Report

Learning Algorithm

The agents are trained using DDPG framework with shared experience.

Model and Hyper Parameters

Both Actor and Critic networks have two hidden layers of 512 and 256 nodes. Inputs are normalized using BatchNorm1d. Each layer comes with RELU activation. The final output activation is tanh.

The actor has a learning rate of 0.0001 and the critic has a learning rate of 0.0003. The agent uses the Ornstein–Uhlenbeck process to explore. The Ornstein–Uhlenbeck process has theta of 0.15 and sigma of 0.2. The memory can hold up to 10^6 timesteps of the game. The batch size is set to 512. Policy is updates every 4 timesteps. Soft update uses a tau of 0.001.

Refer to the `tennis.ipynb` and `config.py` for hyper-parameters

Training Specifications

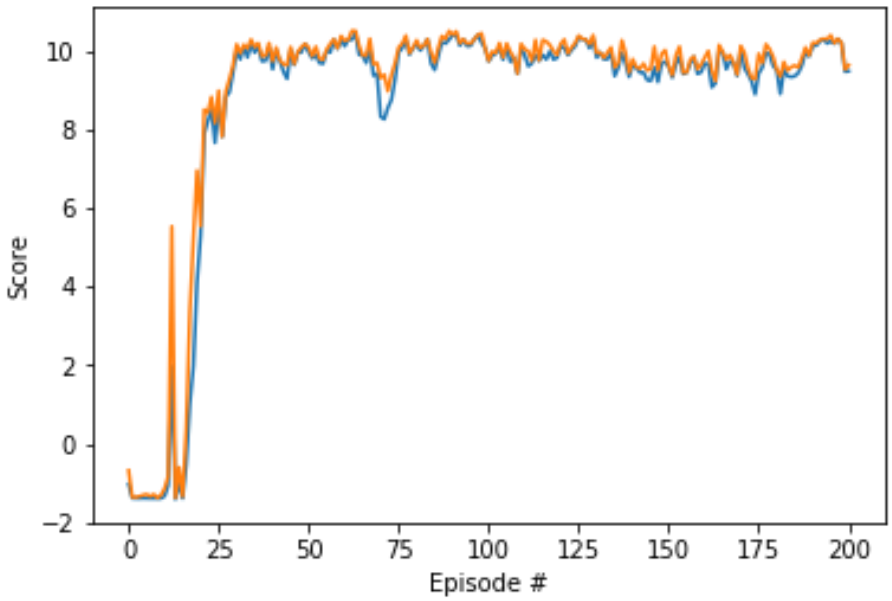
After spending weeks training unsuccessfully, I have found that ending each episode whenever the environment passes [done] would limit exploration. The score would be stuck around 0.2. I suspect this is because the game is inherently unstable. Disregarding the [dones] earlier on would the agent to explore and find ways to improve average performance over many timesteps.

The result was achieved through two steps:

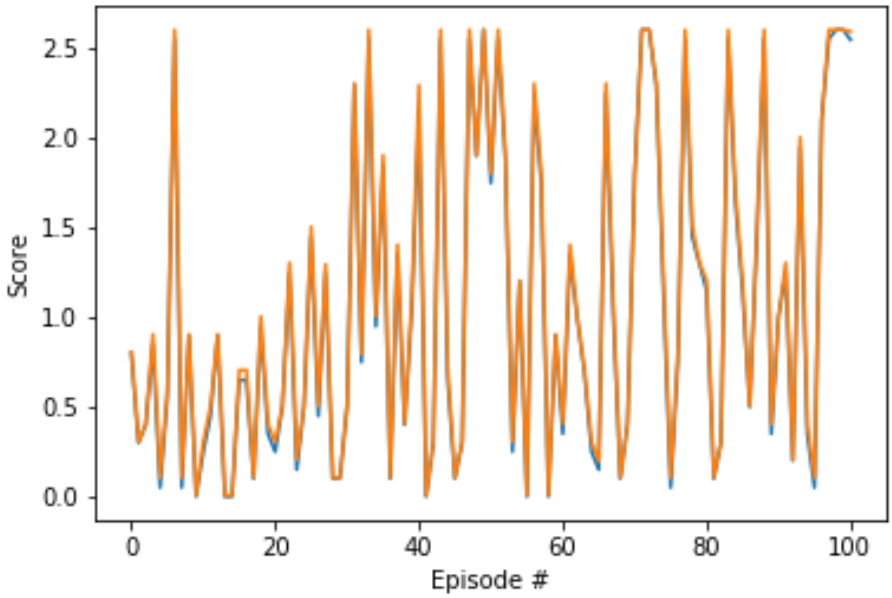
1. Training while disregarding environment [dones] for 200 episodes, 4000 timesteps each.
2. Training while taking into account [dones]

Result

As described in Training Specifications, the first step of the training has brought the agent's performance to roughly 10 per episode



The pre-trained agent then trained for another 100 episode, achieving a final average score of 1.09



Future Ideas

- More training at the first step would improve the agent's performance. The highest score I have gotten is around 2.
- I would also try training iteratively to see if it would improve performance. In other words, repeat step 1 and 2 a few times.
- Consider implementing PPO
- Consider using simple DDPG with the perspective of single player