Deep Reinforcement Learning Arm Manipulation

Cheshta Dhingra

Abstract—The goal of this project is to 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 a 90
2) Have only the gripper base of the robot arm touch the object, with at least a 80

XXX

Index Terms—Deep RL, Robotics Nanodegree Program, Udacity.

1 Introduction

Reinforcement Learning (RL) is a subfield of Machine Learning where an agent learns by interacting with its environment, observing the results of these interactions and receiving a reward (positive or negative) accordingly. This way of learning mimics the fundamental way in which we humans (and animals alike) learn.
As J. Kober, J. Andrew (Drew) Bagnell, and J. Peters point out in Reinforcement Learning in Robotics: A Survey:
Reinforcement learning (RL) enables a robot to autonomously discover an optimal behavior through trial-and-error interactions with its environment. Instead of explicitly detailing the solution to a problem, in reinforcement learning the designer of a control task provides feedback in terms of a scalar objective function that measures the one-step performance of the robot.
This project is based on the Nvidia open source project ”jetson-reinforcement” developed by Dustin Franklin. The goal of the project is to 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 a 90
2) Have only the gripper base of the robot arm touch the object, with at least a 80

2 Reward Functions

The robotic arm was given a positive reward of +1.0 for ”winning” an episode and a negative reward of -1.0 for ”losing” it. A ”win” was defined based on the task: in the first task, the full +10 was rewarded when any part of the arm touched the object (or if the gripper touched the object in task 2) while -10 was rewarded for hitting the ground. To do this, we checked if the tube (COLLISION_ITEM) touched the arm. In the second task, we checked whether the arm element that collided with the tube was the gripper base link.
There were several trigger points at which an episode was terminated:

1) The arm touched the ground

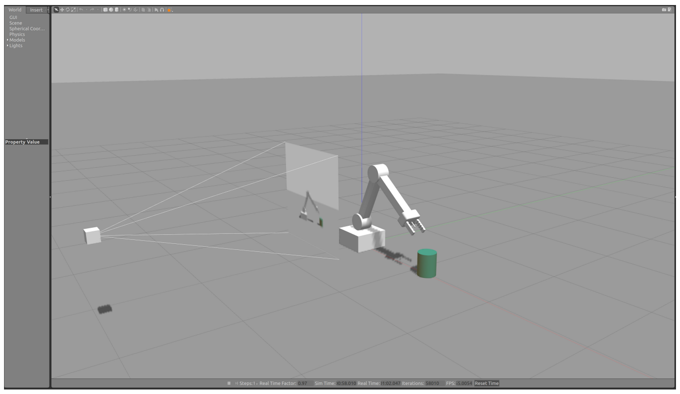


Fig. 1. Deep RL Arm Manipulator Setup.

2) If task 1 was accomplished
3) if task 2 was accomplished
4) if the episode exceeds 100 frames

In order to check whether the arm hit the ground, bounding boxes were used to demarcate the position of the gripper. If the minimum or maximum z-value of the gripper’s bounding box was below a threshold of 0.05, it was considered to have hit the ground and the episode was terminated. This action prompted the arm to incur a REWARD_LOSS if the time ran out (¿100 frames had passed).
A position based joint control was used.
Interim rewards were provided if the robot arm moved in the correct direction (towards the object). We utilized a smoothed moving average of the delta of the distance to the goal to measure the robot’s progress. If there was very minimal movement or movement in the wrong direction, a small penalty was incurred. On the other hand, the closer the gripper moved towards the goal, the more it was rewarded (with the magnitude of the reward proportional to the distance covered). The delta in distance was calculated by subtracting the distance between the gripper and object bounding boxes from the previous distance. The reward and penalty were 0.1 * distance traveled or -0.1 * distance traveled, respectively.

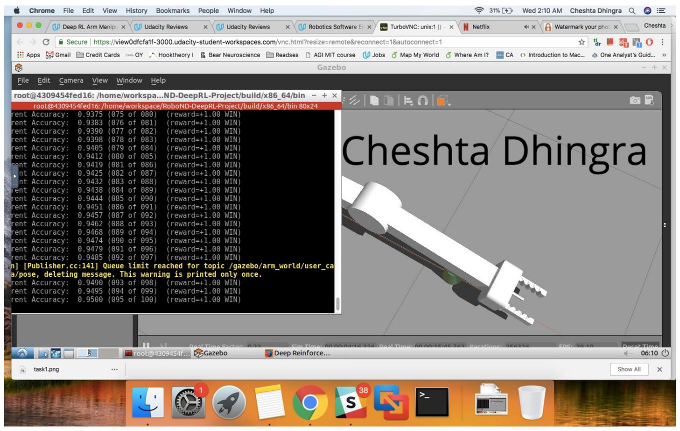
If the episode exceeded 100 frames, the episode was terminated and a penalty of -0.1 was incurred.

2.1 Hyperparameters

The selected hyperparameters are below:

- INPUT_WIDTH 128
- INPUT_HEIGHT 128
- OPTIMIZER ”Adam”
- LEARNING_RATE 0.1f
- REPLAY_MEMORY 20000
- BATCH_SIZE 512
- USE_LSTM true
- LSTM_SIZE 256

Training was completed with 128 x 128 size input images. A Long-short term memory network of size 256 was used with a replay memory of 20000. The optimizer chosen was Adam, learning rate was 0.1 and batch size was 512.
The majority of these parameters were chosen based on the default utilized by Dustin Franklin in the repo that this project is based upon: https://github.com/stBecker/RoboND-DeepRL-Project/blob/master/gazebo/ArmPlugin.cpp. An image size of 512 x 512 was attempted at first, but found to be inefficient and time consuming, especially for task 2, so the size was decreased. It can likely be decreased further to 64 x 64.
Adam optimizer was used instead of RMSprop as a training session with RMSprop was found to be comparatively unstable. The Adam optimizer allowed for a smoother trajectory of Wins and Losses with several in a row, and ultimately a higher Win rate than RMSprop.

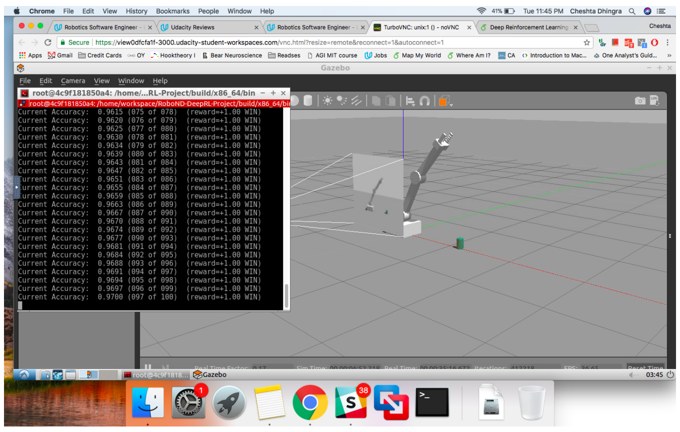


```
root@4309454fed16: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
rent Accuracy:  0.9375 (075 of 080)  (reward=+1.00 WIN)
rent Accuracy:  0.9383 (076 of 081)  (reward=+1.00 WIN)
rent Accuracy:  0.9390 (077 of 082)  (reward=+1.00 WIN)
rent Accuracy:  0.9398 (078 of 083)  (reward=+1.00 WIN)
rent Accuracy:  0.9405 (079 of 084)  (reward=+1.00 WIN)
rent Accuracy:  0.9412 (080 of 085)  (reward=+1.00 WIN)
rent Accuracy:  0.9419 (081 of 086)  (reward=+1.00 WIN)
nt Accuracy:  0.9425 (082 of 087)  (reward=+1.00 WIN)
nt Accuracy:  0.9432 (083 of 088)  (reward=+1.00 WIN)
rent Accuracy:  0.9438 (084 of 089)  (reward=+1.00 WIN)
rent Accuracy:  0.9444 (085 of 090)  (reward=+1.00 WIN)
rent Accuracy:  0.9451 (086 of 091)  (reward=+1.00 WIN)
rent Accuracy:  0.9457 (087 of 092)  (reward=+1.00 WIN)
rent Accuracy:  0.9462 (088 of 093)  (reward=+1.00 WIN)
rent Accuracy:  0.9468 (089 of 094)  (reward=+1.00 WIN)
rent Accuracy:  0.9474 (090 of 095)  (reward=+1.00 WIN)
rent Accuracy:  0.9479 (091 of 096)  (reward=+1.00 WIN)
rent Accuracy:  0.9485 (092 of 097)  (reward=+1.00 WIN)
n] [Publisher.cc:141] Queue limit reached for topic /gazebo/arm_world/user_ca
a/pose, deleting message. This warning is printed only once.
rent Accuracy:  0.9490 (093 of 098)  (reward=+1.00 WIN)
rent Accuracy:  0.9495 (094 of 099)  (reward=+1.00 WIN)
rent Accuracy:  0.9500 (095 of 100)  (reward=+1.00 WIN)
```

Fig. 2. Task 1 at 100 episodes.

2.2 Results

The RL agent was able to successfully learn the Q functions for the 2 tasks. Using this, it was able to maximize its expected rewards given a specific state-action pair.
For the first task, the agent achieved an accuracy of 0.95 by the 100th episode, as shown in Fig. 1. It was able to achieve this by the first 10 or so episodes, after which it smoothly continued to win episodes. The few episodes that were ended were due to the episode exceeding 100 frames.
In the second task, the agent achieved the required accuracy of 0.97 by the 100th episode, as shown in Fig. 3. It achieved the required .80 accuracy by episode 50 or so. Although this task was accomplished with greater accuracy than task 1, the RL agent would frequently move in a jerky manner, so there is room for improvement, described in the Future Work section.



```
root@4c9f181850a4: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bi
Current Accuracy:  0.9615 (075 of 078)  (reward=+1.00 WIN)
Current Accuracy:  0.9620 (076 of 079)  (reward=+1.00 WIN)
Current Accuracy:  0.9625 (077 of 080)  (reward=+1.00 WIN)
Current Accuracy:  0.9630 (078 of 081)  (reward=+1.00 WIN)
Current Accuracy:  0.9634 (079 of 082)  (reward=+1.00 WIN)
Current Accuracy:  0.9639 (080 of 083)  (reward=+1.00 WIN)
Current Accuracy:  0.9643 (081 of 084)  (reward=+1.00 WIN)
Current Accuracy:  0.9647 (082 of 085)  (reward=+1.00 WIN)
urrent Accuracy:  0.9651 (083 of 086)  (reward=+1.00 WIN)
urrent Accuracy:  0.9655 (084 of 087)  (reward=+1.00 WIN)
urrent Accuracy:  0.9659 (085 of 088)  (reward=+1.00 WIN)
Current Accuracy:  0.9663 (086 of 089)  (reward=+1.00 WIN)
Current Accuracy:  0.9667 (087 of 090)  (reward=+1.00 WIN)
Current Accuracy:  0.9670 (088 of 091)  (reward=+1.00 WIN)
Current Accuracy:  0.9674 (089 of 092)  (reward=+1.00 WIN)
Current Accuracy:  0.9677 (090 of 093)  (reward=+1.00 WIN)
Current Accuracy:  0.9681 (091 of 094)  (reward=+1.00 WIN)
Current Accuracy:  0.9684 (092 of 095)  (reward=+1.00 WIN)
Current Accuracy:  0.9688 (093 of 096)  (reward=+1.00 WIN)
Current Accuracy:  0.9691 (094 of 097)  (reward=+1.00 WIN)
Current Accuracy:  0.9694 (095 of 098)  (reward=+1.00 WIN)
Current Accuracy:  0.9697 (096 of 099)  (reward=+1.00 WIN)
Current Accuracy:  0.9700 (097 of 100)  (reward=+1.00 WIN)
```

Fig. 3. Task 2 at 100 episodes.

2.3 Future Work

There are several ways to both improve performance on the tasks at hand as well as complicate the tasks further to make them more challenging. We should tune the parameters to ensure the agent can converge more quickly in task 2. One way could be to reduce the learning rate or use a different optimizer. We could also reduce the size of the input images that are used for training. In terms of further challenges, we could randomize the object's location at the start of each episode and increase the arm's reach by giving it more degrees of freedom.