PROJECT #3 COLLAB-COMPETE TENNIS MULTIAGENT GAME

For this project the goal is to solve a version of the tennis game for multiagent environment, where 2 agents play against each other and learn from each other 'experiences'. For my solution, I used a multi-agent version of the DDPG algorithm since the action space in continuous in the interval [-1, 1]. The goal is to reach an average score of +0.5 over 100 consecutive episodes after taking the maximum over both agents.

The algorithm: multi-agent DDPG is a wrapper for DDPG which uses a replay buffer to train an action, and a target network to stabilize training. Also, multi-agent DDPG is a deterministic policy, thus, it uses a policy gradient method to minimize the cost function. It is also called off-policy because it estimates the Q function which, in turn, it is used to learn the policy The training is done online by sampling past 'experiences' from the replay buffer which are also continuously collected and shared (collaboratively) among the agents. The agents 'learn' by sampling from the replay buffer these shared 'experiences', thus allowing cumulative learning (what one agent learns can be used by the other agent in a future state).

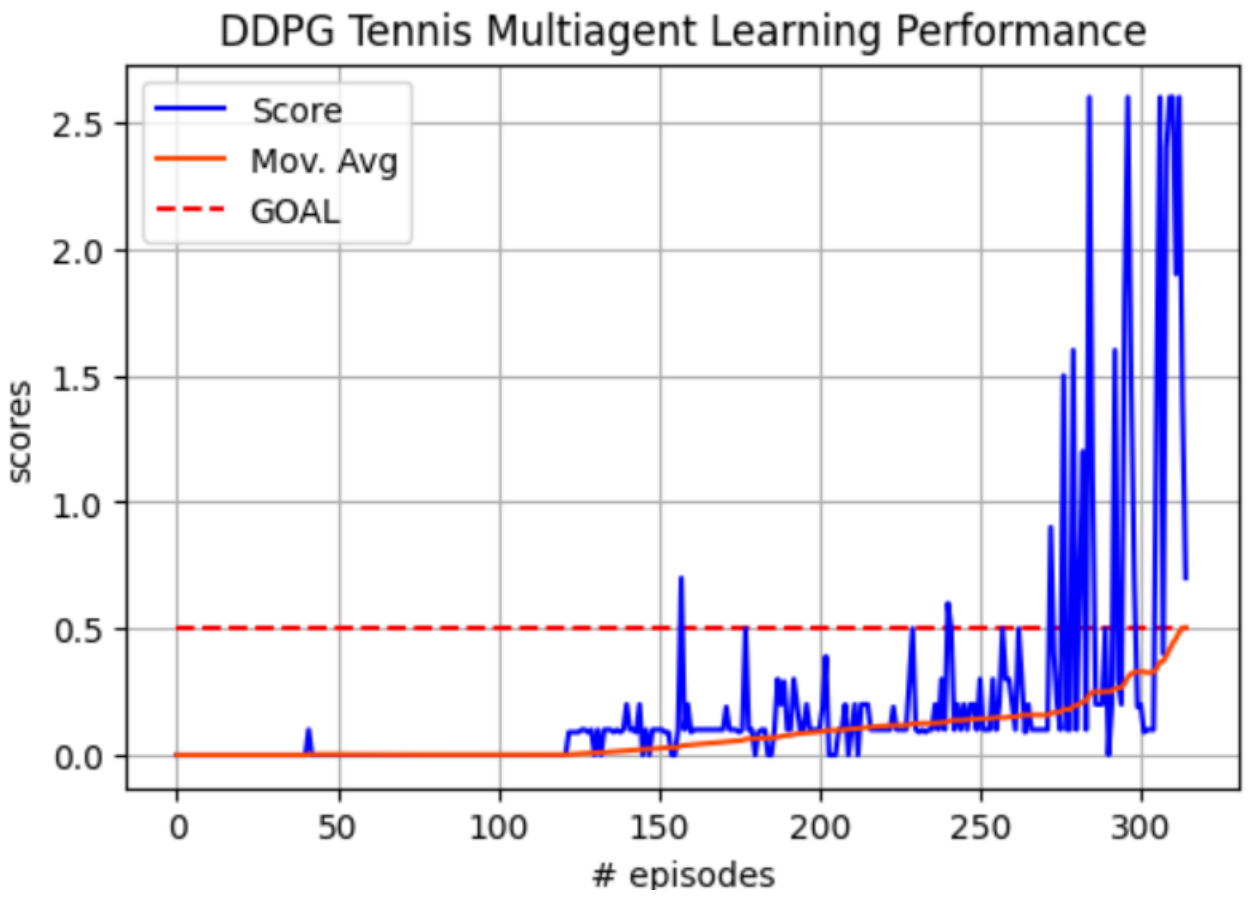
Multi-Agent DDPG Network Architecture: This is an actor-critic algorithm where the actor approximates the optimal policy deterministically and output the best believed action for a given states, the critic learns to evaluate the optimal action value function by using the actor's best believed action. There are 2 agents in the tennis game, each has an observation space of size 8. Each agents' action space is of size 2 and continuous. Their network consisted of 2 fully connected layers with ReLU activation functions and a fully connected output layer. The critic also takes as input the 8 values for the observations space, but in this case, its first hidden layer is concatenated with the actor output layer and passed as input to the critic 2nd hidden layer. This way, the best believed action from the actor is also used in the critic to learn an optimal action-value function $Q(s, \mu(s; \phi); \theta)$.

Hyperparameters:

Each agent has an actor-critic network (2 actors, 2 critics)
Actor First Hidden Layer: 512
Actor Second Hidden Layer: 256

Critic First Hidden Layer: 512
Critic Second Hidden Layer: 256
BUFFER_SIZE = 1.e6
BATCH_SIZE = 128
GAMMA = 0.99
TAU = 1e-3
EPS = 0
EPS_DECAY = 0
LR_ACTOR = 1e-3
LR_CRITIC = 1e-3
WEIGHT_DECAY = 0

Future things to try: Some of the other algorithms to consider are Twin Delayed DDPG (TD3) which is an algorithm based on improvements, and also the PPO algorithm. Now with the results from DDPG I'd like to try these other 2 and compare performances. Other things to try are reducing the number of units in the hidden networks and see if it speeds up learning, (it look about 3.5 hours to achieve 0.5 average score and from there it took another 6 hours to get to 1.0 score) increase the batch_size in replay buffer and decrease the rate of learning from experiences, which should also speed up the process, and finally change the noise process (Ornstein-Uhlenbeck) to Gaussian noise since some literature indicates that either one is effective. Other changes to consider are prioritized experience replay, another would be read the Alpha Zero paper and apply algorithm.


DDPG Tennis Multiagent Learning Performance
Score
Mov. Avg
GOAL
scores
episodes
0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300