

# Deep RL Arm Manipulation

Hytham Tag

**Abstract**—The goal of this project is train a 2 DOF manipulator using reinforcement learning and to optimize it's framework so that the arm can achieve 2 tasks the 1st is to make any part of the arm touch the can with 90% of accuracy at least & the 2nd it to make the gripper touch the can by at least 80%

**Index Terms**—Robot, IEEETran, Udacity, L<sup>A</sup>T<sub>E</sub>X, deep learning.

## 1 INTRODUCTION

THE goal of this project is to create DQN agent and define reward system to teach a robotic arm to carry out two primary tasks:

- Have any part of the robot arm to touch the can , with at least 90%
- Have the gripper of the robot arm to touch to touch the can, with at least 80%

The idea here is to make the robot arm try over and over like a child trying to walk, in the beginning the child try to stand up to get a candy but he fails many times until he get close to the best way to stand up right and get the reward (candy), this is the same as the robot arm as the robot arm tries to get close to the target until it touches it and get a reward for touching it whether for task 1 or 2.

### 1.1 Reinforcement Learning

Reinforcement is a class of machine learning where an agent learns how to behave in the environment by performing actions and thereby drawing intuitions and seeing the results where the robot arm is trying to take the best action to get more postive reward

### 1.2 Q-learning

Q-learning is a model-free reinforcement learning algorithm. The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances. It does not require a model (model-free) of the environment, and it can handle problems with stochastic transitions and rewards, without requiring adaptations.

### 1.3 DQN

Deep reinforcement learning is the combination of reinforcement learning (RL) and deep learning . it uses neural network to train the agent.It uses deep learning and reinforcement learning principles in order to create efficient algorithms that can be applied on areas like robotics Deep Learning + Reinforcement Learning = Deep-Q-Networks Instead of memorizing a different Q-value for each combination of pixels on the screen (theres billions!), we use ConvNets to generalize Q-values across similar states.

## 2 REWARD FUNCTIONS

A system of Rewards and penalties is used to train the Robot ARM DQN Agent. It can be found in Armpulgin.cpp. There Are 3 types of rewards as following:

- Postive Reward for successfully ending the episode
- Negative Reward for unsuccessfully ending the episode
  - reaching more than 100 frames
  - hitting the ground or colliding by it self
  - hitting the can by the arm not gripper (task 2)
- Reward based on distance to objectNegative Reward on time to make the arm reach the object faster
- Negative Reward on small steps ( $\leq 0.002$ ) to prevent the arm from freezing

an interim reward function was designed. Its purpose is to guide the robot arm towards the target. It measure is the distance  $dt$  between gripper and target object at  $t-1$  and  $t$ , the difference  $deta = d(t-1) - d(t)$  is calculated. In order to obtain a smooth reward function, and then the moving average is calculated as following

- $(negativavgGoalDelta = (avgGoalDelta * \alpha) + (distDelta * (1.0f - \alpha)))$

If the moving average is less than 0.002 it get negative reward this prevent the robot arm from freezing

- if ( $avgGoalDelta \leq 0.002$ )
- $rewardHistory = REWARD\_LOSS;$

also there is time penalty the faster the robot reaches the object the less the loss

- $rewardHistory = REWARD\_MUL * distGoal - time\_penalty;$

### 2.1 Hyperparameters

#### 2.2 Task1

- A reward(WIN \*10) is giving If any part of the arm touch the object.
- A reward(WIN) is giving if the arm come close to target
- A penalty(LOSS) is giving if any part of the robot touch

TABLE 1  
Hyperparameters

Parameter	Task 1	Task 2
INPUT_WIDTH	64	64
INPUT_HEIGHT	64	64
OPTIMIZER	Adam	Adam
LEARNING_RATE	0.1	0.1
REPLAY_MEMORY	10000	20000
BATCH_SIZE	256	512
USE_LSTM	true	true
LSTM_SIZE	256	256
Alpha	0.9	0.4

- the ground and the episode end.
- A penalty(LOST \* distance to goal) is provided if the arm robot go away from the target
- Any collision ends the episode.

### 2.3 Task2

- A reward(WIN \*5) is giving If any part of the arm touch
- the object.
- A penalty(LOSS \*5) is giving if any part of the robot touch the object.
- A reward(WIN) is giving if the arm come close to target
- A penalty(LOSS) is giving if any part of the robot touch
- the ground and the episode end.
- A penalty(LOST \* distance to goal) is provided if the arm robot go away from the target
- A penalty loss if the absolute average is less than 0.02
- A time penalty the longer the robot arm takes
- Any collision ends the episode.

## 3 RESULTS

I was able to achieve more than 90% in both tasks and the largest positive effect on the agents performance was observed when lowering the value of the moving average ALPHA parameter from 0.9 to 0.4. also a video showing what i have done <https://www.youtube.com/watch?v=aA83qXZpdi4>

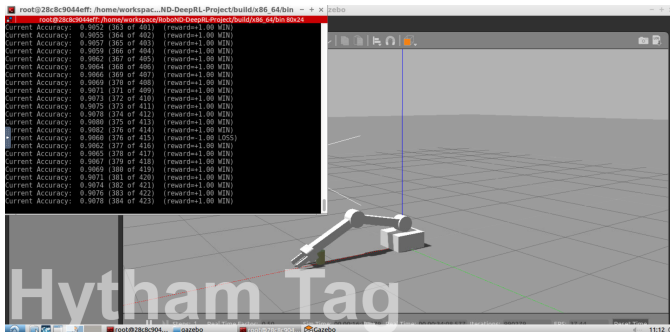


Fig. 1. Task 1.

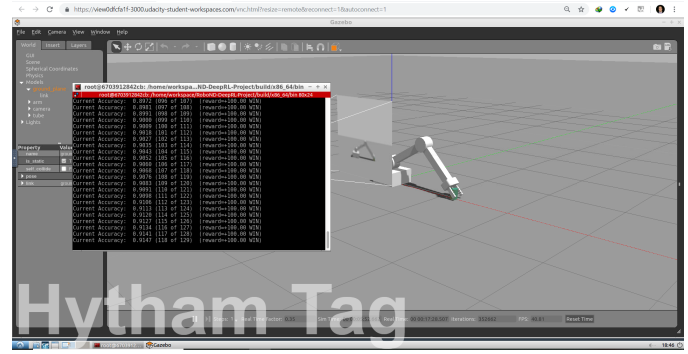


Fig. 2. Task 2.

## 4 FUTURE WORK

I was able to reach very satisfying result but i think we can use trajectory planning as a way to be like a reference and as the robot arm move on it it get higher reward this is better than the distance between the arm and the object also i want to try it on a real robot arm which i will do ti very soon.