Deep RL Arm Manipulation

Hsin-Wen Chang

Abstract—The goal of this project is to create a DQN agent and define reward functions to teach a robotic arm to perform two primary objectives. one is aim to have any part of the robot arm touch the object of interest with at least a 90 percent accuracy. Second is aim for only the gripper base of the robot arm touch the object, with at least a 80 percent accuracy.

Index Terms—Deep Reinforcement Learning Arm Manipulation, DQN, Robotics Software Engineer Nanodegree Program, Robot, IEEEtran, Udacity, LaTeX.

1 Introduction

DEEp Q-Network(DQN) introduced to training a Q value function represented by multi-layer perceptron. The following task will be implemented:
- Subscribe to camera and collision topics.
- Create the DQN Agent.
- Velocity or position based control of arm joints.
- Reward for robot gripper hitting the ground.
- Interim reward based on the distance to the object.
- Reward based on collision between the arm and the object.
- Tuning the hyperparameters.
- Reward based on collision between the arms gripper base and the object.

2 Background

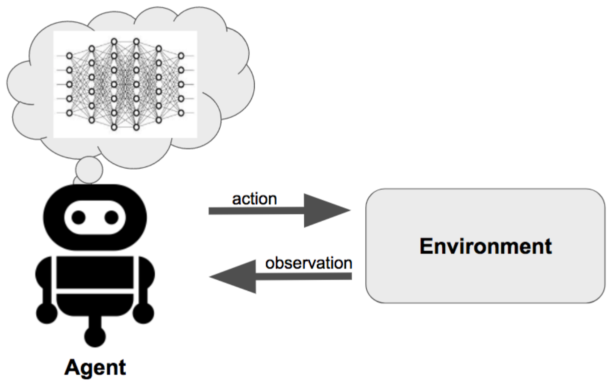


Fig. 1. From RL to Deep RL.

The difference between RL and Deep RL is the use of a deep neural network. Think of the collection of value-action pairs that define what actions an agent should take in any situation as a function of the observations that the agent receives from its environment. Neural network can be used to approximate this function because through its large quantity of parameters that can be learned through trial and error.

2.1 Setup and building from Source (Nvidia Jetson TX2)

```
$ sudo apt-get install cmake
$ conda install -c pytorch pytorch
$ sudo apt-get install libignition-math2-dev
$ git clone
    http://github.com/udacity/RoboND-DeepRL-Project
$ cd RoboND-DeepRL-Project
$ git submodule update --init
$ mkdir build
$ cd build
$ cmake ../
$ make
```

- Subscribe to camera topics:

```
//Subscribe to camera topic
cameraSub =
    cameraNode->Subscribe("/gazebo/arm_world
/camera/link/camera/image",
    &ArmPlugin::onCameraMsg, this);
```

- Subscribe to collision topics:

```
// Subscribe to prop collision topic
collisionSub =
    collisionNode->Subscribe("/gazebo
/arm_world/tube/tube_link/my_contact",
    &ArmPlugin::onCollisionMsg, this);
```

- Create the DQN Agent:

```
// Create DQN Agent
agent = dqnAgent::Create(INPUT_WIDTH,
    INPUT_HEIGHT, INPUT_CHANNELS,
    NUM_ACTIONS, OPTIMIZER,
    LEARNING_RATE, REPLAY_MEMORY,
    BATCH_SIZE, GAMMA, EPS_START,
    EPS_END, EPS_DECAY,USE_LSTM,
    LSTM_SIZE, ALLOW_RANDOM, DEBUG_DQN);
```

- Velocity or position based control of arm joints:
The DQN output is mapped to a particular action such as the control of each joint for the robotic arm. In ArmPlugin::updateAgent(), there are two existing approaches to control the joint movements.

```
const int jointIndex = action / 2;
const int actionDirection = 1 - 2 *
    (action % 2);
//Increase or decrease the joint
    velocity based on whether the action
    is even or odd. Set joint velocity
    based on whether action is even or
    odd.
const float velocity = vel[jointIndex] +
    actionVelDelta * actionDirection;
// Increase or decrease the joint
    position based on whether the action
    is even or odd Set joint position
    based on whether action is even or
    odd.
float joint = ref[jointIndex] +
    actionJointDelta * actionDirection;
```

- Reward for robot gripper hitting the ground:

```
// Get the bounding box for the gripper
const math::Box& gripBBox =
    gripper->GetBoundingBox();
const float groundContact = 0.05f;
// Set appropriate Reward for robot
    hitting the ground.
const bool checkGroundContact = (
    gripBBox.min.z <= groundContact ||
    gripBBox.max.z <= groundContact );
```

2.2 Reward function design

```
const float distDelta = lastGoalDistance -
    distGoal;
// Compute the smoothed moving average of the
    delta of the distance to the goal
avgGoalDelta = (avgGoalDelta * ALPHA) +
    (distDelta * (1.0f - ALPHA));
```

- REWARD LOSS: When robotic arm can't completed it's a round task, this penalty will be issued.
- REWARD WIN: When robotic arm completed its round task, this reward will be issued.
- REWARD MUL: Base on the distance between the arm and the object use this distance to calculate an appropriate reward.

2.3 DQN agent's hyperparameters

- Initial Parameters:

```
#define INPUT_WIDTH 512
#define INPUT_HEIGHT 512
#define OPTIMIZER "None"
#define LEARNING_RATE 0.0f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 8
#define USE_LSTM false
#define LSTM_SIZE 32
```

- Parameters Tuned:
  Input image size is quite large and influence memory usage. Reduce image size will improve performance. Optimizer were changed to RMSprop which is commonly used and produce good result.Small number learning rate will slow learning speed but minimize error. Smaller batch size will reduce more computing power.

```
#define INPUT_WIDTH 64
#define INPUT_HEIGHT 64
#define OPTIMIZER "RMSprop"
#define LEARNING_RATE 0.9f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 128
#define USE_LSTM true
#define LSTM_SIZE 256
```

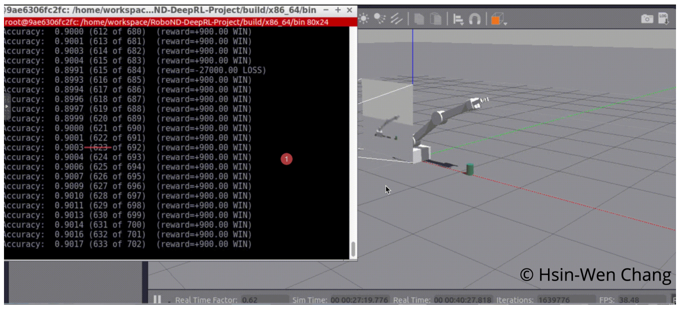
Hyperparameters	Values
INPUT WIDTH	64
INPUT HEIGHT	64
INPUT CHANNELS	3
NUM ACTIONS	DOF*2
OPTIMIZER	RMSprop
LEARNING RATE	0.9f
REPLAY MEMORY	10000
BATCH SIZE	128
GAMMA	0.9f
EPS START	0.9f
EPS END	0.0f
EPS DECAY	250
USE LSTM	true
LSTM SIZE	256
ALLOW RANDOM	true
DEBUG DQN	false

TABLE 1
Table of hyperparameters

3 Results

Both task have trained with the parameters in TABLE 1 and the two primary objectives were achieved:

- Have any part of the robot arm touch the object of interest, with at least a 90 percent accuracy.

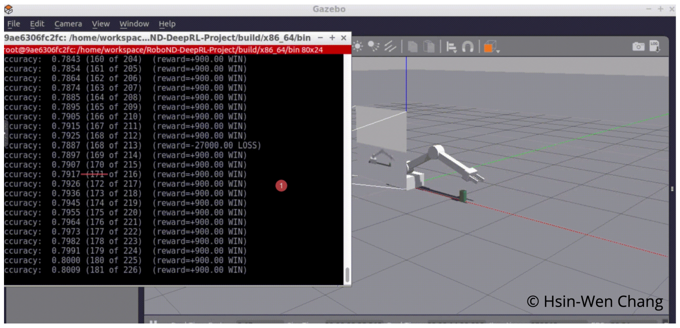


```
@9ae6306fc2fc: /home/workspac...ND-DeepRL-Project/build/x86_64/bin
root@9ae6306fc2fc: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Accuracy:  0.9000 (612 of 680)  (reward=+900.00 WIN)
Accuracy:  0.9001 (613 of 681)  (reward=+900.00 WIN)
Accuracy:  0.9003 (614 of 682)  (reward=+900.00 WIN)
Accuracy:  0.9004 (615 of 683)  (reward=+900.00 WIN)
Accuracy:  0.8991 (615 of 684)  (reward=-27000.00 LOSS)
Accuracy:  0.8993 (616 of 685)  (reward=+900.00 WIN)
Accuracy:  0.8994 (617 of 686)  (reward=+900.00 WIN)
Accuracy:  0.8996 (618 of 687)  (reward=+900.00 WIN)
Accuracy:  0.8997 (619 of 688)  (reward=+900.00 WIN)
Accuracy:  0.8999 (620 of 689)  (reward=+900.00 WIN)
Accuracy:  0.9000 (621 of 690)  (reward=+900.00 WIN)
Accuracy:  0.9001 (622 of 691)  (reward=+900.00 WIN)
Accuracy:  0.9003 (623 of 692)  (reward=+900.00 WIN)
Accuracy:  0.9004 (624 of 693)  (reward=+900.00 WIN)
Accuracy:  0.9006 (625 of 694)  (reward=+900.00 WIN)
Accuracy:  0.9007 (626 of 695)  (reward=+900.00 WIN)
Accuracy:  0.9009 (627 of 696)  (reward=+900.00 WIN)
Accuracy:  0.9010 (628 of 697)  (reward=+900.00 WIN)
Accuracy:  0.9011 (629 of 698)  (reward=+900.00 WIN)
Accuracy:  0.9013 (630 of 699)  (reward=+900.00 WIN)
Accuracy:  0.9014 (631 of 700)  (reward=+900.00 WIN)
Accuracy:  0.9016 (632 of 701)  (reward=+900.00 WIN)
Accuracy:  0.9017 (633 of 702)  (reward=+900.00 WIN)
```

Fig. 2. Achieved 90 percent.

- Have only the gripper base of the robot arm touch the object, with at least a 80 percent accuracy.

As Observed if don't add time penalty the robotic arm will start trembly around it's target and if the LSTM size lower than 256 will take longer time to reach the target accuracy.



```
9ae6306fc2fc: /home/workspac...ND-DeepRL-Project/build/x86_64/bin
root@9ae6306fc2fc: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
ccuracy:  0.7843 (160 of 204)  (reward=+900.00 WIN)
ccuracy:  0.7854 (161 of 205)  (reward=+900.00 WIN)
ccuracy:  0.7864 (162 of 206)  (reward=+900.00 WIN)
ccuracy:  0.7874 (163 of 207)  (reward=+900.00 WIN)
ccuracy:  0.7885 (164 of 208)  (reward=+900.00 WIN)
ccuracy:  0.7895 (165 of 209)  (reward=+900.00 WIN)
ccuracy:  0.7905 (166 of 210)  (reward=+900.00 WIN)
ccuracy:  0.7915 (167 of 211)  (reward=+900.00 WIN)
ccuracy:  0.7925 (168 of 212)  (reward=+900.00 WIN)
ccuracy:  0.7887 (168 of 213)  (reward=-27000.00 LOSS)
ccuracy:  0.7897 (169 of 214)  (reward=+900.00 WIN)
ccuracy:  0.7907 (170 of 215)  (reward=+900.00 WIN)
ccuracy:  0.7917 (171 of 216)  (reward=+900.00 WIN)
ccuracy:  0.7926 (172 of 217)  (reward=+900.00 WIN)
ccuracy:  0.7936 (173 of 218)  (reward=+900.00 WIN)
ccuracy:  0.7945 (174 of 219)  (reward=+900.00 WIN)
ccuracy:  0.7955 (175 of 220)  (reward=+900.00 WIN)
ccuracy:  0.7964 (176 of 221)  (reward=+900.00 WIN)
ccuracy:  0.7973 (177 of 222)  (reward=+900.00 WIN)
ccuracy:  0.7982 (178 of 223)  (reward=+900.00 WIN)
ccuracy:  0.7991 (179 of 224)  (reward=+900.00 WIN)
ccuracy:  0.8000 (180 of 225)  (reward=+900.00 WIN)
ccuracy:  0.8009 (181 of 226)  (reward=+900.00 WIN)
```

Fig. 3. Achieved 80 percent

4 Conclusion / Future work

Training a Deep Reinforcement Learning agent takes a lot of time tuning the LSTM size, image size, learning rate and batch size will influence the time reach it's target accuracy. The advantages of the C/C++ API develope by NVIDIA provide for robotics RL problems include leverage the power of GPU acceleration speed up RL agent training increase execution performance by using C/C++ compilation instead of interpreted python scripts optimized on specific embedded platform NVIDIA Jetson TX2