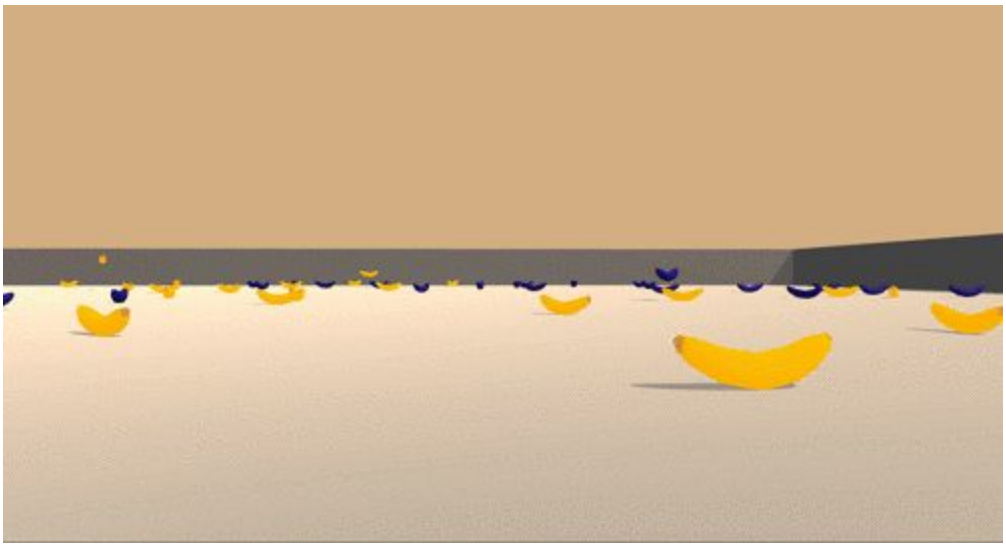
Project 1: Navigation

Jonathan Sullivan



For this project, I trained an agent in a Jupyter notebook to navigate and collect bananas in a large, square world. Given state information, the agent learned how to best select actions. The goal of my agent is to collect as many yellow bananas as possible while avoiding blue bananas.

Environment

The task is episodic, and in order to solve the environment, my agent had to get an average score of +13 over 100 consecutive episodes.

Reward

A reward of +1 is provided for collecting a yellow banana.

A reward of -1 is provided for collecting a blue banana.

State

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction.

Actions

Four discrete actions are available, corresponding to:

0 - move forward.

1 - move backward.

2 - turn left.

3 - turn right.

The Learning Algorithm

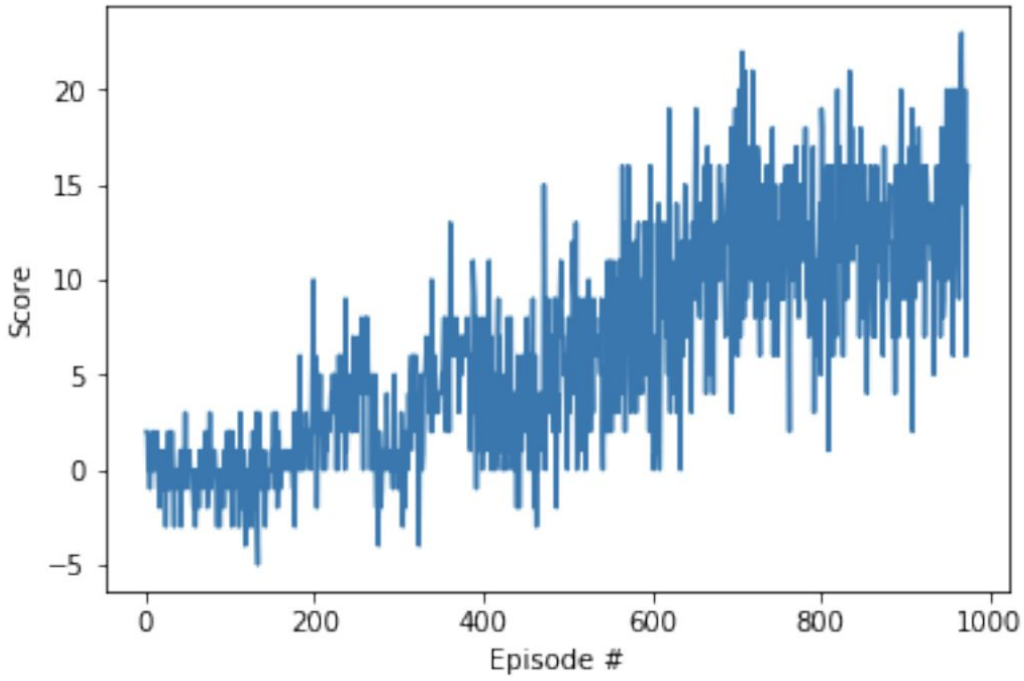
DQN

The algorithm I used is called the Deep Q-Network. Unlike DeepMind's approach which uses a deep convolutional neural network, my agent used networks that contained and an input layer which receives state values, an output layer that corresponds to the 4 action values, and 2 hidden layers using a rectified linear unit as an activation function. The main function of this Value-Based approach is to approximate the true Action-Value function in the form of a neural network.

Due to inherent instability in reinforcement learning that uses nonlinear neural network as function approximator inthatstability comes from correlations the sequences of observations, I used a technique called Experience Replay. This removes the correlations in sequences making the changes to Q-value more stable, by collecting experience tuples then randomly sampling batches for training.

The agent also uses fixed Q target, which help break correlation between the target and parameter the agent is optimizing for. We use 2 separate but identical networks, local network and target network. This way the agent can update the weight of the local network immediately after an action and use soft update adjusts Q values in the target network towards the true target values. This means that Q values in the target network are only updated periodically, further reducing correlations.

I did notice even with these improvements the results were variable even late in the training phase. Even though the results did trend upward, there existed episodes late in training with lower scores than those earlier in training, which is not the case with other deep supervised learning and common in reinforcement learning.



Future Improvements

Even though my algorithm solved the given environment. There are various improvements that I could have made. One improvement I could have used to curb over-estimation of Q is Double DQN. The Double DQN approach which trains two networks selecting one at to choose the action from and the other to use for evaluation at each step. The original algorithm assign the networks their roles randomly at each step. However, this would have been easy to implement since I am using a fixed target network which would be able to double as the evaluation network since the agent uses soft update techniques.

Another technique that I could have used to make learning more stable and efficient is Prioritized Experience Replay. This is similar to Replay Buffer technique describe early except that it doesnt sample values over a uniform random distribution. Instead each experience is provide a priority based on the absolute TD error and it influence on the Q value is mitigated by it freqeuncy in training. This ensures that older and rarer experience influence learning as much as the younger more frequent counterparts.

Another technique that I could have used to make learning more stable and efficient is Dueling DQN. In this technique I would learn the State-Values and Advantage-Values separately then combined them to estimate the Q-value. This technique take advatage of the fact that Action-value function vary very little depending on which action you take. This technique has shown significant improvement over vanilla DQNs.