Project: Collaboration and Competition-Udacity Nano-Degree on Deep Reinforcement Learning

Introduction:

This project revolves around working with Udacity Multi-Agent Reinforcement Learning task. This is the third of the project series on Deep Reinforcement Learning. This task works on competing the two agents which have two rackets to play a game of tennis or table tennis. The agent, if can pass the ball over the net is awarded +1 point and if the agent is unable to pass the ball over the net or does let the ball touch the table or gets lowered gets -1 score. The job of the two agents is continuously passing the ball to each other over the net and being able to score against each other. The observation space comprises of 8 variables corresponding to the position and velocity of the ball. Each of the agent receives its own rewards and two continuous actions are readily available for both of the agents which they continuously move towards and away from the nets and sometimes maybe jumping over the net. This task is episodic and each of the agent must get an average score of 0,5 over the course of consecutive 100 episodes.

- Following the completion of each episode, we aggregate the rewards awarded to each agent, without any form of discounting. This process results in two separate scores, one for each agent. Subsequently, we determine the higher of these two scores by taking the maximum value.
- This yields a single score for each episode.

Solution-Components:

Algorithm:

The algorithm used in this project is a multi-agent deep deterministic policy gradient algorithm. Multi-Agent deep deterministic policy gradient is based on deep deterministic policy gradient (DDPG) that was also used in previous project of Udacity "Continuous-Control". This provided a good starting point and to develop a strong final solution.

The solution is built upon a decentralized actor with a centralized critic architecture. In this setup, there is a single critic that takes inputs consisting of both the actions and state observations from all agents. This inclusion of additional information simplifies the training process and facilitates a training approach where centralization is used, but during execution, each agent independently makes decisions based on its own observations of the environment.

Structure:

Actor:

- First fully connected layer with input size 24 and output size 256.
- Second fully connected layer with input size 256 and output size 128.
- Third fully connected layer with input size 128 and output size 2.

Critic:

- First fully connected layer with input size 24 and output size 256.
- Second fully connected layer with input size (256 + 2) = 258 and output size 128.
- Third fully connected layer with input size 128 and output size 1.
- Batch Normalization layer between first and second layers.

The hidden layers that I started with were 512 and 128 which were then ultimately reduced to 256 and 128 that led to the convergence of model successfully with acceptable running time.

OU-Noise function:

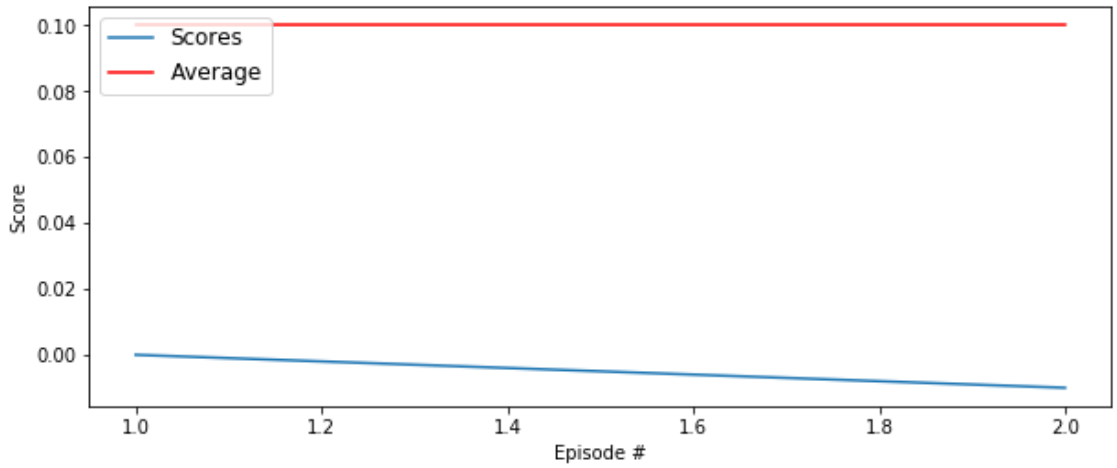
To enable exploration within the environment by the agent Ornstein-Uhlenbeck funciton OU-Noise is introduced in this project.

Experience Replay:

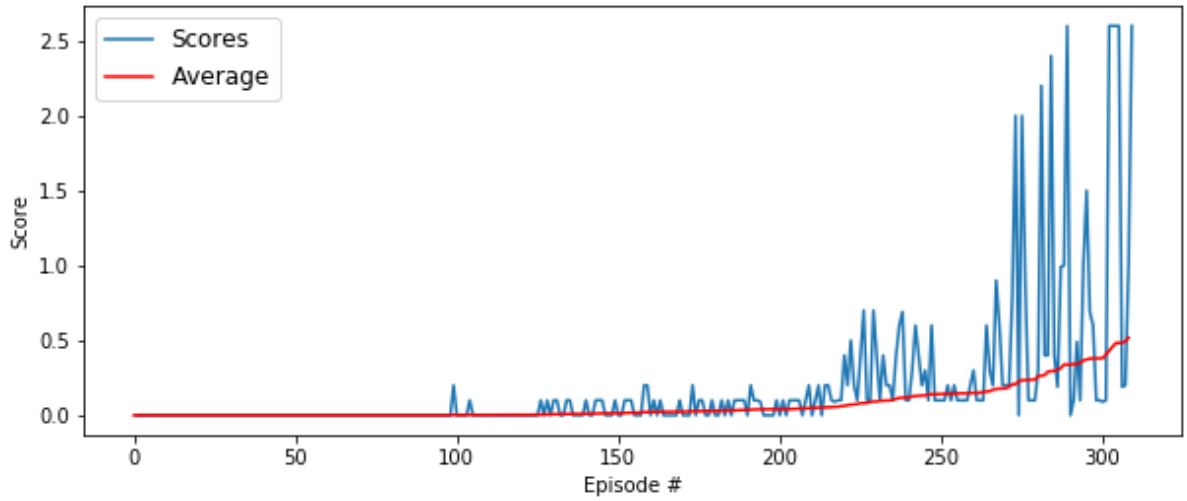
The solution is shared between the agents to improve their performance. The solution is stored in a memory which is called buffer memory. This phenomenon is called Experience Replay.

Results:

During the results, it can be seen that without noise the agents performance are quite plain and they follow a simple linear pattern as shown in the figure.



After we add some noise, then agent gives a sort of mixed behavior with noise clearly being depicted in their performances. This is shown in the below figure:



Conclusion:

Fine tuning of the hyper-parameters would be crucial to see how this algorithm would work. Of-course checking the behavior of noise within the algorithm would also be important. Changing the learning rate of actor and critic can significantly improve the algorithm performance as well.