# Project 3: Collab and Compet

## Learning Algorithm

1. Algorithm

   I used a multi-agent DDPG to solve this Tennis environment. Each actor takes only its agent's state and outputs a vector representing an action chosen from a continuous action space. Critics, on the other hand, get states and actions of all agents and evaluate the action by computing a value function.

   For the first 20,000 episodes, I trained the MADDPG agent using only random actions. After that, I add mean-zero Gaussian noise to actions and I reduce the scale of the noise over the course of training.

2. Hyperparameters

   I used the following hyperparameters in my code:

   - BUFFER_SIZE = int(1e6) # replay buffer size
   - GAMMA = 0.99 # discount factor
   - TAU = 1e-3 # for soft update of target
   - LR_ACTOR = 1e-4 # learning rate of the actor
   - LR_CRITIC = 1e-4 # learning rate of the critic
   - WEIGHT_DECAY = 1e-5 # L2 weight decay
   - UPDATE_EVERY = 20 # how often to update the network
   - BATCH_SIZE = 512 # minibatch size
   - NUM_OF_UPDATES = 1 # how many times to update

## 3. Model architectures

The architecture of Actor-Critic and the number of nodes in each layer is as follows:

***Actor***

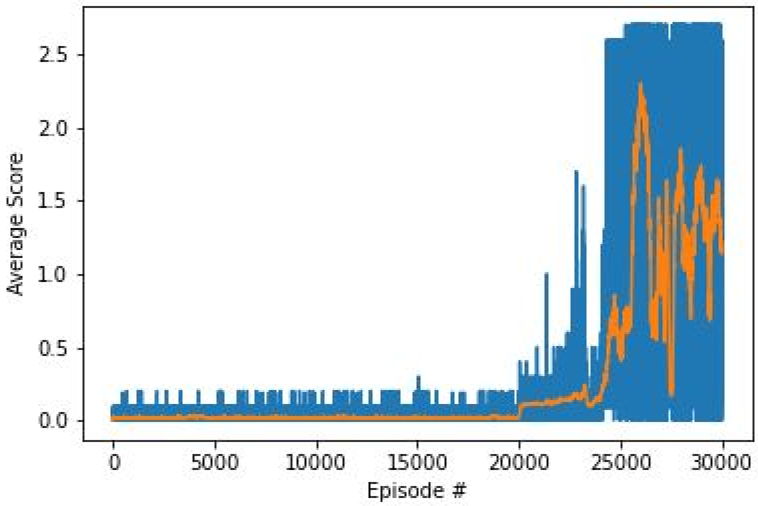
- Input layer = 24 (state_size)
- Hidden layer 1 (relu) = 128
- Hidden layer 2 (relu) = 128
- Output layer (tanh) = 4 (action_size)

***Critic***

- Input layer = 48 (state_size * num_agents)
- Hidden layer 1 (relu) = 128
- Concatenate = 132 (layer 1 size + (action size * num_agents))
- Hidden layer 2 (relu) = 128
- Output layer = 1

## Plot of Rewards

The reward graph is shown below. My agent got an average score (over 100 episodes) of +0.5 in about 25000 episodes.




## Ideas for Future work

1. Trust Region Policy Optimization (TRPO)
2. Truncated Natural Policy Gradient (TNPG)
3. Proximal Policy Optimization (PPO)
4. Distributed Distributional Deterministic Policy Gradients (D4PG)