# Project 3: Collaboration and Competition

## Problem Statement

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.

## Environment Analysis

- 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).
- After each episode, the rewards are added up that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores.
- Maximum of these 2 scores is taken – which 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.

## Implementation Details

The approach followed was:

- Understand the state and action space
- Implementation of MADDPG on top of DDPQ
- Train the agent with the above model over different epochs till the target is achieved
- Plot the results of learning process across episodes

## Algorithm and Implementation

The solution builds up on DDPG algorithm used in the previous project to train the model to solve the environment using multiple agents, which use Actor Critic method to approximate policy function required for our output.

The current implementation is over the top of resource mentioned in one of the Udacity practice exercise: https://github.com/udacity/deep-reinforcement-learning/tree/master/ddpg-pendulum

The algorithm MADDPG (https://arxiv.org/pdf/1509.02971.pdf) builds on top of DQN and DDPG by adding an Actor which estimates optimal policy with max reward using gradient ascent where as a Critic will estimate value using value-based approach – thus complementing learning of actor and enhancing speed of policy estimation. The gents in DDPG collaborate with each other and compete, where actors learn individually to maximize their personal score while critic is trained using experiences of both the agents.

We continue to use Ornstein Uhlenbeck Noise parameters with decay over the previously traditional epsilon greedy approach considering the continuous movement of the action space.

The gradient is also clipped in my Critic learner to prevent the issue of exploding gradients, a traditional problem in gradient based machine learning.

Finally, the first connected layer of NN was batch normalized to speed up the computation (as I saw that the training times were huge and learning was slow).

**Architecture**:
- The Neural Network used for both Actor and Critic are fully connected Neural Network with 3 layers:

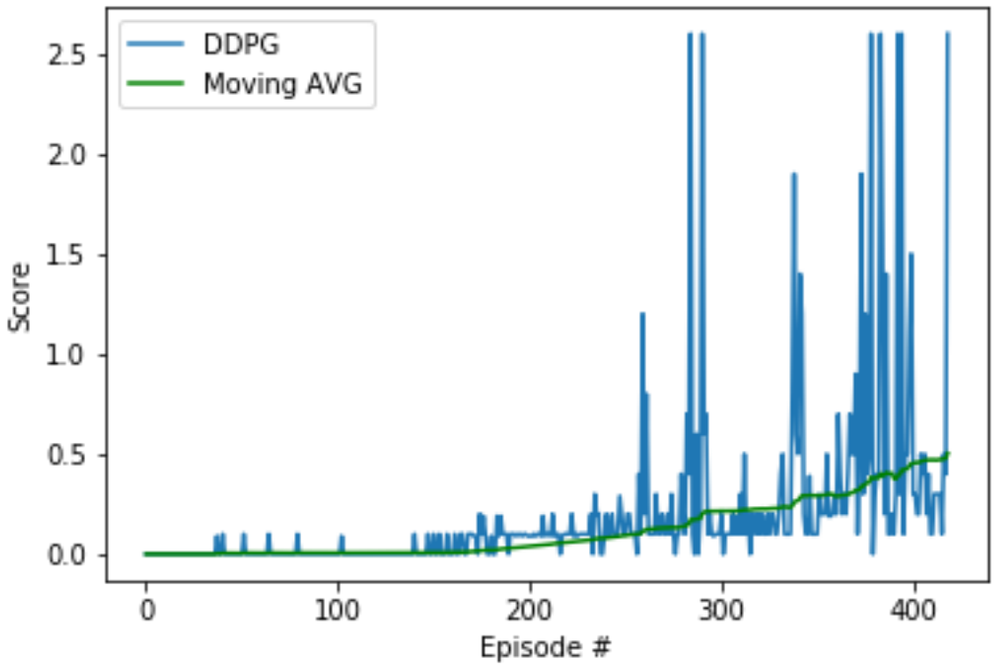
First layer: 400 neurons, Second layer: 300 Neurons, Third layer: Size of Action Space.
- All input connected layers were relu active, with output for first connect layer batch normalized
- The MADDPG used Ornstein Uhlenbeck noise process
- Experience Replay was used to allow agent to learn from previous experiences (like how we did in DQN impmenetation). The replay used a buffer and fetched experiences at random.

**Hyperparameters Used for learning:**

- Replay buffer size = 1e6
- Mini Batch Size = 256
- Discount factor (Gamma) = 0.99
- Soft update for Target Parameters (Tau) = 1e-3
- Learning Rate for actor (For Adam Optimizer) = 1e-3
- Learning Rate for critic (For Adam Optimizer) = 1e-3
- L2 weight decay = 0
- Update Interval = 2
- Learning passes = 10
- Epsilon(for eps greedy action selection) = 1.0
- Epsilon Decay = 1e-6
- Ornstein Uhlenbeck Noise Sigma = 0.2
- Ornstein Uhlenbeck Noise Theta = 0.15
- Gradient Clip = 1.0

# Results

The algorithm was able to achieve the goal in 419 Episodes (Over 100 episodes and at least +0.5 average reward, as specified in the project rubrichttps://review.udacity.com/#!/rubrics/1891/view) for both agents.




Output plot for scores vs episode number for mean and moving average using MADDPG algorithm

## Enhancements for the future

- Add more layers to neural network to improve the efficiency of the training
- Tweak hyperparameters or use hyperparameter search to ensure learning is achieved quickly with better performance
- Enhance the RL learning model with Prioritized Experience replay instead of choosing experiences at random
- Try out different algorithms like multi agent critic for mixed cooperative competitive environments etc