Deep RL Arm Manipulation

Xueyang Kang

Abstract—Reinforcement Learning (RL) has been widely used in lot of applications, due to its robustness and easy implementation, without need to use the training dataset. Furthermore, RL may shed some light on the robotic field for training the robot to perform more complex tasks in real world. In this paper a deep Q-value based neural network is trained to control the manipulator to reach the static tube at ground, either touch by any part of arm or by gripper, the explanation to the design and reward functions, tuning work of hyper-parameters for the network training are presented in this paper to help the reader to shape the thoughts about how to make use of deep neural network for a specific task. In the end, the performance of the network is summarized and future improvement is proposed.

Index Terms—Reinforcement Learning, Robot, deep neural network, manipulator.

1 Introduction

Reinforcement Learning doesn't require the labeled data for training. The agent to determine the control policy is trained with both inputs and outputs from the environment. Normally the camera captures the information of the environment, also reflecting the influence of the agent's output. The output of the agent is mapped into the real state value of the controlling target. The reward is used to encourage the agent to memorize the actions which can generate the goal achievement. A standard machine learning based system, like a manipulator is introduced in [1], no prior knowledge to the task is needed, only raw-pixel images are passed to the neural network to train a Deep Q Network.

2 Reward

The reward function is composed of several piece-wise sub-rewards.

2.1 Task 1

For task 1, the velocity controller is used to control the manipulator, the action is relatively fast.

- Reward for the episode time-out. End of episode is enabled. The punishment also integrates the apart proportional to the distance between gripper and reaching target.

$$reward = -10.0 - DistToGoal \tag{1}$$

- Reward for collision between the ground and any part of the arm. End of episode is enabled. It also includes a proportional part indicating distance to the goal. The farther away from the goal, the more punishment is given to the agent. Because this ground contact will cause heavy damage to the arm, and should be avoided in practice, a less punishment is given compared to the one in time-out case.

$$reward = -5.0 - DistToGoal \tag{2}$$

- Interim reward for the manipulator. No end of episode. The euclidean distance between gripper and tube is applied to control the moving direction of the manipulator. Reward is also dependent on the past reward history. Weight of the change can be tuned by α, ranging from 0 to 1.

$$DistDelta = LastDistToGoal - DistToGoal \tag{3}$$

$$reward = \alpha \cdot AverageGoalDist + (1-\alpha) DistDelta \tag{4}$$

- Reward for collision between the tube and any part of the arm. End of episode is enabled. A big positive reward is assigned if the goal is achieved.

$$reward = +20.0 \tag{5}$$

2.2 Task 2

In this task the position controller is applied to achieve a high accuracy of control.

- Reward for the episode time-out, same as task 1.

$$reward = -10.0 - DistToGoal \tag{6}$$

- Reward for ground contact detection, same as task 1.

$$reward = -5.0 - DistToGoal \tag{7}$$

- Interim reward for the manipulator is more complex compared to the one for task 1. No end of episode is used.

$$reward = \begin{cases} 10 \cdot e^{-DistToGoal} & if DistDelta > 0 \\ -2 - DistToGoal & if DistDelta \leq 0 \end{cases}$$

- Reward for collision between the tube and any part of the arm. End of episode is enabled. But this part is different from the one in task 1. The contact part of the arm should be gripper only, other parts of the arm comes into contact with target should be punished with a negative value, but relatively small to differentiate it from the other bad behaviors.

$$reward = \begin{cases} -2.0 - DistToGoal & if \quad touch \quad by \quad link2 \\ +20.0 & if \quad touch \quad by \quad gripper \end{cases}$$

3 Hyper-parameters

The hyper-parameters are tightly coupled with the training performance. From Table 1, the most hyper-parameters are

TABLE 1: Hyper-parameter comparison

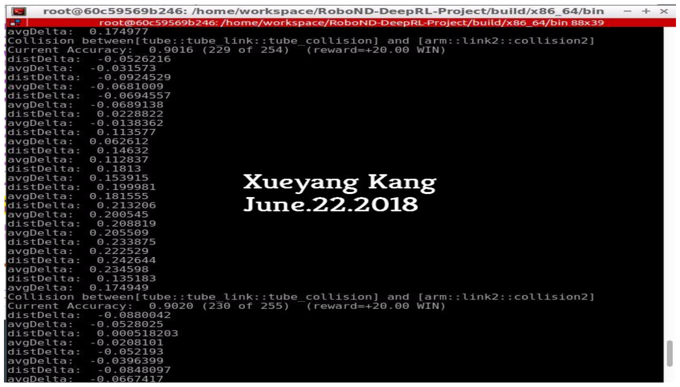
Hyper-parameter	TASK1	TASK2
NUM_ACTIONS	6	6
INPUT_CHANNELS	3	3
GAMMA	0.9f	0.9F
EPS_START	0.9f	0.9F
EPS_END	0.01f	0.01F
EPS_DECAY	80	100
INPUT_WIDTH	64	64
INPUT_HEIGHT	64	64
OPTIMIZER	”RMSprop”	”RMSprop”
LEARNING_RATE	0.08f	0.001f
REPLAY_MEMORY	10000	10000
BATCH_SIZE	8	8
USE_LSTM	true	
LSTM_SIZE	16	16

kept the same in both tasks, the image has three channels and the action number is six, three outputs imposed on three joints, each type has two kinds, increase or decrease, so the number of actions is six. GAMMA determines the decaying rate of the contribution from past rewards, default value is adopted. ”EPS*” values decide the random possibility of exploration, it is dependent on the converging rate of the agent to find the optimal control policy. Initially, the ”EPS” star value should be set to a big value to explore the state space thoroughly, while the end value can be subtle in order to reduce the possibility for manipulator to perform some weird behaviors, after finding the correct actions. The decaying rate of random value is dependent on the training epochs required in practice, here around 100 is enough.

Input and width dimensionality of the image is set to 64, otherwise too big dimensionality will run out of the memory. In terms of optimizer, the ”RMSprop”, which is a mini-batch version of ”RPprop”, dived by a different magnitude number for each mini-batch. SO it keeps a moving average squared gradient for each weight. Learning rate differs in the two cases. In terms of LSTM settings, like replay memory and size are kept with default values, because through experiment these parameter doesn't affect the performance too much. Learning rate for the task 2 is relative small, because the valid touch part is smaller compared to task 1, so to from a proper way to reach the goal, a slow learning rate can assure the learning is not converging too fast to ignore the correct solution.

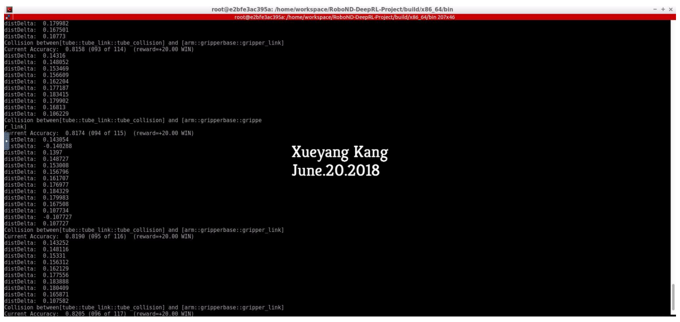
4 Results Discussion

Figure 1 is the terminal result of DQN for task 1, the accuracy can reach 90% after about 200 episodes. Initially the network may repeat with some abnormal motions, like touching the ground. As the time goes, after 30-50 episodes' run, the manipulator can reach the tube with the link2 perfectly. But the velocity control results in the fast motion and hitting the target very hard, even performing some behaviors violating the physic law, this shouldn't occur in reality otherwise the machine will be destroyed.


root@60c59569b246: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin
root@60c59569b246: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 88x39
Xueyang Kang
June.22.2018

```
avgDelta:  0.174977
Collision between[tube::tube_link::tube_collision] and [arm::link2::collision2]
Current Accuracy:  0.9016 (229 of 254)  (reward=+20.00 WIN)
distDelta:  -0.0526216
avgDelta:  -0.031573
distDelta:  -0.0924529
avgDelta:  -0.0681009
distDelta:  -0.0694557
avgDelta:  -0.0689138
distDelta:  0.0228822
avgDelta:  -0.0138362
distDelta:  0.113577
avgDelta:  0.062612
distDelta:  0.14632
avgDelta:  0.112837
distDelta:  0.1813
avgDelta:  0.153915
distDelta:  0.199981
avgDelta:  0.181555
distDelta:  0.213206
avgDelta:  0.200545
distDelta:  0.208819
avgDelta:  0.205509
distDelta:  0.233875
avgDelta:  0.222529
distDelta:  0.242644
avgDelta:  0.234598
distDelta:  0.135183
avgDelta:  0.174949
Collision between[tube::tube_link::tube_collision] and [arm::link2::collision2]
Current Accuracy:  0.9020 (230 of 255)  (reward=+20.00 WIN)
distDelta:  -0.0880042
avgDelta:  -0.0528025
distDelta:  0.000518203
avgDelta:  -0.0208101
distDelta:  -0.052193
avgDelta:  -0.0396399
distDelta:  -0.0848097
avgDelta:  -0.0667417
```

Fig. 1: Result of task 1


root@e2bfe3ac395a: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin
Xueyang Kang
June.20.2018

```
distDelta:  0.179982
distDelta:  0.167501
distDelta:  0.10773
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripper_link]
Current Accuracy:  0.8158 (093 of 114)  (reward=+20.00 WIN)
distDelta:  0.14316
distDelta:  0.148052
distDelta:  0.153469
distDelta:  0.156609
distDelta:  0.162204
distDelta:  0.177187
distDelta:  0.183415
distDelta:  0.179902
distDelta:  0.16813
distDelta:  0.106229
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::grippe
r_link]
Current Accuracy:  0.8174 (094 of 115)  (reward=+20.00 WIN)
distDelta:  0.143054
distDelta:  -0.140288
distDelta:  0.1397
distDelta:  0.148727
distDelta:  0.153008
distDelta:  0.156796
distDelta:  0.161707
distDelta:  0.176977
distDelta:  0.184329
distDelta:  0.179983
distDelta:  0.167508
distDelta:  0.107734
distDelta:  -0.107727
distDelta:  0.107727
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripper_link]
Current Accuracy:  0.8190 (095 of 116)  (reward=+20.00 WIN)
distDelta:  0.143252
distDelta:  0.148116
distDelta:  0.15331
distDelta:  0.156312
distDelta:  0.162129
distDelta:  0.177556
distDelta:  0.183888
distDelta:  0.180409
distDelta:  0.165871
distDelta:  0.107582
Collision between[tube::tube_link::tube_collision] and [arm::gripperbase::gripper_link]
Current Accuracy:  0.8205 (096 of 117)  (reward=+20.00 WIN)
```

Fig. 2: Result of task 2

As for task 2, the work is quite not easy. As the successful case shown in 2, the overall accuracy, 80% is achieved after around 100 episodes. The same parameter configuration For the agent will have different performance after each launching. To make sure a fast converging rate to the right Q value can be realized, during experimentation, if the initial 30-50 episodes will not be done successfully with at least 2-6%, then the training will be terminated manually, in this way, time can be saved to avoid the situation, when many following successful episodes should happen to compensate the episodes failure. So a good start can be applied to make sure the DQN can memorize the right action series to achieve the goal with less training episodes. The DQN will get accustomed to using link2 to touch the tube in most of experimentation time, although a position control is accurate and result in a relative soft contact with the tube. Initially after touch by link2, I let the manipulator continue moving without termination to explore further, but it normally flipped up the tube totally and even threw it away, so the episode is terminated if the arm touched it by the wrong part, in this way it makes the DQN still hard to learn the touch by gripper instead of by link2, but at least the random exploration rate will happen to result in a reaching in proper way. The figure 3 is a demo of task2, the gripper touches the tube properly.

5 Conclusion / Future work

To make a short wrap-up of the work, the DQN can complete the challenging 1 with less episodes within total 100 runs, but some glitches like the velocity at reaching tube is not zero will make the arm crash into the tube and the

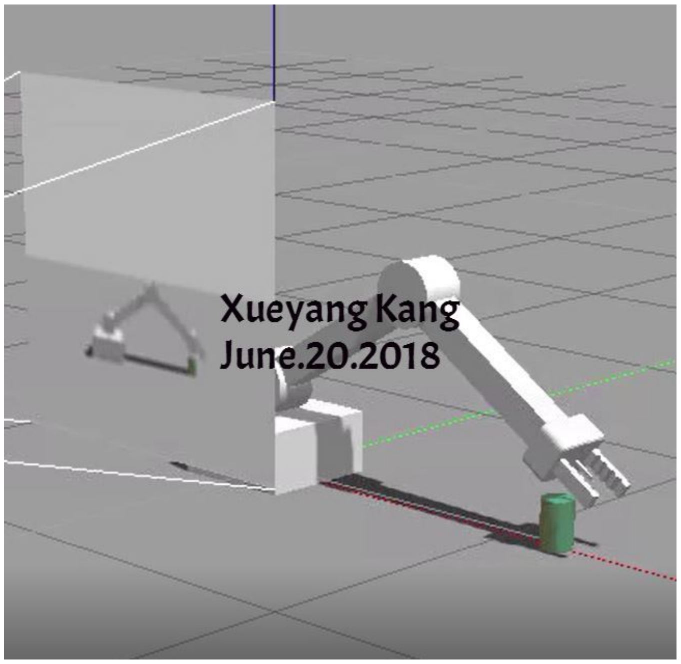


Fig. 3: Reach target with gripper

ground. In terms of task 2, DQN requires several successful episodes within the initial 30-50 runs as a good start to converge to the right values, sometimes the manipulator will consider using the link2 to touch tube as the goal, so some improvement work on reward may help to solve this.

To improve the accuracy, the depth of DQN network can be reduced somewhat, because the training dataset is relatively small for a deep network to extract the features from images, some connected layers within network can even be modified to be adaptive to this specific task.

For task 2, a more complex reward can be tried, especially the interim reward which controls the moving direction to the goal, if this transition can be performed elegantly to be connected to the ultimate goal. Aside from that, more features from input can be added into the training process. Someone in the slack channel even used additional camera from a top viewpoint to capture the additional dimensional information of the tube, or arm with moving base, even the texture or the color information can being added to the arm, to make each part of the arm be more distinctive, much easier to be recognized by the network, these proposed approaches to improve the accuracy can be tried in the future work.

References

[1] M. M. e. a. Zhang F, Leitner J, "Towards vision-based deep reinforcement learning for robotic motion control[j]." arXiv preprint arXiv:1511.03791, 2015., April, 04 2017.