). From these two figures, we notice that the relative x, y distance at the last state is quite different among all the objects. Interestingly, object 0,1 and 8, which our agent has most difficulty in grasping, all locate in the right most in Fig.2(b).

To accommodate this variance among different objects, we introduce a new extrinsic reward using Gaussian estimation. Basically, for each object, we fit a 2D Gaussian distribution with relative x,y distance using successful transitions obtained from DDPG models. To unify the rewards, we remove the scaling in the original Gaussian to guarantee $R_g \in [0, 1]$:

$$R_g^i = \exp(-\frac{1}{2(1-\rho^{i2})}[\frac{(x-\mu_x^i)^2}{\sigma_x^{i2}} + \frac{(y-\mu_y^i)^2}{\sigma_y^{i2}} - 2\rho^i \frac{(x-\mu_x^i)(y-\mu_y^i)}{\sigma_x^i \sigma_y^i}])$$

where R_g^i is the gaussian extrinsic reward for object i; μ_x^i, μ_y^i is the mean x, y distance; σ_x^i, σ_y^i is the standard deviation of relative x, y distance; ρ^i is the correlation between relative x, y distance. Then we combine this extrinsic reward with the original reward: $R_{new} = R + R_g \in [0, 2]$, which makes the reward space much less sparse. Accordingly, we also modify the error bond in PER from 1.0 to 2.0.

5 Results

In discrete action space, PER is added to the baseline method(Double DQN). The resulting training curves are shown in Fig.1. As we expect, PER speeds up the training process and achieves higher cumulative reward than DDQN(12500 vs 10000) in 50000 training episodes. However, the average reward at convergence still retains about 0.47. We deem that discrete action space restrict the agent to learn the complex skills to grasp different object since the number of actions taken is fixed(height hack is on). Some repetitive actions may be used in order to just approach the object and result in few actions left for grasping. So we switch to continuous action space afterwards.

To successfully train the Kuka robotic arm in grasping random objects in continuous action space (we kept height-hack open) we experimented with several off-policy algorithms. The final result is shown in Fig.3, where we make a direct comparison between our DDPG, DDPG+PER and DDPG+PER+GER implementation, our HER implementation is shown in Fig.4. We can see that among all our attempts DDPG+PER+GER performs the best with over 90% success rate during training(actually the best model achieves 95% during testing). The reason we plotted HER separately is that HER requires a significant more actions and optimization steps. A detailed analysis for each of our implementations can be seen in the following section.

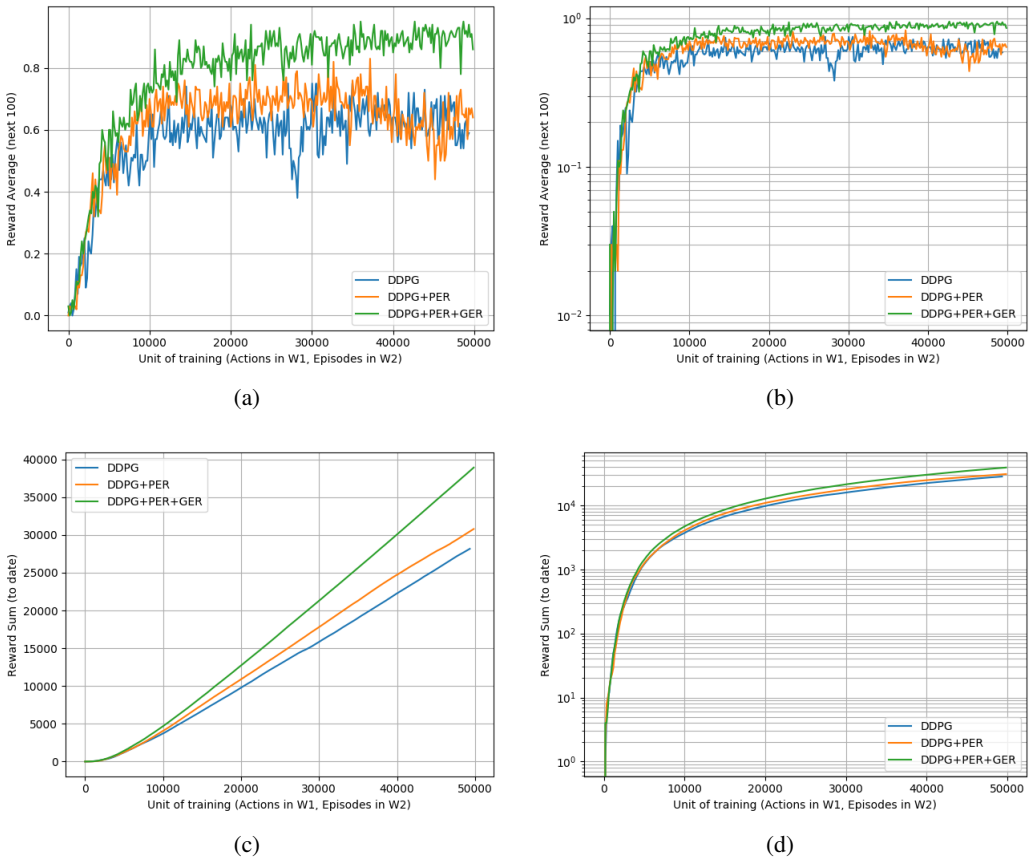


Figure 3: Training curves for DDPG, DDPG+PER and DDPG+PER+GER. (a) average reward during training in linear space; (b) average reward during training in log space; (c) cumulative reward during training in linear space; and, (d) cumulative reward during training in log space.

6 Discussion and Analysis

We can see from the result for the first 10000 training episodes, the performance for DDPG, DDPG+PER and DDPG+PER+GER are similar, then the performance for DDPG and DDPG+PER stagnates while DDPG+PER+GER still improves, though at a lower rate. At the termination, either judging from average reward or cumulative reward, the conclusion is: DDPG+PER+GER > DDPG+PER > DDPG. And from 3(a) we can see that our model still doesn't converge even at 50000 episodes when we terminate the training. This demonstrates that the way we set our Gaussian extrinsic reward indeed aids the training. However, using GER heavily relies on prior knowledge of existing successful states, without such information we wouldn't be able to fit such a Gaussian distribution. Therefore, one assumption we have is having access to either a set of transition tuples of rewarded trajectories or a trained (doesn't need to be fully trained) agent.

Also we can see that the improvement of HER is not as impressive as what is demonstrated in [4], this may result from that the reward is not as sparse as what the paper mentioned (for their environment DDPG receives close to 0 reward), therefore even a vanilla implementation of DDPG can manage to get access to trajectories that results in a successful grasp. Given the height-hack enabled, we actually reduced the problem from a 3D case to a 2D case which greatly shrinks the state space and give random exploration a higher probability to find a local optimum.

7 Further discussion

To illustrate some insights of our method, we plot the intrinsic and extrinsic reward separately in Fig.5. Unlike intrinsic reward which is received only at the last state, the extrinsic reward is given for each state and only depends on the x, y distance relative to the goal at that moment. This enriches the reward space so that the reward is in (0, 1] for non-terminal states and in (0, 2] for terminal states. This increasing of average extrinsic reward means that the agent tries to learn to reach the top of the mean spot, represented by our Gaussian distribution in GER, as fast as possible. This will definitely leave as much rest steps for learning to grasp.

In our implementation, we directly add our extrinsic reward to the original reward to create a new reward. Actually, without any domain knowledge, we cannot guarantee that maximizing this new reward is the same as maximizing the original reward. But [5] proves that if using a form of reward shaping like $R_{new} = R(s) + \gamma R_g(s) - R_g(s')$, the original reward function is optimized at the same time as the shaped reward function is optimized. This will be our next immediate exploration.

As mentioned in the previous section, our approach strongly relies on having a semi-trained agent. To eliminate this dependency, we propose a new approach to form the Gaussian extrinsic reward. The differences are:

- After initially burning-in memory, select all the transitions with non-zero reward to fit a 2D Gaussian distribution for GER and use it to update all the reward in the replay buffer.
- During training, after every N episodes, select all the transitions with non-zero reward in the replay buffer and fit the Gaussian distribution again.

In this manner, our model is able to be trained from scratch. The only assumption is that the burn-in should not stop until there is at least one transition with non-zero reward for each object. Compared to our current approach, this is a much weaker assumption.

Finally, removing height hack is another potential direction we can delve into. In this case, our GER can be achieved with a 3D Gaussian distribution. This reward can guide the agent to rapidly learn how to move downwards, which is usually challenging for models only trained with DQN or DDPG.

References

[1] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

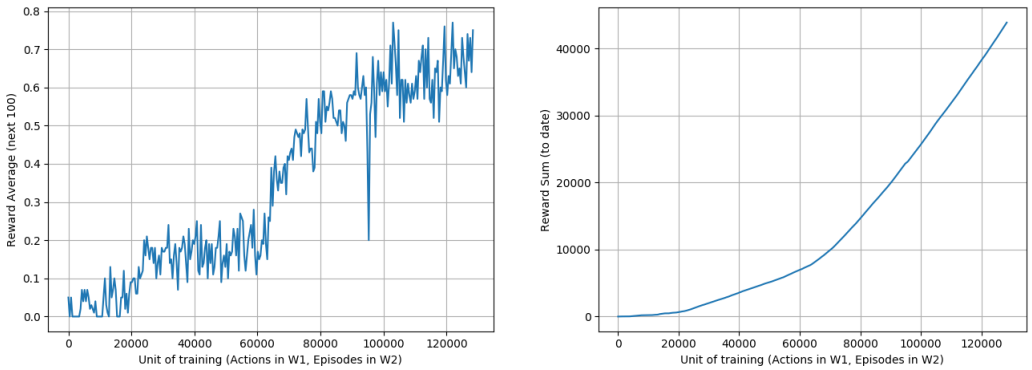
[2] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[3] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

[4] 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.

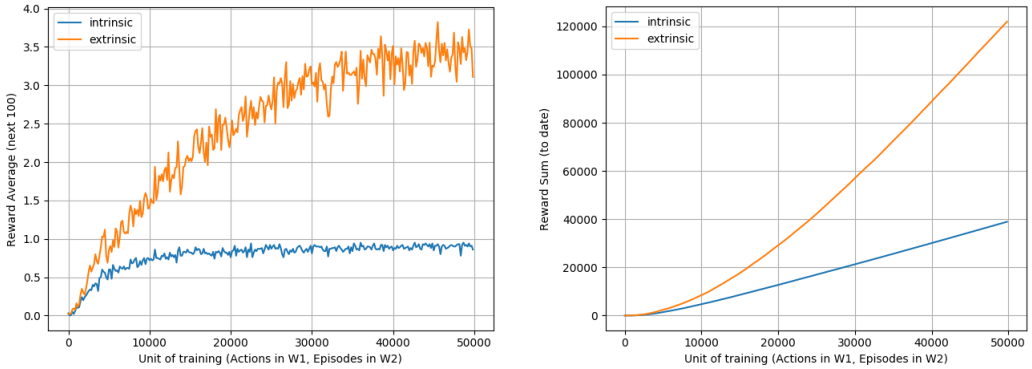
[5] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287, 1999.

A Supplementary Figures



(a) (b)

Figure 4: Training curves for HER (a) average reward during training in linear space; (b) cumulative reward during training in linear space;



(a) (b)

Figure 5: Training curves for GER(intrinsic reward and extrinsic reward) (a) average reward during training in linear space; (b) cumulative reward during training in linear space;