# Project 3: Tennis - Collaboration and Competition

## Environment Details

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, our 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.

## Learning algorithm

We solve the problem here using two separate DDPG agents, one for each agent which share memories of past episodes which are used during training. Each agent can see the states of itself and the other agent. The architecture of the two agent networks are the same.

ACTOR NETWORK - Fully connected layer with two hidden layers.
Input size - 2*state_size(2*24)
Output size - action_size(4)

The first hidden layer has 256 nodes and the second has 128 nodes. The activation function used for the first two layers is a relu activation function. Alternatively, leaky_relu can be used as

well as the smoothness in transition improves training speed. The final output from the last layer is passed through a tanh activation layer as we want the output values to be within -1 to +1.

CRITIC NETWORK - Fully connected layer with two hidden layers.
Input size - 2*state_size(2*24)
Output size - 1(corresponding to Q(s,a))

The first hidden layer has 256+2+2 nodes and the second has 128 nodes. The additional four is the action (2 actions for each agent) as input to the network. The activation function used for the first two layers is the relu activation function. The final output is not passed through any activation function as we want to estimate the value.

The hyperparameters for the problem are as follows,

```
BUFFER_SIZE = int(1e6)  # replay buffer size
BATCH_SIZE = 128        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 6e-2              # for soft update of target parameters
LR_ACTOR = 1e-3         # learning rate of the actor
LR_CRITIC = 1e-3        # learning rate of the critic
WEIGHT_DECAY = 0        # L2 weight decay
NUM_UPDATES = 1
UPDATE_EVERY_T = 1
EPS_START = 6
EPS_END = 0
EPS_DECAY = 250
```

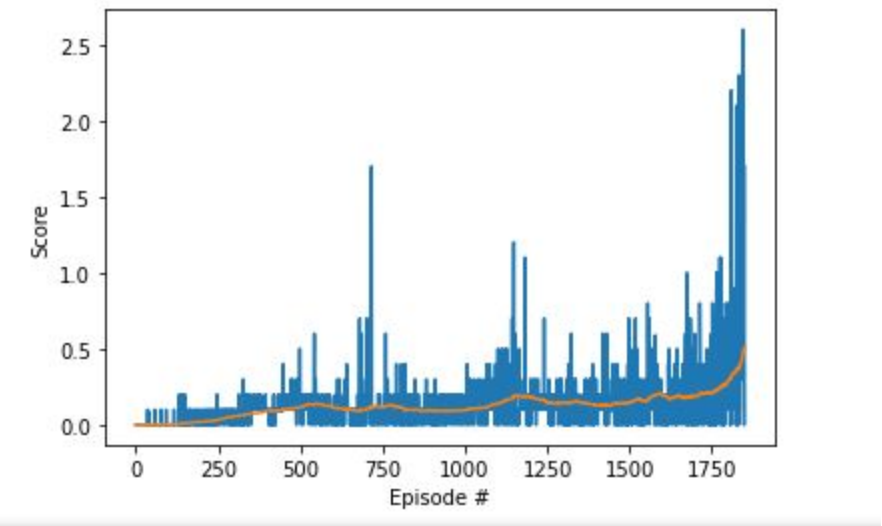
We have taken a batch size of 128 as we have enough data and it is not too high that it would slow down learning.

We have taken a gamma as 0.99 since we want to choose actions that will benefit us long-term.

The learning rate is kept low so we are able to maintain stability and ensure convergence.

The EPS is to control the noise. We reduce the noise added to action gradually to ensure that we rely more on the action choice from our network during the later stages of training.

Below is the plot of the score vs episode number for the trained agent. We solve the problem in 1850 episodes.




# Future Work

I would like to add prioritized experience sampling as well. Will try to use the Sum Tree data structure to improve on it.

I am also currently trying to run the network with different hyperparameters and architecture. And also adding next_action to the list of parameters in the buffer allowing more sharing between the two agents for collaboration.

I would also like to implement the alphaZero algorithm for continuous action space to see if we can get a more stable performance with less variance. Below is the link to paper about A0C paper that proposes an AlphaZero algorithm for continuous action state
https://arxiv.org/abs/1805.09613