Project 7: Deep RL Manipulator Arm

Ahmed Khalid

May 9, 2019

1 Reward Function

The reward function is the same for the 2 objectives:

- +100 if the final objective is achieved
- -100 if the the episode exceeds the maximum number of frames, the arm hits the ground or hits itself
- Interim reward is calculated as follows: reward = 10 * (smoothed moving average) - 0.5

The smoothed moving average is calculated for the distance between the gripper and the can. While the 0.5 is a time penalty that is introduced to encourage the agent to achieve the objective faster and not settle to collecting smaller rewards without actually reaching the can.

Position control was used to solve both of the objectives. The agent was much faster to converge using the position control. Although the movement of the arm was too quick and not realistic in the simulation. Speed control was much slower to converge although the movements of the arm were smoother.

2 Hyperparameters

The hyperparameters were chosen as follows:

- INPUT_WIDTH 64 & INPUT_HEIGHT 64: The dimensions of the input image is set according to 2 criteria, performance and accuracy. If it is too small the details will vanish from the image and decrease the accuracy. If it is too high is will take longer to execute and require more parameters to train. The value of 64 achieved a good balance between performance and accuracy.
- OPTIMIZER "Adam": It was chosen because in general performs better then RMSProp whilst maintaining its advantages.

- LEARNING_RATE 0.01: It was chosen based on experimentation. In general, a low value of learning rate causes slower but less noisy learning. While a higher value may make the learning faster but it may not converge to an optimal minimum of the loss function.
- BATCH_SIZE 64: It was chosen based on trial and error. The higher the batch size the more steady the learning gets but a value too high caused the agent to take longer to converge.
- REPLAY_MEMORY 10000: This value is also chosen based on trial and error. The replay memory is used to keep batch gradient updates stable and to enable the agent to learn from past experiences.
- USE_LSTM true & USE_LSTM 64: The LSTM was used to keep track of the past experiences of the agent and not take actions based only on the current input or frame. The value of USE_LSTM was based on trial and error.

3 Results

3.1 Objective #1: Have any part of the robot arm touch the can

The agent converged after about 20 episodes. Because the task is relatively easy. It achieved an accuracy of 90% after about 130 episodes.

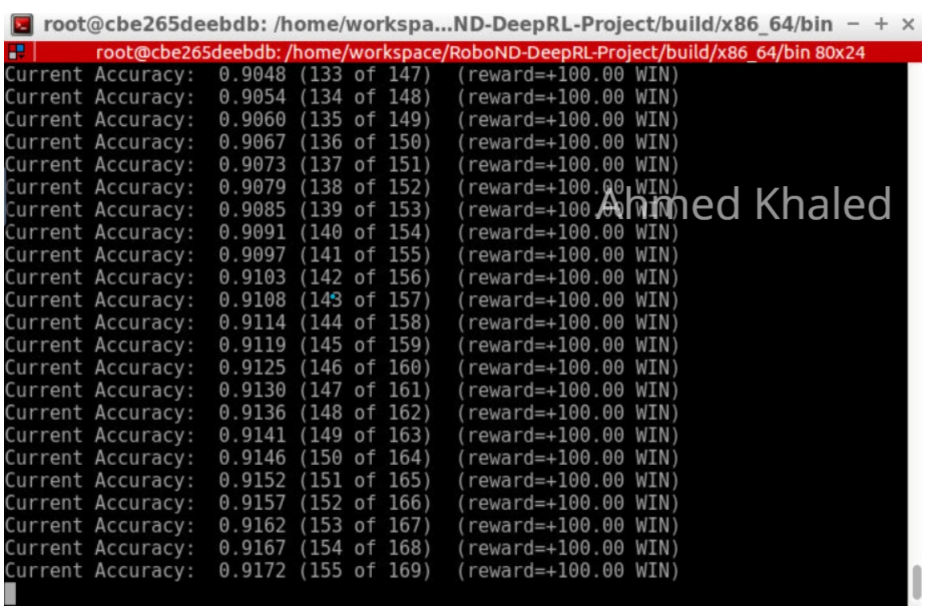

Figure 1: Terminal screenshot of the first objective

3.2 Objective #2: Have only the gripper base of the robot arm touch can

The agent converged after about 50 episodes. Because the task is relatively harder than the first one. It achieved an accuracy of 80% after about 200 episodes. It should be noted that increasing the time penalty too much caused the arm to just slam into the can without actually caring about achieving the original objective.

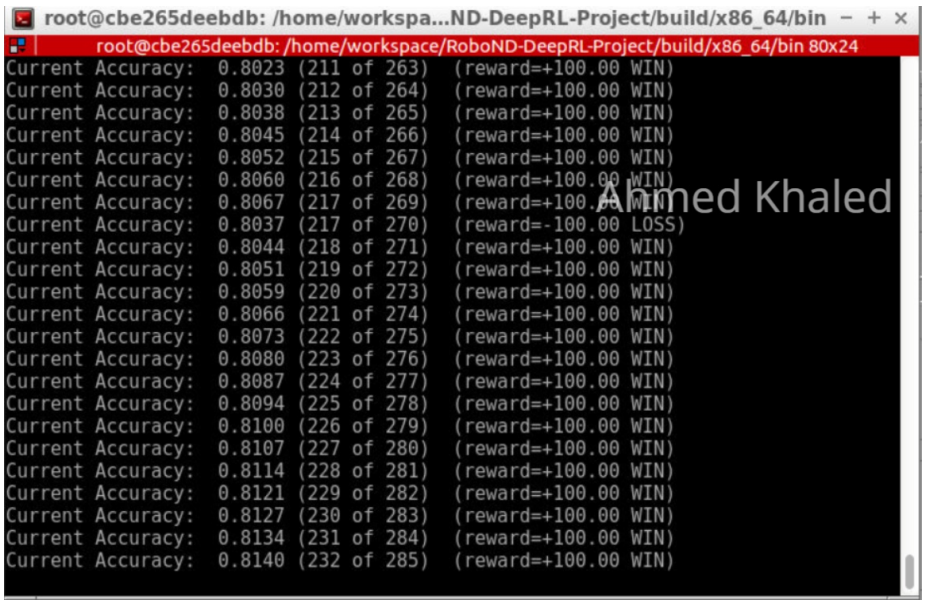

Figure 2: Terminal screenshot of the first objective

4 Future Work

As the current implementation of the reward function has no regard to the actual trajectory that the arm takes. It may be beneficial to experiment on modifying it to prioritize an optimal trajectory. Also implementing more advanced reward functions from research papers could be really beneficial. It can also benefit from using better training techniques, like learning rate annealing with restarts. The annealing part is used to make the network make more steady progress as ti reaches a minimum. The restarts would also make the network get out of a local minimum if gets stuck into one.