

# RoboND DeepRL Project

Zerde Dilmuratkyzy

06/22/2018

## 1 Introduction

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% accuracy.
2. Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy.

## 2 Reward Functions

For this project, Deep Q-Network output is mapped to the control of each joint of the robotic arm. Robotic arm has 3 non-static joints, which can be controlled through velocity control or position control. Position control is chosen to complete the tasks, as it showed better results.

To train the robotic arm to reach each goal following reward functions were defined, for both tasks reward functions were the same, except for some changes in *Collision check* for the second task:

- *Ground contact* A negative reward of 100 was defined for the robot gripper hitting the ground. This action ends the episode.

```
bool checkGroundContact = (gripBBox.min.z <= groundContact)
|| (gripBBox.max.z <= groundContact);
if(checkGroundContact)
{
    if(DEBUG){printf("GROUND CONTACT, EOE\n");}

    rewardHistory = REWARD_LOSS;
    newReward      = true;
    endEpisode     = true;
}
```

- *Interim reward* An interim reward based on the distance between the arm and object was defined as a smoothed moving average of the delta of the distance to the goal, which is calculated as  $average\_delta = (average\_delta * alpha) + (dist * (1 - alpha))$ .  $alpha$  is set to 0.2 .

```
// compute the reward from distance to the goal
const float distGoal = BoxDistance(gripBBox, propBBox);

if( episodeFrames > 1 )
{
    const float distDelta = lastGoalDistance - distGoal;

    // compute the smoothed moving average of the delta of the distance to the goal
    avgGoalDelta = (avgGoalDelta * alpha) + (distDelta * (1 - alpha));
```

```

rewardHistory = avgGoalDelta;
newReward     = true;
    }
}

```

```

lastGoalDistance = distGoal;

```

- *Collision with object* for the first task a positive reward of 1000 is issued when collision between any part of robot arm and object happens. After that episode is ended.

```

if ((strcmp(contacts->contact(i).collision1().c_str(), COLLISION_ITEM ) == 0))
{
rewardHistory = REWARD_WIN;
newReward     = true;
endEpisode    = true;

return;
}

```

For the second task, positive reward of 1000 is given only if the gripper base touches the object, this action ends the episode. And if any other part of the robot arm touches the object negative reward of 10 is issued.

```

if ((strcmp(contacts->contact(i).collision1().c_str(), COLLISION_ITEM ) == 0) &&
(strcmp(contacts->contact(i).collision2().c_str(), COLLISION_POINT) == 0))
{
rewardHistory = REWARD_WIN;
newReward     = true;
endEpisode    = true;

return;
}else //collision with other part
{
    rewardHistory = REWARD_LOSS / 10;
    newReward     = true;
    endEpisode    = false;
}

```

- *Episode timeout* a negative reward of 100 will be issued if frame count exceeds maxEpisodeLength (100), and episode will be ended

```

if( maxEpisodeLength > 0 && episodeFrames > maxEpisodeLength )
{
rewardHistory = REWARD_LOSS;
newReward     = true;
endEpisode    = true;
}

```

### 3 Hyperparameters

Following parameters were tuned in order to achieve objectives of the project:

- **INPUT\_WIDTH, INPUT\_HEIGHT** were reduced to 64x64 as it is enough to perform training and doesn't consume as much memory and computing power as original size 512x512 did.
- **NUM\_ACTIONS** since robot arm had 3 joints and each of them could move in two directions, number of action was set to  $2*3 = 6$ .

- **OPTIMIZER** two optimizer were tested, RMSprop and Adam and it produced similar results, RMSprop was used in both tasks to achieve required results.
- **LEARNING\_RATE** was set to 0.01, when tested with higher values like 0.1-0.3, robot arm did not do well on improving its performance. 0.01 showed the best result.
- **REPLAY\_MEMORY** was set to 20000 to match the **BATCH\_SIZE** which was set to 64, so that there is enough stored information for the model to train on. Different sizes of **BATCH\_SIZE** were tested and 64 was selected at the end, as it was found that it is the best fit based on computer performance.
- **USE\_LSTM**, **LSTM\_SIZE** LSTM was enabled and LSTM\_SIZE was set to 256, as it was enough to reach the required target.
- **EPS\_END** was reduced to 0.1 so that at the end robot will make less random moves, and make decisions based on previously obtained results.

Hyperparameters were set to same values for both task 1 and task 2.

## 4 Results

Before tuning the hyper-parameters and setting the right reward system, DQN network did not perform well enough to reach the objectives, accuracy was very low, and in many cases accuracy kept decreasing with each iteration. But after the tuning of the parameters described in above section it was possible to reach both task's objectives. Above 90% accuracy was reached for both tasks and the accuracy was still increasing.

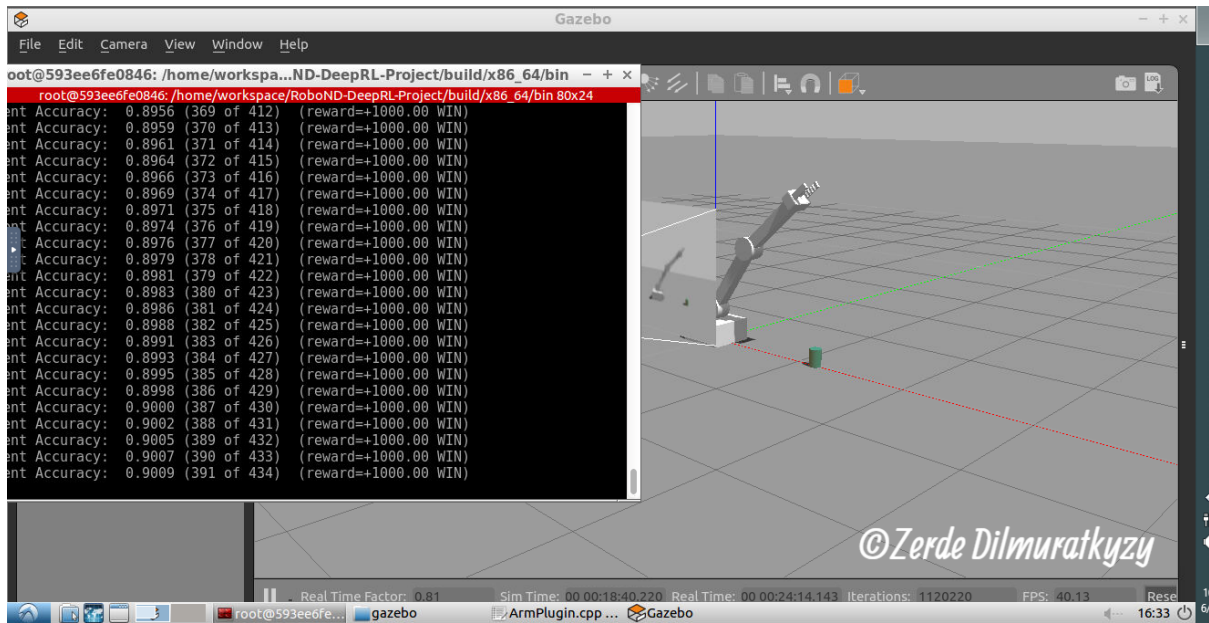


Figure 1: Task 1. Robot Arm touching the object

Video of robot gripper touching the object can be found in /docs/videos folder.

## 5 Future Work

For the future work, other types of motion control can be explored, velocity control for smoother movement and maybe trying out combining the two methods in some way. Further tuning the hyper param-

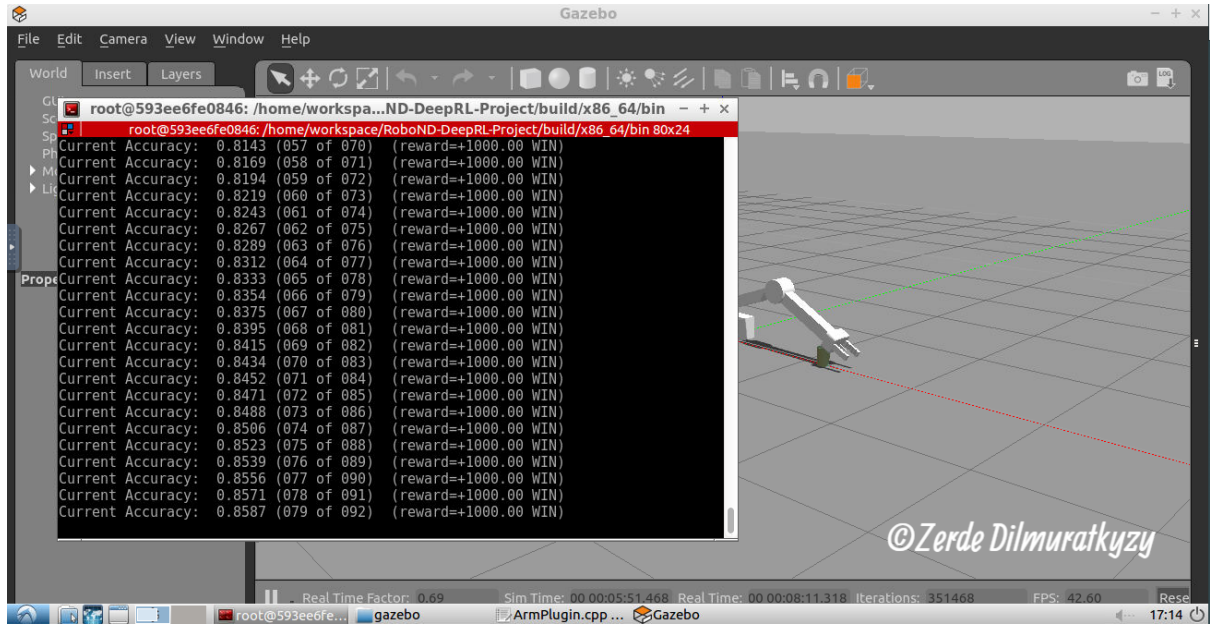


Figure 2: Task 2. Robot Gripper touching the object

eters and rewarding strategies can be considered to achieve the goal within smaller iterations. And to deepen the understanding of how RL works, and how rewarding systems and parameters can really affect agent's performance different analyzing methods or tools can be tested out, like building a graph using debug output informations for better visualization. Furthermore trying out solving additional project challenges and perhaps adding gripper control so that robotic arm can learn to do actions like grabbing and releasing the object would be the next logical steps.