Marko Sarkanj
Udacity Robotics Nanodegree

Deep Reinforcement Learning Robotic Arm Manipulation Writeup

The task of this project was controlling robotic arm with three joints in the Gazebo simulator with help of the Deep Q Network. That has been implemented in C++ programming language by accessing DQN API with the help of dqnAgent class.

Reward Functions

The first reward function defined was negative reward that happens when the gripper hits the ground. There is negative reward of 100 when the gripper's z coordinate reaches bellow or equal to the height of 0.05 . This action cancels the whole episode, as it sets the parameter "endEpisode" to true.

The second reward function defined was the interim reward based on the gripper's distance to the object. This reward function simply measures current and previous distances to the object and gives the positive reward if the distance is getting smaller and the negative reward if the distance is getting larger. There is one smoothing parameter "ALPHA" that smooths the reward over multiple steps, that parameter will be described in the next section.

The third reward defined was the collision reward function. Two positive rewards are the object's collision with the robot's gripper base reward and the object's collision with the robot's tube(arm) reward. The later reward gets activated only if the "GRIPPER_ONLY" parameter is set to "false". Both rewards are 10000 points.

Joint Control

Based on the experimentation, the control of joint's position has been chosen over the control of joint's velocity ("VELOCITY_CONTROL" parameter has been set to false).

Hyperparameters

The first two hyperparameters defined where the input width and the input height. Both values were set to 64 as this is the image format sufficient to perform the training that doesn't consume too much memory resources.

The second parameter defined was the number of possible actions "NUMBER_OF_ACTIONS". This parameter has been set to 2 * degrees of freedom (DOF). There are 3 DOFs in this case and they have been multiplied by 2 as each joint can move in two opposite directions.

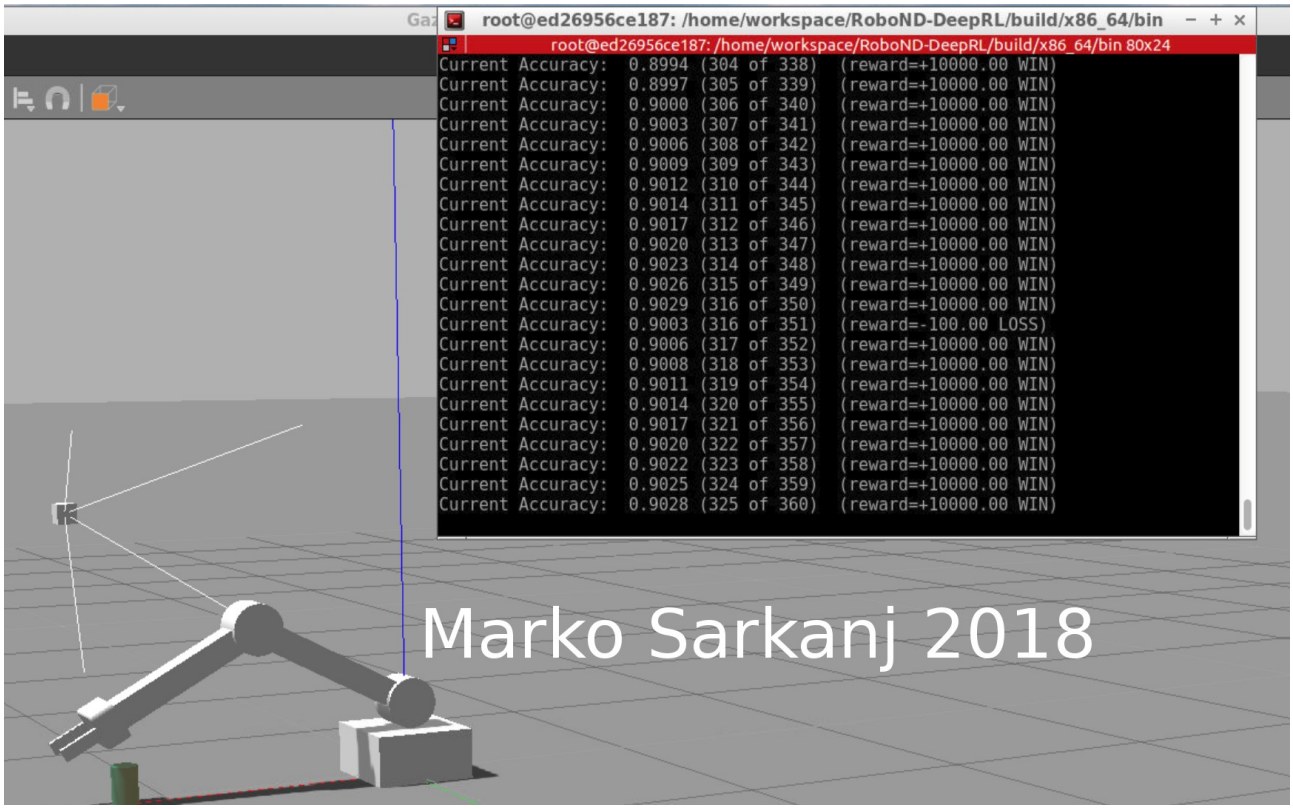
RMSprop has been set as optimizer as it outperformed the Adam optimizer based on experimentation. Learning rate of 0.01 with the batch size of 64 proved to perform best.

Replay memory has been set to 20000, so that there is enough information for the model to train on. LSTM network has been used as well, with the LSTM size set to 128.

The last hyperparameter defined was the "ALPHA" parameter that determined smoothing of the interim reward based on the gripper's distance to the object over multiple steps. This hyperparameter has been set to 0.2 . 20% of the reward is the value from the past rewards and 80% is the current reward.

Results

The RL agent was able to touch the object with the gripper base with over 90% of accuracy over the 340 episodes. Based on this results only one training session has been performed as there is no need to further improve the accuracy by touching the object with the robot's tube as well.



Final results of the RL agent's training (gripper base only)

Advantages/Disadvantages of the RL agent

One of the disadvantages of the agent is that it requires relatively high number of episodes to achieve the accuracy of 90%. The accuracy of 90% is insufficient for many of the practical applications where the agent could be used. If the agent would be trained to grip the object and position it on some desired location the problem would be much more complex and the accuracy of 90% would be much harder to achieve.

The advantage of the agent is that it is able to perform the training without any supervision. It is possible to change the object's location and the agent will be able to adapt to changes after certain amount of episodes.

Improvements or additions

The results could be improved by using a RGB-D camera and sending more information to the RL agent. Another addition would be mounting the camera directly on the gripper's base, to increase the robot's mobility.

Another improvement would be to enable that the robotic arm can turn in 360 degrees. That would require two cameras placed from two different sides to cover all the areas where the object can be placed.

Adding gripping functionality would be the next logical step as well.