For this project, I implemented the DDPG Actor-Critic algorithm for continuous action spaces. It was the right algorithm to implement due to it's combination of policy gradient and value function approximation. The combination of the two lead to better stability and results than either algorithm alone. Below are the steps for implementation.

1. Initialize replay memory buffer
2. Initialize local actor and critic network as pytorch neural nets
3. Implement a soft-weights copy function
4. Train the actor and the critic

The actor and critic network have 3 fully connected layers, with the only difference being a tanh function at the end of the actor network to transform the outputs between -1 and 1. One other minor but crucial difference is that the second layer of the critic network has an input size to include both the output of the first layer AND the actions chosen by the actor network. This enables the critic to evaluate the actor.

The hyperparameters used for training the network are as follows:

BUFFER_SIZE = int(1e5)

BATCH_SIZE = 1000

GAMMA = 0.99

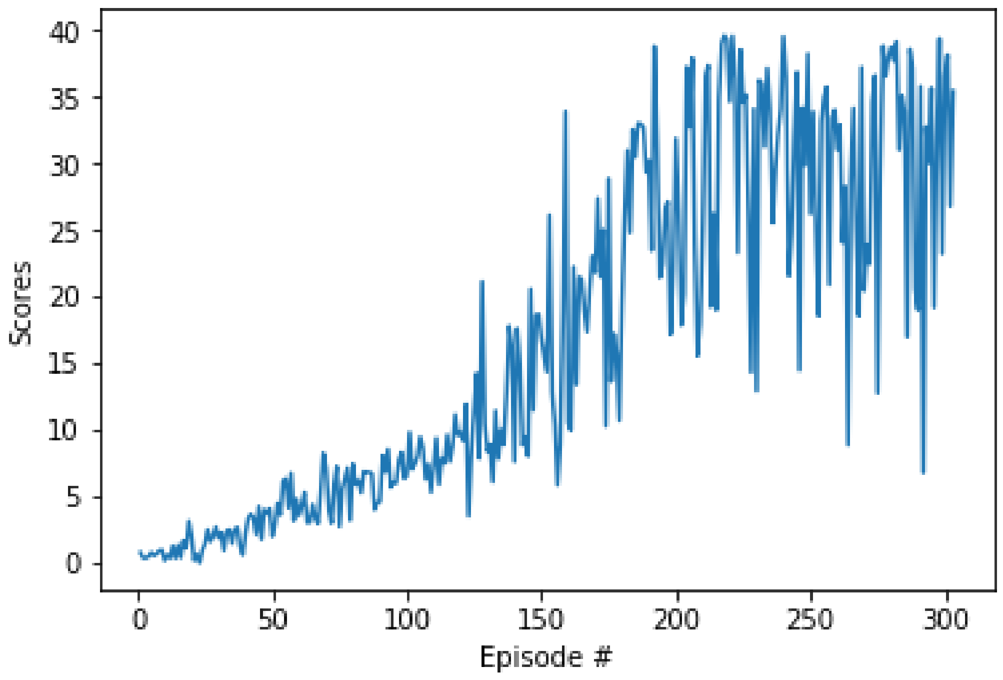
TAU = 1e-3

LR_ACTOR = 1e-4

LR_CRITIC = 1e-3

WEIGHT_DECAY = 0

Here is a plot of the rewards. As we can see, after the 200th episode the environment is considered solved.



Ideas for future work:

1. Prioritized experience replay
2. Asynchronous Actor Critic