The Problem:

In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in play.

The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping.

The task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both 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 2 (potentially different) scores. We then take the maximum of these 2 scores.
- This yields a single score for each episode.

The environment is considered solved, when the average (over 100 episodes) of those scores is at least +0.5.

The Learning Method:

In order to solve the Unity Tennis Environment a MADDPG (Multi Agent Deep Deterministic Policy Gradient) approach was applied. MADDPG extends a reinforcement learning algorithm called DDPG, that was our approach on the last project, taking inspiration from actor-critic reinforcement learning techniques. In this solution each agent in the algorithm is an “actor”, and each actor gets advice from a “critic” that helps the actor decide what actions to reinforce during training. Traditionally, the critic tries to predict the value (i.e. the reward we expect to get in the future) of an action in a particular state, which is used by the agent - the actor - to update its policy. This is more stable than directly using the reward, which can vary considerably. To make it feasible to train multiple agents that can act in a globally-coordinated way, we enhance our critics so they can access the observations and actions of all the agents.

The hyperparameters:

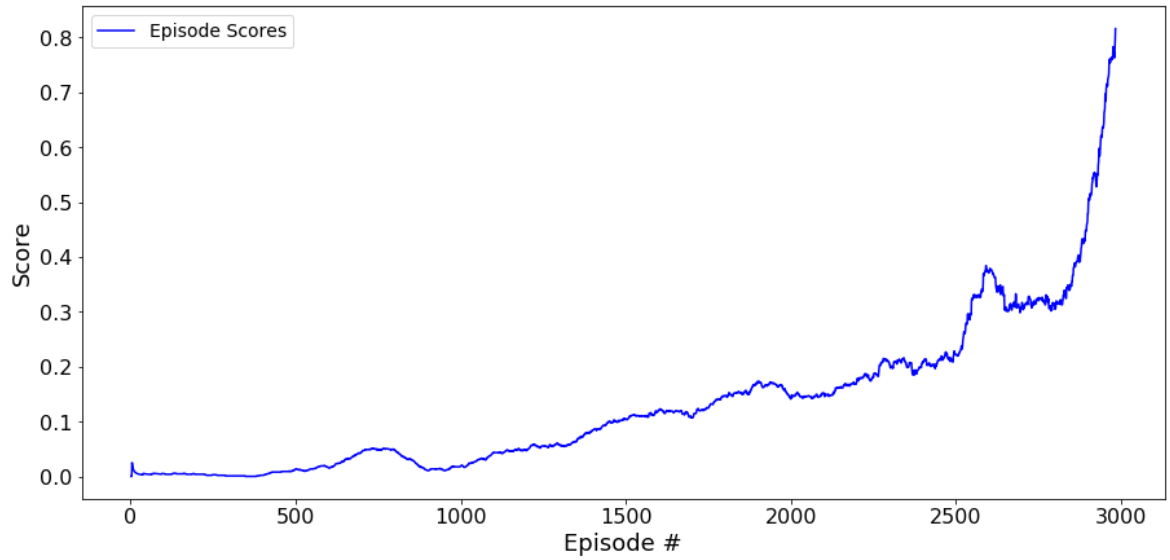
```
buffer_size= int(1e5)
batch_size=512
gamma=0.99
update_every=4
tau=1e-3
lr_actor=1e-4
lr_critic=1e-3
```

The network:

2 fully connected hidden layers with ReLU activation [Size: 400 and 300]
Batch normalization between the hidden layers
Tanh activation for Actor output and ReLU for Critic

The training performance:

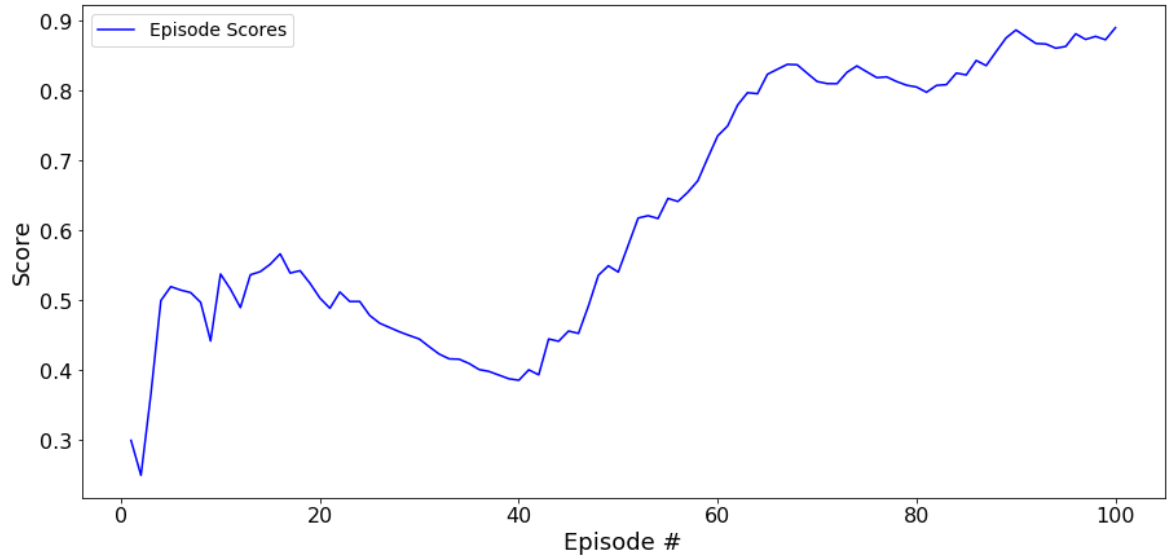
The chart below represents the agents performance during training. Episode scores increased very fast when approximating 3000 episodes reaching 0.8 as average. Threshold of 0.8 was applied to ensure that the agents would perform better during playing mode.



The training was concluded after 2.983 episodes when the agents reached an average score of 0.816.

The agents competition and collaboration performance (Playing mode):

During playing 100 episodes of the match, the agents have got 0.89 as rewards. As following we can see the agents performance:



In order to solve the environment, the agents would have to reach +0.5 as rewards during 100 episodes. So we can claim that the environment was solved.

Future Accomplishments:

- Sove the Soccer Unity environment
- Try to improve the Tennis learn algorithm (both learn method and network)
- Build an Unity Environment by myself

References:

Udacity Reinforcement Learning Nanodregree Program
https://blog.openai.com/learning-to-cooperate-compete-and-communicate/
https://arxiv.org/abs/1706.02275