VERSION [1.0]

APRIL 21, 2019

RoboND-DeepRL-Project

INTRODUCTION

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

Both of these objectives, have associated tasks that you will complete while working on this project. The upcoming sections will cover these tasks in detail.

This project is implemented in Gazebo environment to simulate Reinforcement Learning with DQN agent using Pytorch library.

In this project robotic arm agent is trained to reach the goal object by defining the reward function and control approaches.

Projects tasks:

- Subscribe to camera and collision topics
- Create the DQN Agent
- Choosing velocity or position-based control of arm joints
- Creating reward function:
    - Reward for robot gripper hitting the ground
    - Issue an interim reward based on the distance to the object
    - Issue a reward based on collision between the arm and the object
- Tuning the hyperparameters

REWARD FUNCTION

In Any RL project, Reward function must be clearly defined to cover the required needs.

In this project, Reward function has three parts:

1-ROBOT GRIPPER HITTING THE GROUND

In first objective Robot will get reward of -9000.

In second objective hitting ground reward was -10500 to avoid hitting the ground as possible when approaching to cylindrical object.

2-ROBOT TOUCHES THE GOAL OBJECT

Robot will get reward of 10000 and episode will end.

3-ROBOT FAILS TO REACH OBJECT AND REACHES TIMEOUT

- In first objective Robot will get reward of -500 and episode will end.
- In second objective Robot will get reward of -1000 and episode will end.

4-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.

avgGoalDelta = (avgGoalDelta * ALPHA) + (distDelta * (1 - ALPHA));

HYPERPARAMETERS

INPUT_WIDTH, INPUT_HEIGHT

Tuned values :64 ,64

reasons:

1- project environment doesn't have much details or complicated objects that required high definition input images.
2- to minimize required process power and memory needed.

OPTIMIZER

Tuned values: "RMSprop"

There are several available options like RMSprop, Adam and Adagrad for gradient decent optimizers.

RMSprop only was tested because results were quite sufficient in both objectives when used.

LEARNING_RATE

Tuned values: 0.2, 0.1

1-for first objective 0.1,.05,0.2 was tested and most suitable one to reach the required accuracy was 0.2 to avoid missing times and makes learning faster.
2-for second objective allowed collision area was narrow so robot learning isn't easy like first case and to avoid overfitting 0.1f value was used

REPLAY MEMORY

Tuned values: 10000

It is the memory buffer that is used to store previous states to be used in training later

The more memory allocated for REPLY MEMORY the more the robot will learn from past values too.

In this case values of 100,1000,10000 was tested but the most efficient one was the default value 10000.

BATCH SIZE

Tuned values: 32 ,128

Determines date that will be used in training DQN network increasing DQN efficiency.

Value of 32 was used in first objective and 128 for second one.

32,64,16,128 was tested and the most optimum results was when value =128.

The Only issue here is that high values will increase memory usage.

USE LSTM

Tuned value: true

Enables long short-term memory which Makes robot keep track of both, the long-term memory and the short-term memory.

LSTM_SIZE

Tuned value: 256

Size of each LSTM cell

EPS_END

Tuned value: 0.01f

Was reduced to 0.01f to reduce exploration which makes robot's moves highly based on previous results and decrease random moves resulting accuracy increase.

RESULTS

FIRST OBJECTIVE

At the beginning before tuning parameters results wasn't sloppy and inefficient to meet requirements and robot wasn't learning too much and, in some cases, accuracy ends up decreasing or robot holds in place without moving.

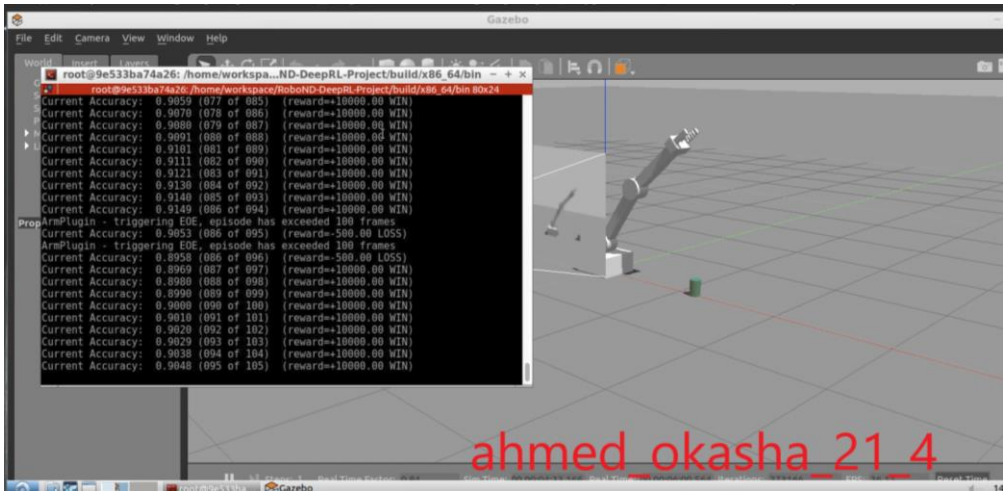
In addition to these problems after tuning parameter robots seems to be learning well until distDelta =0(gripper middle reaches object) and here the problem happens because gripper middle isn't in the collision check object passes through gripper fingers so robot collision is false resulting time out or ground hit after it.

To solve this problem GRIP_NAME was changed to check on gripperbase not gripper_middle resulting distDelta to be between gripper base and object.

Noticeable changes in result happened when increasing the reward of touching object and the punishment when hitting the ground more than timeout punishment.

Also, by increasing learning rate to 0.2 and decreasing random moves robot results become neater and more accurate.

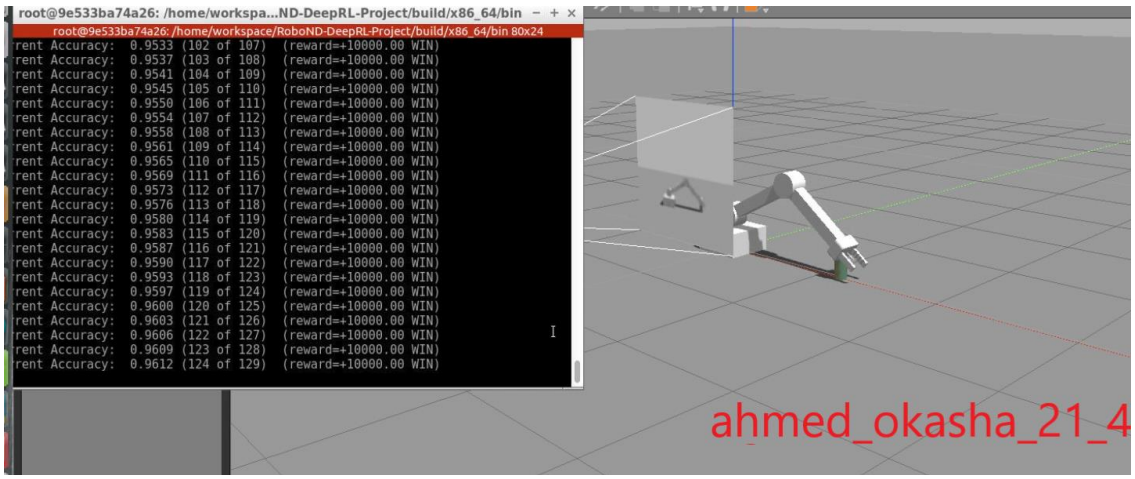
Results video is enclosed in the project files.



SECOND OBJECTIVE

The only changes made to previous objective was to decrease learning rate to 0.1, also batch size was increased to 128 to make DQN deeper and more accurate which increased successful hits to goal object by gripper only.

Strange thing happened her is this objective accuracy is higher than first objective perhaps this can be due to the high batch size or because in this case robot final position is far a bit from ground than previous case resulting less ground hits.



CONTROL MODE

First, I tested but I had some errors and couldn't solve it perhaps the error was in this part or the problem due to the parameters that needed to be tuned at that time. but at the end I changed to position control and it worked fine in both objectives so I left it without trying the other one again.

FUTURE WORK

1- changing position along x axis challenge will be useful to practice more on RL which will increase the understanding by adding more complexity to robot environment.
2- to go further enabling revolving base joint will increase available actions thus increasing optimization that will be needed.
3- Finally, I think it will be more challenging to make this problem general and make goal position is random in xy position but in this case normal cameral should be replaced with range finder laser or RGBD camera.