Deep RL Arm Manipulation

Quah Yu Soon

Abstract—This documentation discusses about the application of Deep Reinforcement Learning on a robotic arm to learn on itself how to achieve specific goal without human intervention. An experiment that implemented Deep Q-Network (DQN) agent in teaching a robotic arm to complete two different objectives would be carried out to evaluate how well DQN agent has performed its role. Tuned hyperparameters of DQN and reward functions would also be discussed.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Deep RL.

1 Introduction

Reinforcement learning suggests that robot can be assigned a goal and it could learn to achieve the goal by trying various available actions, observing changes in environment through raw sensor input and finding the best way to achieve through trial and error. An Reinforcement Learning agent receives observations or states of the system and rewards from the environment and uses some algorithms for instance, Q-Learning to determine the best action to take such that cumulative reward is maximized. This traditional RL approach works well only when the environment could feedback exact readings of any observations for example, joint velocity or position of each robot arm's joint.

Deep Reinforcement Learning has an advantage over the traditional one. It determines the state based on the difference between the current and previous image taken from camera sensor. The word "deep" implies that it uses a neural network called Deep Q-Network (DQN) that takes a state and approximates Q-values for each action based on that state.

In the experiment, a DQN agent and reward functions would be defined to teach a robotic arm to carry out two primary objectives in Gazebo environment (See Figure 1):

1) Have any part of the robot arm touch the object of interest, with at least a 90% accuracy
2) Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy

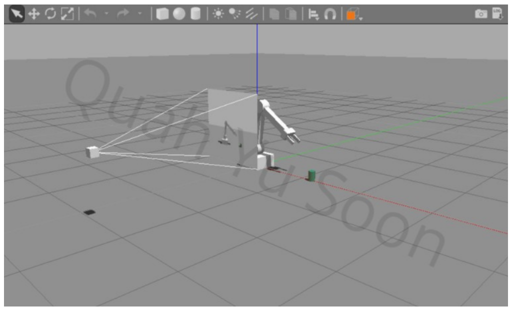


Fig. 1. Robotic Arm in Gazebo

2 Reward Functions

Different amount of rewards would be given according to situations (See Table 1). Whenever objective is achieved, REWARD_WIN +100 would be given. Each episode is allowed to run within 100 maximum frames, once exceeded the limit, episode has to be ended and a penalty of -50 would be given. REWARD_LOSS -100 is given when robot arm hitting the ground. Higher value of penalty for hitting the ground is to encourage the robot arm not to do so as the arm might spoil when collides with the ground. Interim rewards which could be in term of penalty or reward are given based on the distance to the object, penalty to be issued when robot arm is moving farther from the object and reward to be issued when it is moving closer.

Position-based control has been used to control arm joint movements. If the action is an even, joint position would be increased by a constant actionJointDelta of 0.15, whereas an odd would decrease joint position.

TABLE 1
Rewards for Different Situations

Situation	Reward Amount
Collision between the arm and object of interest (For Objective 1)	+100 (REWARD_WIN)
Collision between gripper base and object of interest (For Objective 2)	+100 (REWARD_WIN)
Episode has exceeded max frames	-50
Robot arm hitting the ground	-100 (REWARD_LOSS)
Robot arm getting closer to or farther from object of interest	distance to the goal at previous time step - distance to the goal at current time step

3 Hyperparameters

Hyperparameters have been tuned through trial and error method and final choices are listed in Table 2. These hyperparameters worked for both Objective 1 and 2.

4 Results

At very beginning, robotic arm was taking random actions to explore state-action space as more as possible, meanwhile, the reward functions was helping robotic arm to

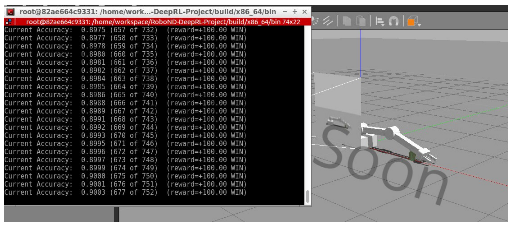
TABLE 2
Tuned Hyperparameters

Hyperparameter	Selected Value	Reason
INPUT_WIDTH & INPUT_HEIGHT	64	Camera images having 64 x 64 dimension.
OPTIMIZER	RMSprop	Popular choice.
LEARNING_RATE	0.2	Tried with rate 0.1, but rate of 0.2 yielded higher accuracy.
BATCH_SIZE	64	A neural network requires large data set to learn. Felt batch size of 64 would be just nice.
USE_LSTM	true	Network could be trained by taking into consideration multiple past frames from the camera sensor instead of a single frame.
LSTM_SIZE	512	Tried with 256, but 512 seemed to work better with batch size 64.

understand that it should move toward the object and ultimately touch it. Within 1st episode, robotic arm could already determine the direction it should move in. Within first 100 episodes of trials and errors, there were a few times robotic arm managed to collide with the object, but accuracy was still low. At certain point of time, the chances of robotic arm meeting its objective became stable and hence accuracy started to increase gradually. Different amount of episodes robotic arm has taken in achieving each objectives.

4.1 Objective 1

For objective of having any part of the robot arm touch the object of interest with at least a 90% accuracy, robotic arm has taken 750 episodes as can be seen from Figure 2.



```
root@82ae664c9331: /home/work...-DeepRL-Project/build/x86_64/bin
root@82ae664c9331: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 74x22
Current Accuracy:  0.8975 (657 of 732)  (reward=+100.00 WIN)
Current Accuracy:  0.8977 (658 of 733)  (reward=+100.00 WIN)
Current Accuracy:  0.8978 (659 of 734)  (reward=+100.00 WIN)
Current Accuracy:  0.8980 (660 of 735)  (reward=+100.00 WIN)
Current Accuracy:  0.8981 (661 of 736)  (reward=+100.00 WIN)
Current Accuracy:  0.8982 (662 of 737)  (reward=+100.00 WIN)
Current Accuracy:  0.8984 (663 of 738)  (reward=+100.00 WIN)
Current Accuracy:  0.8985 (664 of 739)  (reward=+100.00 WIN)
Current Accuracy:  0.8986 (665 of 740)  (reward=+100.00 WIN)
Current Accuracy:  0.8988 (666 of 741)  (reward=+100.00 WIN)
Current Accuracy:  0.8989 (667 of 742)  (reward=+100.00 WIN)
Current Accuracy:  0.8991 (668 of 743)  (reward=+100.00 WIN)
Current Accuracy:  0.8992 (669 of 744)  (reward=+100.00 WIN)
Current Accuracy:  0.8993 (670 of 745)  (reward=+100.00 WIN)
Current Accuracy:  0.8995 (671 of 746)  (reward=+100.00 WIN)
Current Accuracy:  0.8996 (672 of 747)  (reward=+100.00 WIN)
Current Accuracy:  0.8997 (673 of 748)  (reward=+100.00 WIN)
Current Accuracy:  0.8999 (674 of 749)  (reward=+100.00 WIN)
Current Accuracy:  0.9000 (675 of 750)  (reward=+100.00 WIN)
Current Accuracy:  0.9001 (676 of 751)  (reward=+100.00 WIN)
Current Accuracy:  0.9003 (677 of 752)  (reward=+100.00 WIN)
```

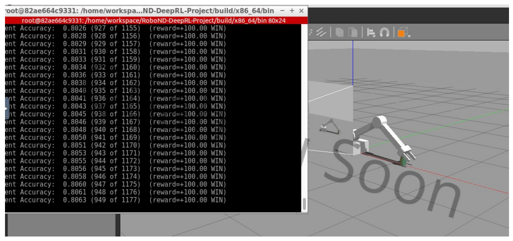
Fig. 2. Objective 1 achieved at 750th episode

4.2 Objective 2

For objective of having only the gripper base of the robot arm touch the object with at least a 80% accuracy, robotic arm has taken around 1155 episodes as can be seen from Figure 3.

5 Discussion

From the aspect of number of episodes robotic arm has taken, Objective 2 seemed to be more difficult for a DQN agent to learn compared to Objective 1. Such result was actually expected. As Objective 2 requiring only the gripper base of the robot arm touch the object, the chances of success attempt would obviously be fewer than the chances



```
root@82ae664c9331: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@82ae664c9331: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
ent Accuracy:  0.8026 (927 of 1155)  (reward=+100.00 WIN)
ent Accuracy:  0.8028 (928 of 1156)  (reward=+100.00 WIN)
ent Accuracy:  0.8029 (929 of 1157)  (reward=+100.00 WIN)
ent Accuracy:  0.8031 (930 of 1158)  (reward=+100.00 WIN)
ent Accuracy:  0.8033 (931 of 1159)  (reward=+100.00 WIN)
ent Accuracy:  0.8034 (932 of 1160)  (reward=+100.00 WIN)
ent Accuracy:  0.8036 (933 of 1161)  (reward=+100.00 WIN)
ent Accuracy:  0.8038 (934 of 1162)  (reward=+100.00 WIN)
ent Accuracy:  0.8040 (935 of 1163)  (reward=+100.00 WIN)
ent Accuracy:  0.8041 (936 of 1164)  (reward=+100.00 WIN)
nt Accuracy:  0.8043 (937 of 1165)  (reward=+100.00 WIN)
nt Accuracy:  0.8045 (938 of 1166)  (reward=+100.00 WIN)
ent Accuracy:  0.8046 (939 of 1167)  (reward=+100.00 WIN)
ent Accuracy:  0.8048 (940 of 1168)  (reward=+100.00 WIN)
ent Accuracy:  0.8050 (941 of 1169)  (reward=+100.00 WIN)
ent Accuracy:  0.8051 (942 of 1170)  (reward=+100.00 WIN)
ent Accuracy:  0.8053 (943 of 1171)  (reward=+100.00 WIN)
ent Accuracy:  0.8055 (944 of 1172)  (reward=+100.00 WIN)
ent Accuracy:  0.8056 (945 of 1173)  (reward=+100.00 WIN)
ent Accuracy:  0.8058 (946 of 1174)  (reward=+100.00 WIN)
ent Accuracy:  0.8060 (947 of 1175)  (reward=+100.00 WIN)
ent Accuracy:  0.8061 (948 of 1176)  (reward=+100.00 WIN)
ent Accuracy:  0.8063 (949 of 1177)  (reward=+100.00 WIN)
```

Fig. 3. Objective 2 achieved around 1155th episode

of touching the object with any part of robotic arm. In addition, DQN is harder to train when it is not provided with sufficient success examples.

6 Conclusion / Future work

Through Deep RL approach, robotic arm has taken 750 episodes and around 1155 episodes to achieve Objective 1 and 2 respectively. However, there should be room for improvement on number of episodes taken. For future work, designing a better reward function could be helpful in reducing number of episodes. Based on experience from this experiment, reward function has significant impact in finding true optimal policy.