Project 3: Collaboration and competition using actor-critic reinforcement learning

Two agents were trained to compete in the Unity Environments Tennis task. The goal was for the agents to keep the ball in the air for as long as possible. As a result of wanting to keep the ball in the air, the agents were actually collaborating to collectively keep the ball in the air for as long as possible. A deep deterministic policy gradient (DDPG) actor critic reinforcement learning approach was used to train the agents to continuously control moving the rackets to successfully hit the ball. Actions were selected on-policy based on the output of an Actor policy approximated with a multi-layered perceptron neural network. The neural network had hidden layers of 256, 128, and 64 with rectified linear unit (ReLU) activation functions between each layer. The output layer consisted of a hyperbolic tangent (TanH) function with the size of the action space for a continuous output. Additionally, batch normalization was used between layers to help prevent exploding gradients.

A Critic network evaluated the agent's actions to help speed up the policy gradient approach. The Critic approximated the value of the agent's action given the current state in the environment with a neural network. This network had hidden layers of 256, 130 (128 + 2 actions), and 64 with an output size of one to estimate a single value of the state and action pair. This network also had ReLU activation functions between each layer and batch normalization to help prevent exploding gradients. A sigmoid was used at the output layer to help stabilize training.

Two separate neural networks with the same structure for both the actor and critic were used to train the agents, namely a primary network that updated at every step and a target network that updated 15 times after every 20 steps. This provided a fixed value function approximation and target policy to avoid correlation between rewards and weight updates. Additionally, a memory replay buffer of 1,000,000 state, action, reward, next state, and done tuples were saved for both agents. During training, tuples were sampled from the memory buffer at random for an uncorrelated experience learning across both agents.

Over the course of 4 seed iterations, an average of 900 ± 221 episodes were required to achieve an average reward goal of 0.5 over 100 episodes (Figure 1). The average reward was determined by selecting the maximum reward out of the two agents for each episode.

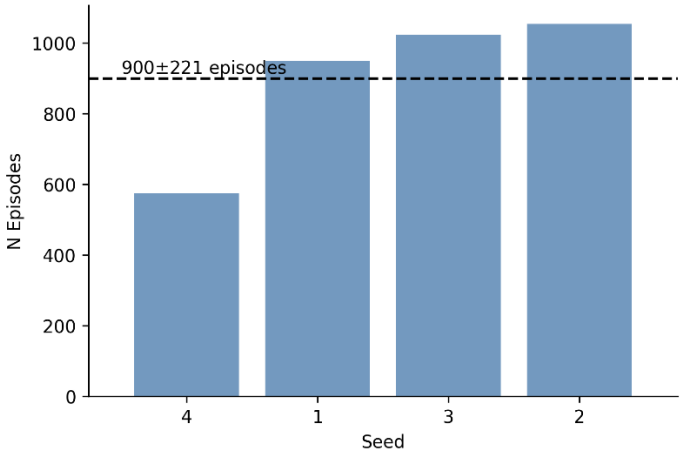

Figure 1: Number of episodes to reach reward goal over 4 seeds

An example of a single seed of the two agents' training rewards (seed=4) over time is shown in Figure 2. Training was sporadic depending on the seed, but in general stabilized after a significant amount of episodes once the exploration with noise was reduced.

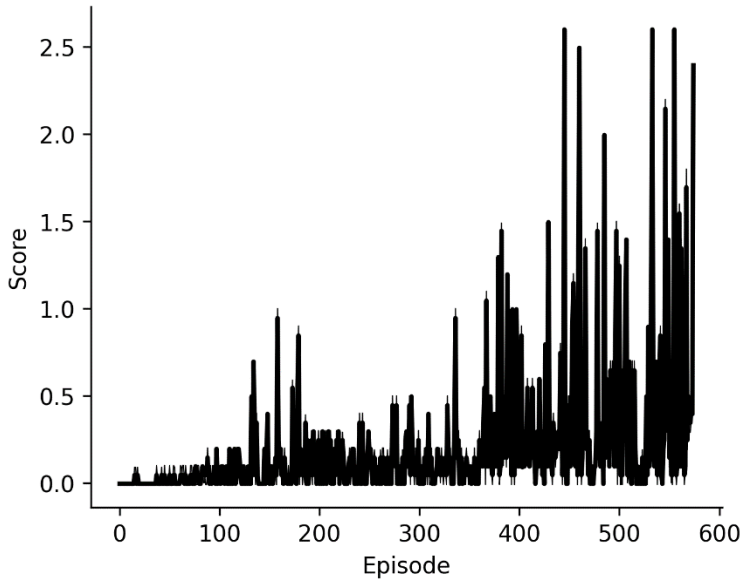

Figure 2: Seed 4 Agent's score throughout training (achieved goal at 575 episodes)

In the future, prioritized experience replay could be considered to help speed up training. Instead of sampling at random from shared buffer experience, samples with higher importance could be weighted heavier than samples with little importance. Additionally, alternative actor-critic methods could be used to better take advantage of the multiple agent's experiences rather than just a combined buffer. Methods such as PPO, A3C, or D4PG that distribute the task of gathering experience may help the agents converge faster with more stable learning.