Deep RL Manipulator

ISRAILOV SARDOR

Abstract—The purpose of this paper is to present the DQN agent who will be able to learn with experience to first touch the object with any part of the robotic arm, then only with the gripper. The basic idea behind is that the camera takes frames from the gazebo environment passes them through neural networks with the help of CuDNN, CuDa, PyTorch and then outputs the actions for various joints to move forwards or backwards.

Index Terms—Robot, IEEEtran, RTAB, SLAM, Udacity, ROS, Gazebo, Rviz, Localization, Mapping.

1 Introduction

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

The goal of this project**DQN** (Deep Q-Network) agent and implement a reward system that permits the robotic arm to accomplish 2 following tasks:

1) have any part of the robotic arm touch the object of interest, with at least a 90% accuracy.
2) have the gripper of the robotic arm touch the object of interest, with at least a 90% accuracy.

The following tasks were accomplished as part of the project:

1) Subscribe to camera and collision topics published by Gazebo.
2) Create the DQN Agent and pass all required parameters to it.
3) Define position-based/velocity-based control strategy for arm joints.
4) Penalize robot gripper hitting the ground.
5) Interim Reward/Penalize based on the arm distance to the object implementing smooth moving average.
6) Reward based on collision between the arm and the object.
7) Reward based on collision between the arm's gripper base and the object.
8) Tuning the hyper-parameters.

2 Package structure

2.1 gazebo-arm.world

In this file, all the links and objects as well as their collisions were defined. the main components are:

1) The robotic arm with a gripper attached to it.
2) A camera sensor, to capture images to feed into the DQN.
3) A cylindrical object or prop.

2.2 C/C++ API

The API provides an interface to the Python code written with PyTorch, but the wrappers use Pythons low-level C to pass memory objects between the users application and Torch without extra copies. By using a compiled language (C/C++) instead of an interpreted one, performance is improved, and speeded up even more when GPU acceleration is leveraged.

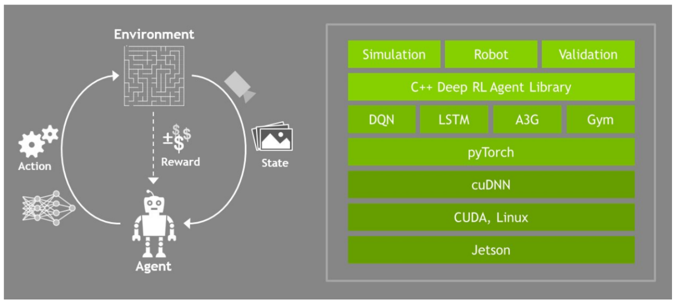


Fig. 1. representation of API stack provided by Nvidea

2.3 dqnAgent

is a class that inherits from the basic rlAgent class. One of the objectives of the project was to tune the parameters of an rl agent:

2.4 Arm plugin

The robotic arm model calls upon a gazebo plugin ArmPlugin. The gazebo plugin shared object file, libgazeboArmPlugin.so. The plugin is defined as ArmPlugin.cpp. Modyfing this file was the main objective of the project.

At every frame, the distance is calculated between two boxes (gripper and object) are calculated and an interim reward/punishement is issued. When there is a collision, a collision message is captured by the subscriber and the correspondent reward/punishement is issued.

3 Reward functions

Let's define a reward as WIN (accomplishing the task), and LOSS (failing). For task one: **WIN +300;LOSS -300**

For task two: **WIN +1500;LOSS -300**

The essence of DQN agent is to learn via reward/punish system. The reward system included:

- any part of the robot touching the cylinder object-WIN
- the gripper of the robotic arm approached the object

The punish system included:

- the robot was too slow, so the camera received 100 frames– end of episode-LOSS
- the robot(gripper) touches the ground-LOSS
- the gripper of a robot is going further away from the object interim punishment
- in case when only the gripper was required to touch the object, the collision between any part of a robot except the gripper and an object was punished (0.1*LOSS).

There exists both **velocity** and **position** control or the mix of both, however only position control was chosen.

4 Hyper-parameters

Following are the hyper-parameters proposed to change in order to optimise the DQN agent

- **USE_LSTM** The usage of the **LONG SHORT TERM MEMORY** is essential while using RNN(recursive neural networks). That permits to have a memory that the DQN agent can tap into.
- **LSTM SIZE** indicates the size of each cell of LSTM. The bigger, the more memory is required. This potentially may improve the overall results. The chosen size was **512**.
- **LEARNING_RATE** tells how far the weights should be updated. For the first task, 0.2 was enough. The learning rate was diminished 0.05 in order to attain the required accuracy for the 2nd task.
- **OPTIMISER** Although there are many different optimisers(Adagrad, SGD, Adadelta...), best results were with RMSprop and Adam. In coparision to other optimisers, RMSprop scales the learning rate so the algorithms goes through saddle point faster than most
- **REPLAY MEMORY** A circular buffer that stores the transitions that the DQN agent observes. That number stayed constant 10000.
- **BATCH SIZE** is the size of the sample that we take every time while training, similar to CNN. So the greater this number, the less iteration we have to do.
- **INPUT width & INPUT height** every camera frame is fed to DQN agent. 64x64 was used for the 1st case, and 128 x 128 was used for the second application. This is one of the most evident parameters to tune in order to obtain good results.
- REWARD_WIN(+300 1st case; +1500 2nd) is the reward issued while winning, or accomplishing the task.

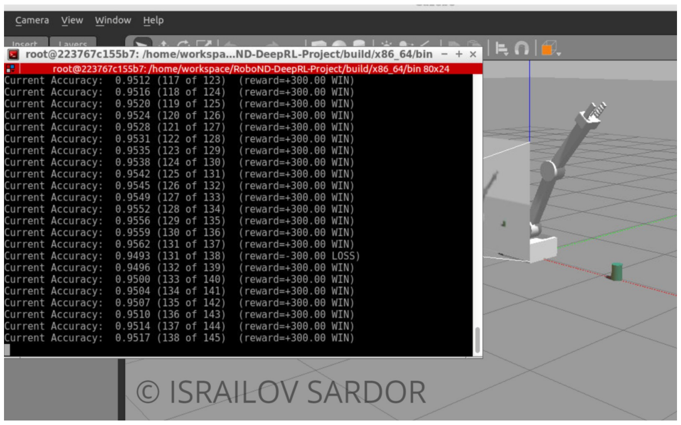
Other hyper-parameters include:

- **INPUT CHANNELS=3**. The number of input channels RGB image.
- **GAMMA**=0.9 is DISCOUNT RATE. It is responsible for the value of future rewards.
- **EPS START=0.9f; EPS END=0.05f; EPS DECAY=200** are the coefficient of the exponential decay.

INPUT

5 Results

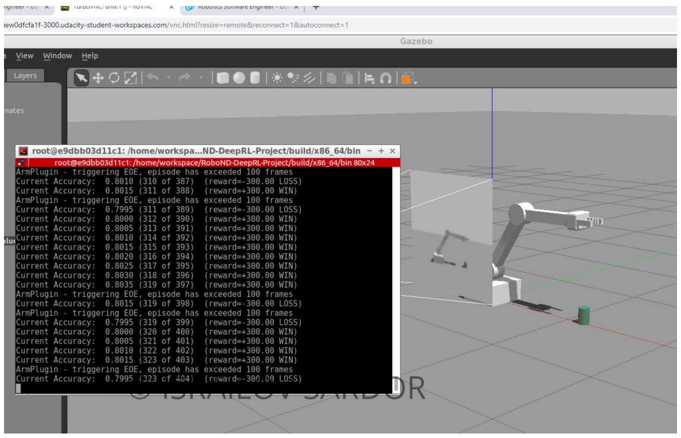
Both the requirement were fulfilled:



```
root@223767c155b7: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@223767c155b7: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Current Accuracy:  0.9512 (117 of 123)  (reward=+300.00 WIN)
Current Accuracy:  0.9516 (118 of 124)  (reward=+300.00 WIN)
Current Accuracy:  0.9520 (119 of 125)  (reward=+300.00 WIN)
Current Accuracy:  0.9524 (120 of 126)  (reward=+300.00 WIN)
Current Accuracy:  0.9528 (121 of 127)  (reward=+300.00 WIN)
Current Accuracy:  0.9531 (122 of 128)  (reward=+300.00 WIN)
Current Accuracy:  0.9535 (123 of 129)  (reward=+300.00 WIN)
Current Accuracy:  0.9538 (124 of 130)  (reward=+300.00 WIN)
Current Accuracy:  0.9542 (125 of 131)  (reward=+300.00 WIN)
Current Accuracy:  0.9545 (126 of 132)  (reward=+300.00 WIN)
Current Accuracy:  0.9549 (127 of 133)  (reward=+300.00 WIN)
Current Accuracy:  0.9552 (128 of 134)  (reward=+300.00 WIN)
Current Accuracy:  0.9556 (129 of 135)  (reward=+300.00 WIN)
Current Accuracy:  0.9559 (130 of 136)  (reward=+300.00 WIN)
Current Accuracy:  0.9562 (131 of 137)  (reward=+300.00 WIN)
Current Accuracy:  0.9493 (131 of 138)  (reward=-300.00 LOSS)
Current Accuracy:  0.9496 (132 of 139)  (reward=+300.00 WIN)
Current Accuracy:  0.9500 (133 of 140)  (reward=+300.00 WIN)
Current Accuracy:  0.9504 (134 of 141)  (reward=+300.00 WIN)
Current Accuracy:  0.9507 (135 of 142)  (reward=+300.00 WIN)
Current Accuracy:  0.9510 (136 of 143)  (reward=+300.00 WIN)
Current Accuracy:  0.9514 (137 of 144)  (reward=+300.00 WIN)
Current Accuracy:  0.9517 (138 of 145)  (reward=+300.00 WIN)
```

Fig. 2. accomplishing MORE THAN 90% after 100th try

The overall score of 95% was achieved, and still increasing. The inevitable divergence will occur at some point, so the best technique is to adapt the learning rate the the precision wanted and stop the learning when the algorithm overfits.



```
root@e9dbb03d11c1: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@e9dbb03d11c1: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
ArmPlugin - triggering EOE, episode has exceeded 100 frames
Current Accuracy:  0.8010 (310 of 387)  (reward=-300.00 LOSS)
Current Accuracy:  0.8015 (311 of 388)  (reward=+300.00 WIN)
ArmPlugin - triggering EOE, episode has exceeded 100 frames
Current Accuracy:  0.7995 (311 of 389)  (reward=-300.00 LOSS)
Current Accuracy:  0.8000 (312 of 390)  (reward=+300.00 WIN)
Current Accuracy:  0.8005 (313 of 391)  (reward=+300.00 WIN)
Current Accuracy:  0.8010 (314 of 392)  (reward=+300.00 WIN)
Current Accuracy:  0.8015 (315 of 393)  (reward=+300.00 WIN)
Current Accuracy:  0.8020 (316 of 394)  (reward=+300.00 WIN)
Current Accuracy:  0.8025 (317 of 395)  (reward=+300.00 WIN)
Current Accuracy:  0.8030 (318 of 396)  (reward=+300.00 WIN)
Current Accuracy:  0.8035 (319 of 397)  (reward=+300.00 WIN)
ArmPlugin - triggering EOE, episode has exceeded 100 frames
Current Accuracy:  0.8015 (319 of 398)  (reward=-300.00 LOSS)
ArmPlugin - triggering EOE, episode has exceeded 100 frames
Current Accuracy:  0.7995 (319 of 399)  (reward=-300.00 LOSS)
Current Accuracy:  0.8000 (320 of 400)  (reward=+300.00 WIN)
Current Accuracy:  0.8005 (321 of 401)  (reward=+300.00 WIN)
Current Accuracy:  0.8010 (322 of 402)  (reward=+300.00 WIN)
Current Accuracy:  0.8015 (323 of 403)  (reward=+300.00 WIN)
ArmPlugin - triggering EOE, episode has exceeded 100 frames
Current Accuracy:  0.7995 (323 of 404)  (reward=-300.00 LOSS)
```

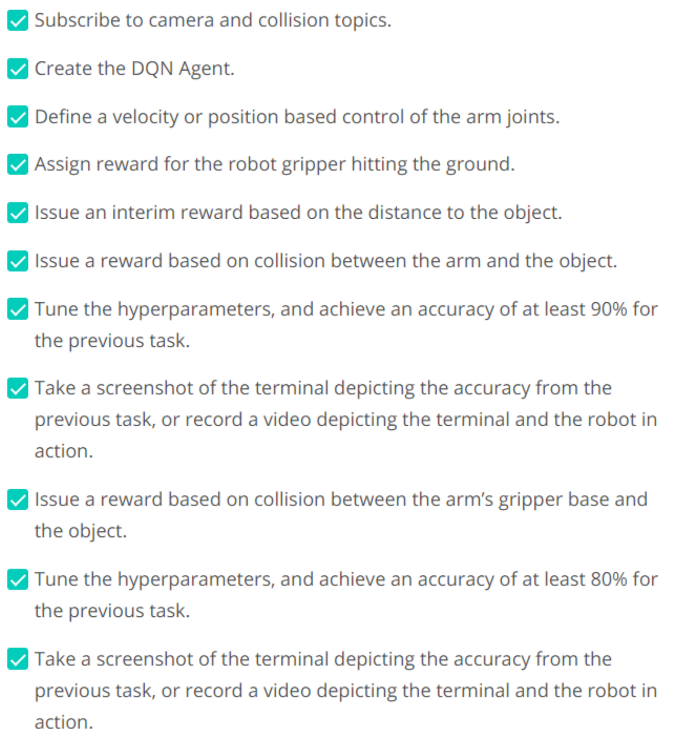
Fig. 3. the score when only the gripper was required to touch

5.1 Discussion

The overall accuracy was satisfactory for both tasks. For task#1, the accuracy stabilised around 95%, while for task#2 the accuracy was about 80%. There are several ways to improve the accuracy:

1) Use smaller learning_rate and take more time to converge to the increased accuracy
2) increase the buffer size allocated for camera image as well as INPUT_WIDTH AND INPUT_HEIGHT of DQN for 256x256.

3) if memory permits increase the LSTM cell size to 1024. In my case that caused the memory issues.
4) Tune precisely the REWARD_WIN reward_loss as well as alpha for the accuracy.



✓ Subscribe to camera and collision topics.

✓ Create the DQN Agent.

✓ Define a velocity or position based control of the arm joints.

✓ Assign reward for the robot gripper hitting the ground.

✓ Issue an interim reward based on the distance to the object.

✓ Issue a reward based on collision between the arm and the object.

✓ Tune the hyperparameters, and achieve an accuracy of at least 90% for the previous task.

✓ Take a screenshot of the terminal depicting the accuracy from the previous task, or record a video depicting the terminal and the robot in action.

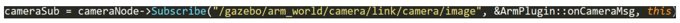
✓ Issue a reward based on collision between the arm's gripper base and the object.

✓ Tune the hyperparameters, and achieve an accuracy of at least 80% for the previous task.

✓ Take a screenshot of the terminal depicting the accuracy from the previous task, or record a video depicting the terminal and the robot in action.

Fig. 4. The overall tasks accomplished: task list provided by udacity

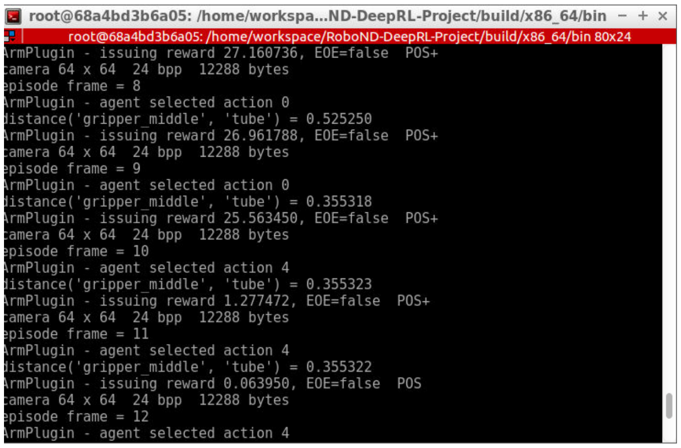
6 Code extracts



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

Fig. 5. subscriber example

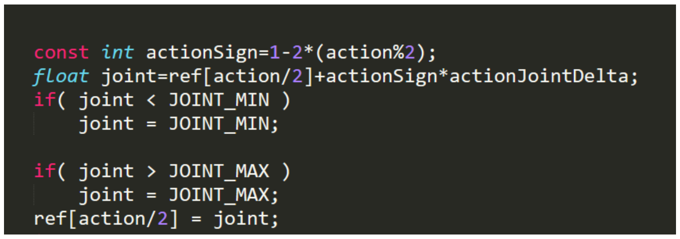
The input arguments of the method are: the topic name, callback function, and the class instance.



root@68a4bd3b6a05: /home/workspa...ND-DeepRL-Project/build/x86_64/bin

```
root@68a4bd3b6a05: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
ArmPlugin - issuing reward 27.160736, EOE=false  POS+
camera 64 x 64  24 bpp  12288 bytes
episode frame = 8
ArmPlugin - agent selected action 0
distance('gripper_middle', 'tube') = 0.525250
ArmPlugin - issuing reward 26.961788, EOE=false  POS+
camera 64 x 64  24 bpp  12288 bytes
episode frame = 9
ArmPlugin - agent selected action 0
distance('gripper_middle', 'tube') = 0.355318
ArmPlugin - issuing reward 25.563450, EOE=false  POS+
camera 64 x 64  24 bpp  12288 bytes
episode frame = 10
ArmPlugin - agent selected action 4
distance('gripper_middle', 'tube') = 0.355323
ArmPlugin - issuing reward 1.277472, EOE=false  POS+
camera 64 x 64  24 bpp  12288 bytes
episode frame = 11
ArmPlugin - agent selected action 4
distance('gripper_middle', 'tube') = 0.355322
ArmPlugin - issuing reward 0.063950, EOE=false  POS
camera 64 x 64  24 bpp  12288 bytes
episode frame = 12
ArmPlugin - agent selected action 4
```

Fig. 6. terminal in a debug mode



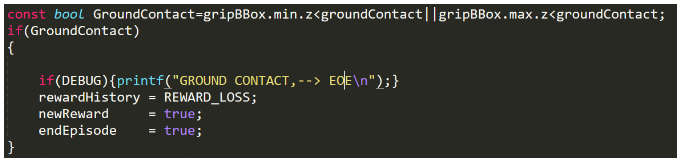
```
const int actionSign=1-2*(action%2);
float joint=ref[action/2]+actionSign*actionJointDelta;
if( joint < JOINT_MIN )
    joint = JOINT_MIN;

if( joint > JOINT_MAX )
    joint = JOINT_MAX;
ref[action/2] = joint;
```

Fig. 7. POSITION CONTROL with a smoothing technique

Debug mode helped to visualise the messages, for instance: the distance between the object and the arm is ... In some cases, it served to eliminate errors.

Overall there are 6 actions, either to increase or decrease any 3 of joints.



```
const bool GroundContact=gripBBox.min.z<groundContact||gripBBox.max.z<groundContact;
if(GroundContact)
{

    if(DEBUG){printf("GROUND CONTACT,--> EOE\n");}
    rewardHistory = REWARD_LOSS;
    newReward     = true;
    endEpisode    = true;
}
```

Fig. 8. collision check

The collision check with the ground and gripper. There is the method in gazebo API, **getBoundingBox()** that enabled to calculated the distance of two objects with the helper method. **BoxDistance()**.

7 Future work&hardware deployment

The future work would include the randomnisation of an object and implementing the robotic arm with more axes of freedom(DOF).

The hardware deployment would include either jetson tx2 with robotic arm such as kuka, staubli etc.

8 References

1) Retrieved september 12, 2019, from http://wiki.ros.org
2) Retrieved september 12, 2019, from https://udacity.com