Deep RL Arm Manipulation Project – Udacity: Deep Reinforcement Learning

Introduction :

This project aims to create a Deep Q-Learning Network (DQN) and, with it, train a robotic arm to meet certain objectives. The Robotic Arm used for this project is simulated on Gazebo and run and trains (learns) on the Nvidia Jetson TX2 Deep Reinforcement Learning platform. This project mainly leverages an existing DQN that gets instantiated with specific parameters to run the Robotic Arm. For Deep Reinforcement Learning to occur, reward functions and hyper parameters must be defined. To test the capabilities of DQN, two objectives were established: a. Have any part of the robot arm touch the object of interest, with at least a 90% accuracy for a minimum of 100 runs. b. Have only the gripper base of the robot arm touch the object, with at least an 80% accuracy for a minimum of 100 runs.

Reward Functions:

To achieve both objectives, rewards functions were configured slightly different as you'll see in the following.

2.1. Objective 1

For this task the robotic arm collide with the target object at any point, the approach taken was to use "velocity control" instead of position. To control velocity, a simple check was performed to observe whether the action was even or odd. Depending on the action check, actionVelDelta w However, to hone in on velocity control, the reward was issued as follows: if (avgGoalDelta > 0) { if(distGoal > 0.001f){ rewardHistory = REWARD_WIN / distGoal; } else if (distGoal < 0.001f || distGoal == 0.0f){ rewardHistory = REWARD_WIN / 2000.0f; } } This approach encourages the arm to quickly reach the goal by increasing the reward (inversely proportional) as it gets close to the goal. However, once this arm gets to a certain point (close enough), the reward stops increasing and only issues a small reward, which should encourage the arm to slow down as it reaches the target.

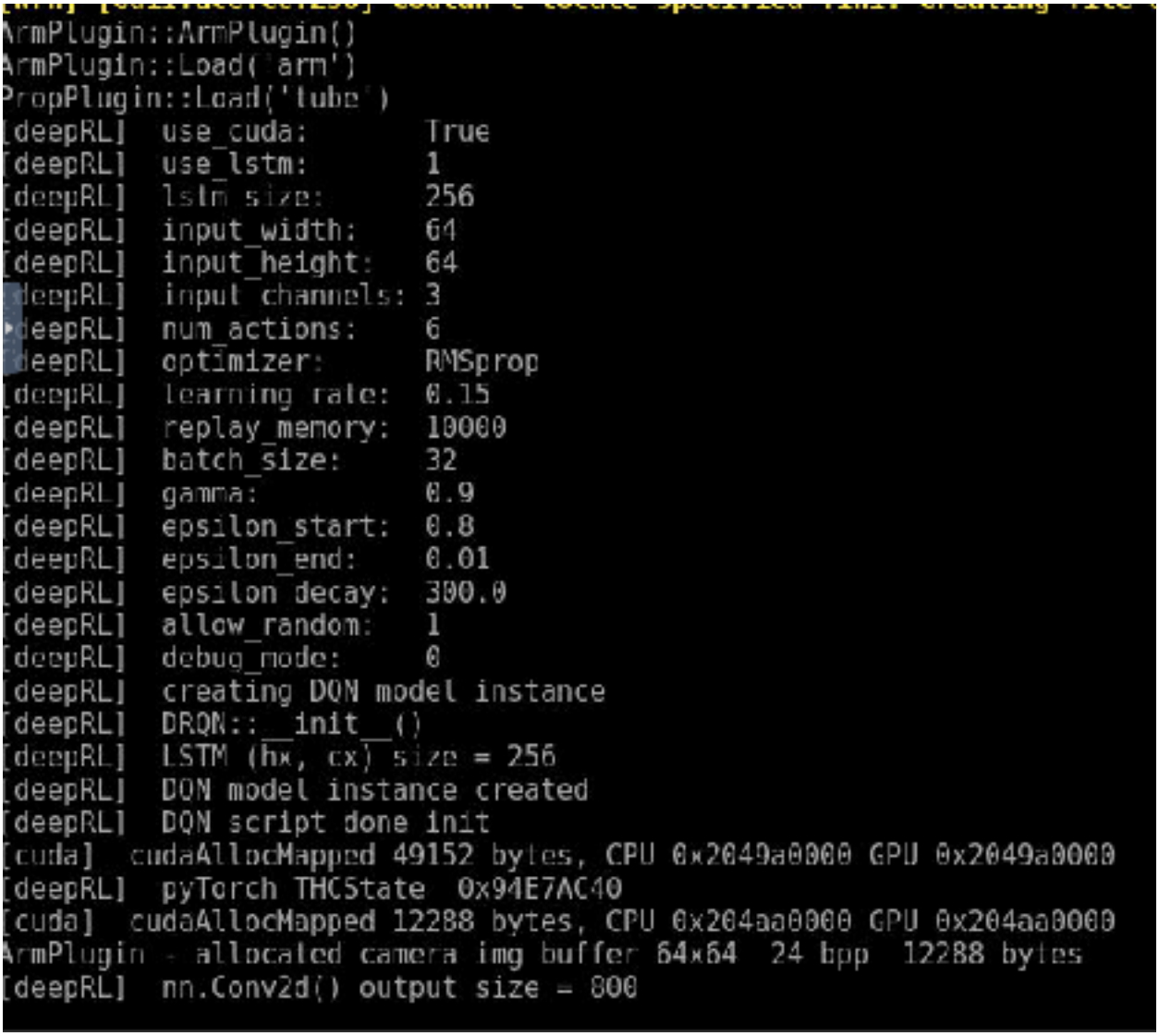
2.2.Objective 2

In this task have the robotic arm collide with the target object only with the gripper, velocity control was initially tested. After many attempts, it was decided to use "position control" instead of velocity. Since touching the target solely with the gripper, more finesse and joint control seemed to require more emphasis. To control position, a simple check was performed to observe whether the action was even or odd. Depending on the action check, actionJointDelta was either added or subtracted. The robotic arm must never collide with the ground, therefore, a REWARD_LOSS of 1 * 20 will be issued upon ground collision. However, if the gripper

collided with the target object, it would issue a REWARD_WIN of 1 * 100, meeting the objective and resulting in a “win” for that run. For this objective, the emphasis was to get it to touch with the gripper (solely), which is why the reward is higher than for the previous objective. Also, another limitation was added, to encourage the arm to accomplish its objective before the episode was over (frame span). If the run exceeded the maximum number of frames per episode (maxEpisodeLength), which was set to 100, a REWARD_LOSS of 1 would be issued. This would result in a “loss” for that run. Although the objective is still to collide with the target within one run/episode, some emphasis was taken away from this limitation given that velocity control has been replaced by position control. However, since position control will be used to achieve objective 2, the reward function setup for the process will be different this time around. As part of the generic solution, the average delta (avgGoalDelta) was calculated just as stated in the tasks lesson (the same as for objective 1): avgGoalDelta = (avgGoalDelta * alpha) + (distDelta * (1.0f - alpha)); However, this time around instead of using the distance from the goal (distGoal) to drive the rewards, the smoothed moving average will be used, as follows: if (avgGoalDelta > 0) { if(distGoal > 0.0f){ rewardHistory = REWARD_WIN * avgGoalDelta; } else if (distGoal == 0.0f){ rewardHistory = REWARD_WIN * 10.0f; } } else { rewardHistory = REWARD_LOSS * distGoal; } This approach focuses on the movement, encouraging the robotic arm to move more. The more it moves towards the goal, the more rewards it will get (proportional to avgGoalDelta). However, once the arm at 0 distance away from the it would provide an extra reward. Furthermore, if not moving towards the goal, it would issue REWARD_LOSS proportional to the distance to the goal (distGoal). At least that’s the theory behind this approach.

3 Hyperparameters

List of hyperparameters and corresponding descriptions are shown in the below picture.

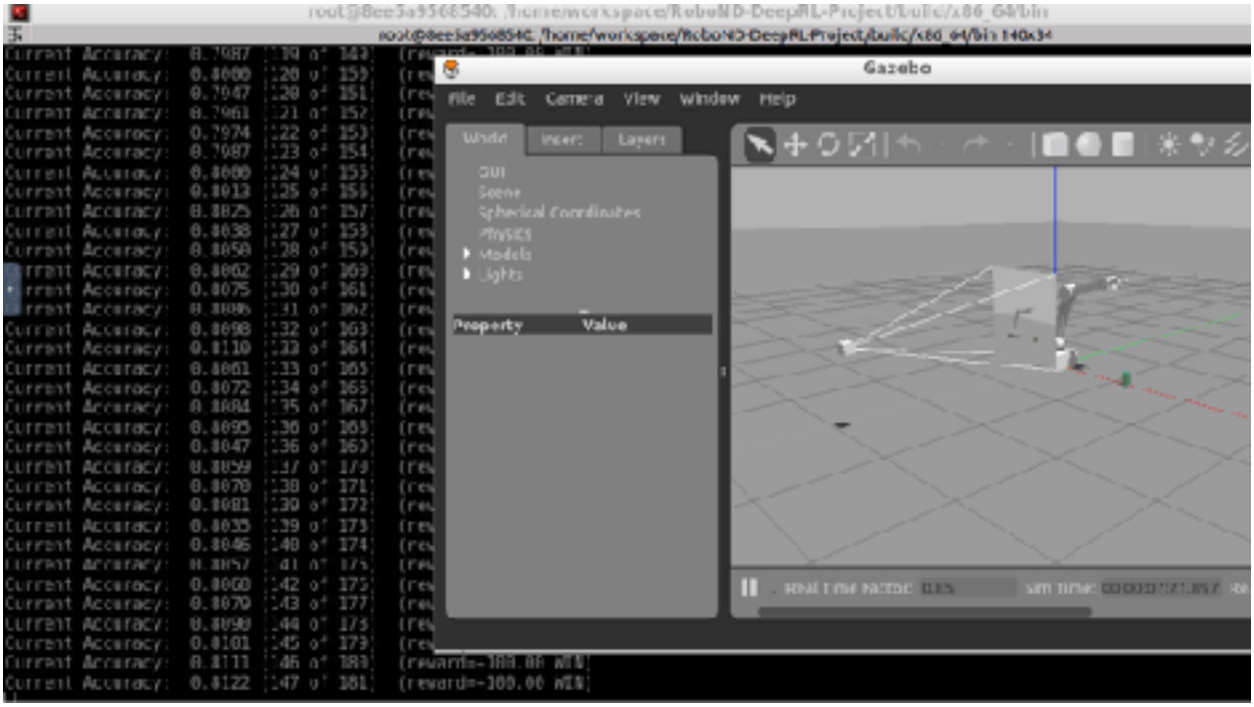


4. Results

Objective 1

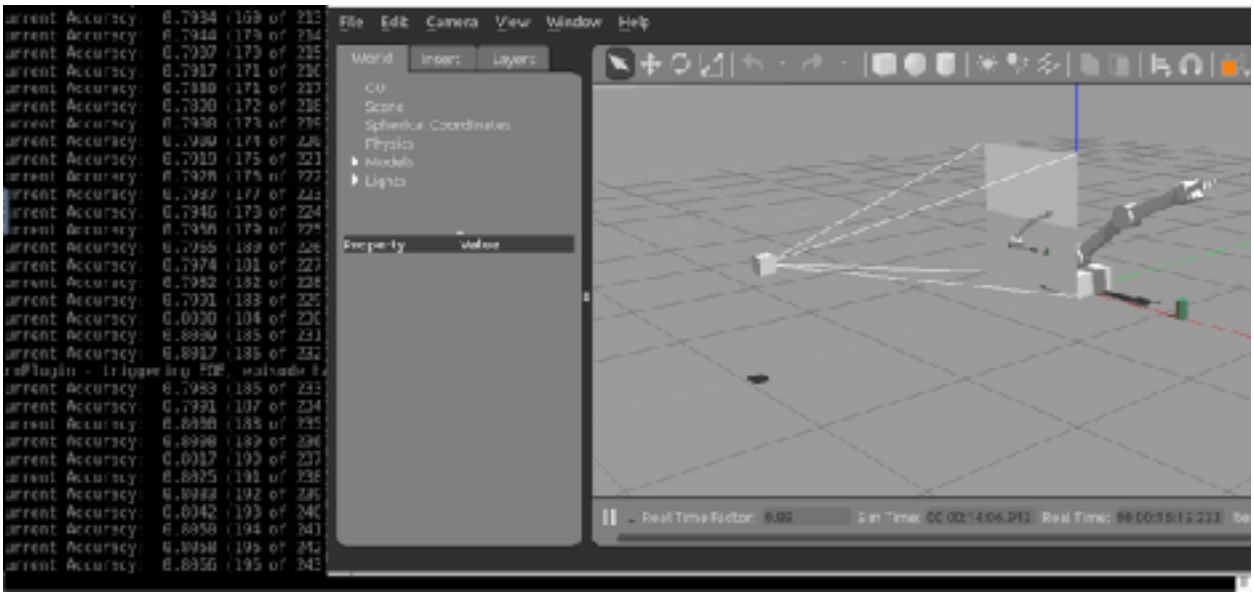
This was relatively simple to train, the robotic arm intuitively started learning better as hyperparameters were tuned and the reward values were increased. At almost each iteration of testing, at the beginning before learning the objective, the arm started colliding with the ground, seeming to break multiple times. Increasing the negative rewards for ground collisions didn't

dramatically improve this behavior. This could definitely be a problem for real-life solution. However, this behavior did not happen as much on the last iteration, and learned fairly quickly, which is why it became the final version for this objective.



4.2. Objective 2

Normally, for objective 2, required more accuracy in its behavior. This was dramatically harder to train than objective 1. After iterating and testing many times, by tuning hyperparameters, it was noticed that the agent was trying to explore too much. However, there are so many approaches that can be taken to touch the target with just one point. For that reason, the DQN API Settings were changed to decrease exploration slightly compared to objective 1. This helped on teaching the robotic arm agent to perform a solution approach quicker.



Also, the fact that the learning rate was decreased and the batch size increased, seemed to have helped the agent to train and learn deeper than objective 1. However, as with objective 1, the agent was also displayed hitting the ground hard enough to sometimes break, at least in the initial stages of the iteration.As expected, the second objective was harder to reach, as training the robotic arm to have finesse and be more precise, required finer tuning of hyperparameters.

5 Future Work

There were clear arcs that once found achieved a win quickly. Such that it would be worthwhile investigating an interim reward system based on not just the distance from the goal but also distance from an ideal arc trajectory as the arm approached.

Further using centre points to calculate distance from goals becomes less accurate the closer to the goal the arm is. Such that other points like the end of the gripper_middle and top of prop cylinder, would be worthwhile experimenting with.