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:

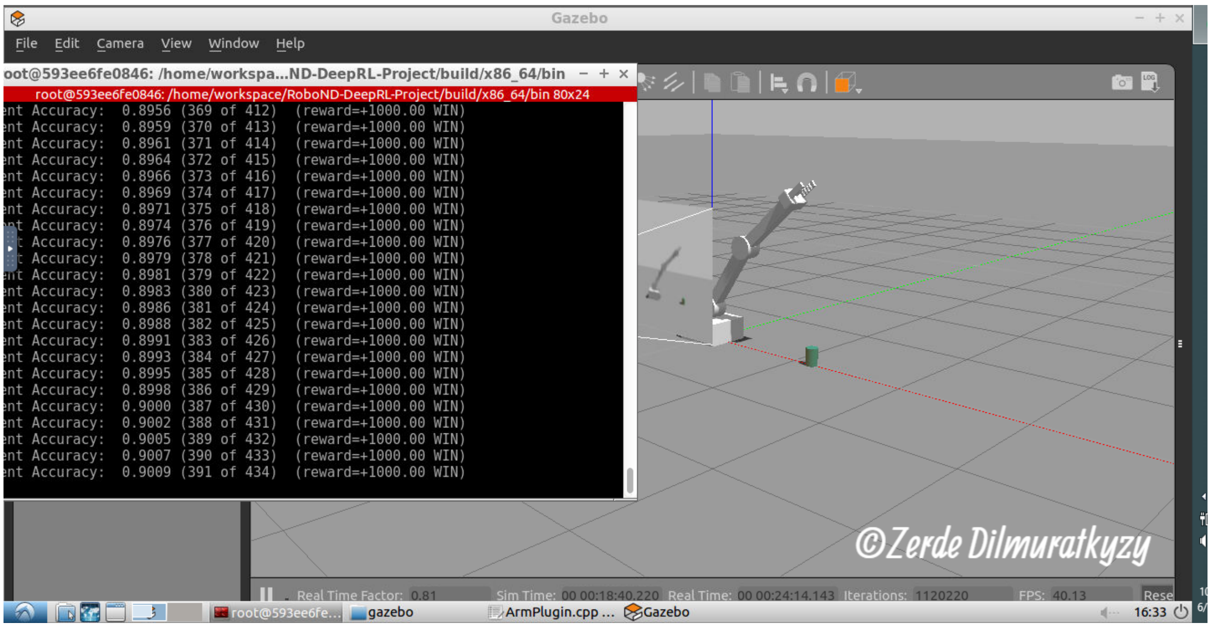
- **INPUT_ WIDTH, INPUT_HEIGHT** were reduced to 64x64 as it is enough to perform training and doesn't consume as much memory and computing power as original size 512x512 did.
- **NUM_ACTIONS** since robot arm had 3 joints and each of them could move in two directions, number of action was set to 2*3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


Gazebo
File Edit Camera View Window Help
oot@593ee6fe0846: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@593ee6fe0846: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
©Zerde Dilmuratkyzy
Real Time Factor: 0.81
Sim Time: 00 00:18:40.220
Real Time: 00 00:24:14.143
Iterations: 1120220
FPS: 40.13
Rese
root@593ee6fe...
gazebo
ArmPlugin.cpp ...
Gazebo
16:33

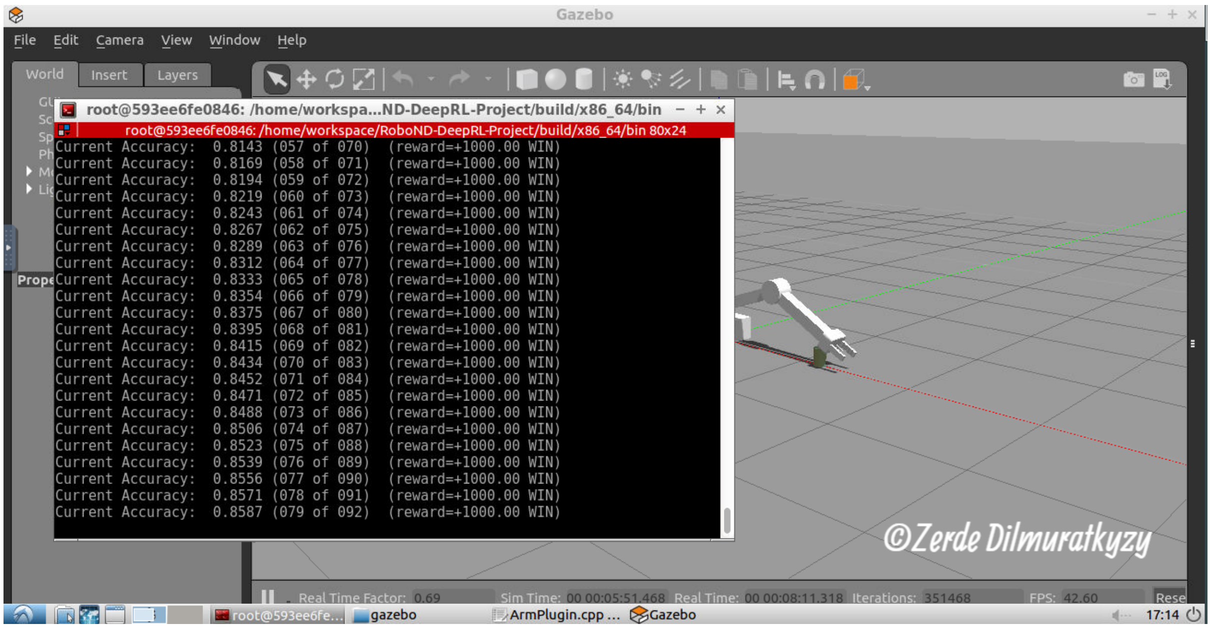
```
ent Accuracy:  0.8956 (369 of 412)  (reward=+1000.00 WIN)
ent Accuracy:  0.8959 (370 of 413)  (reward=+1000.00 WIN)
ent Accuracy:  0.8961 (371 of 414)  (reward=+1000.00 WIN)
ent Accuracy:  0.8964 (372 of 415)  (reward=+1000.00 WIN)
ent Accuracy:  0.8966 (373 of 416)  (reward=+1000.00 WIN)
ent Accuracy:  0.8969 (374 of 417)  (reward=+1000.00 WIN)
ent Accuracy:  0.8971 (375 of 418)  (reward=+1000.00 WIN)
nt Accuracy:  0.8974 (376 of 419)  (reward=+1000.00 WIN)
t Accuracy:  0.8976 (377 of 420)  (reward=+1000.00 WIN)
t Accuracy:  0.8979 (378 of 421)  (reward=+1000.00 WIN)
ent Accuracy:  0.8981 (379 of 422)  (reward=+1000.00 WIN)
ent Accuracy:  0.8983 (380 of 423)  (reward=+1000.00 WIN)
ent Accuracy:  0.8986 (381 of 424)  (reward=+1000.00 WIN)
ent Accuracy:  0.8988 (382 of 425)  (reward=+1000.00 WIN)
ent Accuracy:  0.8991 (383 of 426)  (reward=+1000.00 WIN)
ent Accuracy:  0.8993 (384 of 427)  (reward=+1000.00 WIN)
ent Accuracy:  0.8995 (385 of 428)  (reward=+1000.00 WIN)
ent Accuracy:  0.8998 (386 of 429)  (reward=+1000.00 WIN)
ent Accuracy:  0.9000 (387 of 430)  (reward=+1000.00 WIN)
ent Accuracy:  0.9002 (388 of 431)  (reward=+1000.00 WIN)
ent Accuracy:  0.9005 (389 of 432)  (reward=+1000.00 WIN)
ent Accuracy:  0.9007 (390 of 433)  (reward=+1000.00 WIN)
ent Accuracy:  0.9009 (391 of 434)  (reward=+1000.00 WIN)
```

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-


Gazebo
File Edit Camera View Window Help
World Insert Layers
©Zerde Dilmuratkyzy
Real Time Factor: 0.69 Sim Time: 00 00:05:51.468 Real Time: 00 00:08:11.318 Iterations: 351468 FPS: 42.60
root@593ee6fe... gazebo ArmPlugin.cpp ... Gazebo 17:14

```
root@593ee6fe0846: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@593ee6fe0846: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Current Accuracy:  0.8143 (057 of 070)  (reward=+1000.00 WIN)
Current Accuracy:  0.8169 (058 of 071)  (reward=+1000.00 WIN)
Current Accuracy:  0.8194 (059 of 072)  (reward=+1000.00 WIN)
Current Accuracy:  0.8219 (060 of 073)  (reward=+1000.00 WIN)
Current Accuracy:  0.8243 (061 of 074)  (reward=+1000.00 WIN)
Current Accuracy:  0.8267 (062 of 075)  (reward=+1000.00 WIN)
Current Accuracy:  0.8289 (063 of 076)  (reward=+1000.00 WIN)
Current Accuracy:  0.8312 (064 of 077)  (reward=+1000.00 WIN)
Current Accuracy:  0.8333 (065 of 078)  (reward=+1000.00 WIN)
Current Accuracy:  0.8354 (066 of 079)  (reward=+1000.00 WIN)
Current Accuracy:  0.8375 (067 of 080)  (reward=+1000.00 WIN)
Current Accuracy:  0.8395 (068 of 081)  (reward=+1000.00 WIN)
Current Accuracy:  0.8415 (069 of 082)  (reward=+1000.00 WIN)
Current Accuracy:  0.8434 (070 of 083)  (reward=+1000.00 WIN)
Current Accuracy:  0.8452 (071 of 084)  (reward=+1000.00 WIN)
Current Accuracy:  0.8471 (072 of 085)  (reward=+1000.00 WIN)
Current Accuracy:  0.8488 (073 of 086)  (reward=+1000.00 WIN)
Current Accuracy:  0.8506 (074 of 087)  (reward=+1000.00 WIN)
Current Accuracy:  0.8523 (075 of 088)  (reward=+1000.00 WIN)
Current Accuracy:  0.8539 (076 of 089)  (reward=+1000.00 WIN)
Current Accuracy:  0.8556 (077 of 090)  (reward=+1000.00 WIN)
Current Accuracy:  0.8571 (078 of 091)  (reward=+1000.00 WIN)
Current Accuracy:  0.8587 (079 of 092)  (reward=+1000.00 WIN)
```

Figure 2: Task 2. Robot Gripper touching the object

eters to achive the goal within smaller iterations, and trying out solving additional project challenges would be the next steps.