Project 3: Collaboration and Competition

Description of the implementation

The solution to this challenge was inspired by the **Multi-Agent Deep Deterministic Policy Gradient (MADDPG)** algorithm, as presented in the paper Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments.

However, in this implementation:

a) Each has it's own critic and each critic does not get inputs from the states and actions from the other agent. Hence, the environment is still "non-stationary" from the perspective of any individual agent (in a way that is not explainable by changes in the agent's own policy).

b) To ensure each agent can learn from the experience of all participating agents, a shared replay buffer was also implemented.

c) Each agent's actor network will only receive inputs (observations) from its own agent.

As a start, the DPPG algorithm used to solve the Udacity Project 2 Continuous Control was used as a starting point. This version of DDPG of has 2 fully-connected (FC) layers [256, 128] for both actor and critic.

Starting network architecture:

- ✓ The Actor Network was built to receive input size of 24 and generate output of numbers representing the actions to be taken for that observed state. In this case, the Actor is used to approximate the optimal policy π deterministically.
- ✓ Although the Critic Network was also built to receive input size of 24, the second hidden layer of the Critic Network receives both the result of the Critic's first hidden layer and the 2 actions which came from the Actor Network. In this case, the Critic Network is used to approximate the target value based on the given state and the estimated best action , Q(s, a)

Key actions taken to tweak hyperparameters

i. Increased max_t from 1000 to 2000.

ii. Increase batchsize was kept relatively high at 1024 with the hope of learning from more experiences in 1 timestep

iii. Keeping critic learning rate (1e-3) slightly higher than actor's learning rate (1e-4) to help critic to converge faster

iv. Did batch normalization at 1st hidden layer for all 4 networks (both Actor and Critic) to help stabilize training

v. Implemented "Delayed Intensive Update" by updated network 10 times every 20 timesteps to speed up training and stabilize learning.

vi. Added noise decay of 0.999 to reduce unnecessary exploration after agent has gathered sizeable amount of experience

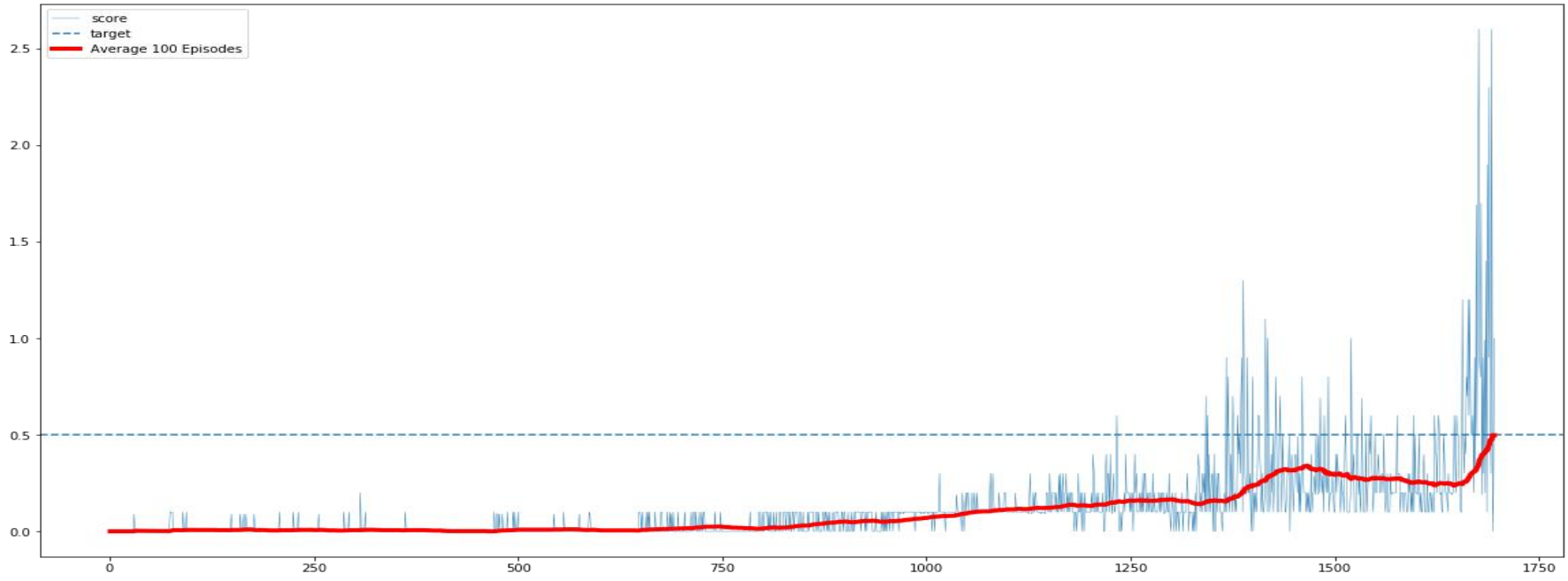
Final Hyperparameters used:

- ✓ BUFFER_SIZE = int(1e6) # replay buffer size
- ✓ BATCH_SIZE = 256 # minibatch size
- ✓ GAMMA = 0.99 # discount factor
- ✓ TAU = 1e-3 # for soft update of target parameters
- ✓ LR_ACTOR = 1e-4 # learning rate of the actor
- ✓ LR_CRITIC = 1e-3 # learning rate of the critic
- ✓ WEIGHT_DECAY = 0 # L2 weight decay

- ✓ NOISE_DECAY=0.9999
- ✓ UPDATE_EVERY=20
- ✓ FREQ_UPDATES=10

Final outcome

Environment was solved in 1596 episodes as we met the target threshold of 0.5 in 1696 episodes!



Ideas for Future Work

a. The most important next step is to configure each agent's critic network to receive input (both actions and states) from all participating agents to establish a stable environment.
b. There could be better hyperparameters out there to solve the environment faster. The only constraint was time.
c. While experiences were randomly sample from the replay buffer in this implementation, prioritized replay buffer might deliver a better result.