Project 3: Collaboration and Competition

This report includes a summary of the learning algorithm for the project titled "Collaboration and Competition", its outcome and ideas for follow-up work.

1 – Environment

The environment is the Tennis from the Unity ML-Agents, see Figure 1. 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.

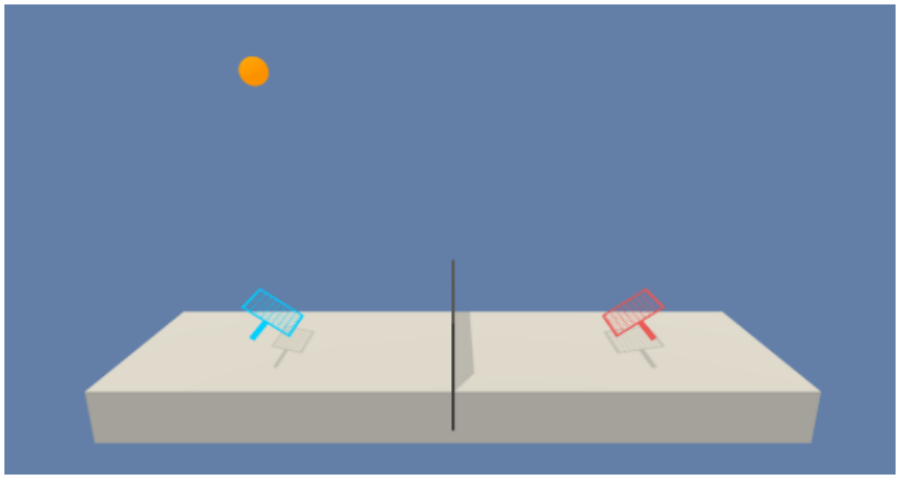

Figure 1: The Tennis environment from the Unity ML-Agents. The goal is to send the ball over the net without letting it touch the ground or go out of bounds.

The task is episodic, and in order to solve the environment, the 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.

2 – Learning Algorithm

In this project, I used the Multi-agent Deep Deterministic Policy Gradient (MADDPG) algorithm [https://arxiv.org/abs/1706.02275]. I used two DDPG agents [https://arxiv.org/pdf/1509.02971] with the same network configuration in this algorithm.

DDPG is an actor-critic, model-free algorithm based on the deterministic policy gradient which can be used with continuous action spaces. DDPG utilizes the strong suits of the DQN agent, and adapts them to the continuous action domain.

MADDPG adopts the decentralized actor, centralized critic approach. Each actor takes actions based on its own observation of the environment. However, each critic observes the tuples of both agents in order to train more effectively. As a result, training is centralized by using a shared experience replay buffer.

The following information is shared by both agents since their configurations are identical. I used Adam for learning the neural network parameters with the learning rate of 2×10^{-4} for both the actor and critic. For Q I used no weight decay. I used a discount factor of $\gamma = 0.99$. For the soft target updates I used $\tau = 0.01$. The neural networks used the ReLU for all hidden layers. The final output layer of the actor was a Tanh layer, to bound the actions. For the actor, I used batch normalization in the first two layers, and for the critic only in the first layer. The networks had 2 hidden layers with 128 and 64 units respectively. Actions were not included until the 2nd hidden layer of Q. The final layer weights and biases of both the actor and critic were initialized from a uniform distribution $[-3\text{x}10^{-3}, 3\text{x}10^{-3}]$. This was to ensure the initial outputs for the policy and value estimates were near zero. The other layers were initialized from uniform distributions $\left[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}\right]$ where f is the fan-in of the layer. I trained with a batch size of 128 using a replay buffer size of 10^5.

For the exploration noise process I used temporally correlated noise in order to explore well in physical environments that have momentum. I used an Ornstein-Uhlenbeck process with $\theta = 0.15$ and $\sigma = 0.2$.

3 – Results

Figure 2 shows the average score per episode and the moving mean of the average score over the last 100 episodes. It can be seen that after 666 episodes, the average score of 0.5135 was obtained. At that episode, the training was stopped as the requirement for the agent's success was achieved.

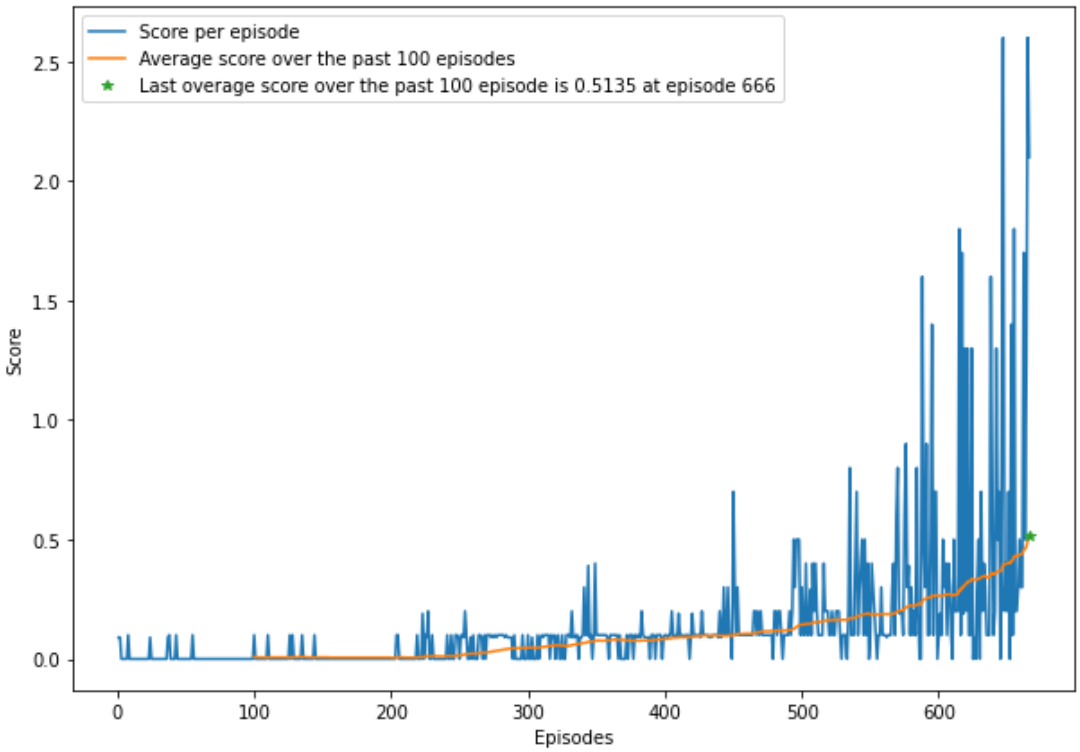

Figure 2: Average score per episode.

4 – Ideas for follow-up work

The following ideas come to mind for improving the agent's performance:

- Hyperparameter tuning
- Use Prioritized experience replay
- Adding noise to the parameter of the neural networks
- Try the soccer environment