Overview:

This project controls two tennis racket to hit call.

Reward: +0.1

Penalty: -0.1

Aim is to get the maximum combined score, hence play as long as possible.

## Approach:

I am using DDPG to solve this multi agent problem.

Parameters:

* BATCH\_SIZE = 128 # Minibatch size
* GAMMA = 0.99 # Discount factor
* TAU = 1e-3 # For soft update of target parameters
* LR\_ACTOR = 2e-4 # Actor learning rate
* LR\_CRITIC = 2e-4 # Critic learning rate
* WEIGHT\_DECAY = 0 # L2 weight decay

I use two linear layer then a batch normalization layer, and then finally a linear layer. Idea is not to make the model too complex so that it can train at a fair speed.

In this model I am using a new type of weight initializer to see if it can make it faster : ***torch.nn.init.xavier\_uniform(self.fc1.weight)***

My Policy network is designed like this:

def \_\_init\_\_(self, state\_size, action\_size, seed, fc1\_units=300, fc2\_units=200):

* super(Actor, self).\_\_init\_\_()
* self.seed = torch.manual\_seed(seed)
* self.fc1 = nn.Linear(state\_size, fc1\_units)
* self.bn1 = nn.BatchNorm1d(fc1\_units)
* self.fc2 = nn.Linear(fc1\_units, fc2\_units)
* self.bn2 = nn.BatchNorm1d(fc2\_units) ## added 2nd batch norm Saurabh
* self.fc3 = nn.Linear(fc2\_units, action\_size)
* self.reset\_parameters()

Hope this explanation helps.

## Training Graphs