Project 2 (Continuous Control) Report

Learning Algorithm

The algorithm I chosen is Deep Deterministic Policy Gradient, as outlined in this paper. This algorithm has been known to do well in simulated physics tasks that operate over continuous spaces, which is exactly what this project is about.

Model Architecture

Both Actor and Critic share a similar network architecture:

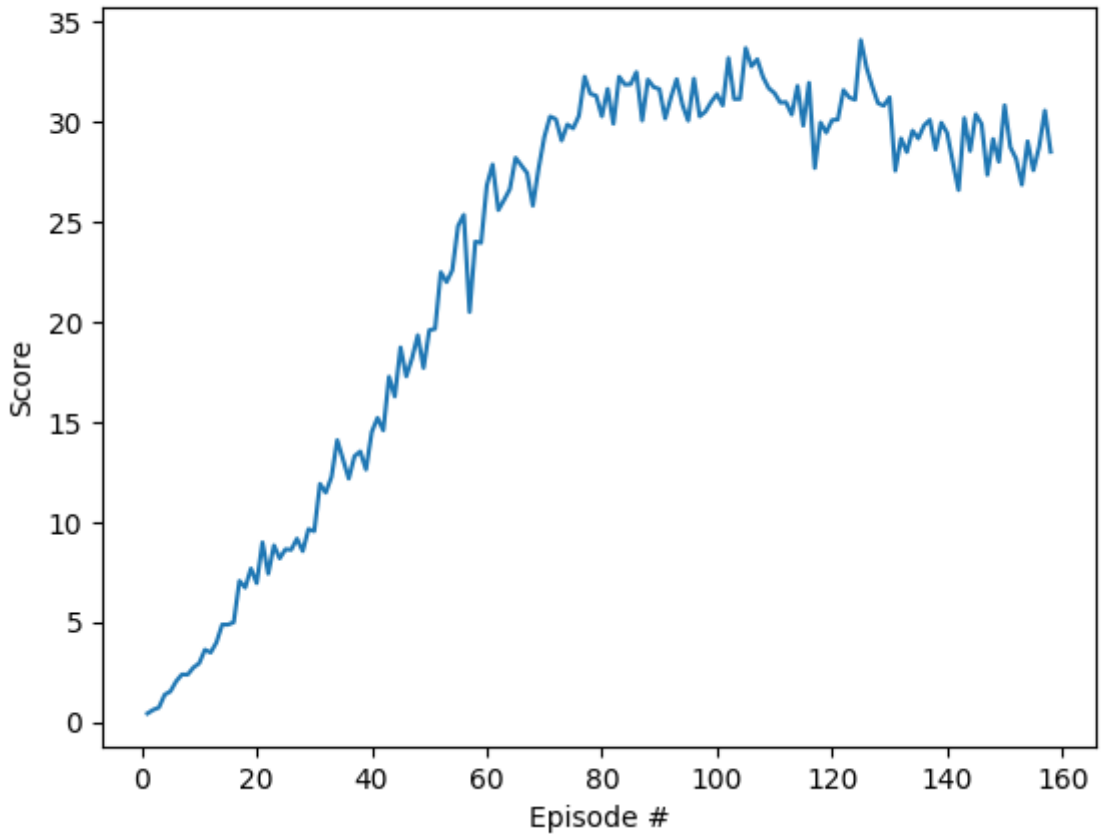
Layers	# Nodes
Input	33
Fully Connected 1	256
Fully Connected 2	256 (+4 for Critic)
Fully Connected 3	64
Output	4

I used Leaky ReLUs for all the fully connected layers.

Hyperparameters

```
BUFFER_SIZE = int(1e6)  # replay buffer size
BATCH_SIZE = 512 # mini-batch size
GAMMA = 0.99 # discount factor
TAU = 1e-3 # for soft update of target parameters
LR_ACTOR = 1e-4 # learning rate of the actor
LR_CRITIC = 3e-4 # learning rate of the critic
WEIGHT_DECAY = 0.000 # L2 weight decay
```

Plot of Rewards



Lessons Learnt

Training the agent took far longer than I expected, with many false starts. Here are some of the lessons learnt:

Don't start with too big a network

I initially made the network relatively large. However, the average scores for the first 10 episodes weren't increasing and stagnated at a very low number.

Learning Rate

As expected with deep neural networks, the scores obtained are very sensitive to the learning rate. Trying learning rates of `0.01` didn't yield very good results as the scores would at first

increase then decrease and get stuck.

Learning Every X Time Steps

The agent samples from the replay buffer every X time steps. Initially, I set this number to equal the number of agents. This means that for every episode, the agent samples from the replay buffer 20 times. However, this made training extremely slow.

Going to the other extreme and setting it to 1 wasn't very helpful either, because it wasn't enough for the agent to improve it's average scores across episodes. Even setting to 5 didn't yield very good results.

Setting to 10 did the trick, which a nice balance of speed and increasing scores.

Ideas for Future Work

If I have time, I would like to implement Asynchronous Advantage Actor-Critic (A3C) and Proximal Policy Optimization.