Deep RL Project

Danny Fisher

1 Introduction

Deep Reinforcement Learning techniques are of high usage in commercial applications. [1] Teaching an agent to complete a particular task requires action-based conditioning. To implement a self-learning agent various elements need to be addressed (Fig. 1) These elements are:

- Reward functions- Rewarding/Penalising feedback from previous actions
- Policy- Determining optimum strategy for highest cumulative reward
- Deep Neural Network- Used to map individual feedback to actions. Learned over time.
- Hyper-parameters- used to optimise learning performance

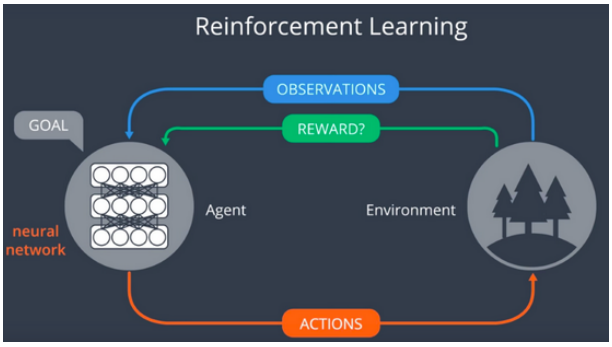

Fig. 1. Reinforcement Learning Combined a Deep Neural Network

The goal of the project is to create a DQN agent and define a reward system to allow a robot to carry out the objectives below:

- Have any part of the robot arm touch the object of interest, with at least a 90% accuracy.
- Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy.

2 Reward Function

The reward values that were used as a baseline were:

- REWARD_WIN: 300
- REWARD_LOSS: -300
- REWARD_MULTIPLIER: 100
- alpha: 0.3

The criteria for achieving both objectives were determined by the following functions:

- Arm Contact- For Objective 1 arm contact with object is positive; value: REWARD_WIN. For Objective 2 arm contact is not desired. Therefore the reward is negative. Value: REWARD_LOSS/10.
- Gripper Contact- For Objective 2, each parts of the gripper that make contact are equally rewarded. Value: REWARD_WIN;
- Timeout - In order to penalise a solution that takes longer a negative reward is awarded if 100 iterations pass without reaching the goal. Value: REWARD_LOSS/10.
- Ground Contact - If the gripper touches the ground this is undesirable. And so a negative reward is given. Value: REWARD_LOSS.
- Moving Toward Goal- In order to encourage the robot to reach further an incremental goal is added. If the robot moves away from the target object, a negative reward is given. The reward is calculated as a value between -1 and 1 using the smoothing average function.

3 Hyper-parameters

The following hyper-parameters were selected for the following reasons:

- INPUT_WIDTH & INPUT_HEIGHT - These two parameters determine the size of the image which is fed into the DQN agent. The initial values of 512x512 were conducive to poor performance. 64x64 was chosen without an observed deterioration in accuracy.
- OPTIMIZER - This was a choice between Adam or RMSProp, both of which are adaptive learning rates used in gradient-based optimisation. The optimiser chosen was RMSProp.
- LEARNING_RATE this determines the rate of convergence. Initially this value was set to 0.2, but was adjusted to 0.12
- REPLAY_MEMORY - size of memory allocated to storing the observations from the agent. Value set to 11000. Given the batch size of 32.
- BATCH_SIZE- affects the number of training examples- The larger the product of batch size and number of iterations the more memory and computational performance required. The optimum batch size was found to be 32.
- USE_LTSM - using the long Short Term Memory (LSTM) allows the training in the network to learn form previous frames, not just the current frame. As this enchanced the accuracy of the agent it was set

to true. LTSM_SIZE - This determines the number of frames saved to improve accuracy of training. Due to computational requirements the LSTM size was set to 256.

-

4 Results

After the setting the reward function and the hyper-parameters the robotic arm was able to achieve a 91% accuracy within 250 cycles for objective 1.

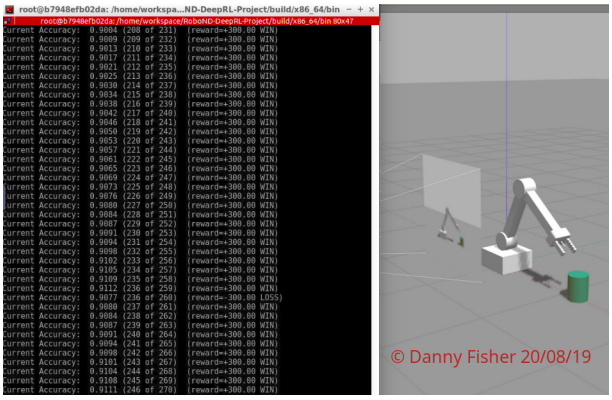

Fig. 2. Objective 1- Final result of 91%

There were no parameters that required changing for the second task except for the learning rate; the value was changed to 0.012. This gave the robot a final accuracy of 93% after 290 cycles.

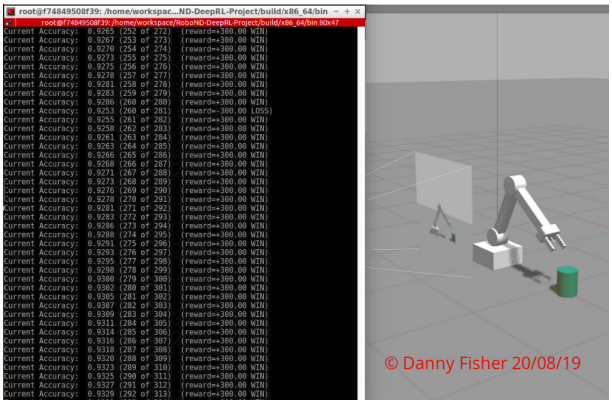

Fig. 3. Objective 2- Final result of 93%

5 Future Work

This current solution could be modified to investigate the success of controlling the motion through velocity control for smoother movement. The hyper-parameters and reward functions could be investigated further to provide a more aggressive and equally accurate learning process; one with less iterations. The DQN could be further utilised if computation power and memory was increased it would allow for larger input images. This would allow the DQN to learn more complicated games. Rewards can be score based and other penalties based on the obstacles.

References

[1] Re-Work, "Deep learning in production warehousing with amazon robotics." https://medium.com/@teamrework/deep-learning-in-production-warehousing-with-amazon-robotics-571e69fea721, 2017. Accessed: 20/07/19.