DEEP RL ARM MANIPULATION

Abstract:

The deep RL project consists of training a 3 degree of freedom arm to learn to touch the object. The project uses OpenAI Gym environment and a deep Q learning network. The arm is simulated in Gazebo. The agent interacts with the environment using a plugin for gazebo named Armplugin.cpp. Two main learning task are covered in this project. First task is to train the agent to touch any part of the arm to the object with at least 90% accuracy for a minimum of hundred runs. The second task is to train the agent to touch the gripper base to the object with at least 80% accuracy for minimum of hundred runs. The parameters and reward functions that were used are discussed. In the end, the possible future improvements are discussed.

Reward Functions:

For the first task, a simple reward function proportional to the change in distance from the object was used. This function is given below -

distDelta = lastGoalDistance - distGoal
*reward = distDelta * 1000*

The distDelta calculates how much did the distance of arm from the base of the arm change. Then that distance is multiplied by 1000 to assign a reward. Therefore, if the distance from the goal increases, the agent is penalised with a proportionate negative reward. If the distance is positive, the agent is rewarded with a proportionate positive reward. The reward parameters were set as given below -

REWARD_WIN was set to 1000.
REWARD_LOSS was set to -1000.

Velocity control was used to train the agent as it performed better than position control. Changing the multiplier of distDelta from 1000 to 100 reduced accuracy to 80%.

For the second task, the interim reward function is the moving average of the change in distance to goal. This is as given below -

avgGoalDelta = (avgGoalDelta * alpha) + (distDelta * (1.0 – alpha)) where alpha = 0.8.

When the gripper base failed to touch the object, a reward of REWARD_LOSS*(1+distGoal) was given and the episode was terminated. When the gripper base touched the object, a reward equal to REWARD_WIN was given.

Hyperparameters:

For the first tasks below are the hyperparameters that were used -

INPUT_WIDTH   64
INPUT_HEIGHT 64
OPTIMIZER "RMSprop"
LEARNING_RATE 0.04f

```
REPLAY_MEMORY 10000
BATCH_SIZE 32
USE_LSTM true
LSTM_SIZE 256
```

Here, the choice of parameters is explained. The input width and height were reduced to 64 in order to reduce the computational load and increase the accuracy. For higher values such as 512, the accuracy decreases to around 50% and the agent takes longer to train. A standard RMSprop optimizer was used. Replay memory was set to a high value of 10000. This increases the learning from previous experience. Use LSTM was set to true.The LSTM memory size was set to 256. Increasing this from 32 improved the overall accuracy as the agent now learns from the previous actions in a better way. Instead of using random previous action, LSTM stores the important learnings along the way. This helps agent in avoiding the costly actions in future. The batch size was experimented with values 8, 16, 32, 64,128 and 512. A batch size of 32 was found to be optimal. Similarly, learning rate was experimented from 0.01 to 0.1 and a value of 0.04 was found to be optimal.

For the second task, the hyperparameters were set as below.

```
INPUT_WIDTH   128
INPUT_HEIGHT  64
OPTIMIZER "Adam"
LEARNING_RATE 0.05f
REPLAY_MEMORY 20000
BATCH_SIZE 256
USE_LSTM true
LSTM_SIZE 256
```

Adam optimizer was used instead of RMSprop as it converges better. Also, a different height and width were used to improve the accuracy (more resolution was required along the width). Learning rate was reduced to 0.05 as compared to 0.1 in the first task. Also Replay memory was increased for better learning. Velocity control was used instead of position control.

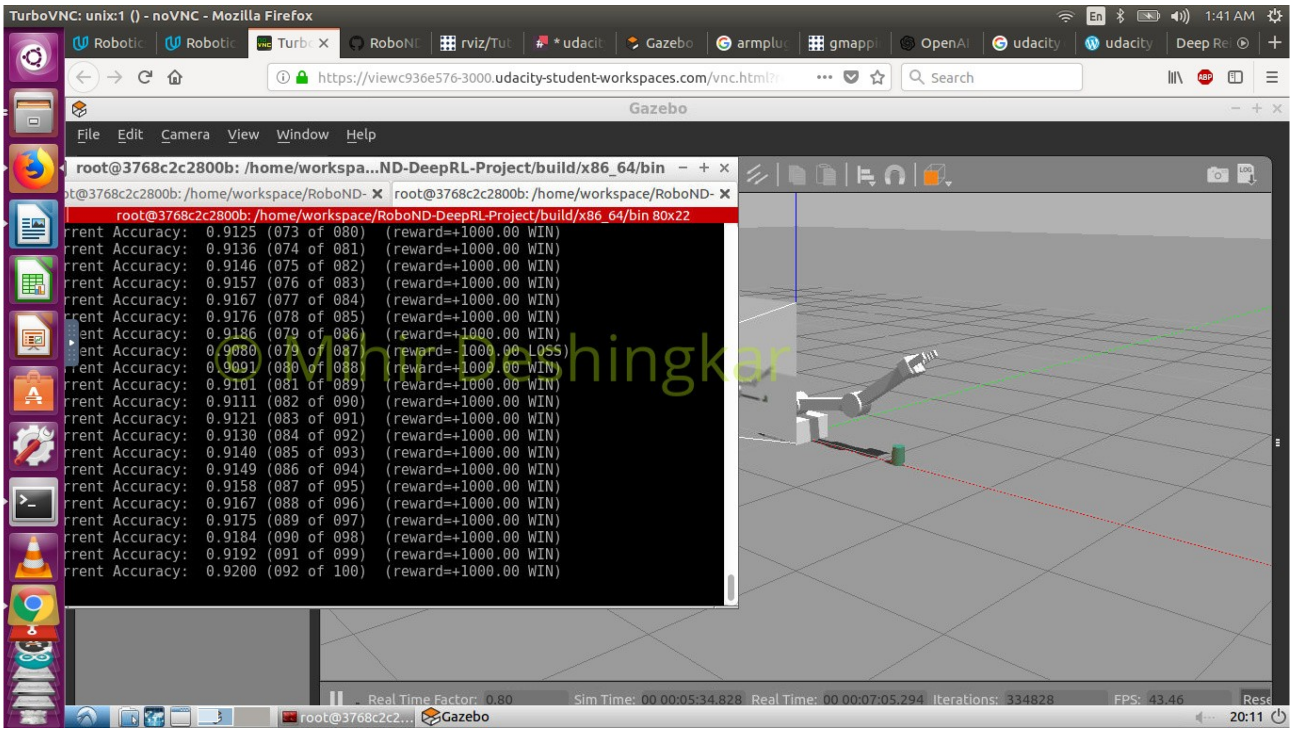
Results:

The images are included on the next page.

For the first objective, the agent successfully maintained an accuracy over 90%. The accuracy varied between 85% to 95% till completion of 100 runs. However, for the most part, the agent had an accuracy of about 92%. This indicated that the agent quickly learnt the objective and maintained a high accuracy. However, the agent still made mistakes in later stages, which means that agent is still exploring.

For the second objective, it took 1333 runs to reach an accuracy of 79.97%. Therefore, the agent took much longer time to achieve the required accuracy.

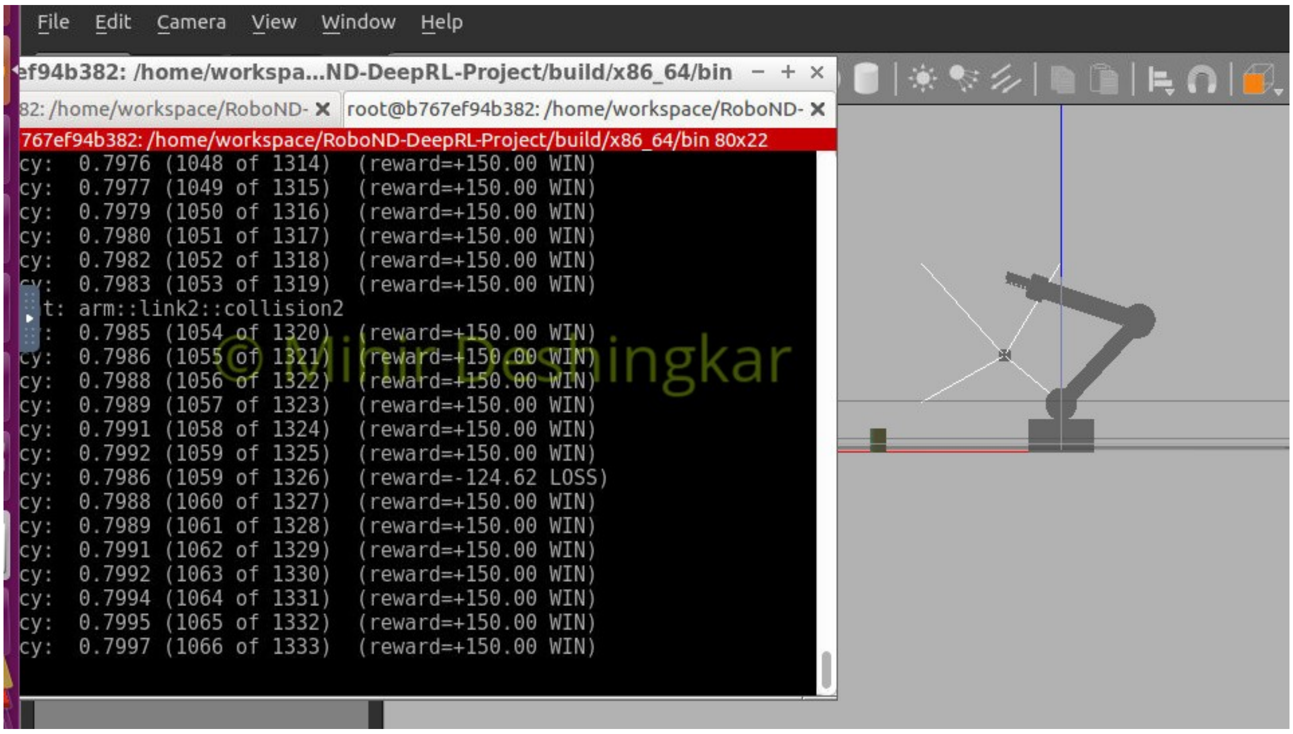
First objective -



```
root@3768c2c2800b: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x22
rrent Accuracy:  0.9125 (073 of 080)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9136 (074 of 081)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9146 (075 of 082)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9157 (076 of 083)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9167 (077 of 084)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9176 (078 of 085)  (reward=+1000.00 WIN)
ent Accuracy:  0.9186 (079 of 086)  (reward=+1000.00 WIN)
ent Accuracy:  0.9080 (079 of 087)  (reward=-1000.00 LOSS)
rent Accuracy:  0.9091 (080 of 088)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9101 (081 of 089)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9111 (082 of 090)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9121 (083 of 091)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9130 (084 of 092)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9140 (085 of 093)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9149 (086 of 094)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9158 (087 of 095)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9167 (088 of 096)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9175 (089 of 097)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9184 (090 of 098)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9192 (091 of 099)  (reward=+1000.00 WIN)
rrent Accuracy:  0.9200 (092 of 100)  (reward=+1000.00 WIN)
```

TurboVNC: unix:1 () - noVNC - Mozilla Firefox
https://viewc936e576-3000.udacity-student-workspaces.com/vnc.html
Gazebo
File Edit Camera View Window Help
Real Time Factor: 0.80 Sim Time: 00 00:05:34.828 Real Time: 00 00:07:05.294 Iterations: 334828 FPS: 43.46

Second objective -



```
767ef94b382: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x22
cy:  0.7976 (1048 of 1314)  (reward=+150.00 WIN)
cy:  0.7977 (1049 of 1315)  (reward=+150.00 WIN)
cy:  0.7979 (1050 of 1316)  (reward=+150.00 WIN)
cy:  0.7980 (1051 of 1317)  (reward=+150.00 WIN)
cy:  0.7982 (1052 of 1318)  (reward=+150.00 WIN)
cy:  0.7983 (1053 of 1319)  (reward=+150.00 WIN)
t: arm::link2::collision2
:  0.7985 (1054 of 1320)  (reward=+150.00 WIN)
cy:  0.7986 (1055 of 1321)  (reward=+150.00 WIN)
cy:  0.7988 (1056 of 1322)  (reward=+150.00 WIN)
cy:  0.7989 (1057 of 1323)  (reward=+150.00 WIN)
cy:  0.7991 (1058 of 1324)  (reward=+150.00 WIN)
cy:  0.7992 (1059 of 1325)  (reward=+150.00 WIN)
cy:  0.7986 (1059 of 1326)  (reward=-124.62 LOSS)
cy:  0.7988 (1060 of 1327)  (reward=+150.00 WIN)
cy:  0.7989 (1061 of 1328)  (reward=+150.00 WIN)
cy:  0.7991 (1062 of 1329)  (reward=+150.00 WIN)
cy:  0.7992 (1063 of 1330)  (reward=+150.00 WIN)
cy:  0.7994 (1064 of 1331)  (reward=+150.00 WIN)
cy:  0.7995 (1065 of 1332)  (reward=+150.00 WIN)
cy:  0.7997 (1066 of 1333)  (reward=+150.00 WIN)
```

File Edit Camera View Window Help
ef94b382: /home/workspa...ND-DeepRL-Project/build/x86_64/bin

Future Work:

The accuracy for the first objective can probably be increased by changing two main parameters. Firstly, increase the epsilon decay value so that agent expolores less and less in later runs. Secondly, increase the LSTM size so that agent does not make the same costly mistakes in the future. The arm movement can be smoothened and a better accuracy can be achieved with a better reward function for interim rewards, which takes into account not just the distance between the arm and the object, but also penalizes the arm if too much change in velocity is observed for each joints.

For the second objective, a different interim reward functions such as exponential function should be tried out. This can both improve the overall accuracy and decrease the time required to achieve the accuracy. Another function could be where one maintains the distance of gripper middle, gripper base and link 2 of the arm and give rewards accordingly.