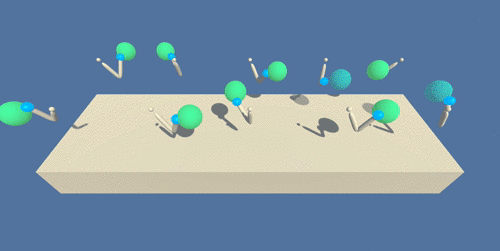
# Project 2 – Continuous Control

## Introduction:

The purpose of this project is to train a double-jointed robotic arm to move to a target location (Marked with a green ball). A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of the robotic arm (“The agent” onwards) is to maintain its position at the target location for as many time steps as possible.

For this project, we will work with the [Reacher](https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Examples.md#reacher) environment.

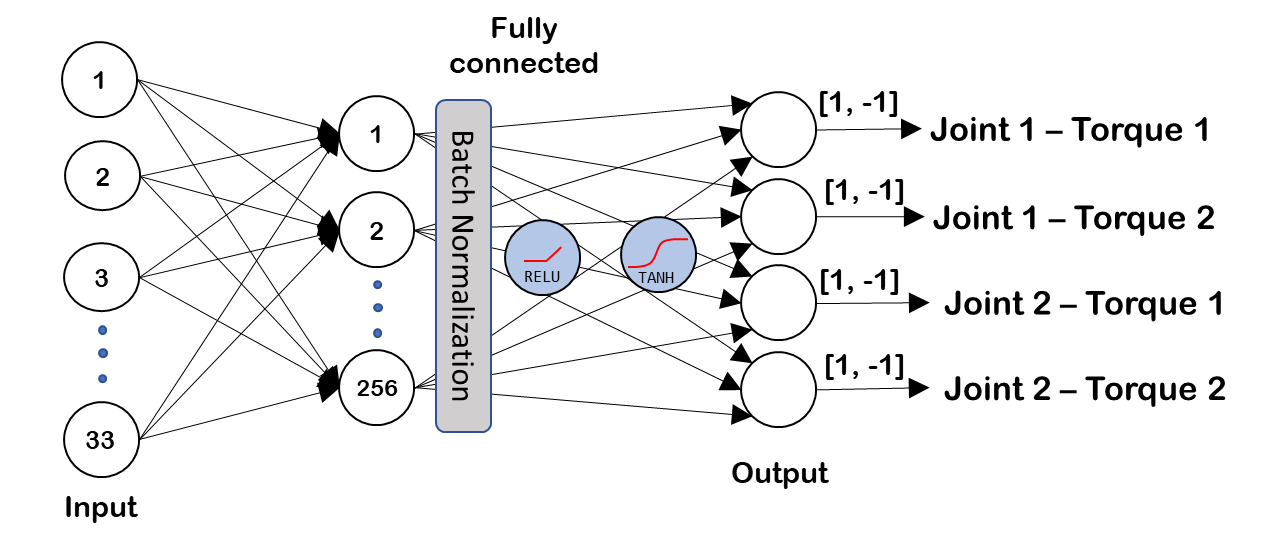
The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the robotic arm. Each action is a vector with four numbers, corresponding to torque applicable to two joints. Every entry in the action vector should be a number between -1 and 1.

Usually to calculate a robotic arm to move to a place there are a lot of calculations to be done to, translate the desired position from the end effector to the base of the arm (Forward kinematics) translate this position to each joint (Inverse kinematics) and calculate the necessary movement of each torque to get to the desired positions (Differential Kinematics) which takes a lot of calculations (Calculate translation matrixes, calculate matrix rotations and translations, find Jacobian of matrixes, calculate the determinant of these Jacobians, …), thanks to Reinforcement Learning we avoid all these calculation, which is implicitly learnt by the RL agent.

## Learning Algorithm:

The algorithm is an implementation of the DDPG (Deep deterministic policy gradient) algorithm with soft updates. There are two neural network for the DDPG implementation:

## Actor Neural Network:

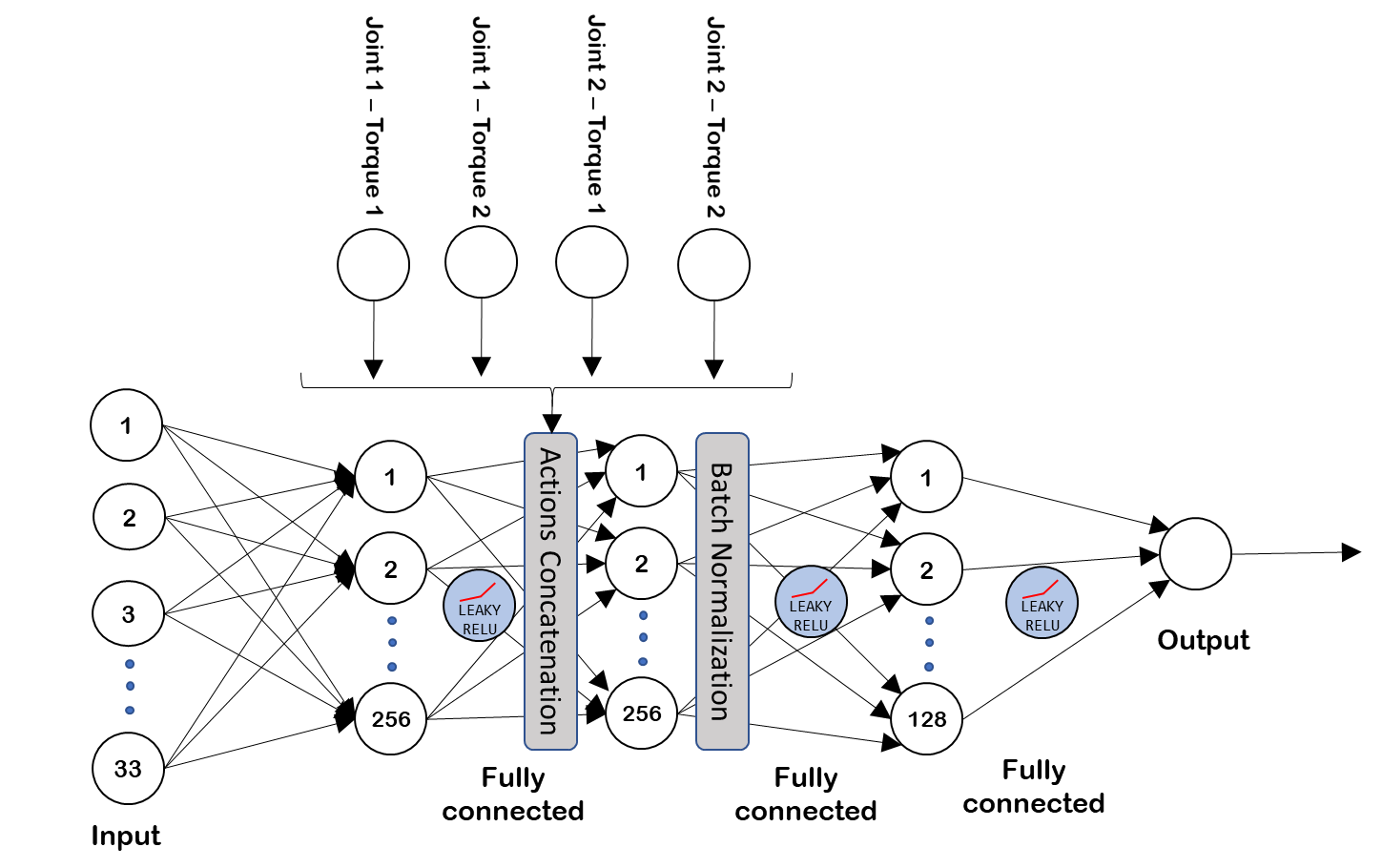


Explicar el Actor The Actor neural network hace esto, esto y esto

Explicar el continuous space.

## Critic Neural Network:

Explicar Critic.



Revisar Experience Replay.

Thanks to the implementation of **experience replay**, the algorithm stores states which are rare and actions

that are costly to be able to recall them, each experience (Which is formed by State, Action, Reward and Next

State) is stored in a buffer as the agent is interacting with the environment, then sample a small batch of these

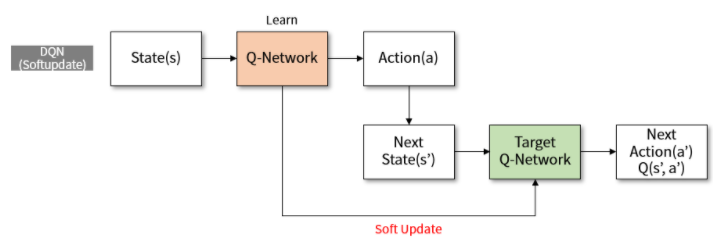
experiences in order to learn from them. Thanks to this, it learns from individual state-actions multiple times,

recall rare occurrences and make a better use of the experience obtained.

The buffer of experiences is sample randomly, to this helps breaking the correlation and prevent actions from

oscillating or diverging in wrong ways.

The algorithm includes a **soft update** to the target network. Explicar Regular->Target network. (Soft Updates)



The implementation of this soft update means that the algorithm has two neural networks. The called “Regular” neural network and the “Target” neural network.

Every **[cada step o tengo una variable?]** step the networks are blended with a percentage of merging considering the TAU variable, so considering the TAU variable is set to 0,05%, every step a 0,05% of the regular network will be merged with the target network [Especificar lo que hemos puesto y la evolución si hemos ido cambiando y ha arreglado cosas]

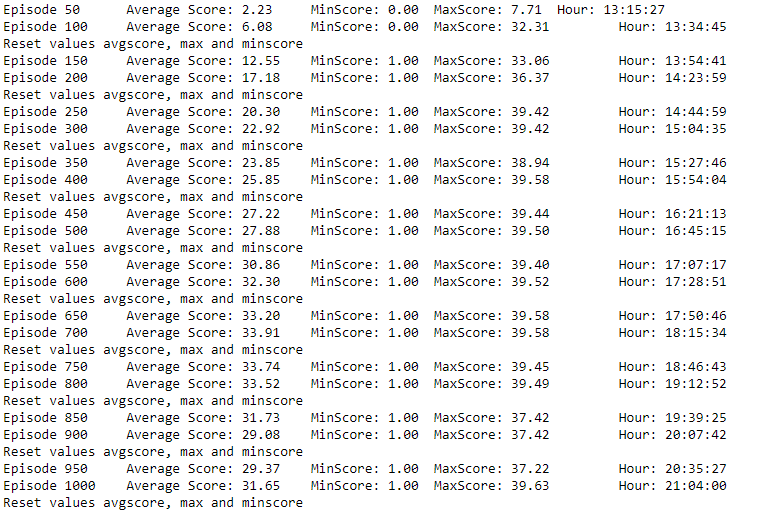
That means that this algorithm is an off-policy algorithm [estoy Seguro de esto?] asa

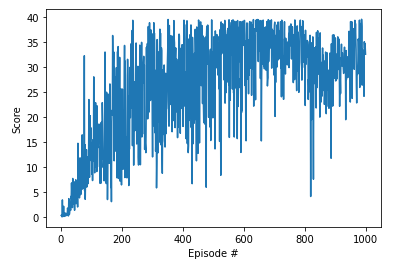
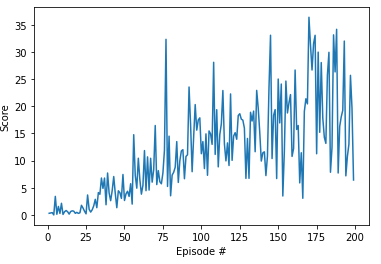
Revisar/explicar Actions concatenation?

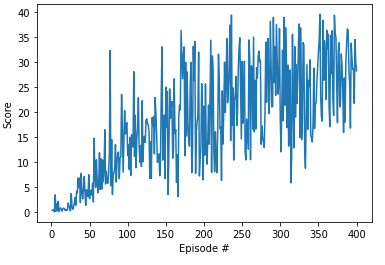
Revisar/explicar Batch normalization?

Explicar el uso de Leaky Relu.

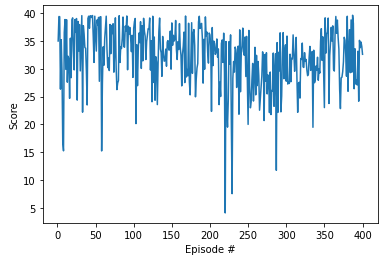
## Algorithm performance:



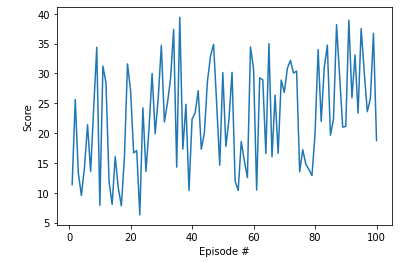




Del 600 al 1000:



Del 200 al 300:



## Future improvements:

¿?