Network Architecture Description:

The algorithm described in the attached code is an implementation of the Multi-Agent Proximal Policy Optimization algorithm. This network consists of an Actor (which generates the mean and sigma for the subsequent Normal action sampling distribution) and a Critic (which evaluates the Actor's actions based on estimated value and suggests improvements to the Actor).

The Actor network consists of 2 base Fully Connected layers and 2 final branches for mu and sigma. The first layer takes in input of state_size dimensions and its output consists of fc1_units dimensions. The second layer takes in fc1_units and outputs fc2_units. The output of this layer then gets input into a mu FC Layer and a sigma FC Layer which both take in fc2_units and output action_size dimension output.

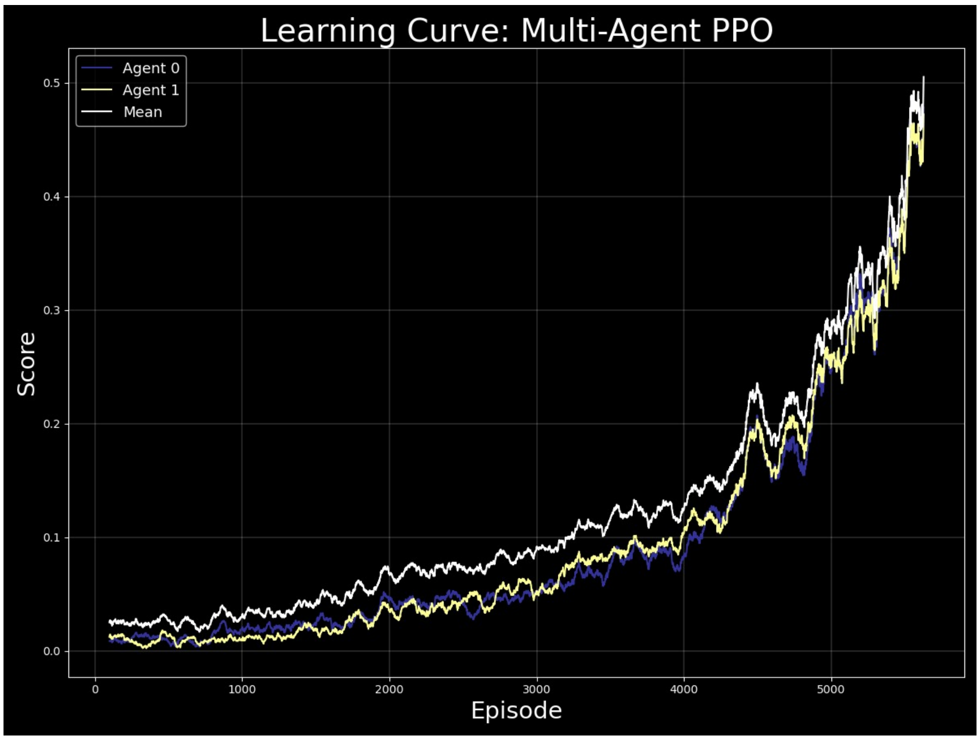
The Critic network consists of 3 Fully Connected Layers. The first takes in state_size dimension input and outputs fc1_units output. The next takes in fc1_units and outputs fc2_units. Finally, the last layer takes in fc2_units and outputs a single value, designating the goodness of the action taken by the Actor model.

Learning Algorithm Description:

In the environment, the state space consists of `8` variables that correspond to the position and velocity of the ball and the racket. The action space consists of `2` continuous actions of moving toward or away from the net and jumping. A reward of `+0.1` is gained each time an agent is able to hit the ball over the net. If the agent allows the ball to touch the ground or if the agent hits the ball out of bounds, it receives a reward of `-0.01`. These score allocations promote the overall objective of keeping the ball in play. The environment is considered solved once the mean maximum episodic reward reaches `+0.5` for the past 100 episodes.

After each time-step, the states, actions, log_probs, rewards, and dones are stored in experiences from which the Actor and Critic networks utilize a random batch to learn after a predefined number of time-steps. The Critic evaluates the next state to offer a critique of the action which the Actor selected. This input allows the Actor to augment its action sampling distribution.

As experiences are gained, and the Actor and Critic networks optimize themselves jointly, the action sampling distribution becomes better and better allowing for more optimized game play. Below you may see the learning curve graph for the mean maximum episodic reward over the prior 100 episodes. In the case below, the environment was solved in 5,627 episodes.



Hyperparameters:

The most influential hyperparameters seemed to be the eps_clip which designated the range for clipping the surrogate objective as well as the gamma which designates the discount factor and the entropy_bonus which designates the Actor’s tendency for exploration.

All of the chosen hyperparameters in the implementation were chosen in order to optimize the training process. Fc1 and fc2 units of 512 and 256 for both the Actor and Critic networks seemed to be ideal, as well as the learning rate of around 1e-4 for both. A large gamma and small entropy_bonus also seemed to optimize training. Seemingly because tennis may be too straight forward to allow for too much exploration of the agent!

An eps_clip of 0.3 seemed rather large, but resulted in an improved performance of both agents!

Future Ideas:

To further improve this project, I would like to evaluate the performance if a single Actor/Critic were used for both agents. Although, this would possibly make the learning process too easy as they have information for the entire environment.
I would also like to evaluate different neural network architectures as well as further fine-tune the hyper-parameters used in the algorithm.