## Learning Algorithm Overview

The learning algorithm present in the program is Deep Deterministic Policy Gradient, or DDPG, which comes from and improves upon Deep Q-Learning. The algorithm makes use of two sets of neural networks, one deciding the expected reward from taking certain actions, and one which proposes which actions to use. They both iterate off of the same incentive (that being the rewards from the environment).

## Learning Algorithm In-Depth

DDPG ends up being a more valuable tool than Deep Q-Learning because of its capabilities of predicting rewards as well as its prediction of best actions. Though Deep Q-Learning is capable of predicting its own actions, DDPG tunes and recalculates the possible reward for an action mostly without seeing the state before in its lifetime. This makes for a much more robust system that can determine the best rewards and the best actions *independently* from each other. DDPG also works off of the same mechanics as Deep Q-Learning such as replay buffers and soft updates, which both help to dissociate states from other states.

## Hyperparameters

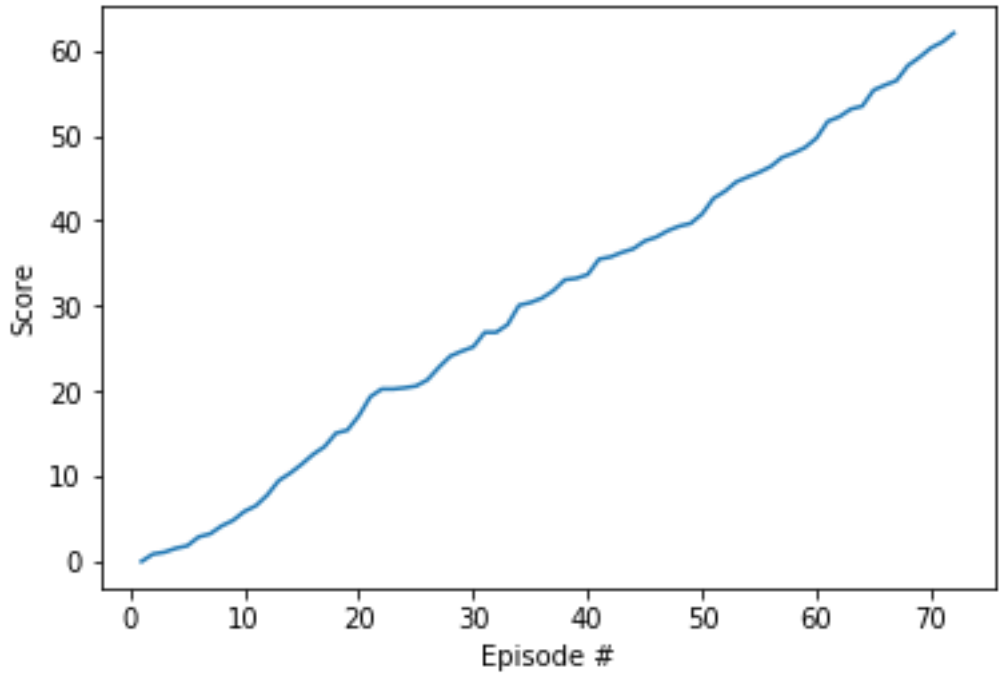
The chosen hyperparameters are a buffer size of 10^5, batch size of 128, gamma of .99, tau of 10^-3, learning rate of 10^-4 (for the actor), learning rate of 10^-3 (for the critic), a weight decay of 0, max episodes of 1000, max timesteps of 300. The buffer size is how large the replay buffer is, the 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 weight decay is a regularization parameter.

## Architecture

The architecture is state size to 400 hidden nodes with relu activation to 300 hidden nodes with relu activation to action size for the actor (with a tanh function applied at the end). The architecture is state size to 400 hidden nodes with relu activation to a concatenation of action space size to 300 hidden nodes with relu activation to action size for the critic (with a relu function applied at the end)

## Rewards per Episode

Episode was solved in 72 episodes! Detailed graph of total rewards over each episode below.




## Future Work

This is a strong program that can be used for a multitude of functions. It solved the episodes here relatively quickly, but could have been optimized better. The parameters were pretty fixed from the beginning as the program was doing well, but it took longer than 30 minutes to run which can be undesirable in many cases.

Running the program with 20 agents (the second option to the project) would work well to reduce the learning time as a whole of the agents. The program is already pseudo-tailored to multiple agents and would simply need a few tweaks to be fully operational. Adding 20 agents would be a decent feat and would be sure to improve the program.