Continuous Control

Architecture:

A total of four neural network is used in project. Two networks, for each of Actor and Critic. Both models (Actor and Critic) have 24 nodes for input layer, two hidden layers and an output layer. For the first hidden layer, actor model has 250 nodes whereas critic model has 250 state nodes plus 2 action nodes. For the second hidden layer, both have 150 nodes. Finally, the output layer for actor model has 2 nodes ranging between -1 to 1 therefore hyperbolic tangent is used for the activation function whereas critic model has only 1 node without activation function.

self.actor_local = Actor(state_size, action_size, random_seed).to(device)

self.actor_target = Actor(state_size, action_size, random_seed).to(device)

self.critic_local = Critic(state_size, action_size , random_seed).to(device)

self.critic_target = Critic(state_size, action_size , random_seed).to(device)

Following techniques were used to enhance the learning capability/stability:

- **Experience Replay**: The class ReplayBuffer is responsible for adding and sampling of experience.

- **Fixed Q-Target**: The function approximators are defined in class Agent. They are initialized in constructor as, network and network_target. The network_target approximator is used for finding TD target and the network approximator is used for finding TD expected value. Before starting new time step, the network_target is updated in accordance with network approximator.

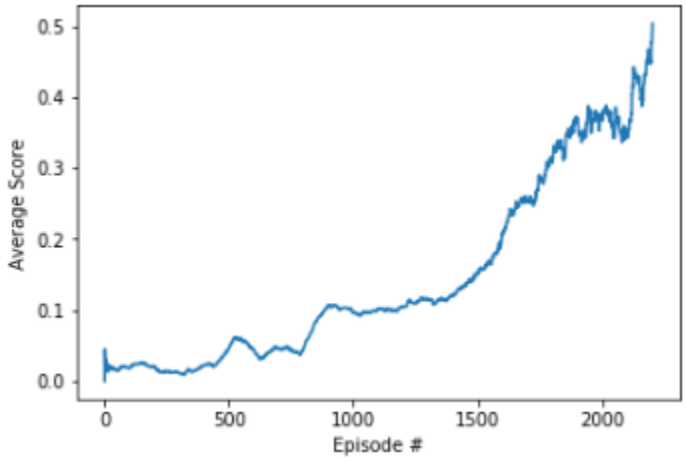
  ```
  self.actor_local = Actor(state_size, action_size, random_seed).to(device)
  self.actor_target = Actor(state_size, action_size, random_seed).to(device)
  ```

- **Actor-Critic Method**: We are using value-based techniques to further reduce the variance of policy-based methods. Basically actor-critic are a hybrid version of the policy- and value- based methods, in which the actor estimates the policy and the critic estimate the value function.

- **Deep Deterministic Policy Gradient (DDPG) algorithm**: DDPG, short for Deep Deterministic Policy Gradient, is a model-free off-policy actor-critic algorithm, combining DPG with DQN. DQN stabilizes the learning of the Q-function by using experience replay and a fixed target network. The DQN works in discrete space and DDPG extends it to continuous space with the actor-critic framework while learning deterministic policy.

- **Noise**: Instead of Normal distribution, Ornstein-Uhlenbeck noise process is used to generate the behavior policy for better exploration.

- **Plot of Rewards**



Hyperparameters:

Parameter	Value	Description
BUFFER_SIZE	int(1e5)	replay buffer size
BATCH_SIZE	2.50E+02	minibatch size
GAMMA	9.90E-01	discount factor
TAU	1.00E-03	for soft update of target parameters
LR_ACTOR	1.00E-04	learning rate
LR_CRITIC	1.00E-03	learning rate
n_episodes	2.00E+03	number of episodes
EXPLORE_TIMESTEPS	5.00E+03	timesteps until the agents just explore

Future ideas for improving the agent's performance:

- **Prioritized Experience Replay** could be implemented to further enhance the performance of the agent. Its suggested to use '**Sum Tree**' data structure as it helps in a faster implementation of Prioritized Experience Replay.
- Tweak hyperparameter to achieve goal within less episode.