Robotic Arm Manipulation Using Deep Q-Learning

Vijayasri Iyer

Abstract—This project report is a brief overview of a serial manipulator which manipulates itself to touch, grasp and pick up objects in a simulated ROS (Robot Operating System) environment. The arm uses Deep Reinforcement Learning; particularly a technique called Deep Q-learning (DQN) to achieve this feat. The objectives of this project are :

- Have any part of the robot arm touch the object of interest, with at least 90 percent accuracy.
- Have only the gripper base of the robot arm touch the object, with at least 80 percent accuracy

Index Terms—DQN, IEEEtran, Udacity RoboND, Serial Manipulator

1 Introduction

The past few decades have seen robots being used as assistive labour to do heavy industrial assembly line jobs which are dangerous and repititive. In these cases, robots were mostly programmed by hand with a detailed description of the task. But now since robots are moving into industries such as healthcare and autonomous transport where it isn't possible to hardcode all of the features required for the task. This led to the use of machine learning. Machine learning is a set of algorithms that allows the computer to learn the features involved in a task. A particular paradigm of machine learning known as reinforcement learning where an agent will learn from the environment by interacting with it and receiving rewards for performing actions. Reinforcement algorithms that incorporate deep learning, have shown significant promise in this avenue. One such algorithm called Deep Q-learning was introduced in 2015 by Google's DeepMind which beat the world's best players at complex games like Atari and Go. In this paper, the DQN algorithm is used to teach a robotic arm to manipulate its surrounding objects using just 2D images of the arm and the object. Performing this task, will help fuel further development of robots to perform pick and place tasks using DeepRL. An illustration of the DQN algorithm is shown below in fig 1.

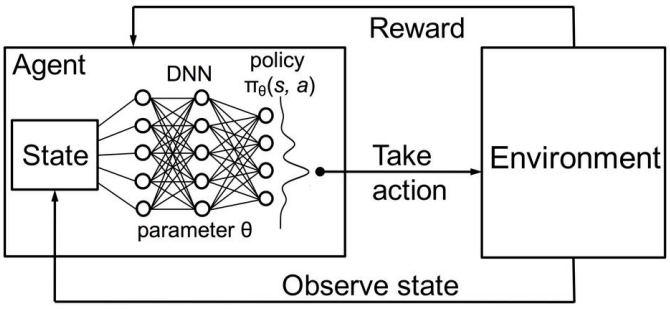

Fig. 1. Deep Q-Learning

1.1 Deep Q-Networks

The DQN algorithm is a spin-off of the original Q-Learning algorithm which uses deep learning instead of the Q-table (lookup table). Q-Learning uses a table to keep track of all discrete Q-values calculated for each state-action pair in the scenario. The algorithm is iteratively run over multiple episodes until the "best-action" for each state converges to a consistent choice. The equation for calculating Q-values is shown below in fig 2.

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

Fig. 2. Deep Q-Learning

DNNs (Deep Neural Networks) have been known to have excellent capability for extracting feature representations through backpropagation and the use of multiple layers for representation learning. After a sufficient amount of training iterations, DNN knows which information like color or shape is important to do the task. The DNN performs the task of mapping the 2D images to appropriate actions that the agent should take in order to reach the goal state. This is especially useful in an continuous state space, since the DNN can output Q-values for every possible action in one training pass. The DNN acts as a function approximator in the Q-Learning process. Deep Q-Learning is suited for environments which are partially observable, stochastic environments.

The Deep Q Network proposed in the paper, uses a CNN-LSTM network for representation learning. This model is a combination of an RNN (Recurrent Neural Network) with a CNN instead of a regular Multilayer Perceptron. The advantage of using this network is that the CNN is ideal for learning representations of images and an RNN (LSTM- Long Short Term Memory network) is ideal for learning time sequences. So this makes the network ideal for processing and training on video data, which is an ordered sequence of images.

2 Reward Policy

This arm was controlled using the joint position control method. As per this method, the joint position is increased based on a function of the variable action. The arm can perform 2 actions for each dof, 6 actions in total. So, the variable joint is assigned as a function of ref array containing the position of each joint and the direction variable multiplied by the actionJointDelta variable. The reward policy of the arm is as follows :
Giving positive reward if the arm touches the object and higher reward if the gripper touches the object.The arm condition is verified by checking if the contact sensor touches the COLLISION_ELEMENT, whereas the gripper condition is checked with the COLLISION_POINT variable.
Giving reward based on distance to the object. For this the smoothed average of the delta of the distance towards the goal was calculated. Here, alpha is set to 0.09.
Giving negative reward if the arm touches the ground. Checking contact sensor distance with the COLLISION_FILTER variable.
Reward at the end of an episode.
Negative reward if an episode exceeds too many frames.

3 Hyperparameters

The dataset consists of RGB (3 input channels) images of dimesions 128x128 which the LSTM takes as input. This data was collected using a camera sensor placed in front the arm in the gazebo world. The agent uses an LSTM of size 128 with a replay memory of 10000. The optimizer used is RMSprop with a learning rate of 0.1 and a batch size of 128.

4 Results

Case 1 : Any part of the arm touches the object.

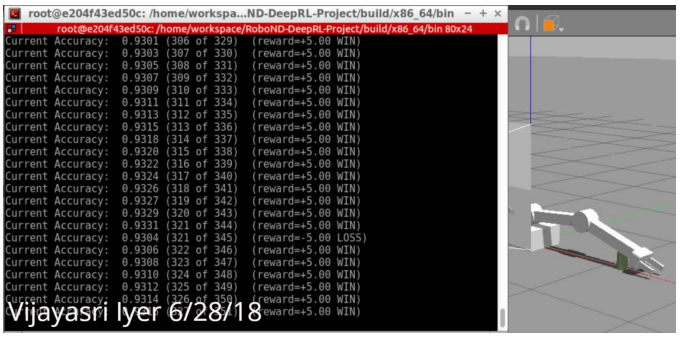

Fig. 3. Deep Q-Learning

In this case, the model took a little time to converge and improve it's accuracy. However, it did manage to achieve over 94 percent accuracy in this task. This was done in under 350 episodes of training.

Case 2 : Only gripper touches the object.

In the second case, the model converged quite fast, taking only a total of 75 episodes to achieve an accuracy of 95 percent in the task. The rewards provided were mainly of two types +5.00 for winning and -5.00 as negative or loss reward.

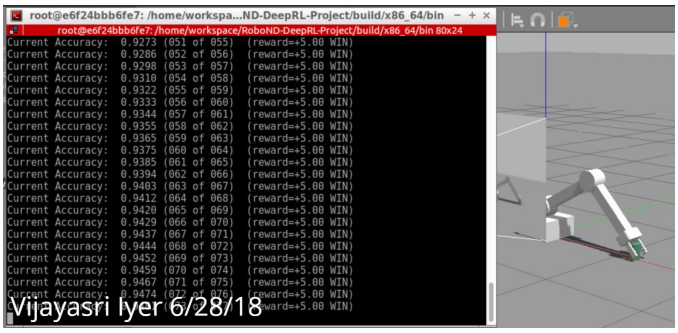

Fig. 4. Deep Q-Learning

5 Discussion

The arm is able to touch the object attempt to grasp it by navigating to the correct grasp position. But, this requires some imporvement if it needs to achieve the task of grasping the object and picking it up. This can be due to a very basic reward function and also due to the fact that DQN is not the state of the art algorithm used for such tasks. In such cases, using better training techniques such as experience replay and better algorithms such as Actor-critic or Policy-gradients may help in improving this project.

6 Conclusion / Future work

As mentioned above, this project needs to be extended more since it can only touch an object using the gripper at this point. Training it for more number of episodes, along with some changes to the source script reward functions and parameters is one option along with using better algorithms. This can become a commercially viable product if the arm is able to pick and place objects successfully in a given environment and it can also transfer whatever it has learnt to another task, or the same task in another environment.