Enviroment

In this environment, a double-jointed arm can move to target locations. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of your agent is to maintain its position at the target location for as many time steps as possible.

The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the arm. Each action is a vector with four numbers, corresponding to torque applicable to two joints. Every entry in the action vector should be a number between -1 and 1.

belnds ## Learning Algorithm I used [Deep Deterministic Policy Gradient (DDPG)](https://arxiv.org/abs/1509.02971) algorithm. DDPG is an actor critic method where contains 4 networks: local and target Actor and local and target critic.

- The Actor specifies the current policy by deterministically mapping states to a specific action (approximate maximizer).
- The critic in ddpg is use to approximate the mazimizer over the Q values of the next state. The critcis learns to eveluate the optimal action value function by using the actors best believed action.
- DDPG uses areply buffer.
- DDPG soft upates the target networks. Soft updates are used to update target networks of actor and critic. Soft updates are used to slowly blends local netwroks wights woth target netwrok weights.
- When we are training we are training the local netwroks therefore local netwroks are the most up-to-date network.
- We use target network for predication to stablize strain.
- I used gradient clipping to prevent exploding gradients when training the critic network.
- I also updated the network after 20 steps as suggested in the benchmark implementation

Actor Network Architecture

- Input : 33 (state size)
- Output : 4 (action size)

- ```
  Actor(
        (fc1): Linear(in_features=33, out_features=256, bias=True)
        (fc2): Linear(in_features=256, out_features=128, bias=True)
        (fc3): Linear(in_features=128, out_features=4, bias=True)
     )
  ```

Critic Network Architecture

```
- Input : 33 (state size)
```

- Output : 4 (action size)
```

-

```
Critic(
(fcs1): Linear(in_features=33, out_features=256, bias=True)
(fc2): Linear(in_features=260, out_features=128, bias=True)
(fc3): Linear(in_features=128, out_features=1, bias=True)
)
```

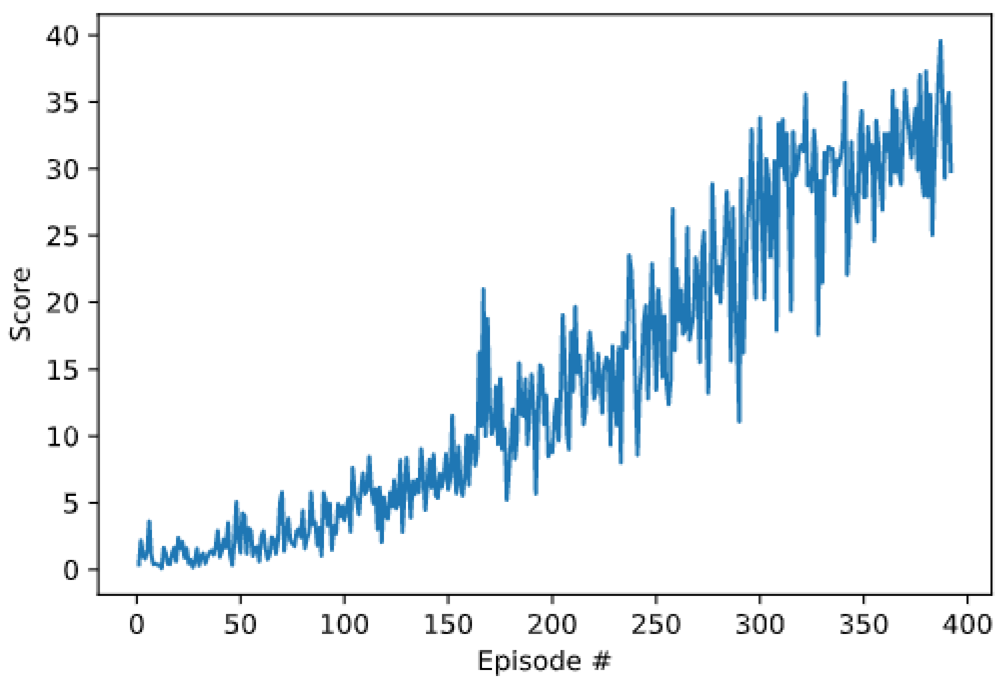
Hyperparameters

```
BUFFER_SIZE = int(1e6)  # replay buffer size
BATCH_SIZE = 128        # minibatch size
GAMMA = 0.94            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1e-4         # learning rate of the actor
LR_CRITIC = 1e-4        # learning rate of the critic
WEIGHT_DECAY = 0.0      # L2 weight decay
TRAIN_EVERY = 20        # How many iterations to wait before updating ta
rget networks
```

Training result:

Option 1

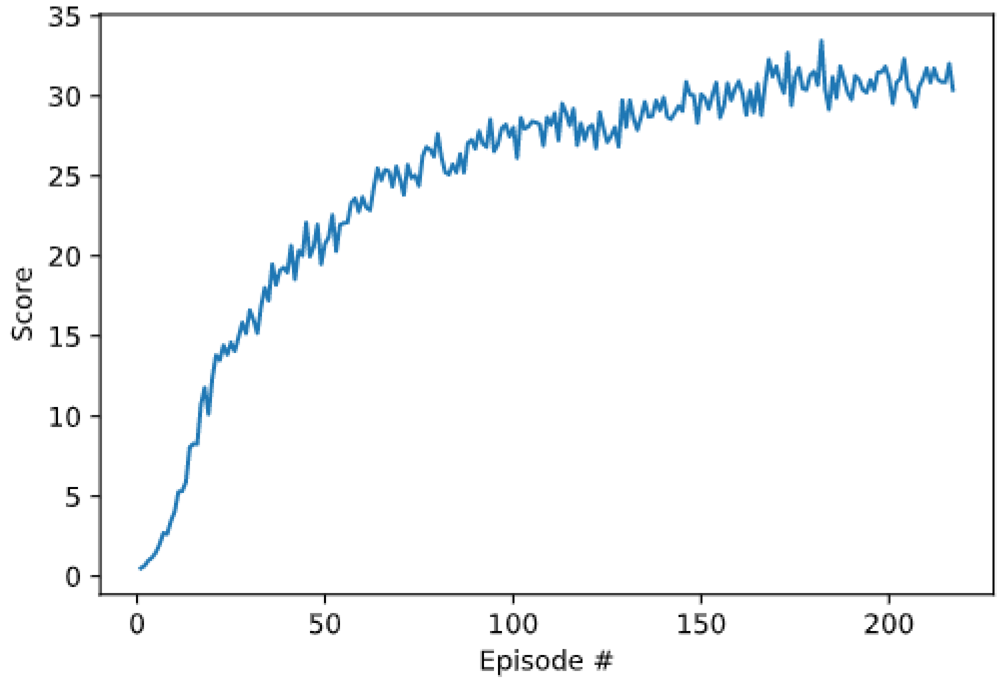
One agent enviroment was solved in 392 Episodes with average score: 30.08




```
Episode 100    Average Score: 2.09
Episode 200    Average Score: 8.24
Episode 300    Average Score: 18.15
Episode 392    Average Score: 30.08
```

Option 2

Multi agent enviroument is sloved in 217 Episodes with average Score: 30.03




```
Episode 100     Average Score: 18.97
Episode 200     Average Score: 29.57
Episode 217     Average Score: 30.03
```

Test Results

option 1 - One Agent

```
Episode:     0     Score:     31.29
Episode:     1     Score:     36.03
Episode:     2     Score:     36.61
Episode:     3     Score:     30.97
Episode:     4     Score:     28.38
```

option 2 - Multi Agent

```
Episode:     0     Score:     31.60
Episode:     1     Score:     32.11
Episode:     2     Score:     32.80
Episode:     3     Score:     32.55
Episode:     4     Score:     32.23
Episode:     5     Score:     32.77
Episode:     6     Score:     31.62
Episode:     7     Score:     33.29
Episode:     8     Score:     32.15
Episode:     9     Score:     31.52
```

Future Ideas

- Work to improve performance of the muti agent model.
- Add bacth normalisation to the netwroks
- Look into other models such as A3C , and PPO