Enviroment

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.

Learning Algorithm

I used Multi Agent [Deep Deterministic Policy Gradient (DDPG) (https://arxiv.org/abs/1509.02971)](https://arxiv.org/abs/1509.02971) algorithm. I used a wrapper class around ddpg agent class from my previous project. In my implementation agents have their own reply buffer. An they train separately.

DDPG is an actor critic method where contains 4 networks: local and target Actor and local and target critic.

- The Actor specifies the current policy by deterministically mapping states to a specific action (approximate maximizer).
- The critic in ddpg is use to approximate the mazimizer over the Q values of the next state. The critcis learns to eveluate the optimal action value function by using the actors best believed action.
- DDPG uses areply buffer.
- DDPG soft upates the target networks. Soft updates are used to update target networks of actor and critic. Soft updates are used to slowly blends local netwroks wights woth target netwrok weights.
- When we are training we are training the local netwroks therefore local netwroks are the most up-to-date network.
- We use target network for predication to stablize strain.
- I used gradient clipping to prevent exploding gradients when training the critic network.
- I also updated the network after 20 steps as suggested in the benchmark implementation

Actor Network Architecture

- Input : 24 (state size)
- Output : 2 (action size)

- ```
  Actor(
      (fc1): Linear(in_features=24, out_features=128, bias=True)
      (fc2): Linear(in_features=128, out_features=128, bias=True)
      (fc3): Linear(in_features=128, out_features=4, bias=True)
    )
  ```

Critic Network Architecture

- Input : 24 (state size)
- Output : 1

- ```
  Critic(
  (fcs1): Linear(in_features=24, out_features=128, bias=True)
  (fc2): Linear(in_features=132, out_features=128, bias=True)
  (fc3): Linear(in_features=128, out_features=1, bias=True)
  )
  ```

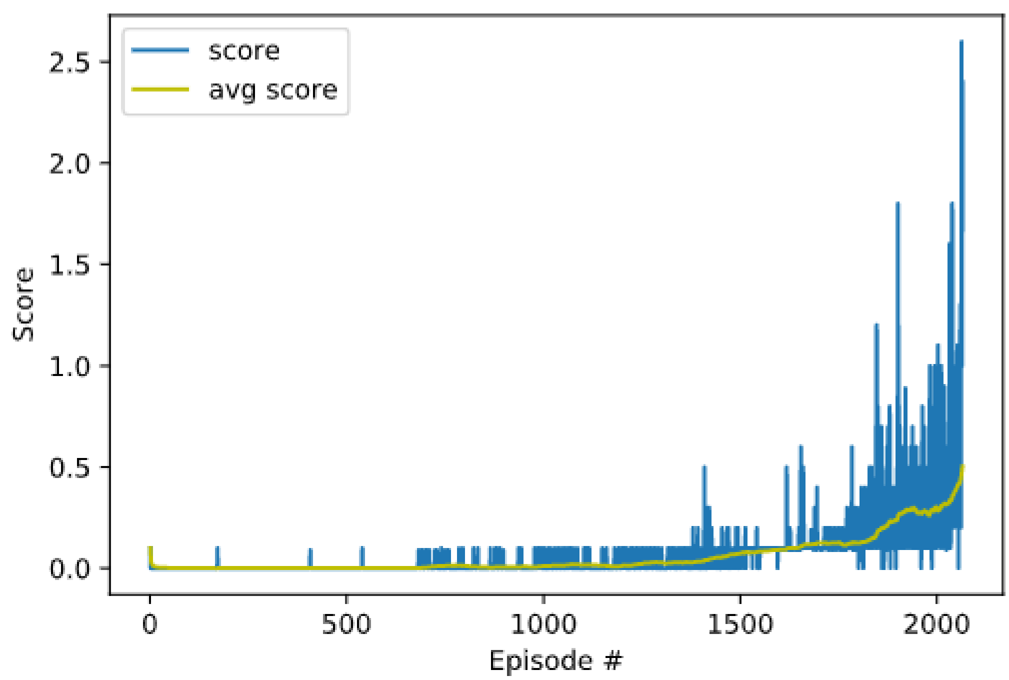
Hyperparameters

```
[NOISE]
addnoise = true
mu = 0.
theta = 0.15
sigma = 0.1
noise = 1.0

[DDPG]
gamma = 0.995
tau = 1e-3
seed = 25

[AGENT]
lr_actor = 2e-4
lr_critic = 2e-4
weight_decay = 0.0
train_every = 4
buffer_size = 1000000
batch_size = 128
learn_per_episode = 1
```

Training result:

enviroument is sloved in 217 Episodes with average Score: 30.03

```
Episode 100	Average Score: 0.0010
Episode 200	Average Score: 0.0010
Episode 300	Average Score: 0.0000
Episode 400	Average Score: 0.0000
Episode 500	Average Score: 0.0009
Episode 600	Average Score: 0.0010
Episode 700	Average Score: 0.0036
Episode 800	Average Score: 0.0129
Episode 900	Average Score: 0.0069
Episode 1000	Average Score: 0.0099
Episode 1100	Average Score: 0.0150
Episode 1200	Average Score: 0.0130
Episode 1300	Average Score: 0.0280
Episode 1400	Average Score: 0.0340
Episode 1500	Average Score: 0.0734
Episode 1600	Average Score: 0.0917
Episode 1700	Average Score: 0.1241
Episode 1800	Average Score: 0.1257
Episode 1900	Average Score: 0.2407
Episode 2000	Average Score: 0.2998
Episode 2065	Episode Score: 2.4000	Average Score: 0.5027
Environment solved in 2065 Episodes	Average Score: 0.5027
```



Test Results

```
Episode:	0	Score:	1.40
Episode:	1	Score:	1.75
Episode:	2	Score:	1.25
Episode:	3	Score:	1.05
Episode:	4	Score:	2.45
```

Future Ideas

- Work to improve performance of the mutli agent model.
- experiment with shared memory.
- expriment with environments with multiple agents and brains.