1. Environment

The environment contains two agents playing tennis. The two agents control rackets to bounce a ball over a net. The action space of each agent is 2 dimensions which is continuous. The agent can move towards (or away from) the net and jump. Every time an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets the ball hit the ground or out of bound, it will receive a reward of -0.01. The observation space is 8 variables containing the position, velocity of the ball and the racket. Each agent receives its own local observation. The goal of each agent is to keep the ball in play. The problem is considered solved if the agents get an average score of +0.5 over 100 consecutive episodes, after taking the maximum over both agents.

2. Algorithm

We implement Multi Agent Deep Deterministic Policy Gradient (MADDPG). In our algorithm, we have two DDPG agents. The DDPG agent in MADDPGt had a local actor and a global critic. Moreover, the two agents share the replay buffer. DDPG was more suitable than DQN because the environment of the tennis had a continuous action space. DDPG has two deep neural networks: an actor network and a critic network. The actor is used to determine the optimal policy deterministically. The critic evaluates the best belief action from the actor. For each network, there is a real network and a target network. The real network and the target network is used to increase stability. The real network is actively trained and soft update is applied to the target network, slowly blending the network weights. For training, the DDPG uses a replay buffer to store experiences.

3. Hyper parameters

The following are the hyper parameters

In ddpg.py,
```
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1e-4         # learning rate of the actor
LR_CRITIC = 1e-3        # learning rate of the critic
WEIGHT_DECAY = 0.0000   # L2 weight decay
```

In maddpg.py,
```
BATCH_SIZE = 600                #Batch size for training
BUFFER_SIZE = int(1e7)          #Buffer size of the replay buffer
RANDOM_SEED = 1                 #Random seed
GAMMA = 0.95            #Learning rate
```

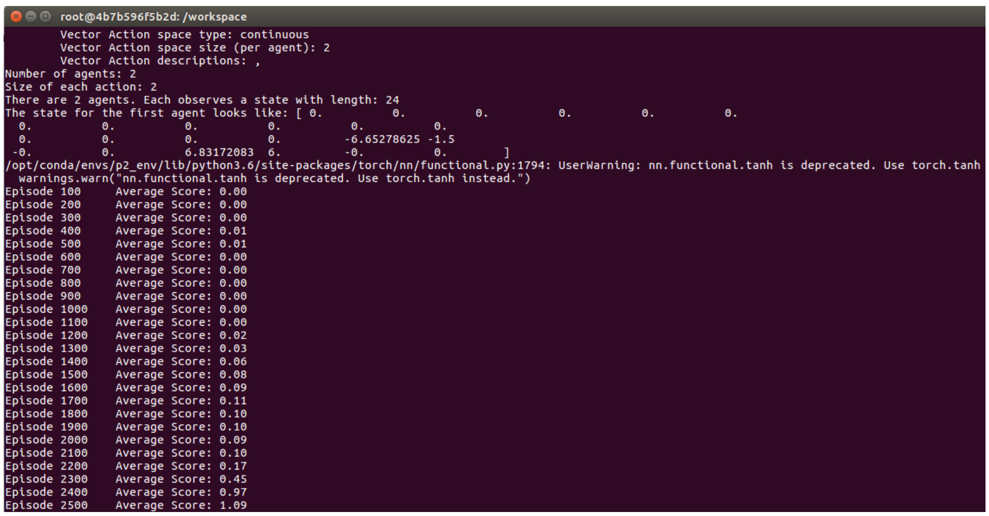
In main.py,
```
n_episodes = 100000     #Number of episodes
max_t = 1000000                 #Number of steps
print_every=100                 #when to print
Train = 0               #1 to train, 0 to test
```

4. Architecture

The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) consists of actor networks and a critic networks. The actor maps the state to action. The actor network has two hidden layers. The size of the hidden layers are 512 and 128. The critic maps the state and action pairs of all the agents to q-values. The critic network has three hidden layers. The size of the hidden layers are 256, 512 and 64. For stability, each network has a real and target network.

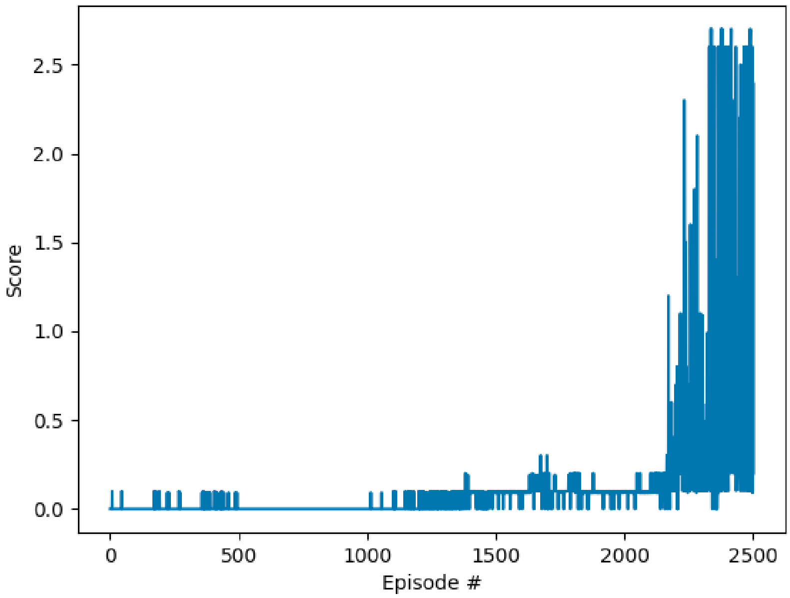
5. Results & Figures

The MADDPG agents were trained for 2500 episodes. After 2300 episodes, the average score was greater than 0.5. We observed that during training, the score didn't increase very much in the beginning. After 1300 episodes, the average scores started to become better. After training for 2200 episodes, the scores became above 0.5. The average scores are printed on the terminal is shown below.



```
root@4b7b596f5b2d: /workspace
        Vector Action space type: continuous
        Vector Action space size (per agent): 2
        Vector Action descriptions: ,
Number of agents: 2
Size of each action: 2
There are 2 agents. Each observes a state with length: 24
The state for the first agent looks like: [ 0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.         -6.65278625 -1.5
 -0.          0.          6.83172083  6.         -0.          0.        ]
/opt/conda/envs/p2_env/lib/python3.6/site-packages/torch/nn/functional.py:1794: UserWarning: nn.functional.tanh is deprecated. Use torch.tanh
  warnings.warn("nn.functional.tanh is deprecated. Use torch.tanh instead.")
Episode 100     Average Score: 0.00
Episode 200     Average Score: 0.00
Episode 300     Average Score: 0.00
Episode 400     Average Score: 0.01
Episode 500     Average Score: 0.01
Episode 600     Average Score: 0.00
Episode 700     Average Score: 0.00
Episode 800     Average Score: 0.00
Episode 900     Average Score: 0.00
Episode 1000    Average Score: 0.00
Episode 1100    Average Score: 0.00
Episode 1200    Average Score: 0.02
Episode 1300    Average Score: 0.03
Episode 1400    Average Score: 0.06
Episode 1500    Average Score: 0.08
Episode 1600    Average Score: 0.09
Episode 1700    Average Score: 0.11
Episode 1800    Average Score: 0.10
Episode 1900    Average Score: 0.10
Episode 2000    Average Score: 0.09
Episode 2100    Average Score: 0.10
Episode 2200    Average Score: 0.17
Episode 2300    Average Score: 0.45
Episode 2400    Average Score: 0.97
Episode 2500    Average Score: 1.09
```

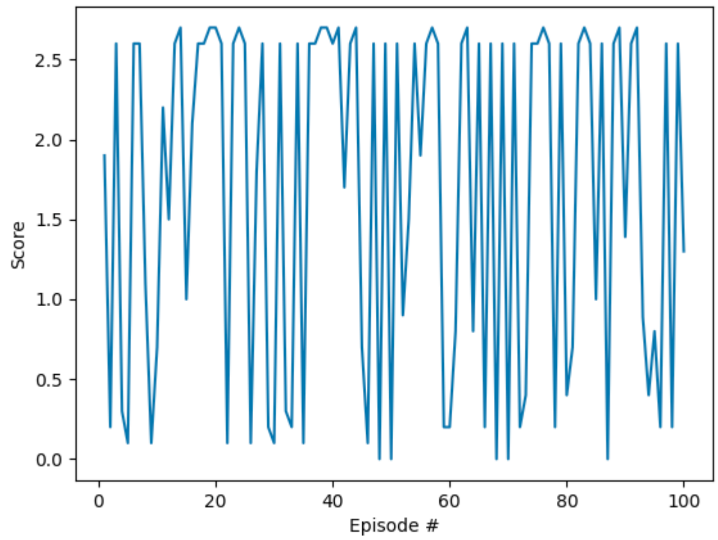
We also plotted the scores during training below. The score up until 1000 episodes didn't have any significant increase in score. After 2000 episodes, the average scores seemed to have significantly increase.



When we average the 100 episodes while testing, the average score of the agent was 1.68 like shown below. We can see that after training the agent, the agent is able to score above 0.5 in average.

```
root@4b7b596f5b2d: /workspace
Episode 66 Score: 0.20
Episode 67 Score: 2.60
Episode 68 Score: 0.00
Episode 69 Score: 2.60
Episode 70 Score: 0.00
Episode 71 Score: 2.60
Episode 72 Score: 0.20
Episode 73 Score: 0.40
Episode 74 Score: 2.60
Episode 75 Score: 2.60
Episode 76 Score: 2.70
Episode 77 Score: 2.60
Episode 78 Score: 0.20
Episode 79 Score: 2.60
Episode 80 Score: 0.40
Episode 81 Score: 0.70
Episode 82 Score: 2.60
Episode 83 Score: 2.70
Episode 84 Score: 2.60
Episode 85 Score: 1.00
Episode 86 Score: 2.60
Episode 87 Score: 0.00
Episode 88 Score: 2.60
Episode 89 Score: 2.70
Episode 90 Score: 1.39
Episode 91 Score: 2.60
Episode 92 Score: 2.70
Episode 93 Score: 0.89
Episode 94 Score: 0.40
Episode 95 Score: 0.80
Episode 96 Score: 0.20
Episode 97 Score: 2.60
Episode 98 Score: 0.20
Episode 99 Score: 2.60
Episode 100 Score: 1.30
Episode 100     Average Score: 1.68
```

We also plotted the scores during testing below. There are outliers, but in general the scores are above 0.5.



6. Future Idea

Training with a higher number of episodes - In future work, we should check if performance increases as the number of training episodes increase. When we trained the MADDPG, it took us a very long time. It could be that it takes even longer.

Configure the hyper parameters - In future work, more testing should be done with the hyper parameters. When we were experimenting with the hyper parameters, we found that the score was sensitive to the number of hidden layers in the actor/critic network and to the buffer size. In the future, we need to try configuring the parameters more.

Attempt the optional challenge - In this work, we used the tennis environment. The soccer environment is more complicated to solve. The current MADDPG needs to be improved to be acceptable to more agents. The MADDPG is hardcoded to have 2 agents. This needs to be improved to accommodate more agents. Also, the current code may not work with a more complicated environment.