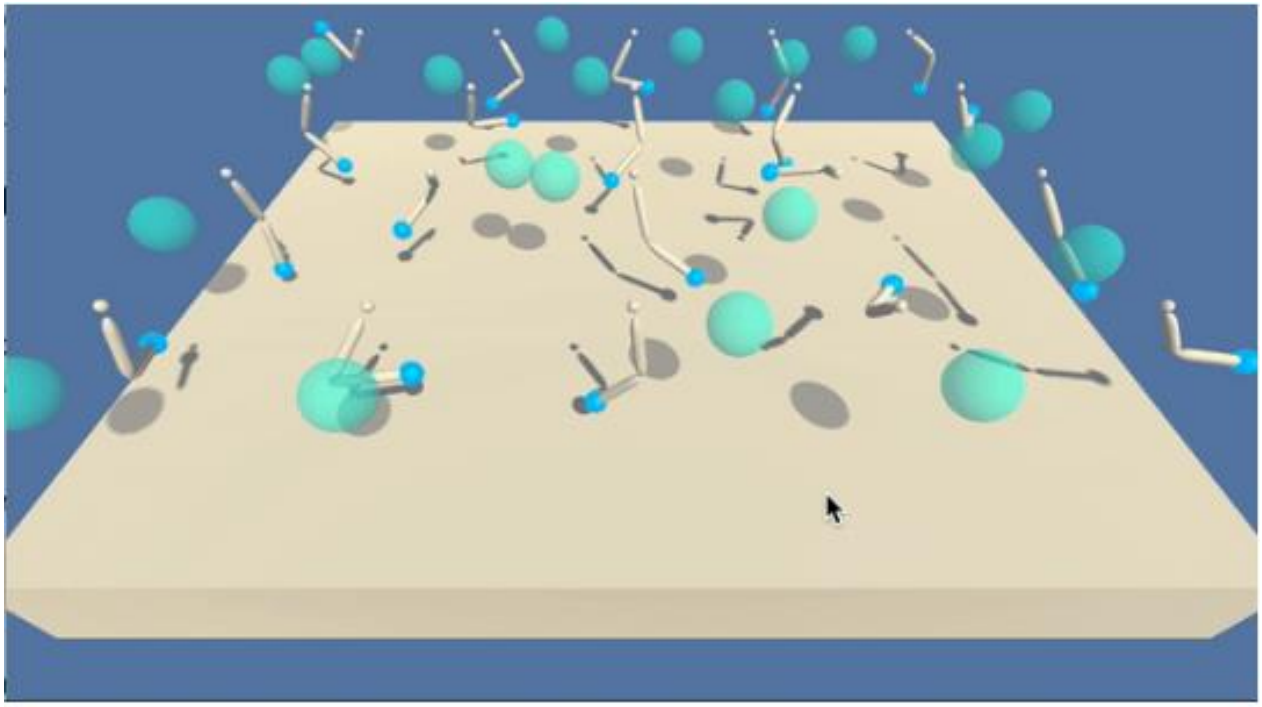
Continuous Control



Problem Statement

Using reinforcement learning techniques to train an agent to maintain its position at the target location for as many time steps as possible.

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.

Implementation

The learning algorithm takes into account the presence of many agents, specifically there are 20 in this project implementation. In particular, the goal is to train the agents such that they get an average score of +30 (over 100 consecutive episodes, and over all agents).

Specifically, after each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 20 (potentially different) scores. We then take the average of these 20 scores.

This yields an average score for each episode (where the average is over all 20 agents).

The Algorithm used to train the agent is DDPG Algorithm. DDPG uses four neural networks: a Q network, a deterministic policy network, a target Q network, and a target policy network.

The Q network and policy network is very much like simple Advantage Actor-Critic, but in DDPG, the Actor directly maps states to actions (the output of the network directly the output) instead of outputting the probability distribution across a discrete action space

Following depicts the pseudocode of the same.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Figure 1:DDPG Algorithm

Results

The agents were able to obtain an average reward of value greater than +30 within 139 episodes of learning.

```
In [15]: from collections import deque
         import matplotlib.pyplot as plt
         import time

         scores = ddpg()
```

```
Training Started.......
/home/ubuntu/anaconda3/envs/drlnd/lib/python3.6/site-packages/t
tanh is deprecated. Use torch.tanh instead.
  warnings.warn("nn.functional.tanh is deprecated. Use torch.ta
Episodes 10, AVG Score: 1.12
Episodes 20, AVG Score: 1.41
Episodes 30, AVG Score: 1.96
Episodes 40, AVG Score: 3.22
Episodes 50, AVG Score: 5.21
Episodes 60, AVG Score: 7.42
Episodes 70, AVG Score: 9.75
Episodes 80, AVG Score: 12.15
Episodes 90, AVG Score: 14.26
Episodes 100, AVG Score: 16.41
Episodes 110, AVG Score: 20.09
Episodes 120, AVG Score: 23.75
Episodes 130, AVG Score: 27.26
Episodes 139, AVG Score: 30.03

Environment solved in 139 episodes!     Average Score: 30.03
```

Figure 2: Training using DDPG

Plots

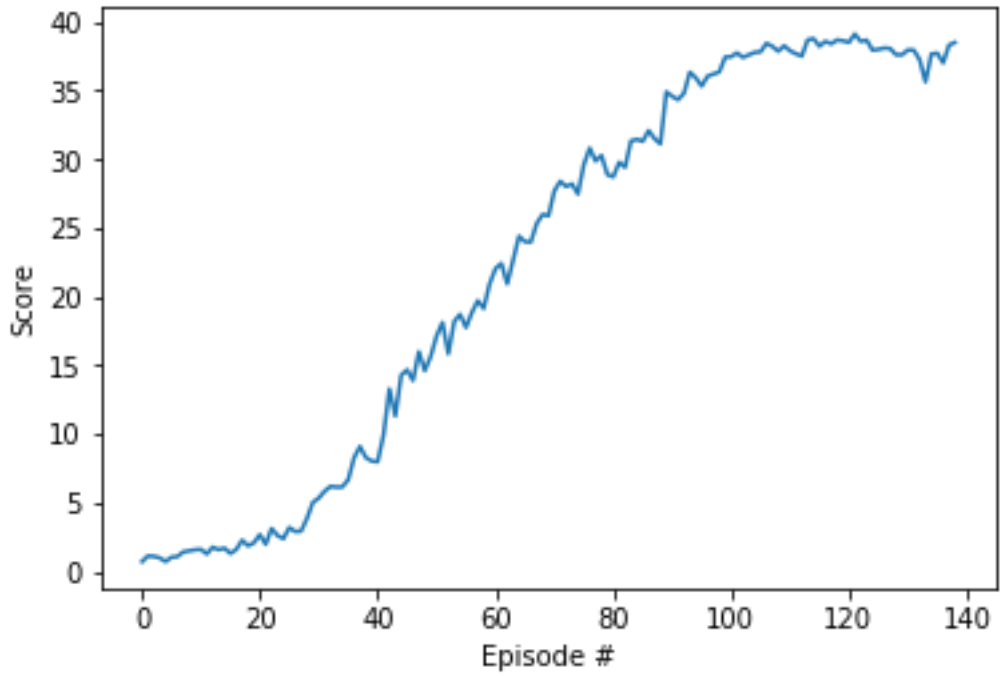


Figure 3: Episode # V/S Score

It can be observed that after about 90 episodes of learning, the agents are slowly able to accumulate a reward of +30.0. And a training upto 139 episodes finally solved the environment.

Conclusion and Future Work

- The agents are able to accumulate an average reward of +30 over 139 episodes of learning.
- We can experiment with our model architecture to further observe if there can be any improvements.
- There are several more algorithms like D4PG, PPO etc, we can use these algorithms and performs comparisons to see if it improves the results any further.
- Hyperparameter tuning seemed to be most effective for me during the implementation of this project. Experimenting with the parameters like epsilon, epsilon decay, updating factor etc. can be done further to observe if the results improve.

References

- https://arxiv.org/abs/1509.02971
- https://towardsdatascience.com/deep-deterministic-policy-gradients-explained-2d94655a9b7b