CSE 546: Reinforcement Learning
Final Project Report

Name: Amal S Namboodiri
Roll Number: 506070302

Environment:

I have used a table tennis environment where two adversarial agents must learn to defeat the opponent agent. The environment also includes a point counter which I will be using as an indicator of how well the agent performs.

- Each agent is characterized by 24 different states that include the agent bat positions (x,y,z), ball positions (x,y,z), ball velocities (x,y,z) at three consecutive timestamps.
- In my implementation, I have allowed the agent to make 4 possible actions: moving bat up, down, left or right.
- A reward of 0.05 is received if the agent is able to successfully hit the ball into the opponent's area. A reward of -0.005 is received if the agent fails to hit the ball.

Model Architecture:

DQN Model: I have made changes to this model after the checkpoint to give better results.

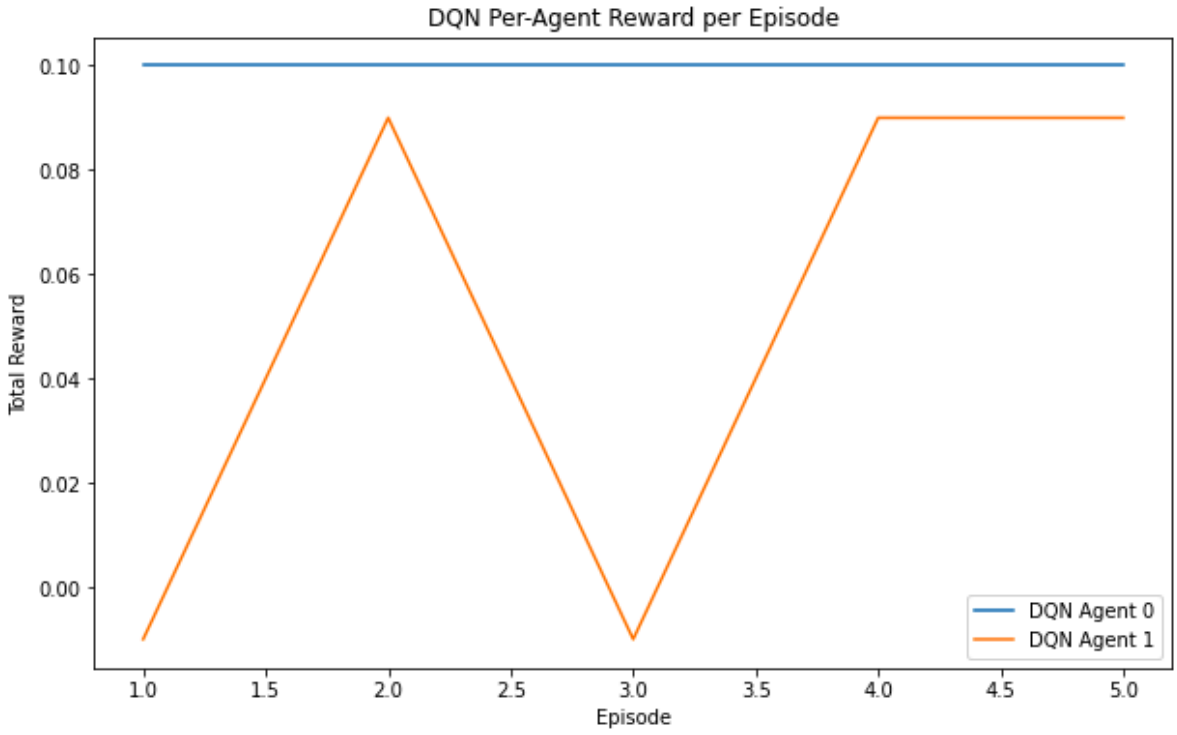
- The model has 24 input nodes, 128 hidden nodes and 4 output nodes.
- The agent’s actions are discretized as DQN can only handle discrete actions. Each agent can move either up, down, left or right.
- Learning rate: 5e-4
- Batch size: 128
- Epsilon Decay: 0.999
- Replay Buffer: 100,000

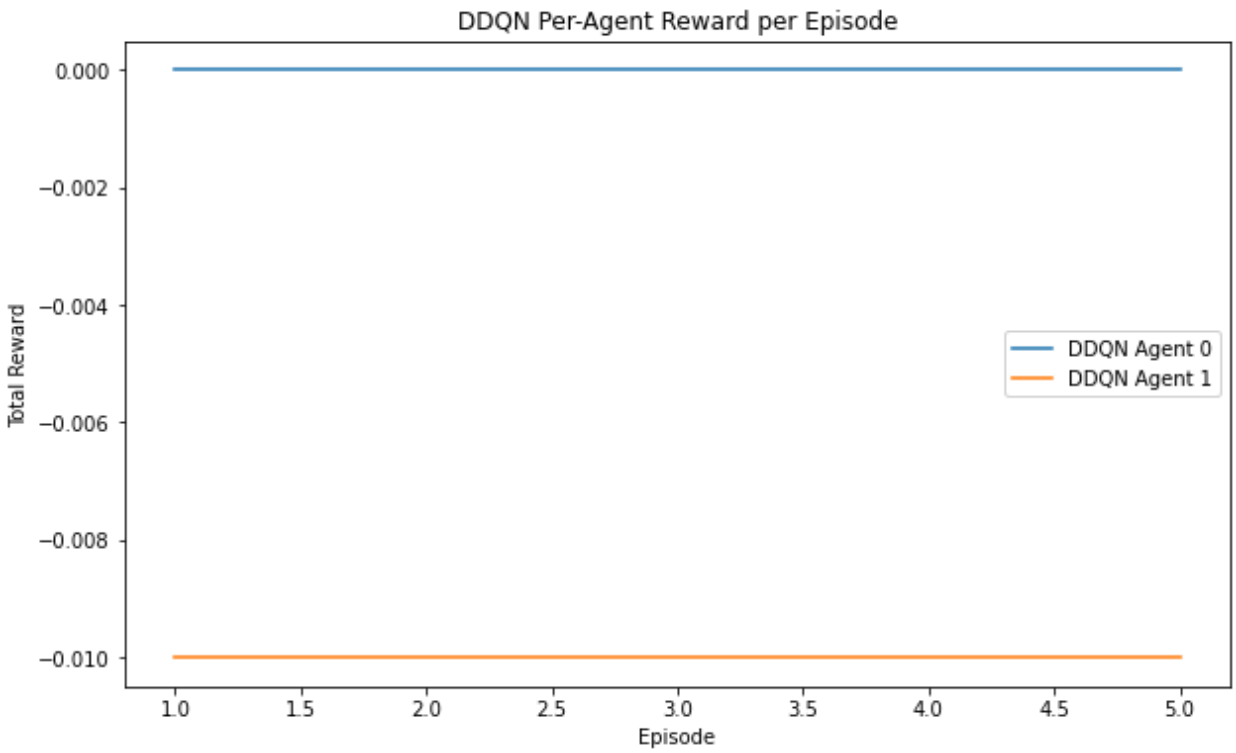
DDQN Model: The model and hyperparameters remain the same as the DQN model. This model also requires discretized actions

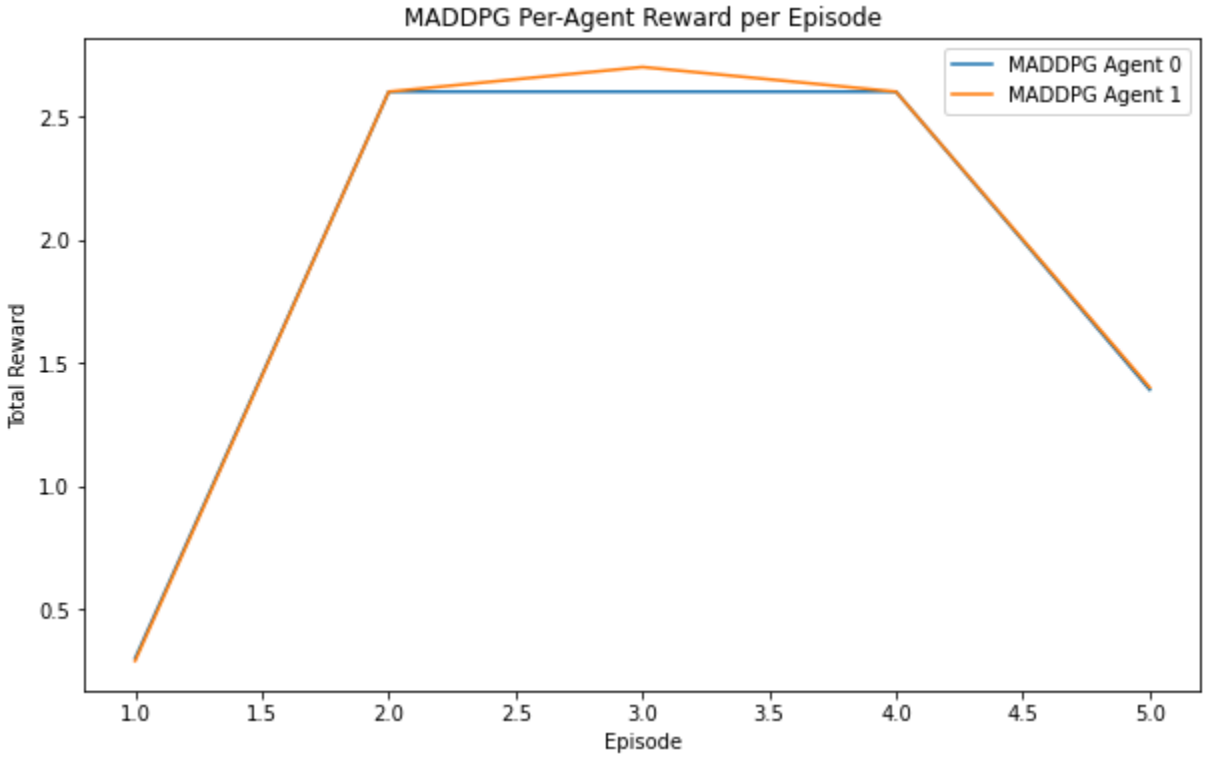
DDPG Model:

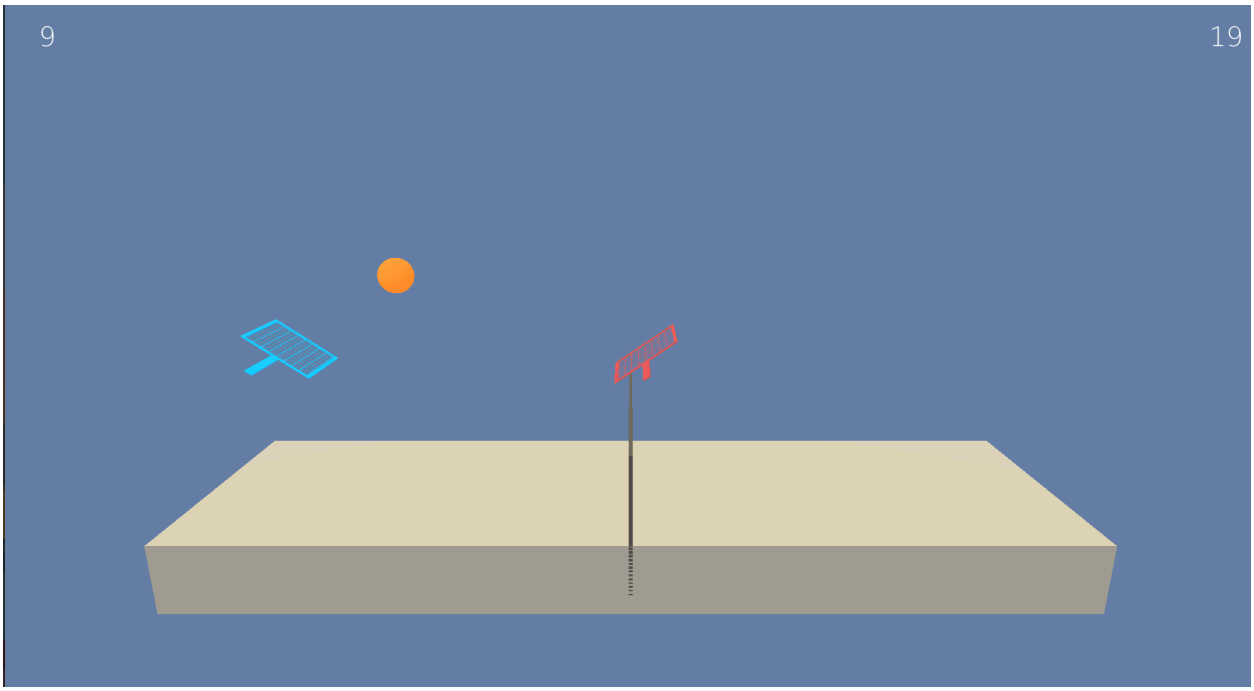
- Two models are used:
  - Actor network: Outputs a continuous action vector
  - Critic network: Outputs a scalar Q-value
- No discretization is required as this model can handle continuous data.
- Learning rate: 1e-4
- Batch size: 128
- Replay Buffer: 1,000,000
- Soft update of target parameters (tau): 0.001
- L2 decay coefficient: 1e-6

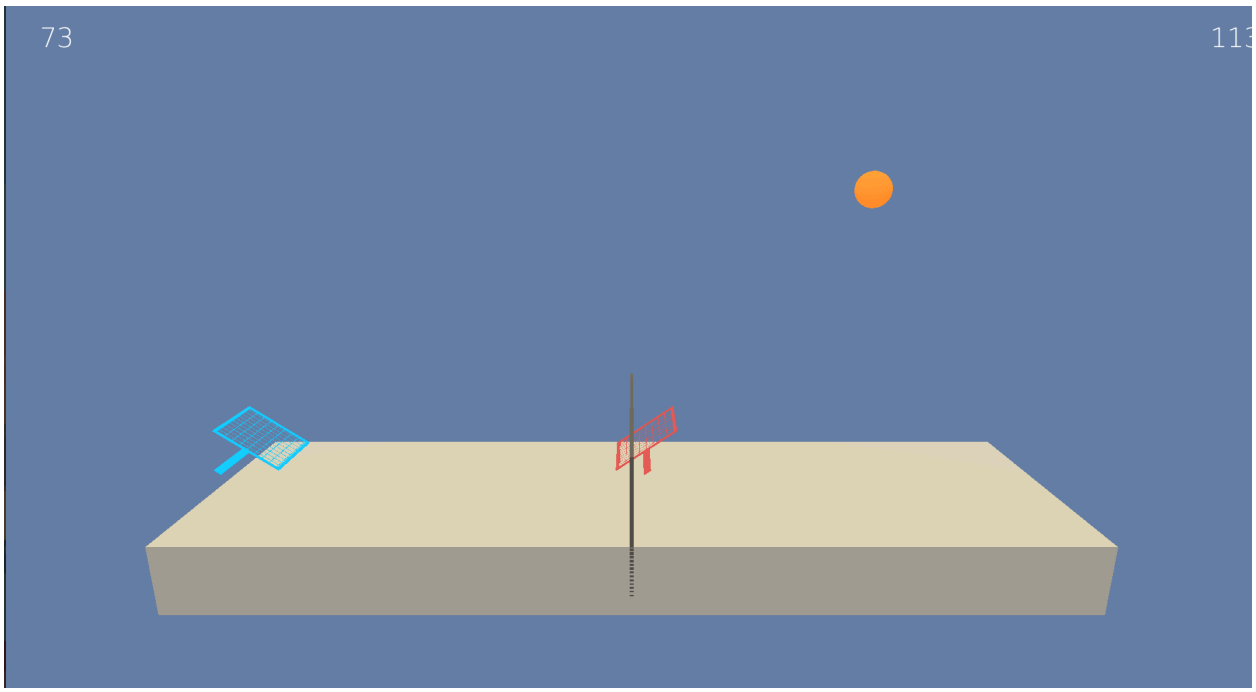
Results:




DDQN Per-Agent Reward per Episode
0.000
−0.002
−0.004
−0.006
−0.008
−0.010
Total Reward
DDQN Agent 0
DDQN Agent 1
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Episode


MADDPG Per-Agent Reward per Episode
2.5
2.0
1.5
1.0
0.5
Total Reward
MADDPG Agent 0
MADDPG Agent 1
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Episode





RESULTS:

We can see that the ddpg model is performing way better than DQN and DDQN model. For the DDQN model it looks like only the one agent has

learned. DDPG performs better because it can handle continuous data easily.

References:

- **For Environment:** https://github.com/ravishchawla/Reinforcement-Learning-NanoDegree/tree/master/Project%203%20-%20Collaboration%20and%20Competition

- https://pytorch.org/docs/stable/index.html