Deep RL Arm Manipulation

Jason, Yu-Chieh Huang

Abstract—A deep Q-learning network, DQN, receives sensor data and rewards, and it makes actions to maximize cumulative rewards. In this project, a robot arm in Gazebo simulation environment is given. Rewards are modified for different scenario, and a DQN agent is trained. Good results for two cases are reached.

Index Terms—Robot, Deep Reinforcement Learning, DQN, ROS, Gazebo.

1 Introduction

Reinforcement, RL, adopts a similar concept of how animals learn to response to the world through interaction. After grasping ideas of cause and effect, animals can use knowledge to achieve a specific goal. Nowadays, the reinforcement learning is an active field in which main applications include robotics, stock predictions, self-driving car, etc.

There are two roles in the reinforcement learning: an agent and an environment. The agent makes decisions base on states of the environment, and executes actions. If actions are beneficial to the goal, the environment will return positive rewards to encourage the agent. On the contrary, if actions cause negative effects to the goal, the environment will return negative punishments to discourage the agent. As time goes by, the agent will learn, hopefully, how to make actions to approach the goal and also maximize cumulative rewards. The mechanism of making decisions is called policy, and it is to map sets of states to sets of actions. As a result, the reinforcement learning is to optimize the policy which leads to maximize cumulative rewards.

2 Background

In a simple environment, there are few different states and different actions, and the policy can be optimized base on the Q-learning. It is a model-free algorithm, and a state-action table is updated iteratively by updating parameters with observations and rewards.

However, in a more complex environment, there are many different states and actions, and it is difficult to describe images into few states. Deep RL, deep reinforcement learning, integrates deep neural network to solve reinforcement problems under complex environments. Deep neural network is a general function approximator, and we can leverage it's ability to take raw sensor data and directly choose an action for an end-to-end solution.

When collecting interactions among the environment and the agent, it is resource consuming. A technique, called experience replay, stores observations and interactions, like states, actions, rewards, etc in a buffer, and the deep neural network is trained from sampled data in the buffer. This makes data usage efficiently, and de-correlates sequence between each time steps. As a result, the deep neural network is generalized.

3 Simulations

In the Gazebo simulation environment, there are a robot arm and a tube. The goal is to build a DQN agent to control the arm to touch the tube with an accuracy more than 90% in the first scenario. The second one is to get an accuracy more than 80% when the gripper base of the arm touches the tube.

A camera-image node is subscribed, and 2 dimensional images containing the robot arm and the tube are sent as an input to the DQN agent. Therefore, convolutional neural networks are trained by capturing patches from images, and constructs kernels to identify specific image features.

The agent outputs actions to control joints positions. Each degree of freedom, DOF, contributes to positive and negative moving actions, and the amount of action indexes is $DOF * 2$. Therefore, a joint position is specified by $action\_index/2$. Action indexes with even integer are positive motions while odd action indexes are negative motions.

When the arm collides with the ground, it is easily to cause damage to the structure. Therefore, colliding with the ground is considered a fail case. A bounding box captures boundary coordinates of the arm, and a ground collision is detected when the minimum z value is very close to the ground within $groundContact = 0.05$. When the ground collision is raised, an episode is terminated, and a negative $REWARD\_LOSS$ is issued to discourage the agent.

During the arm's moving, it is better to make the arm close to the target. A $distDelta$ is a difference between the last arm-to-object distance and the current arm-to-object distance. It is positive when the arm is approaching the target, and the arm's leaving the target will lead to a negative value. A smoothed moving average of the delta of the distance to the goal is calculated, and is used as a reward. To avoid the arm staying at the same place or moving very slowly, a negative reward is added when the situation is detected.

To detect an arm-to-target collision, a collision message is checked whether it contains $COLLISION\_ITEM$ ($tube :: tube\_link :: tube\_collision$). Because the environment has only the arm and the tube, the arm must touch the tube when the tube is collided. On the other case, a gripper-base-to-target collision is found when the message also includes $COLLISION\_POINT$ ($arm :: gripperbase :: gripper\_link$).

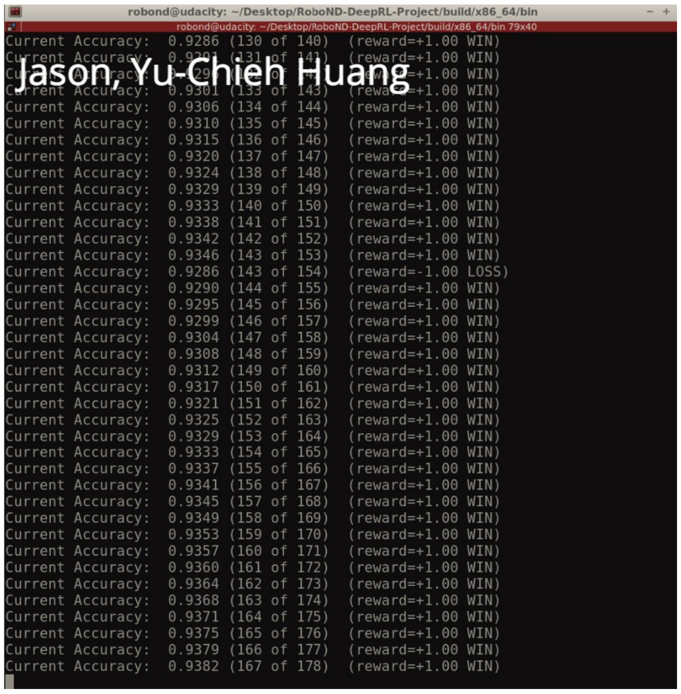
In both cases, when $collisionCheck$ is $true$, i.e. the goal is achieved, the environment issues a positive $REWARD\_WIN$ value and terminates the episode.

4 Hyperparemeters

In convolutional neural networks, a larger image consumes more computation resource. On the contrary, a smaller image degrades the resolution and leads to accuracy impact. There is a trade-off needed to be considered. $INPUT\_WIDTH$ and $INPUT\_HEIGHT$ are decreased from 512 to 128.

A $RMSprop$ optimizer is added. In the future, different optimizers can be tested. A learning rate must be a value larger than zero, and neural network parameters, like weighting and bias, are modified base on multiplications of back-propagation values and the learning rate. Nonetheless, the learning rate can not be a large value, or the network will oscillate around the optimal point rather than converge to it. The learning rate is set to 0.05. In the future, different values can be analyzed by grid search. If a large amount of data is send to GPU once, it may be running out of the memory. The batch size is set to 512. The memory loading is within the limit. Here a long-short-term memory (LSTM) recurrent neural network (RNN) is used to gain information of time sequence. In each training example, a series of time-sequenced images is input to the agent. Standard neural networks, like multi-layer perceptrons or convolutional neural networks, don't integrate time sequence information well. The RNN trains neurons base on the current input and previous states, and time factors are included since previous and current information both processed.

$REWARD\_WIN$ is set to 1 while $REWARD\_LOSS$ is set to -1. Beneficial situations are: the arm approaches or touches the tube. Positive rewards encourage to continue these beneficial actions. A negative reward discourages the agent to avoid: the arm leaves away the tube, touches the ground, or the episode is terminated without reaching the goal.


robond@udacity: ~/Desktop/RoboND-DeepRL-Project/build/x86_64/bin
robond@udacity: ~/Desktop/RoboND-DeepRL-Project/build/x86_64/bin 79x40
Jason, Yu-Chieh Huang

```
Current Accuracy:  0.9286 (130 of 140)  (reward=+1.00 WIN)
Current Accuracy:  0.9291 (131 of 141)  (reward=+1.00 WIN)
Current Accuracy:  0.929[illegible] (132 of 142)  (reward=+1.00 WIN)
Current Accuracy:  0.9301 (133 of 143)  (reward=+1.00 WIN)
Current Accuracy:  0.9306 (134 of 144)  (reward=+1.00 WIN)
Current Accuracy:  0.9310 (135 of 145)  (reward=+1.00 WIN)
Current Accuracy:  0.9315 (136 of 146)  (reward=+1.00 WIN)
Current Accuracy:  0.9320 (137 of 147)  (reward=+1.00 WIN)
Current Accuracy:  0.9324 (138 of 148)  (reward=+1.00 WIN)
Current Accuracy:  0.9329 (139 of 149)  (reward=+1.00 WIN)
Current Accuracy:  0.9333 (140 of 150)  (reward=+1.00 WIN)
Current Accuracy:  0.9338 (141 of 151)  (reward=+1.00 WIN)
Current Accuracy:  0.9342 (142 of 152)  (reward=+1.00 WIN)
Current Accuracy:  0.9346 (143 of 153)  (reward=+1.00 WIN)
Current Accuracy:  0.9286 (143 of 154)  (reward=-1.00 LOSS)
Current Accuracy:  0.9290 (144 of 155)  (reward=+1.00 WIN)
Current Accuracy:  0.9295 (145 of 156)  (reward=+1.00 WIN)
Current Accuracy:  0.9299 (146 of 157)  (reward=+1.00 WIN)
Current Accuracy:  0.9304 (147 of 158)  (reward=+1.00 WIN)
Current Accuracy:  0.9308 (148 of 159)  (reward=+1.00 WIN)
Current Accuracy:  0.9312 (149 of 160)  (reward=+1.00 WIN)
Current Accuracy:  0.9317 (150 of 161)  (reward=+1.00 WIN)
Current Accuracy:  0.9321 (151 of 162)  (reward=+1.00 WIN)
Current Accuracy:  0.9325 (152 of 163)  (reward=+1.00 WIN)
Current Accuracy:  0.9329 (153 of 164)  (reward=+1.00 WIN)
Current Accuracy:  0.9333 (154 of 165)  (reward=+1.00 WIN)
Current Accuracy:  0.9337 (155 of 166)  (reward=+1.00 WIN)
Current Accuracy:  0.9341 (156 of 167)  (reward=+1.00 WIN)
Current Accuracy:  0.9345 (157 of 168)  (reward=+1.00 WIN)
Current Accuracy:  0.9349 (158 of 169)  (reward=+1.00 WIN)
Current Accuracy:  0.9353 (159 of 170)  (reward=+1.00 WIN)
Current Accuracy:  0.9357 (160 of 171)  (reward=+1.00 WIN)
Current Accuracy:  0.9360 (161 of 172)  (reward=+1.00 WIN)
Current Accuracy:  0.9364 (162 of 173)  (reward=+1.00 WIN)
Current Accuracy:  0.9368 (163 of 174)  (reward=+1.00 WIN)
Current Accuracy:  0.9371 (164 of 175)  (reward=+1.00 WIN)
Current Accuracy:  0.9375 (165 of 176)  (reward=+1.00 WIN)
Current Accuracy:  0.9379 (166 of 177)  (reward=+1.00 WIN)
Current Accuracy:  0.9382 (167 of 178)  (reward=+1.00 WIN)
```

Fig. 1. Case 1: the arm touches the tube - accuracy.

5 Results

5.1 Case 1: the arm touches the tube.

A training result for the arm touching the tube is shown in the figure 1. After around 150 runs, the accuracy exceeds 93%. In figure 2, a Gazebo environment when the arm touching the tube is illustrated.

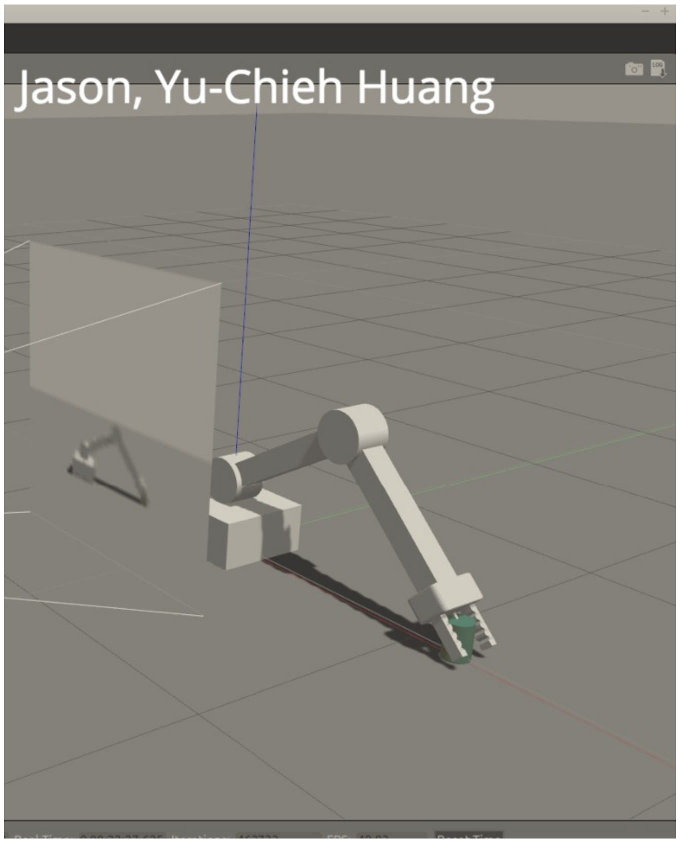


Fig. 2. Case 1: the arm touches the tube - gazebo.

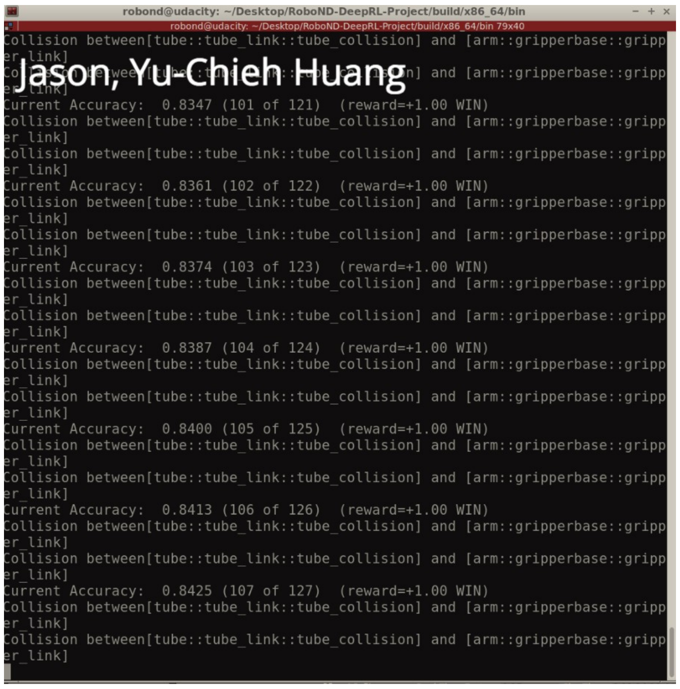
5.2 Case 2: the gripper base touches the tube.

A training result for the gripper base touching the tube is shown in the figure 3. After around 100 runs, the accuracy reaches 83%.

6 Conclusion / Future work

In this project, a Gazebo environment is given. Rewards are fine-tuned base on different criteria, and a DQN agent is trained to get target accuracy for two tasks.

In the future, a different environment can be constructed and tested. For example, a self driving car can drive inside a lane base on images.


robond@udacity: ~/Desktop/RoboND-DeepRL-Project/build/x86_64/bin
robond@udacity: ~/Desktop/RoboND-DeepRL-Project/build/x86_64/bin 79x40
Jason, Yu-Chieh Huang

```
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Current Accuracy:  0.8347 (101 of 121)  (reward=+1.00 WIN)
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Current Accuracy:  0.8361 (102 of 122)  (reward=+1.00 WIN)
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Current Accuracy:  0.8374 (103 of 123)  (reward=+1.00 WIN)
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Current Accuracy:  0.8387 (104 of 124)  (reward=+1.00 WIN)
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Current Accuracy:  0.8400 (105 of 125)  (reward=+1.00 WIN)
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Current Accuracy:  0.8413 (106 of 126)  (reward=+1.00 WIN)
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Current Accuracy:  0.8425 (107 of 127)  (reward=+1.00 WIN)
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripp
er_link]
```

Fig. 3. Case 2: the gripper base touches the tube - accuracy.