# CONTINUOUS CONTROL

## INTRODUCTION:

The goal of the project is to make a double-jointed arm to move to a target location. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of your agent is to maintain its position at the target location for as many time steps as possible.

## ENVIRONMENT:

The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the 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.

The task is designed to be episodic and the task ends if a reward of 30 is achieved.

## ALGORITHM:

The Deep Deterministic policy gradient (DDPG) algorithm over Deep Q Network (DQN) is used for this project. This is because DDPG works on both continuous state and action space whereas DQN is meant to solve discrete action space problems as well.

The structure of the agent’s network is as follows.

Actor network:

Given the current state the actor is trained to output an approximation of the optimal policy action deterministically.

Critic network:

It computes the Q value for any given state or action by using actor’s best trusted action.

Four MLP layers are used for both actor and critic network.

Replay buffer:

Just like the DQN , this algorithm uses a reply buffer and sample from it randomly in order to break the correlation that exist because of the consecutive experiences .

Soft updates:

DDPG consists of 4 networks.

A regular (local) copy of actor network.

A target copy of the actor network.

A regular (local) copy of the critic network.

A target copy of the critic network.

The regular network is the updated network that is used for training as well as target which is used for prediction to stabilize the network.

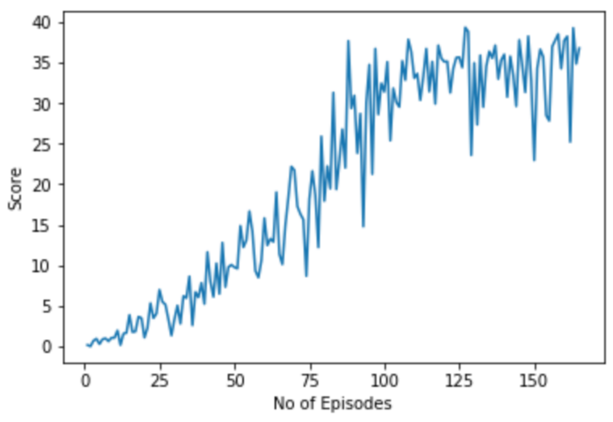
**HYPERPARAMETER:**

- GAMMA = 0.99 # discount factor
- TAU = 1e-3 # soft update of target network weights
- LR_ACTOR = 1e-4 # learning rate of the actor
- LR_CRITIC = 1e-4 # learning rate of the critic
- BUFFER_SIZE = int(1e5) # reply buffer
- BATCH_SIZE = 128 # minibatch size
- WEIGHT_DECAY = 0 # L2 weight

- n_episodes (int): maximum number of training episodes
- max_t (int): maximum number of timesteps per episode
- num_agents: number of agents in the environme

**RESULT:**

Environment was solved in 165 episodes with a average reward of 30.04.



**IMPROVEMENTS:**

The improvements that can be made,

1. Trying out algorithms like PPO, A3C, and D4PG that use multiple (non-interacting, parallel) copies of the same agent to distribute the task of gathering experience.
2. Trying out dropouts, various weight initialization and further hyper-parameters like cost functions that may yield better results.