Project 1 Navigation (Collecting Bananas) Report

Udacity Deep Reinforcement Learning Nanodegree

Algorithm Used

This project uses Deep-Q Learning to train a 2-hidden-layer neural network (Multilayer Perceptron) to learn how to control an agent in the ML-Agents Unity environment to collect yellow bananas and avoid blue bananas. The environment state space has 37 dimensions and responds to 4 possible actions (move forward, move backward, turn left, turn right). The hidden layers employ rectified linear activation functions and, after training, represent the Q table to control the agent. A first hidden layer of 128 neurons and a second hidden layer of 64 neurons exhibited excellent performance (slightly better than using a first hidden layer of 64 neurons). The network is depicted in Figure 1.

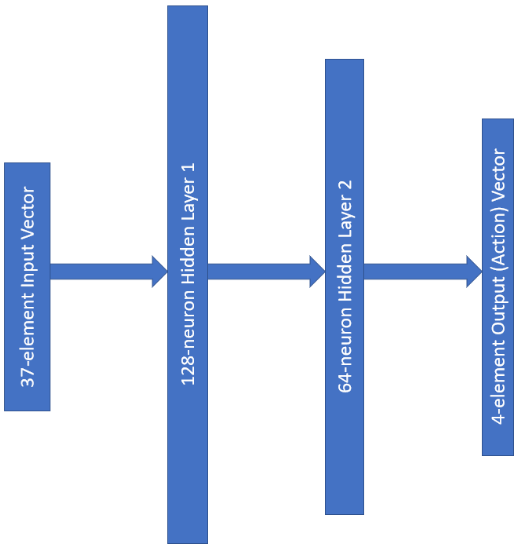

Figure 1. A representation of our neural network

The network is trained over a maximum of 2000 episodes using Deep-Q learning while using epsilon decay, a replay buffer of size 100000 steps with minibatches of size 64 (to prevent action values from oscillating or diverging), and a modest discount factor of 0.99 seen to be useful in the earlier miniprojects. A learning rate of 0.0005 allowed for stable learning over this time, while a tau value of 0.001 was used for soft-updating the target network values with those of the primary network (i.e., updating w^- in Equation 1 periodically to avoid oscillations or diverging):

$$\Delta w = \alpha \cdot \overbrace{(\underbrace{R + \gamma \max_a \hat{q}(S', a, w^-)}_{\text{TD target}} - \underbrace{\hat{q}(S, A, w)}_{\text{old value}})}^{\text{TD error}} \nabla_w \hat{q}(S, A, w)$$

Equation 1

Using these parameters, the environment was solved (i.e., achieved a score of at least +16 over 100 consecutive episodes) in 1288 episodes. +13 was required for the assignment which was actually achieved after 600 episodes, but +16 was used out of curiosity since it is inclusive of the main requirement. A plot of the scores over these episodes is shown in Figure 2 below.

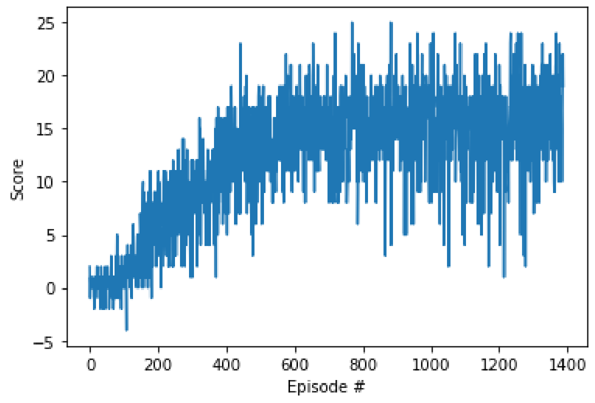

Figure 2. A plot of our score over training episodes

Future Work

The Deep-Q Learning algorithm used to train this agent would likely benefit from applying the Rainbow technique; i.e., applying Double DQN, prioritizing the experience replay, using dueling DQN, multi-step bootstrap agents, distributional DQN, and noisy DQN. As an *immediate next* step, the easiest implementation modification would likely be prioritizing the experience replay by applying a priority queue to the ReplayBuffer class. One possible value to prioritize could be the density of bananas in view – in a wide open area actions matter less, whereas navigating dense regions requires careful movement! Using the pixel-based implementation may be necessary for this modification.