## Learning Algorithm Overview

The learning algorithm present in the program is the Deep Q-Learning Algorithm, which contains many moving parts. The overview of the algorithm is that it keeps memory of past actions, samples them at random, then updates the scores of certain actions according to a continuous string of actions based on how far away the target score is and how much the developer wants the new actions to adjust existing scores.

## Learning Algorithm In-Depth

The replay buffer holds a certain number of past experiences to be randomly sample, and only samples them after a certain amount of steps. Since the program does not heavily rely on what order the actions are taken in, the replay buffer allows harmful correlations from not being formed by having them chosen at random.

There are two neural networks at work (albeit of the same architecture and one simply a few steps behind the other). The network to be updated provides the prediction of how much score each action will generate, and the newest one is updated to fit this prediction. The first is known as the target network and the second is the local network. Like the replay buffer, this is implemented to restrict the program from making harmful correlations between actions where none or very little may apply. The target buffer shows how much error there is between the networks prediction and the actual reward, and the local network is the network updated.

## Hyperparameters

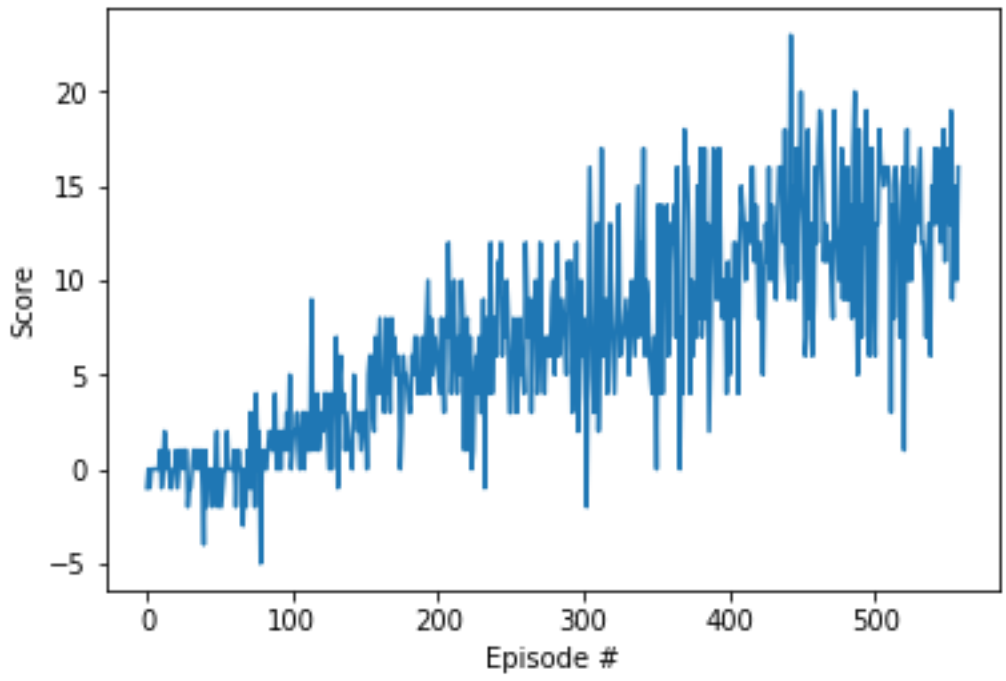
The chosen hyperparameters are a buffer size of 10^5, batch size of 64, gamma of .99, tau of 10^-3, learning rate of 5*10^-4, an update to the target network every 4 steps, max episodes of 2000, max timesteps of 1000, epsilon of 1 with a decay of .995 till .01. The buffer size is how large the replay buffer is, the batch size is how large of a sample is pulled from the buffer, the gamma is the discount factor for how far in the future the reward is, tau is how much the weights of the target network are updated by (from the local network), learning rate is how quickly the model converges / does not converge, and epsilon is the chance that an agent will act randomly (as it learns).

## Architecture

The architecture is state size to 128 hidden nodes with relu activation, to a dropout of .01, to 64 hidden nodes with relu activation, to a dropout of .01, to 32 hidden nodes with relu activation, to action size.

## Rewards per Episode

Episode was solved in 458 episodes! Detailed graph of total rewards over each episode below.




## Future Work

This is not by far a sophisticated program. It is detailed enough to solve this simple challenge, but may falter in other regions that include more randomness and could solve the challenge faster with a different architecture.

In the lesson in Udacity, double DQN, dueling DQN, and prioritized experience replay were all mentioned. These would all be worthy candidates to at least apply to the program, as having cooperating / competing networks may solve some of the issues in having the challenge solved in less time steps. Different architectures of larger hidden nodes at the start of the neural network could also be used (one that went up and down variably in hidden nodes did not work as intented).