Arm Manipulation with Deep RL

Esraa Magdy

Abstract—This paper presents a Deep learning based approach to control the motion of a robotic arm using images from a single camera sensor. This approach utilizes a Deep Q Network (DQN) to train the robotic arm in a simulation environment.

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

1 Reward Functions

There are four rewards that the arm can have during any episode:

1- A reward for hitting the ground : This is a simple reward loss minus a factor that represents how early the robot hits the ground to encourage the robot to not hit it early.

$$rewardHistory = REWARD\_LOSS * 10 -$$

$$((maxEpisodeLengthepisodeFrames)/maxEpisodeLength)$$

2- A reward loss for when the episode terminates. 3- A reward win when the right part of the arm touches the object, This was the same for both objective functions. It's simply a reward win times a multiplier plus a factor that is inversely proportional to how many frames have already done before wining. this factor to encourage the robot to finish fast.

$$rewardHistory = REWARD\_WIN * 100$$

$$+(maxEpisodeLength - episodeFrames) * 100$$

4- the last reward function is the interim reward function: For task 1, it was a simple smoothed average of the delta distance between the robot and the object times a multiplier as this average is usually a small number so it was better to add a multiplier. the delta distance will be a positive number if the robot is moving towards the object but a negative number if it's moving away from it. so this interim reward serves as a reward win and a reward loss at the same time based on in which direction the arm is moving.

$$avgGoalDelta = (avgGoalDelta * alpha)$$

$$+(distDelta * (1 - alpha))$$

Different values for alpha were experimented but at the end the value for alpha was 0.3, that means that the recent values for delta distance were more important than the history. the reward function was defined as :

$$rewardHistory = tanh(avgGoalDelta) * 100$$

For task2, Multiple rewards were experimented but finally a simple reward function was able to make it. The reward function used here was first presented in [1]. And states that:

$$if deltadistance > 0$$

$$rewardHistory = REWARD\_WIN * 10 + avgGoalDelta$$

$$if deltadistance < 0$$

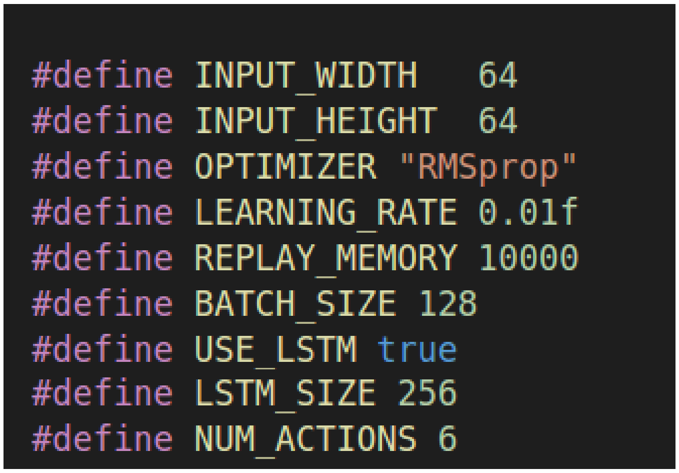
$$rewardHistory = REWARD\_LOSS * 10$$

$$elserewardHistory = 0.$$

Other reward functions could achieve a similar results but maybe with different hyper parameters . It depends on how fast the robot discovers the target object and learns that touching it results in a great reward. As long as the reward function is logically right, the robot will be able to achieve the objectives but it might take long time. More complex reward functions might produce the same results faster.

2 Hyperparameters

A starter guess for the hyperparameters was based on fruit and catch examples.The hyperparameters were the same for task one and task2.



```
#define INPUT_WIDTH   64
#define INPUT_HEIGHT  64
#define OPTIMIZER "RMSprop"
#define LEARNING_RATE 0.01f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 128
#define USE_LSTM true
#define LSTM_SIZE 256
#define NUM_ACTIONS 6
```

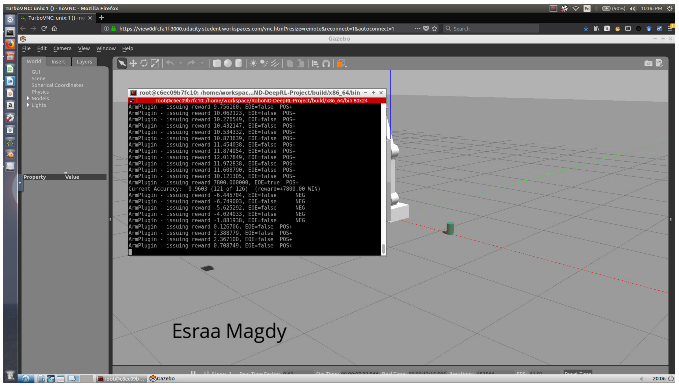
Fig. 1. HyperParams

The width and the height was set to 64 as Gazebo reported that this was the size of the image. beside a square image would reduce the complexity for the calculations. The Optimizer used was RMSprop. Other optimizers like SGD or Adam could have been used as well. The learning rate was set to 0.01. Usually increasing the batch size have the same effect as decreasing the learning rate, so instead of decreasing the learning rate below 0.01, the batch size was increased to 128. the agent was learning slowly but it manged to converge smoothly after some time. Increasing the LSTM size was a good practice to help it remember the recent good moves.

3 Results

3.1 Task1

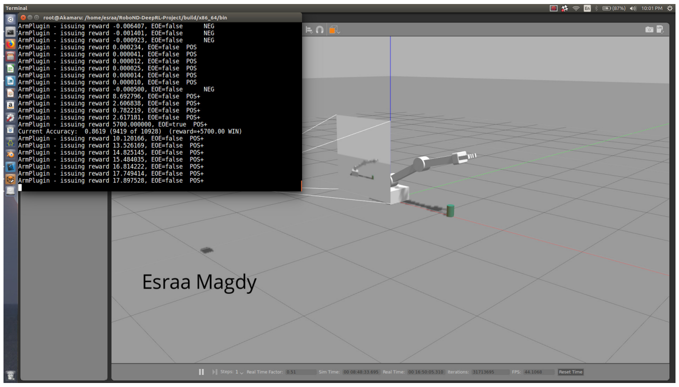
For task1, Velocity control was first used on the Udacity workspace environment and achieved 96% accuracy.


root@c6ec09b7fc10: /home/workspac...ND-DeepRL-Project/build/x86_64/bin
root@c6ec09b7fc10: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Gazebo
Esraa Magdy

```
ArmPlugin - issuing reward 9.756160, EOE=false  POS+
ArmPlugin - issuing reward 10.062123, EOE=false  POS+
ArmPlugin - issuing reward 10.276549, EOE=false  POS+
ArmPlugin - issuing reward 10.432147, EOE=false  POS+
ArmPlugin - issuing reward 10.534332, EOE=false  POS+
ArmPlugin - issuing reward 10.873639, EOE=false  POS+
ArmPlugin - issuing reward 11.454038, EOE=false  POS+
ArmPlugin - issuing reward 11.874954, EOE=false  POS+
ArmPlugin - issuing reward 12.017849, EOE=false  POS+
ArmPlugin - issuing reward 11.972838, EOE=false  POS+
ArmPlugin - issuing reward 11.608790, EOE=false  POS+
ArmPlugin - issuing reward 10.121305, EOE=false  POS+
ArmPlugin - issuing reward 7800.000000, EOE=true  POS+
Current Accuracy:  0.9603 (121 of 126)  (reward=+7800.00 WIN)
ArmPlugin - issuing reward -6.445704, EOE=false      NEG
ArmPlugin - issuing reward -6.749003, EOE=false      NEG
ArmPlugin - issuing reward -5.625292, EOE=false      NEG
ArmPlugin - issuing reward -4.024033, EOE=false      NEG
ArmPlugin - issuing reward -1.881938, EOE=false      NEG
ArmPlugin - issuing reward 0.126706, EOE=false  POS+
ArmPlugin - issuing reward 2.388779, EOE=false  POS+
ArmPlugin - issuing reward 2.367100, EOE=false  POS+
ArmPlugin - issuing reward 0.708749, EOE=false  POS+
```

Fig. 2. Velocity Control

Then, position control was experimented. Position control movements are smoother than velocity control movements but it took more time to reach a good accuracy.


Terminal
root@Akamaru: /home/esraa/RoboND-DeepRL-Project/build/x86_64/bin
Esraa Magdy

```
ArmPlugin - issuing reward -0.006407, EOE=false      NEG
ArmPlugin - issuing reward -0.001401, EOE=false      NEG
ArmPlugin - issuing reward -0.000923, EOE=false      NEG
ArmPlugin - issuing reward 0.000234, EOE=false  POS
ArmPlugin - issuing reward 0.000041, EOE=false  POS
ArmPlugin - issuing reward 0.000012, EOE=false  POS
ArmPlugin - issuing reward 0.000025, EOE=false  POS
ArmPlugin - issuing reward 0.000014, EOE=false  POS
ArmPlugin - issuing reward 0.000010, EOE=false  POS
ArmPlugin - issuing reward -0.000500, EOE=false      NEG
ArmPlugin - issuing reward 8.692796, EOE=false  POS+
ArmPlugin - issuing reward 2.606838, EOE=false  POS+
ArmPlugin - issuing reward 0.782219, EOE=false  POS+
ArmPlugin - issuing reward 2.617181, EOE=false  POS+
ArmPlugin - issuing reward 5700.000000, EOE=true  POS+
Current Accuracy:  0.8619 (9419 of 10928)  (reward=+5700.00 WIN)
ArmPlugin - issuing reward 10.120166, EOE=false  POS+
ArmPlugin - issuing reward 13.526169, EOE=false  POS+
ArmPlugin - issuing reward 14.825145, EOE=false  POS+
ArmPlugin - issuing reward 15.484035, EOE=false  POS+
ArmPlugin - issuing reward 16.814222, EOE=false  POS+
ArmPlugin - issuing reward 17.749414, EOE=false  POS+
ArmPlugin - issuing reward 17.897528, EOE=false  POS+
```

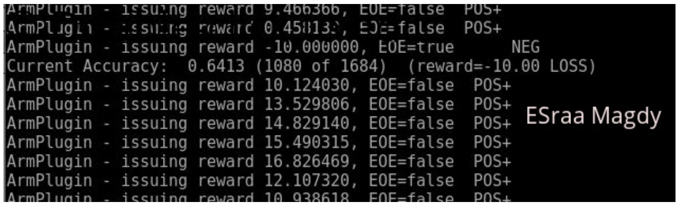
Fig. 3. Position Control

3.2 Task2

At first the arm couldn't get beyond 65%.That was odd but might be a result to the exploration and trying random values even after converging to the good move. Then After some tweaks with the reward function and decreasing EPS_End to 0.01 the arm reached the required accuracy but the convergence was really slow. For task2, position control was better in that case as gripper base collision depends on reaching a precise position.

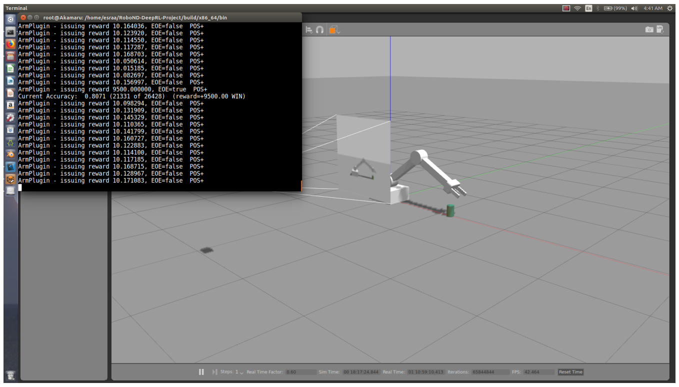
4 Future Work

Further improvements could concentrate on how to implement a complex reward function that would make the robot touch the object faster to increase the convergence rate. Also, Other optimizers could be tested to see the effect of each optimizer on the convergence rate.



```
ArmPlugin - issuing reward 9.466366, EOE=false  POS+
ArmPlugin - issuing reward 0.458133, EOE=false  POS+
ArmPlugin - issuing reward -10.000000, EOE=true      NEG
Current Accuracy:  0.6413 (1080 of 1684)  (reward=-10.00 LOSS)
ArmPlugin - issuing reward 10.124030, EOE=false  POS+
ArmPlugin - issuing reward 13.529806, EOE=false  POS+
ArmPlugin - issuing reward 14.829140, EOE=false  POS+
ArmPlugin - issuing reward 15.490315, EOE=false  POS+
ArmPlugin - issuing reward 16.826469, EOE=false  POS+
ArmPlugin - issuing reward 12.107320, EOE=false  POS+
ArmPlugin - issuing reward 10.938618, EOE=false  POS+
```

Fig. 4. Results


Terminal
root@Akamaru: /home/esraa/RoboND-DeepRL-Project/build/x86_64/bin

```
ArmPlugin - issuing reward 10.164036, EOE=false  POS+
ArmPlugin - issuing reward 10.123920, EOE=false  POS+
ArmPlugin - issuing reward 10.114550, EOE=false  POS+
ArmPlugin - issuing reward 10.117287, EOE=false  POS+
ArmPlugin - issuing reward 10.168703, EOE=false  POS+
ArmPlugin - issuing reward 10.050614, EOE=false  POS+
ArmPlugin - issuing reward 10.015185, EOE=false  POS+
ArmPlugin - issuing reward 10.082697, EOE=false  POS+
ArmPlugin - issuing reward 10.156997, EOE=false  POS+
ArmPlugin - issuing reward 9500.000000, EOE=true  POS+
Current Accuracy:  0.8071 (21331 of 26428)  (reward=+9500.00 WIN)
ArmPlugin - issuing reward 10.098294, EOE=false  POS+
ArmPlugin - issuing reward 10.131909, EOE=false  POS+
ArmPlugin - issuing reward 10.145329, EOE=false  POS+
ArmPlugin - issuing reward 10.110365, EOE=false  POS+
ArmPlugin - issuing reward 10.141799, EOE=false  POS+
ArmPlugin - issuing reward 10.160727, EOE=false  POS+
ArmPlugin - issuing reward 10.122883, EOE=false  POS+
ArmPlugin - issuing reward 10.114100, EOE=false  POS+
ArmPlugin - issuing reward 10.117185, EOE=false  POS+
ArmPlugin - issuing reward 10.168715, EOE=false  POS+
ArmPlugin - issuing reward 10.128967, EOE=false  POS+
ArmPlugin - issuing reward 10.171083, EOE=false  POS+
```

Fig. 5. Task2 Results

5 References

[1]https://arxiv.org/pdf/1511.03791.pdf