Project 2: Continuous Control

Description of the implementation

1. Starting algorithm

In order to solve this challenge, **Deep Deterministic Policy Gradient algorithm (DDPG)** was selected as the algorithm of choice. (Description of this algorithm can be found here)

As a start, original hyperparameters values from Udacity Deep Q-Network solution codes (which was used to solve the pendulum environment) was used as a starting point. This version of DDPG of has 2 fully-connected (FC) layers, containing 400 and 300 nodes.

Starting network architecture:

- ✓ The Actor Network was built to receive input size of 33 and generate output of numbers representing the actions to be taken for that observed state. In this case, the Actor is used to approximate the optimal policy π deterministically.
- ✓ Although the Critic Network was also built to receive input size of 33, the second hidden layer of the Critic Network receives both the result of the Critic's first hidden layer and the 4 actions which came from the Actor Network. In this case, the Critic Network is used to approximate the target value based on the given state and the estimated best action , Q(s, a)

Starting hyperparameters:

- ✓ BUFFER_SIZE = int(1e5) # replay buffer size
- ✓ BATCH_SIZE = 128 # minibatch 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 = 1e-3 # learning rate of the critic
- ✓ WEIGHT_DECAY = 0 # L2 weight decay
- ✓ Max_t =300 #maximum timesteps allowed in 1 episode

Starting results:

Average score kept oscillating between 0.3 to 0.45. Even after 30mins, model was hitting only 75 episodes. Keeping in mind of the limited GPU hours granted by Udacity, model training was ended abruptly at episode 75.

```
In [*]: scores = ddpg()
        Episode 75      Average Score: 0.31
```

2. **Next iteration**

During the start, I was tweaking 1 hyperparameter or 1 aspect of model architecture at a time with the intention of isolating the impact of each change on the final model performance. However, too many iterations ended up with poor performances. Towards the end, I was updating multiple hyperparameters/aspects of model architecture at once to save time. Here are the sumamry of the key changes made in hyperparameter and model architecture, all of which were aimed at solving 2 problems:

a. **How to speed up time taken to complete each episode:**

   i. Increased max_t from 300 to 1000. With reward of 0.1, keeping max_t at 300 means agent was expected to act perfectly every timestep in order to hit average score of 30

   ii. Increase batchsize from 128 to 1024 with the hope of learning from more experiences in 1 timestep

   iii. Tried tweaking neural network architechture from [300,400] to [512,256] for both actor and critic

b. **How to stabilize learning and make sure agent has sufficient data to learn from before ending each episode**

   i. Increase critic learning rate from 1e-4 to 1e-3 to help agent to converge faster

   ii. Do batch normalization at 1st hidden layer for all 4 networks (both Actor and Critic) to help stabilize training

iv. Increase replay buffer size from 1e5 to 1e6 since there are more agents in Reacher environment vs pendulum

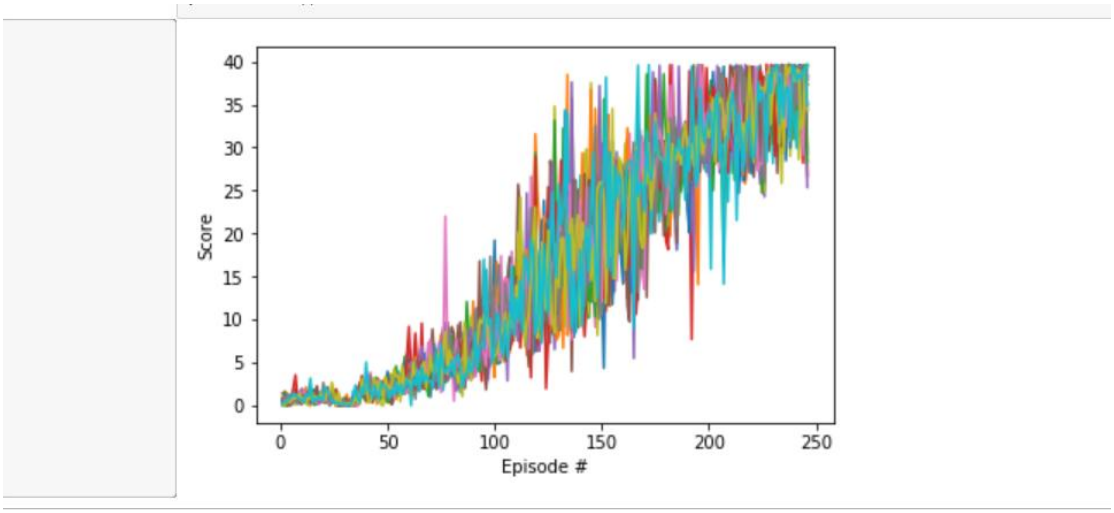
v. Implemented “Delayed Intensive Update” by updated network 10 times every 20 timesteps. This was the step which significantly moved the needle in speeding up training and stabilizing learning.

vi. Added noise decay of 0.999 to reduce unnecessary exploration after agent has gathered sizeable amount of experience

3. Final outcome

Environment was finally solved in 146 episodes.

```
Episode 20      Average Score: 0.80
Episode 40      Average Score: 0.79
Episode 60      Average Score: 1.25
Episode 80      Average Score: 2.01
Episode 100     Average Score: 2.88
Episode 120     Average Score: 5.08
Episode 140     Average Score: 8.25
Episode 160     Average Score: 11.96
Episode 180     Average Score: 16.25
Episode 200     Average Score: 20.89
Episode 220     Average Score: 25.18
Episode 240     Average Score: 29.05
Episode 246     Average Score: 30.14
Environment solved in 146 episodes!     Average Score: 30.14
```



4. Ideas for Future Work

a. There could be better hyperparameters out there to solve the environment faster. The only constraint was time.
b. While experiences were randomly sample from the replay buffer in this implementation, prioritized replay buffer might deliver a better result.
c. Other actor-critic algorithms, which had not been experimented in this implementation, could have also been better suited to solve the Reacher environment. Some examples which also use multiple (non-interacting, parallel) copies of the same agent to distribute the task of gathering experience are as follow:
    a) PPO - Proximal Policy Optimization
    b) D4PG - Distributed Distributional Deterministic Policy Gradients
    c) A3C - Asynchronous Advantage Actor-Critic