Project3: Collaboration and Competition

In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in play.

The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping.

The task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents). Specifically,

- After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores. We then take the maximum of these 2 scores.
- This yields a single **score** for each episode.
- The environment is considered solved, when the average (over 100 episodes) of those **scores** is at least +0.5.

Implementation

For this environment, a modification of DDPG was used i.e. MADDPG (Multi-Agent Deep Deterministic Policy Gradient). Components from the paper *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments* were implemented to cater for the multi-agent environment. Where traditional actor-critic methods have a separate critic for each agent, this approach utilizes a single critic that receives as input the actions and state observations from all agents. This extra information makes training easier and allows for centralized training with decentralized execution. Each agent still takes actions based on its own unique observations of the environment.

Few techniques implemented which improved the overall performance were:

1. Batch Normalization – The neural network weights are normalized according to the batch size, which speeds up training and helps in better convergence
2. Soft Updates - Instead of updating after every N step like in DQN, we update a small percentage from the target network
3. Fixed Targets - 2 targets for actor and critic model similar to DQN
4. Shared Experience Replay Buffer - We maintain a replay buffer from which we randomly sample experiences to train our model in order to avoid unwanted correlation between the sequential observations. Both the agents can learn from the experiences of each other
5. Ornstein-Uhlenbeck was implemented which is correlated to previous noise and therefore tends to stay in the same direction for longer durations without cancelling itself out. This allows the agent to maintain velocity and explore the action space

with more continuity. A higher number of EPS_START was used to allow more random actions in the beginning in-order to pick up a good signal i.e. When ball touches racket.

Network Architecture

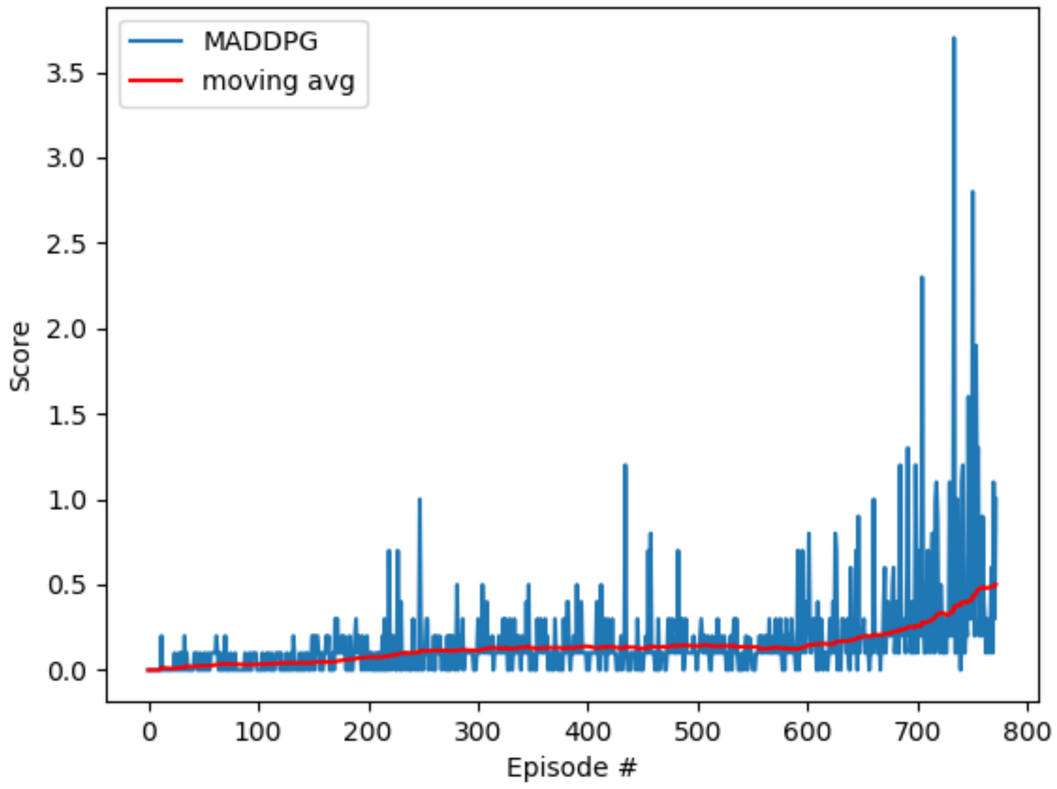
Actor Model	**Shared Critic Model**
Input Layer = 24 units * 2	Input Layer = 24 units * 2
Hidden Layer = 256 units with RELU and Batch Norm	Hidden Layer = 256 units with RELU and Batch Norm
Hidden Layer = 256 units with RELU and Batch Norm	Action input = 2 units * 2
Action Output with tanh activation (2 units)	Hidden Layer with input from layer 2 + action input with RELU and Batch Norm
	Q-Value output = 1 unit

Hyper Parameters

The following hyper parameters were used:

Hyperparameter	**Value**
Replay buffer size	1e6
Batch size	128
Discount factor/Gamma	0.99
Actor Learning rate	1e-3
Critic Learning rate	1e-3
Number of episodes	2000
Max number of timesteps per episode	1000
OU_SIGMA	0.2
OU_THETA	0.15
EPS_START	5

Results



The environment gets solved in 672 episodes!

Suggestions for improvements

1. Adding Prioritized Experience Replay might improve the performance by taking advantage of experiences that lead to higher reward
2. Exhaustive grid search on a few different set of hyper parameters. It’s very hard to fine-tune RL agents, as small differences to hyper params in different combinations make a lot of difference. Further research is need in this direction to be able to hypothesize and choose hyper params more intuitively.
3. Experiment with variations of other algorithms such as PPO, DP4G etc.