Deep RL Arm Manipulation

Gaurav Saxena

Abstract—This paper present approach to control a robotic arm with grip to pick up a object using RL Deep Q Network for training. Using Gazibo enviroment.

Index Terms—Robot, ROS, RL, Deep Q Network

1 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
2) Have only the gripper base of the robot arm touch the object, with at least a 80

Will be using a Gazebo simulated environment with a plugin written in C++ acting as a interface between simulated robot in Gazebo environment and the DQN agent with following task to be completed.

- Subscribe to camera and collision topics
- Create the DQN Agent and passing the required parameters.
- Decide to use Velocity or Position control and the provide the needed value for increment.
- 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
- Issue a reward based on collision between the arms gripper base and the object.

2 Reward Functions

As discussed in the project guideline we have to provide the reward to the RL system at different episodes:

1) Reward for robot gripper hitting the ground: If the robot gripper hit the ground the reward is set to the value defined in REWARD_LOSS and the episode is ended.
2) Issue an interim reward based on the distance to the object: Using this function, calculate the distance between the arm and the object. Then, use this distance to calculate an appropriate reward as well. average_delta = (average_delta * alpha) + (dist * (1 - alpha));
3) Episode Timeout: Once the episode has time out the reward is set to the value defined in REWARD_LOSS.

3 Hyperparameters

Below is the list of Hyperparameters with configured values reached for successful achievement of the goals:

- INPUT_WIDTH x INPUT_HEIGHT: Default values were 512 x 512 which are quite high in memory utilization so the values were changes to 64 x 64 and provided satisfactory results.
- OPTIMIZER : Used RMSprop as provided the value in c/C++ API. Other available optimizers are Adam, Adadelta, RMSprop etc.
- LEARNING_RATE : Learning rate is kept as it is to 0.1. item BATCH_SIZE : Batch size is kept to 32. Which gave good results.
- REPLAY_MEMORY: Replay memory is kept to 1000. With batch size 32 we maintained 312 stage(REPLAY_MEMORY
  BATCH_SIZE).
- USE_LSTM : kept value as true. So that DQN network will allow training the network by taking into consideration past frames.
- LSTM_SIZE : Size of each cell. Having larger value will result in more computing power. So tried with 256 and received satisfactory results.
- REWARD_WIN : Kept it as 500.
- REWARD_LOSS : Kept it as -500.
- REWARD_MULT : Kept it as 10.
- alpha : Kept as 0.9

4 Results

Running the solution without fine tuning any parameters resulted in DQN network having non success. But after fine tuning the parameters with multiple tries reached success rate of 90After changing objective from ”arm touching the object” to ”gripper base touching the object” only changed the learning rate to 0.01 and the DQn performance was very good and consistent.

5 Conclusion / Future work

This project provided hand-on experience Deep RL using DNQ network and how to fine tune the hyper parameters. Going forward we can set up a simple robotics arm with two motors controlled by Arduino and a camera to take the images and deploy the DNQ network on Jetson nano to run it there.

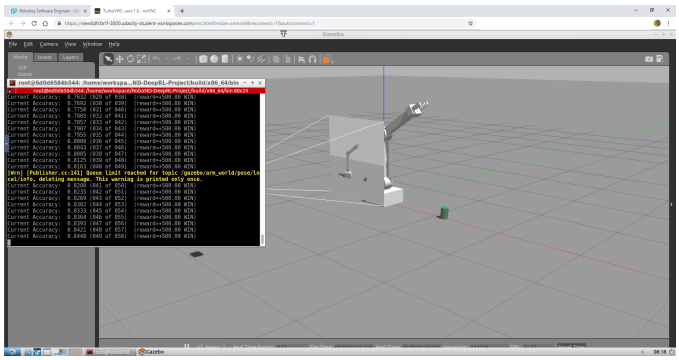

Fig. 1. Initial Run

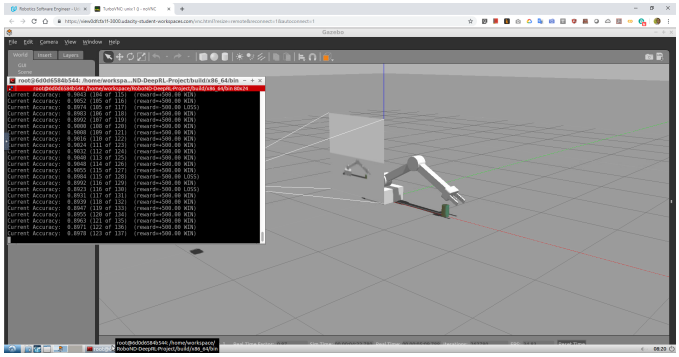

Fig. 2. Robot arm touching the object

REFERENCES