**Udacity Collaboration and Competition Tennis Project**

**1 Learning Algorithm**

**Model Architectures**

I used two different neural networks each for the Actor and critic but each one has two different network – the regular network and the target network. This is where we use soft updates to train the most up-to-date network – the regular network and use the target network – for predicting to stabilise the training in the regular network. The soft updates I used in the network was chosen to be Tau = 0.001.

**Actor Network**

For the Actor neural network, we used 2 hidden layers with 400 and 300 units respectively, with each hidden layer we used a rectified non-linearity but for the output layer we used tanh activation function to bound the actions – control pendulum problem. The following is three fully connected layers for the actor network:

* 1. 1st Fully connected layer - input: 24
  2. 2nd Fully connected layer - input: 400
  3. 3rd Fully connected layer - input: 300

**Critic Network**

The critic network was like the Actor network with 2 hidden layers, 400 and 300 units respectively but with the difference of the output layer returns a value without the constraint of the tanh function imposed on the Actor network. This mean all layers are activated with ReLU with the last layer not activated. The following is three fully connected layers for the critic network:

* 1. 1st Fully connected layer - input: 33
  2. 2nd Fully connected layer - input: 400
  3. 3rd  Fully connected layer - input: 300

**Hyperparameters**

All the hyperparameters are specified in the tennis\_agent module where I used many of the same hyperparameters in project two – Continuous control, but with added parameters to fit the current project:

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

EPSILON\_MIN = 0.1

EPSILON\_MAX = 1.0

EPSILON\_DECAY = 0

LEARN\_START = 0

UPDATE\_EVERY = 10

UPDATES\_PER\_STEP = 10

num\_workers = 2

episode\_count = 4000

buffer\_size = int(1e5) # replay buffer size

mini\_batch\_size = 1024

Chart, histogram

Description automatically generated**2 Plot of rewards**

***Figure 1: Plot of rewards from training***

**3 Future works**

Since we are operating in a non-stationary environment this proposes many challenges for our RL Agent. This brings in more complexity, which means experience replay suffers and we need new ways of alleviating our instability in learning [1]. One way to tackle this problem has been presented by the team in openAI in their paper Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, where they used a multi-agent deep deterministic policy gradient (MADDPG). algorithm, which outperformed many of the state of the art and traditional RL models.



***Figure 2: Plot of rewards from different algorithms***

One of the algorithms being compared is the algorithm I have used to solve the current environment and as can be seen above in figure 2 – (figure taken from the paper) MADDPG outperformed DDPG, REINFORCE and many other well-known algorithms by a large margin [1].

**MADDPG Algorithm**

My current methods use a decentralised method for updating RL agents where each agent learns from its own observation, state, and reward. However, the proposed approach will be using a centralised critic with deterministic policies where we append further information about the state and other policies of all agents into the critic [1].

**4 References**

**[1] R. Lowe McGill University OpenAI, Y. Wu UC Berkeley, A. Tamar UC Berkeley, J. Harb McGill University OpenAI, P. Abbeel UC Berkeley OpenAI, I. Mordatch OpenAI. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments.**