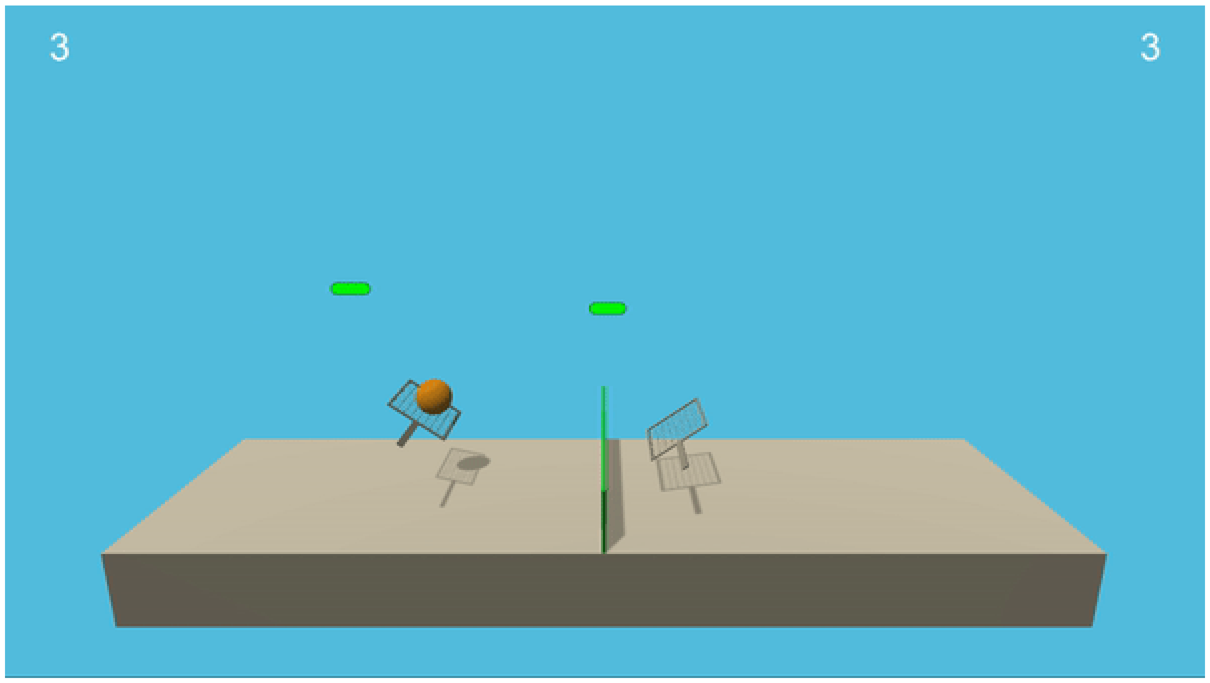
Project 3: Multi-Agent Tennis RL



I: Project Summary

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.

II: Learning Algorithm

We use a Multi-Agent Deep Deterministic Policy Gradient (DDPG) Actor-Critic algorithm to teach each agent in the table tennis environment how to interact with the environment to learn how to achieve its goal of keeping the ball in play.
The main idea and details behind DDPG can be found in this paper here, but in general, in DDPG, we have to train two neural networks, one for the actor, and another for the critic.

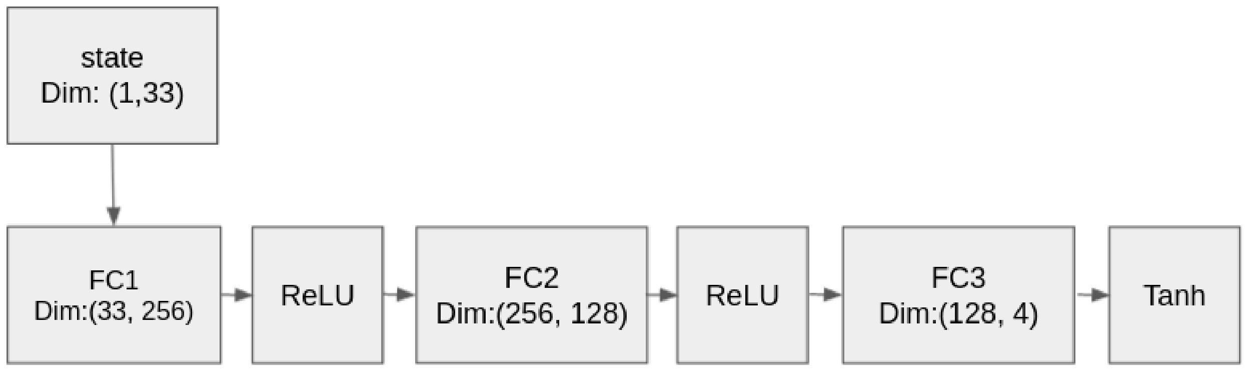
In addition to the vanilla DDPG implementation, the experience relay technique is used to learn from individual experience tuples multiple times, recall rare occurrences, and make better use of our experience. For experience relay, a replay buffer of size 100,000 is used. Since we have multiple agents in this environment, we use a shared buffer between both agents, and with each learning step, each agents learns from multiple experiences.

The table below shows the hyperparameters used to train the model:

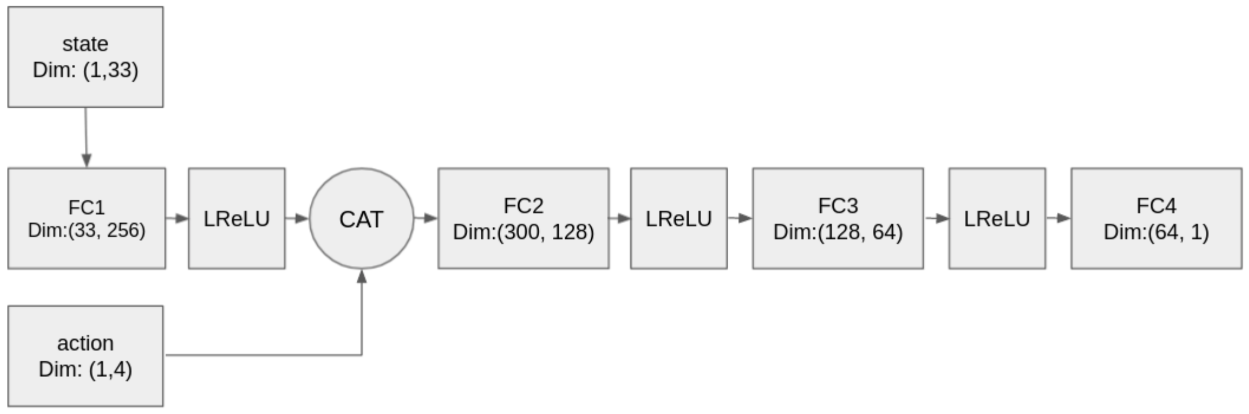
BUFFER_SIZE (replay buffer size)	100,000
BATCH_SIZE (minibatch size)	128
GAMMA (discount factor)	0.99
TAU (for soft update of target parameters)	1e-3
Actor LR (learning rate)	1e-4
Critic LR (learning rate)	1e-4
LEARN_FREQUENCY (how often to update the network)	3
Noise Epsilon Initial Value	0.9
Noise Epsilon Decrement Value	2e-6
N_LEARNING_EXP	4

The figure below shows the deep neural network architecture used for the actor and critic networks, where:
FC: Fully Connected Layer with its dimensions
ReLU: Rectified-Linear Unit
LReLU: Leaky Rectified-Linear Unit

A: Actor Deep Neural Network



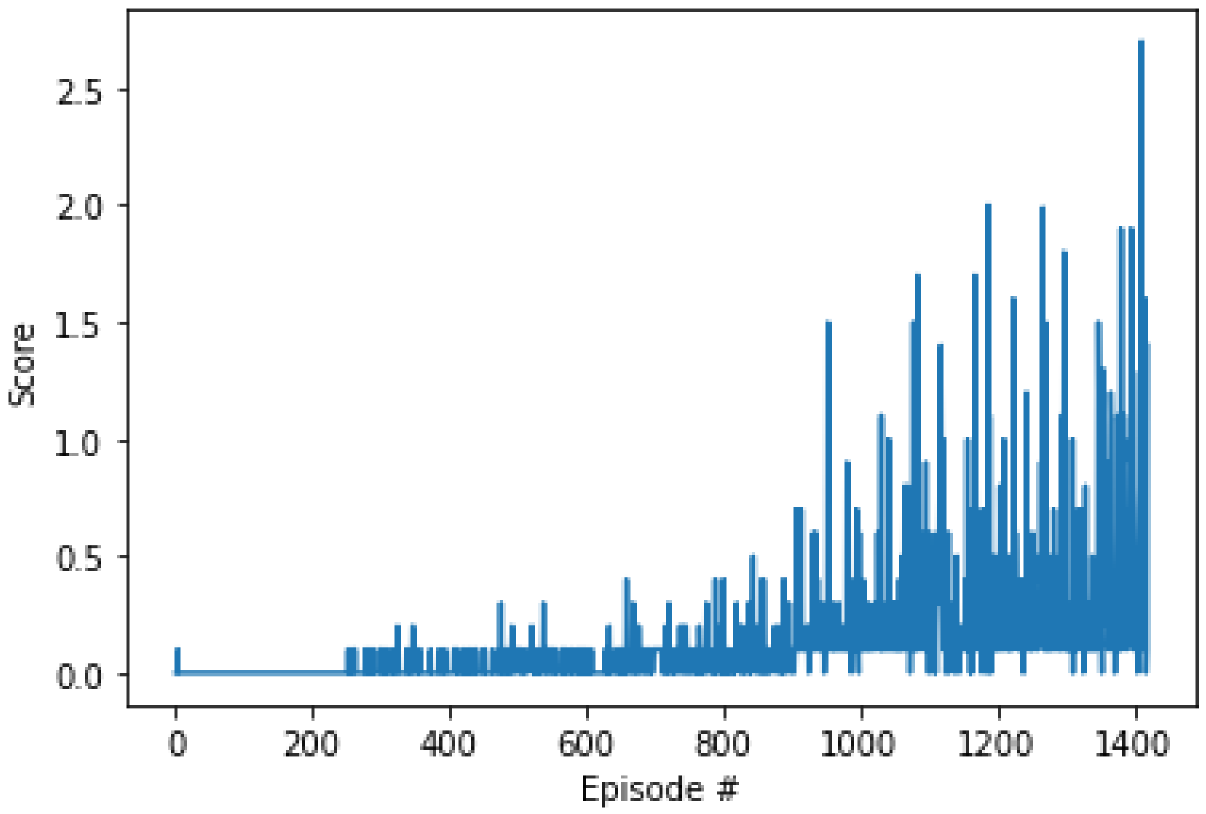
B: Critic Deep Neural Network



III: Rewards Plot

A: Training Plot

The plot below shows the rewards per episode that the agent received an average reward over training for 912 episodes to solve the environment.



IV: Future Work

Although the agents are able to keep the ball in play for a significant number of time steps, the agents fail to keep the ball when the ball is at certain locations, or when transitioning from one state to another.
It would also be interesting to try other actor-critic methods such as A2C or A3C to see how they affect the agents' performances.