Multi Agent Tennis Report

Learning Algorithm

The learning algorithm used is the **Deep Deterministic Policy Gradient (DDPG).**

The implementation builds upon the original DDPG algorithm and extends it to a multi-agent setting. It follows an actor-critic architecture suitable for continuous action environments, where each agent independently learns a policy while sharing a common critic network for value estimation.

Each agent maintains two actor networks: a local actor for selecting actions and a target actor for computing stable targets during learning. These networks take the agent's observed state as input and output a continuous action. All agents share a single critic network, which estimates Q-values based on the current agent's state and action. A corresponding target critic is used to compute stable target Q-values using the Bellman equation.

To encourage exploration, each agent's actions are perturbed using Ornstein-Uhlenbeck noise, which generates temporally correlated noise suitable for physical control tasks. A shared replay buffer is used to store experiences from all agents in the format (state, action, reward, next_state, done), enabling off-policy learning via experience replay.

During learning, a batch of experiences is sampled from the replay buffer. The critic network is trained to minimize the mean squared error (MSE) between predicted Q-values and target Q-values. The actor networks are updated using the deterministic policy gradient, based on the critic's evaluation of predicted actions. To ensure training stability, soft updates are performed on the target networks, gradually updating their weights toward the local networks.

This multi-agent DDPG setup allows agents to learn continuous control policies independently, while benefiting from a shared critic for more stable and coordinated learning.

Model Architecture: **Critic Network**

Critic Network is a fully-connected feedforward neural network with the following layers:

1. Layer 1: Linear(state_size, 256), followed by **Leaky ReLU**.
2. Layer 2: Linear(256+action_size, 256), followed by **Leaky ReLU**.
3. Layer 3: Linear(256, 128), followed by **Leaky ReLU**.
4. Layer 4: Linear(128, 1), no activation (outputs Q-value).

Model Architecture: **Actor Network**

Actor Network is a fully-connected feedforward neural network with the following layers:

1. Layer 1: Linear(state_size, 256), followed by ReLU.

2. Layer 2: Linear(256, action_size), followed by Tanh activation. Then the values are clipped so that the actions have a value in the range [-1, 1]

In the tennis multi-agent environment, the state size is of length 8 per agent (i.e. 16 in total) while the action size is 2 continuous actions for each agent (i.e. 4 in total).

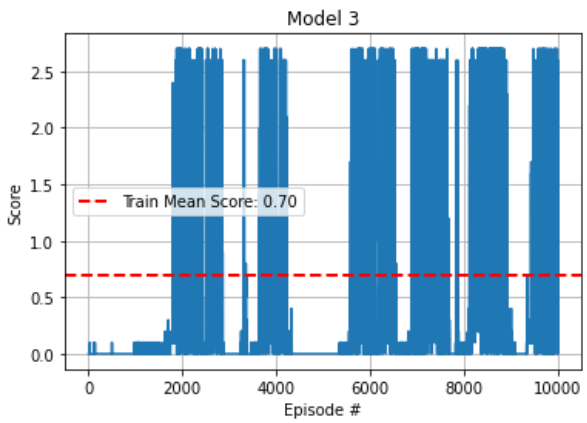
Chosen Hyperparameters

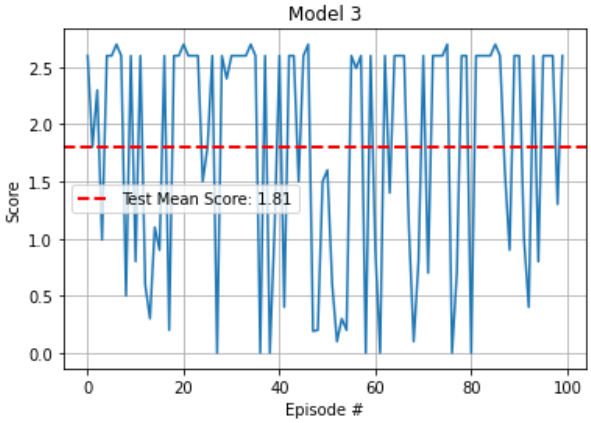
After training 3 models with different hyperparameters, the hyperparameters in the table below achieved the best results:

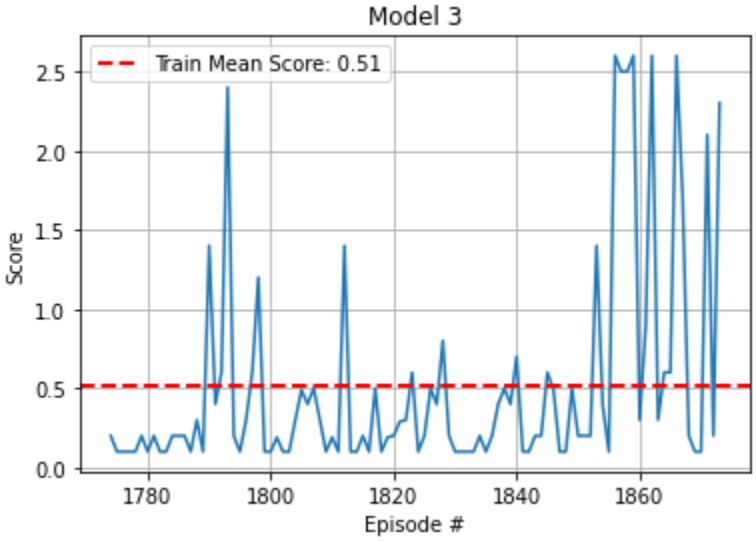
Hyperparameter	Value	Description
n_episodes	10000	Number of training episodes
max_t	1000	Max steps per episode
BUFFER_SIZE	1,000,000	Size of replay memory
BATCH_SIZE	256	Mini-batch size for training
GAMMA	0.99	Discount factor for future rewards
TAU	0.001	Soft update interpolation parameter
LR_ACTOR	0.0001	Learning rate for Adam optimizer for the Actor Network
LR_CRITIC	0.001	Learning rate for Adam optimizer for the Critic Network

Best Model Results

The following tables show the mean reward score in the best model for the training and the testing respectively. The environment was solved at episode 1774 as can be seen in the zoomed in view of the training episodes.







Ideas for Future Work

- Centralized Critic: Extend the current implementation to use a fully centralized critic that receives the observations and actions of all agents, not just one. This could improve coordination in cooperative environments. Note that the experience replay buffer is already centralized.
- Prioritized Experience Replay: Instead of uniform sampling, use prioritized sampling based on TD error to improve sample efficiency.
- Exploration Strategies & Hyperparameter Tuning: Explore other forms of noise (e.g., Gaussian, parameter noise), and experiment with different values for TAU, LR_ACTOR, and LR_CRITIC to further improve learning stability and convergence speed.