# Project: Collaboration and Competition

## 1) Goal:

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. A snapshot of the environment is as follows:

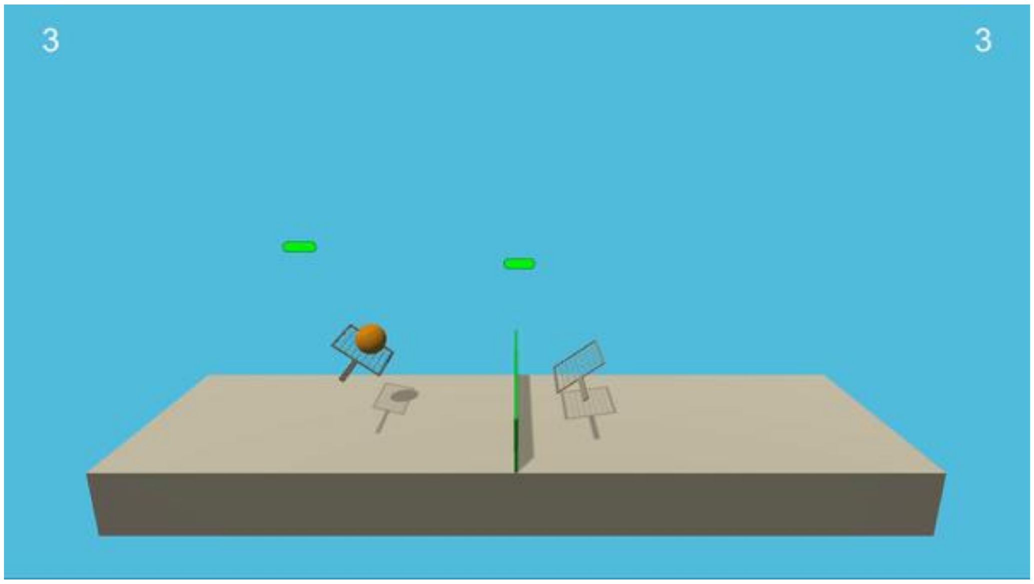

*Figure 1 Tennis Environment.*

## 2) Algorithm:

We use Deep Deterministic Policy Gradient (DDPG) algorithm to solve this environment. DDPG is an actor-critic algorithm where a critic estimates the state value function via Temporal difference estimate. The architecture for actor and critic is as shown in the following figure

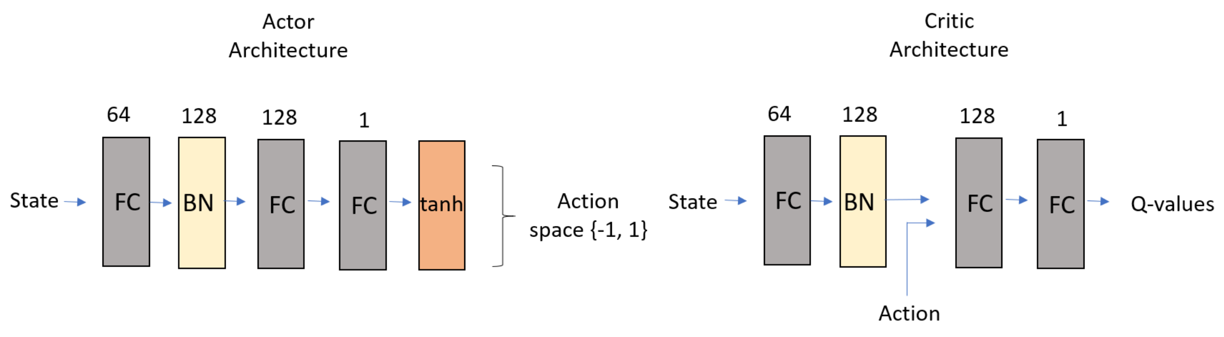

*Figure 2 Model Architecture.*

Two agents have their own set of actor/critic networks and store their experiences in a shared replay buffer memory.

Hyperparameters chosen for this experiment:

| | |
|---|---|
| BUFFER_SIZE | 1e5 (replay buffer size) |
| BATCH_SIZE | 128 (minibatch size) |
| GAMMA | 0.99 (discount factor) |
| TAU | 1e-3 (for soft update of target parameters) |
| LR_ACTOR | 1e-3 (learning rate of the actor) |
| LR_CRITIC | 2e-4 (learning rate of the critic) |
| WEIGHT_DECAY | 0 (L2 weight decay) |

## 3) Results:

The rewards plot is shown in Figure 3.

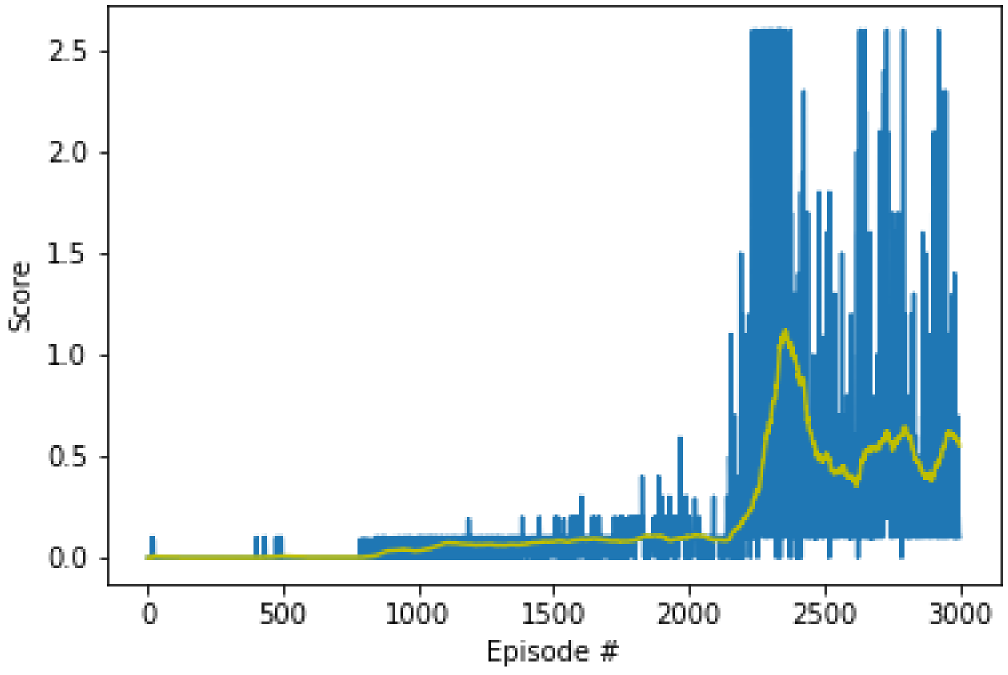

*Figure 3 Rewards Plot.*

The environment was solved in 2279 episodes with average reward of 0.51. In Figure 3, the blue lines indicate the scores after each episode and the yellow line indicates the moving average score for the agents. The performance of the agents decreased after 2400 episodes. The highest average reward obtained was +0.94 at 2400th episode.

## 4) Future work:

a) DDPG with different actor/critic networks for two agents convergence is very slow and unstable. Implementing Prioritized experience replay and other popular algorithms have to be explored to compare their performance with DDPG baseline.

b) Parallelize the existing code by taking advantage of multicore environments and running multiple agents for faster learning. Follow implementation of MADDPG lab to improve this code.