# Report

## Learning Algorithm - MADDPG

This algorithm is best for problems where actions and states are in continuous space. It is an extension of the normal DDPG Algorithm but for 2 or more agents. The DDPG Algorithm worked fine for the single agent problem so extending it to a multiagent problem was easy. Although the soft-actor critic and other distributed algorithms may perform better, but this was simple to implement so it was used. In this project, the 2 Agents had different replay buffers because they will never be in the same state due to how the environment is created so it doesn't make sense to train both agents with the same buffer. Noise was added to the agent's actions to because continuous actions are used. I found a lot of clipping happening initially, so the time varying noise helped add exploration and stabilize training.

The Hyperparameters where chosen from the paper but also a lot of decision was made from trial and error due to slow and unstable initial training. The Architecture worked great for a single agent so extending it to a multiagent wasn't difficult.

### Hyperparameters

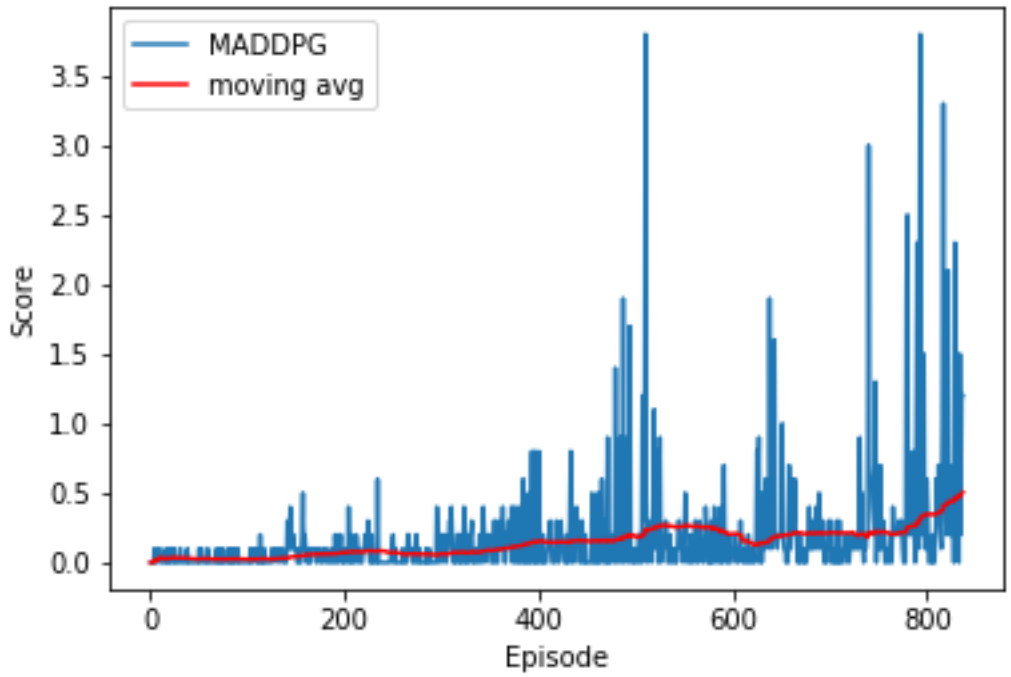
BUFFER_SIZE = int(1e6) # replay buffer size
BATCH_SIZE = 128 # minibatch size
GAMMA = 0.95 # discount factor
TAU = 1e-3 # for soft update of target parameters
LR_ACTOR = 1e-4 # learning rate of the actor
LR_CRITIC = 1e-3 # learning rate of the critic
WEIGHT_DECAY = 0 # L2 weight decay
EPOCHS = 1 UPDATE_EVERY = 5

## Model Architecture

- 3 Fully connected layers with 256, 128, 64 units respectively batch normalization on the input

An average score of +0.5 was required to solve the problem, the agent gradually attained this value at 1660 episode and a day of training.

## Reward Plot


MADDPG
moving avg
Score
Episode
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800