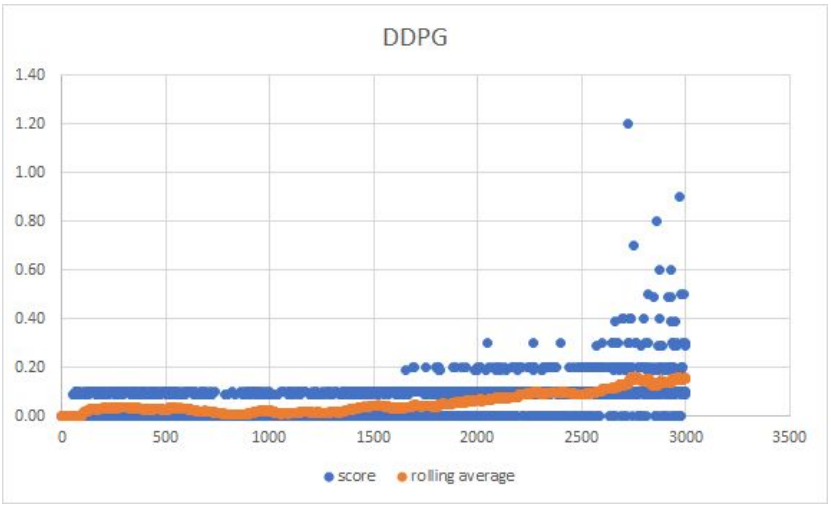
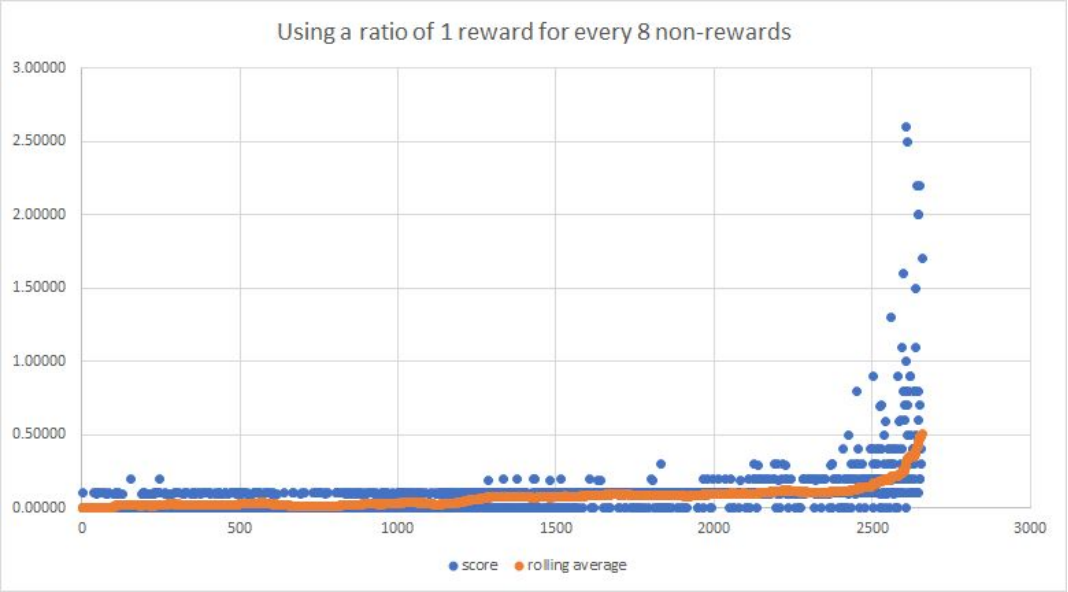
Learning Algorithm

A DDPG agent was trained to play tennis against itself in a unity environment. Even though there are 2 rackets that operate independent, a single critic and actor can be used since the states (ball velocity, locations, etc) are presumably provided with respect to the net.

It was observed that it takes 1600 episodes to start getting better than luck points. As pointed out in this paper, in supervised learning it is common to under or over sample. Early in training there was 3 positive rewards in 128 experiences. I was curious if overall training would be faster if I always took experiences with positive or negative rewards as well as 8x the number in zero reward experiences.



When that was implemented the environment was solved in 2657 episodes which is better than without. It learned slowly for the the first 2300 episodes but then learned fast. The main benefit was expected to be speed since an instance of learning might only have 27 experiences instead of 128. Below took 41 minutes to run on my computer.



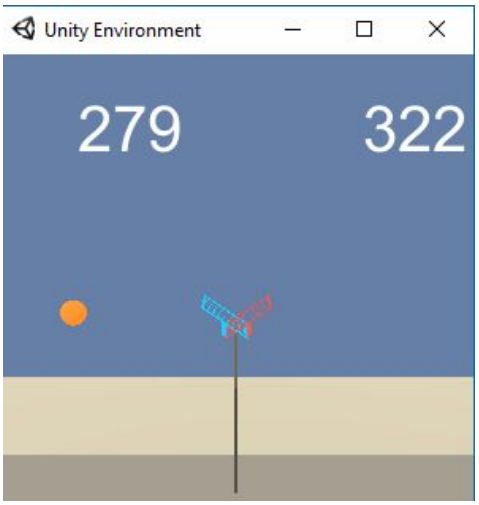
Unfortunately, I wasn't tracking the hyperparameters and re-discovering them is time consuming.

Hyperparameters used

Hyperparameter	Value
BUFFER_SIZE	100000
BATCH_SIZE	128
GAMMA	0.99
TAU	1.00E-03
LR_ACTOR	1.00E-04
LR_CRITIC	1.00E-04
WEIGHT_DECAY	0
AGENT_LEARN_COUNT	10
UPDATE_TARGET_EVERY	10
Actor hidden layer 1	256
Actor hidden layer 2	128

Critic hidden layer 1	200
Critic hidden layer 2	100

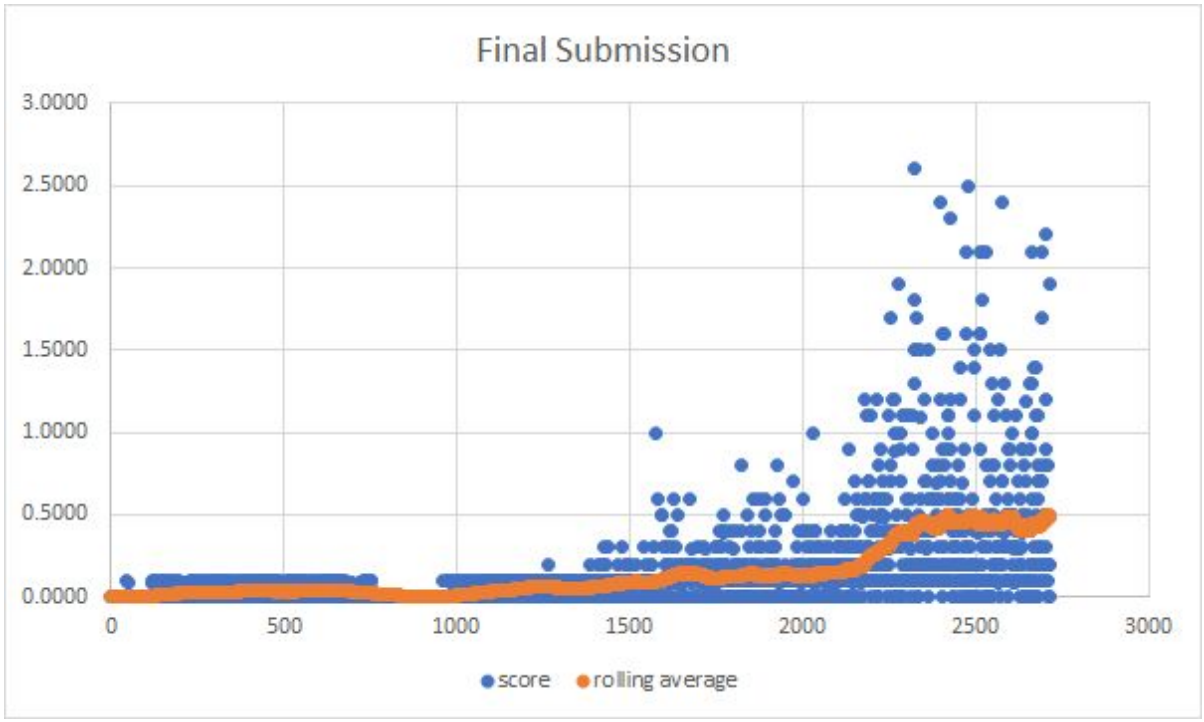
Items experimented with

- Considering how small the end batch size is after downsampling I expected to be able to increase batch size but that did not work. Possibly I'd have to increase the learning rate at the same time?
- Batch normalization was expected to allow for higher training rates and reduce the rolling average score declines but I got mixed results.
- It was observed that early in training agents will meet at the center of the net and do that many times (presumably since it happened to hit a ball one time while moving to the net). 

I thought that collecting data from from 300 episodes of random actions would provide a better base to learn from but that did not result in shorter train time.

Plot of Rewards

The environment was solved in 2714 episodes.



Ideas for Future Work

It'd be useful to understand why batch normalization is not working.
Using Tensorboard's visualization tools that show model states could help diagnose model problems.