## Learning Algorithm:

Deep Deterministic Policy Gradient (DDPG), is a model free algorithm, they can deal with low and high dimensional continuous action spaces, this can be unstable for large challenge problems. The algorithm is an off – policy actor – critic using deep function approximators updates the policy as it explores the environment. It applies gradient descent to the deterministic policy with minibatch data sampled from a replay buffer, where the gradient is computed via:

$$\widehat{\nabla_\theta \eta(\mu_\theta)} = \sum_{i=1}^{B} \nabla_a Q_\phi(s_i, a)|_{a=\mu_\theta(s_i)} \nabla_\theta \mu_\theta(s_i)$$

Here B is the batch size, the critic net Q is trained via gradient descent with Bellman error:

$$L = \frac{1}{B}\sum_{i=1}^{B}(y_i - Q_\phi(s_i, a_i))^2 \quad \text{where} \quad y_i = r_i + \gamma Q'_{\phi'}(s'_i, \mu'_{\theta'}(s'_i))$$

To improve the performance of the algorithm DDPG uses target networks for actor and critic networks. DDPG uses two innovations from DQN (Deep Q Network) [1]:

1) Use a replay buffer to train the network of-policy to reduce sample correlations
2) "The network is trained with a target Q network to give consistent targets during temporal difference backups"

The pseudo code algorithm is shown in the next figure [1]:

**Algorithm 1** DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
**for** episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  **for** t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
    Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$
    Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  **end for**
**end for**

## Hyperparameters:

the learning rate is selected to be the same proportion than DDPG article [1], with a value of 1-4 and 1-3 for the actor and critic respectively, this allows reduce instability although increasing the number of episodes, weight decay is set in 1-6, for higher values the system did not seem to converge. The discount factor is close to 1 with value equal to 0.995. For actor and critic net initializations DDPG uses a [-3x10-3, 3x10-3] uniform distribution in final layers, the batch size is set in 256 and buffer with 1e6 size.

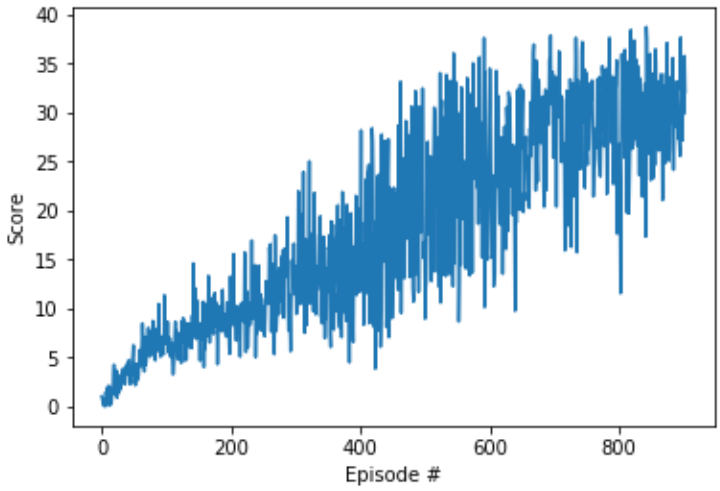
## DNN Model Architecture:

The deep model includes activation functions in the hidden layers of neural network (MLP multi-layer perceptron 400,300) used into actor-critic architecture, for agent the output activation function is a tanh activation. The train of net uses mini batch samples from large replay buffer, this is an off-policy method that would reduce the correlated batches.

## Rewards:

In the next two images I show the results for version 1 and version 2 of the project.

- Version 1: The agent receives an average reward (over 100 episodes) of at least +30



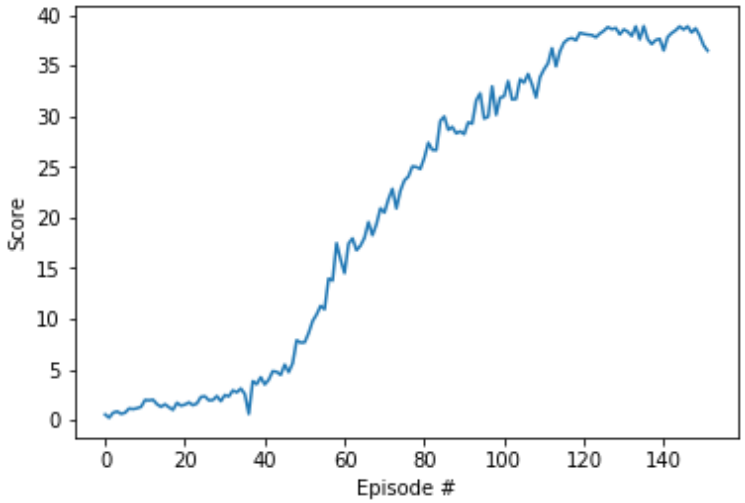

The last episodes from the scores report is shown below:

**Episode 899 Average Score: 29.73**

**Episode 900 Average Score: 29.82**
**Episode 901 Average Score: 29.85**
**Episode 902 Average Score: 29.98**
**Episode 903 Average Score: 30.18**

**Environment solved in 903 episodes! Average Score: 30.18**

- Version 2: The agent is able to receive an average reward (over 100 episodes, and over all 20 agents) of at least +30




The last episodes from the scores report is shown below:

**Episode 147 Average Score: 28.63**
**Episode 148 Average Score: 28.95**
**Episode 149 Average Score: 29.26**
**Episode 150 Average Score: 29.57**
**Episode 151 Average Score: 29.86**
**Episode 152 Average Score: 30.14**

**Environment solved in 152 episodes! Average Score: 30.14**

## Future Work:

For this project the DDPG algorithm can deal successfully with the environment for continuous control however, there is an instability in the learning process which varies widely for each training experiment. There are challenges to determining the parameters for a specific problem that is relative to the limitations for generalization common to current developments in I.A. However, R.L is more complex due to the sensitivity to the

adjustment of its hyperparameters and initialization parameters, which makes it difficult to reproduce a certain learning process. With the incorporation of Deep Learning architectures, the generalization capacities increase at the expense of the increase in the amount of data and consequently the number of episodes, but also the effort in the exploration is reduced. For the particular case of continuous states and actions, neural network approach functions have the problems of learning from a set of samples (from the replay buffer) whose labels are variable from the point of view of supervised learning, so the continuous exploration It is fundamental, hence, for example, the modification of learning rates is critical. The policy-based methods (on-policy) allow to reduce the variability of the data used during the training by restricting the samples to those consistent with the current policy; however, updating the policy leads to a rethinking of prior learning, in accordance with the above for the next assessments , I would explore algorithms like TRPO , PPO , ACKTR, 4DPG and as shown in this project could be tested multi-agent architectures.

## References:

[1] Continuous control with deep reinforcement learning, Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, Google Deepmind, 9 sept 2015.

[2] Deep Reinforcement Learning that Matters, Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, David Meger, The Thirty-Second AAAI Conference on Artificial Intelligence, Association for the Advancement of Artificial Intelligence (www.aaai.org), 2018.