Deep RL Arm Manipulation

Harrison Seung

Abstract—The following report outlines the Deep Reinforcement Learning Arm Manipulation project of Term 2 for Udacity's Robotic Software Engineering nanodegree. The goal of the project is to create a DQN agent and define the reward functions to teach a robotic arm to carry out two objectives. The first objective is for any part of the robot arm to touch an object with 90%accuracy. The second objective is for only the gripper base of the robot to touch an object with an accuracy of 80%. Each task is completed using a a C++ API to run the RL agent and simulated in a Gazebo world.

Index Terms—Deep Reinforcement Learning.

1 Introduction

Deep Reinforcement Learning is an approach to robotics where a solution to a problem is found by focusing on the goal rather than the steps to reach the goal. This is achieved by training an agent (robot) to perform a task by providing incentives for positive actions and/or discouragement for negative actions. The following project presents an application for training a robotic arm to perform a task of locating an object/prop using deep reinforcement learning.

1.1 Problem statement

Create a DQN agent and define reward functions to teach a robotic arm to carry out two primary objectives:

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

2 Reward Functions

In Reinforcement Learning, the reward functions provide feedback to the agent on the performance of its actions. For this project there were two types of Rewards: End of Episode and Interim Rewards.

2.1 End of Episode (EoE) Rewards

The end of episode rewards are received when an event occurs that ends the current episode. This includes a collision with an object, a collision with the ground, or exceeding the max frame count for the episode. For Task 1, the goal was to have any part of the robot arm make contact with the prop. The reward for any collision with the arm was set at +100. Any other collisions were set at a loss of -100. Additionally, collisions with the ground and exceeding the maximum episode length were rewarded a loss of -100. For Task 2, the goal is to have ONLY the gripperbase of the robot arm contact the prop. This adds complexity as a collision with the gripper fingers, gripper middle prong, and robot arm linkages are all considered losses. To complete Task2, the reward for colliding with ONLY the gripperbase was set at +1000, an order of magnitude greater than the worst case loss conditions to strongly encourage the agent to go for the goal. For the loss conditions, colliding with another object was set to 50% less than the worst case scenarios of ground contact or episode timeout. This indicated colliding with another part of the arm was not as significant as a ground contact or episode time out.

2.2 Interim Rewards

The interim reward provides positive reward for movements toward the goal and negative reward for moving away. This reward is calculated using a smoothed average (SMA) function equation shown below. The SMA retains a

```
average_delta  = (average_delta * alpha) + (dist * (1 - alpha));
```

certain percentage of the previous goal distance to minimize the volatility of the interim reward function. The percentage is controlled by alpha, a constant between 0 and 1, where a greater value correlates to an emphasis on the previous goal distance and a lower value correlates to an emphasis on the current goal distance.

2.3 Joint Control

Although not as accurate in real world applications, a position based controller was chosen for this project over the velocity based controller as it was simpler to implement.

3 Hyperparameters

When selecting the hyperparameters for Task1, the initial settings chosen were modeled after the Fruit Game sample as this had a similar end goal objective of finding a "fruit/prop". Next the camera frame was scaled down from 512x512 to match the camera sensor output of our model at 64x64. The learning rate was set at an initial value of 0.01 and reduced by 50% until the robot arm began to converge at 0.001.

For Task 2, after over 1000 runs with the Task1 hyperparameters, it was observed that the accuracy of the DQN agent could not exceed 78%. This appeared as though the agent had found an optimal path but would slightly adjust every couple of moves to explore a new path.

```
// Define DQN API Settings

#define INPUT_CHANNELS 3
#define ALLOW_RANDOM true
#define DEBUG_DQN true
#define GAMMA 0.9f // 0.9
#define EPS_START 0.99f // 0.9
#define EPS_END 0.01f // 0.05
#define EPS_DECAY 200 // 200
#define NUM_ACTIONS 2*DOF // 6

// Define hyperparameters
#define INPUT_WIDTH   64 // 512
#define INPUT_HEIGHT  64 // 512
#define OPTIMIZER "RMSprop" // none
#define LEARNING_RATE 0.001f // 0.0
#define REPLAY_MEMORY 10000 // 10000
#define BATCH_SIZE 32 // 8
#define USE_LSTM true // false
#define LSTM_SIZE 256 // 32
```


Fig. 1. Hyperparameters (default in comments)

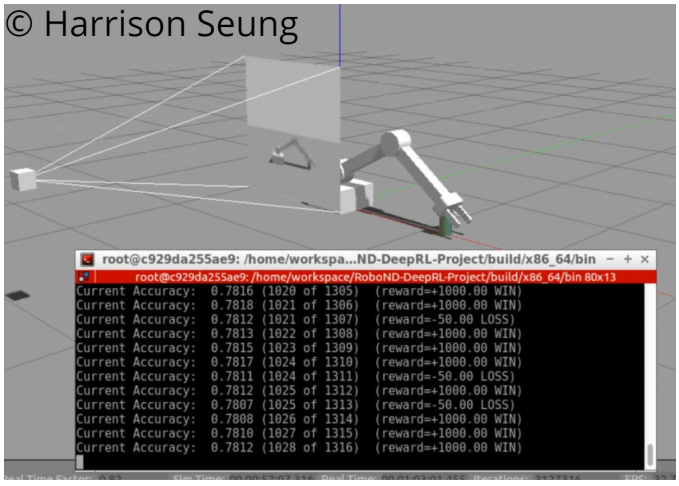

Fig. 2. Trial run for Task 2

Instead of adjusting the hyperparemeters, the DQN API settings for EPS_START and EPS_END were adjusted. EPS_START and EPS_END control the amount of exploration the DQN agent performs at the beginning and end of the training. As the agent was very close to the goal, intuition indicated only a small modification to each setting was required. An increase to 0.99 for the EPS_START and a reduction to 0.01 to EPS_END was sufficient for the DQN agent to complete the task.

4 Results

After significant tuning of the reward functions, hyperparameters, and DQN agent settings, the goal for both tasks were achieved. For Task 1, the DQN agent was able to quickly adjust itself to accurately hit the prop achieving an accuracy of 90.67% after 237 attempts. For Task 2, the DQN

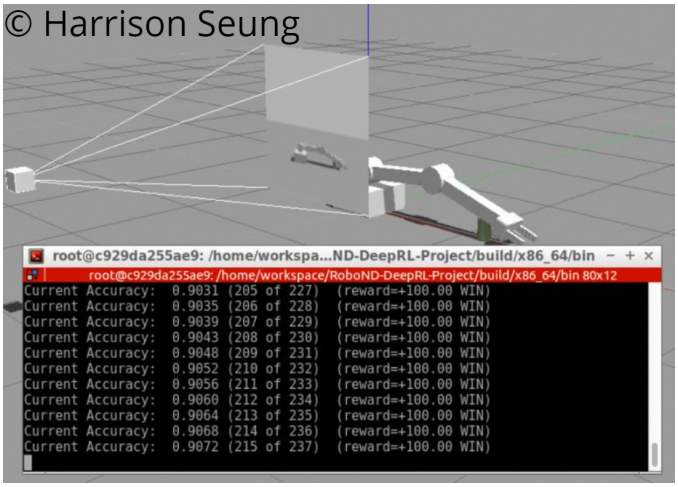

Fig. 3. Task 1: Accuracy greater >90% for collision with any part of robot arm (Note: video of task provided with submission)

agent was able to achieve an accuracy of 91.15% after 113 attempts. A notable shortcoming of the DQN for Task 2 is

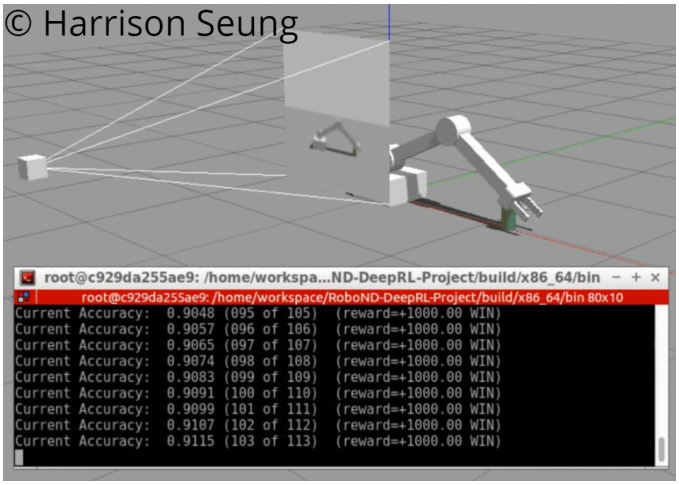

Fig. 4. Task 2: Accuracy >80% for collision with ONLY gripperbase of robot arm (Note: video of task provided with submission)

a frequency of hitting local minimums when approaching the prop goal causing the robot arm to stall and runout the episode length. This primarily occurred on attempts where the robot had not yet found a path to the prop.

5 Future Work

Areas for further improvement of the DQN agent would be a modification of the reward function to address the local minimums during the exploration stage. On some instances the robot arms approach to the goal would be jerky and the collision would be excessive enough to knock the prop off screen. Switching to a velocity based controller would help smooth out the movements.