Learning Algorithm

I used Deep Deterministic Policy Gradient agent for this environment because it can be used for continuous action space and we can apply some of the DQN techniques to improve performance. Local and target networks have been used for avoiding over-estimation. Also, e-greedy Ornstain-Uhlenbeck noised action selection and experience replay is used for proper exploration.

Hyperparameters

I tried several combinations of hyperparameters as;
Actor Learning Rate: [0.0001, 0.0005, 0.001, 0.005]
Critic Learning Rate: [0.0001, 0.0005, 0.001, 0.005]
Epsilon: [1]
Epsilon Decay: [0.99, 0.999]
Minimum Epsilon : [0.01]
Batch Size : [128, 256]
Gamma : [0.99]
TAU : [0.001]
Buffer Size = 100000
Weight Decay=[0, 0.0001]

Most successful parameters are;

Actor Learning Rate: 0.001
Critic Learning Rate: 0.0001
Epsilon: 1
Epsilon Decay: 0.999
Minimum Epsilon : 0.01
Batch Size : 256
Gamma : 0.99
TAU : 0.001
Buffer Size = 100000
Weight Decay = 0

Model Structure

There is a pre-trained model which is DDPG with:

Actor Network has;
- Fully Connected Layer (state_size, 256)
- ReLU
- Fully Connected Layer (256, action_size)
- Tanh

Critic Network has;
- Fully Connected Layer (state_size, 256)
- Leaky ReLU
- Fully Connected Layer (256 + action_size, 128)
- Leaky ReLU
- Fully Connected Layer (128, 128)
- Leaky ReLU
- Fully Connected Layer (128, 1)

In the beginning, 2. and 3. hidden layer size was 256 but learning was slow. Therefore, I went for a simpler network by changing them from 256 to 128. I tried 64 but it was not sufficient for faster learning.
I used Adam optimizer for both actor and critic networks.
MSE loss function is used for critic model.
Min(-Q value from critic with the action that is generated from actor model) loss function is used for actor model.

Gradient clipping is used for critic model training because MSE loss can be a very high or low values for some states. This can lead to a wrong direction of gradient with no returning back. When I saw, at the mid stage of the training, robotic arm started making circles horizontally or vertically continuously and never tries another move. After that, I thought it may be a good idea.

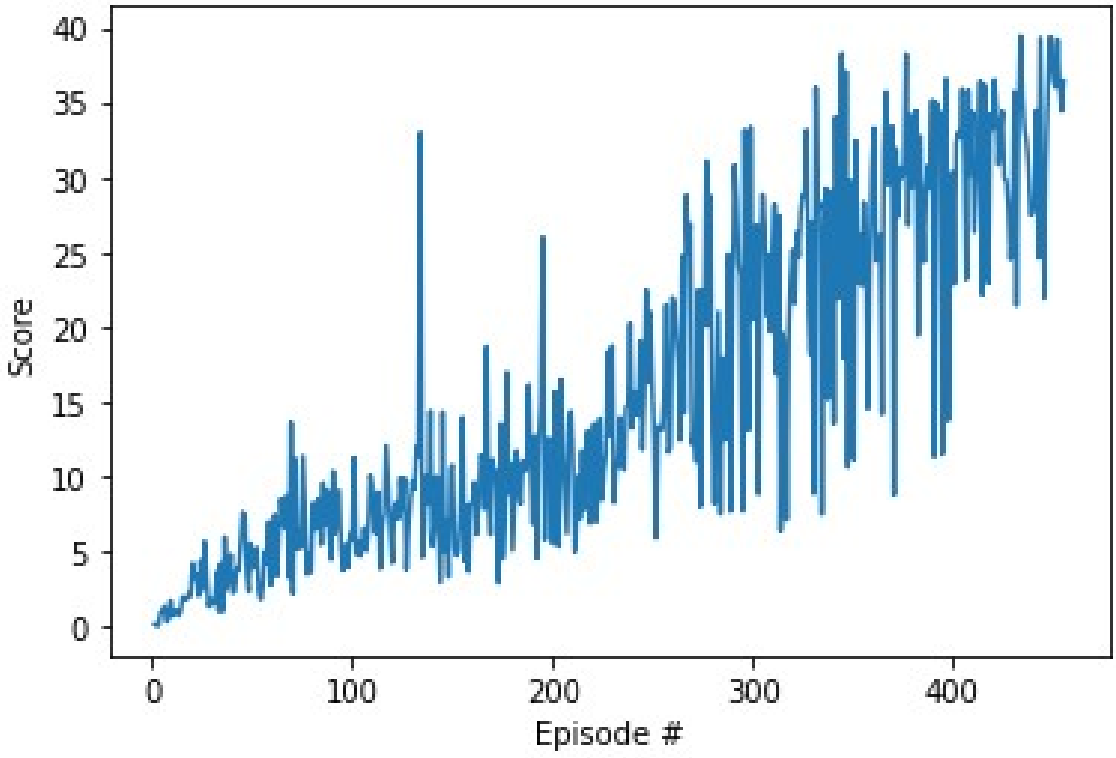
Plots

Training process result:

6. Start Training

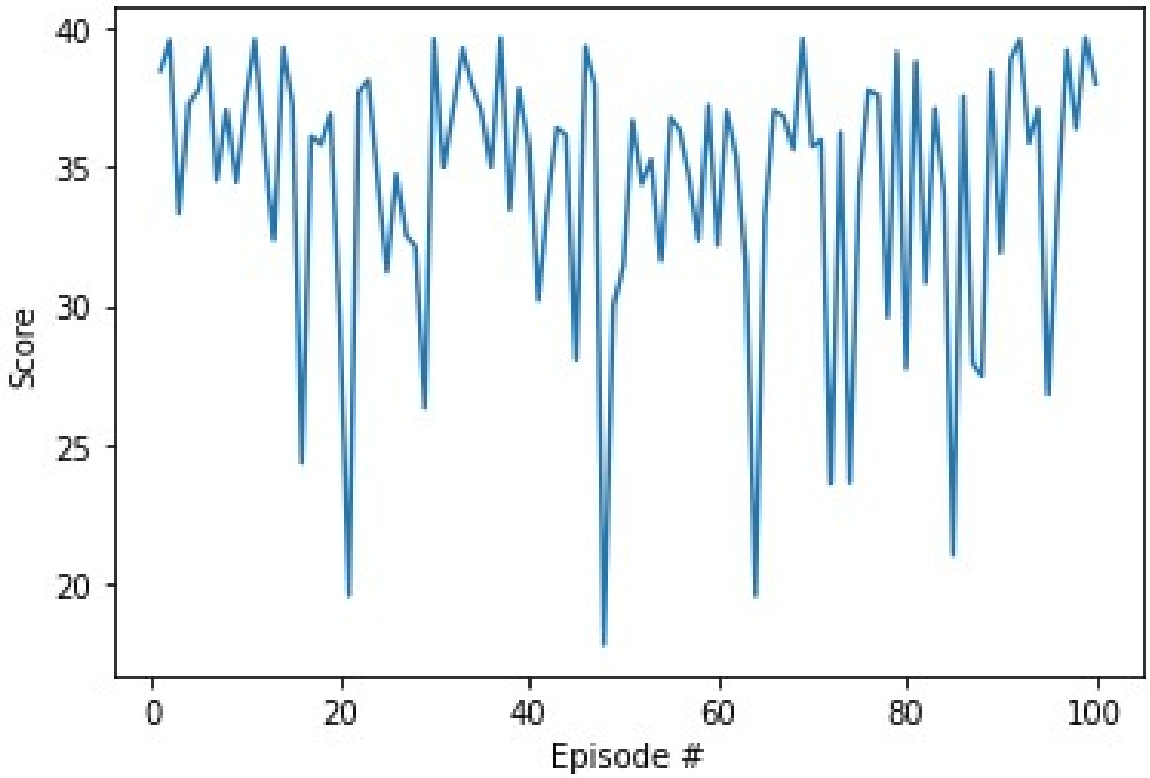
```
In [8]: scores = ddpg(ddpgAgent)

Episode 100     Average Score: 4.32
Episode 200     Average Score: 8.85
Episode 300     Average Score: 15.37
Episode 400     Average Score: 25.34
Episode 456     Average Score: 30.08    Score: 36.39
Episode solved at episode 456
```



Testing process result:

Average over 100 episodes: 34.37529923165217



Future Work

Sometimes, when the green ball is at some location agent could not reach it for lots of episodes. Then, what I observed is that agent became less successful at that moments. Therefore, I can use priotized experience replay, I can give more importance to these states and that results a better training overall. However, in Udacity discussions, one of the mentors said that it is gonna be a lot of calculation cost and results may not be satisfactory. Still, I can give PER a try.
In addition, I can try other algorithms and compare them with the same seed in an another report.

Updating target networks every X episodes can help a bit.