Deep RL Arm Manipulation

Seongyoon Cho

Reward Functions

Reward Constants

The constants for winning and losing rewards are defined like these.

```
REWARD_WIN = 2000.0f
REWARD_LOSS = -2000.0f
```

Goal Reward

`reward = REWARD_WIN`

If the robot achieves the goal, `REWARD_WIN` is issued and the episode ends. This value is set to a relatively high to give a significant amount of reward compared to **interim reward** introduced later. The reward is the same for both tasks, a task for any part of a robot arm touches the object, and the other task for gripper base touches the object.

Reward for Touching the Ground

`reward = REWARD_LOSS * distGoal`

If the gripper base touches the ground, a negative reward is issued, and the episode ends. This value is multiplied by a distance between the gripper and the target object. i.e., if the gripper touches the ground far from the target object, a more significant negative value is issued. If the gripper just slightly misses the target object and hit the ground, a small negative value is issued.

Interim Reward

`avgGoalDelta = (avgGoalDelta * alpha) + (distDelta * (1.0f - alpha))`

While the robot is in motion, an interim reward is issued to encourage robot to move towards to the target. The reward is smoothed with an alpha value of 0.2.

Joint Control

Both velocity control and position control is implemented for this project. But only position control made a promising result. Velocity control shows smoother movements but somehow failed to achieve over 0.7 of accuracy for task 2.

Hyperparameters

`INPUT_WIDTH` and `INPUT_HEIGHT`

512 is so big that it causes memory problems. 64 is chosen because it's small enough to run smoothly and big enough to get information.

`OPTIMIZER`

`RMSprop` is selected because it's a popular choice. It's a gradient descent with momentum, so it reduces oscillations and converges faster.

`LEARNING_RATE`

After many trial and errors, 0.05 is selected. If I set it higher, sometimes the accuracy oscillates. If I set it lower, it takes too long to achieve the desired accuracy or even it gets stuck on low accuracy.

`REPLAY_MEMORY` and `BATCH_SIZE`

Replay memory is left to 10000 as it is. Batch size is slightly increased to 64. There were no memory issues.

`USE_LSTM` and `LSTM_SIZE`

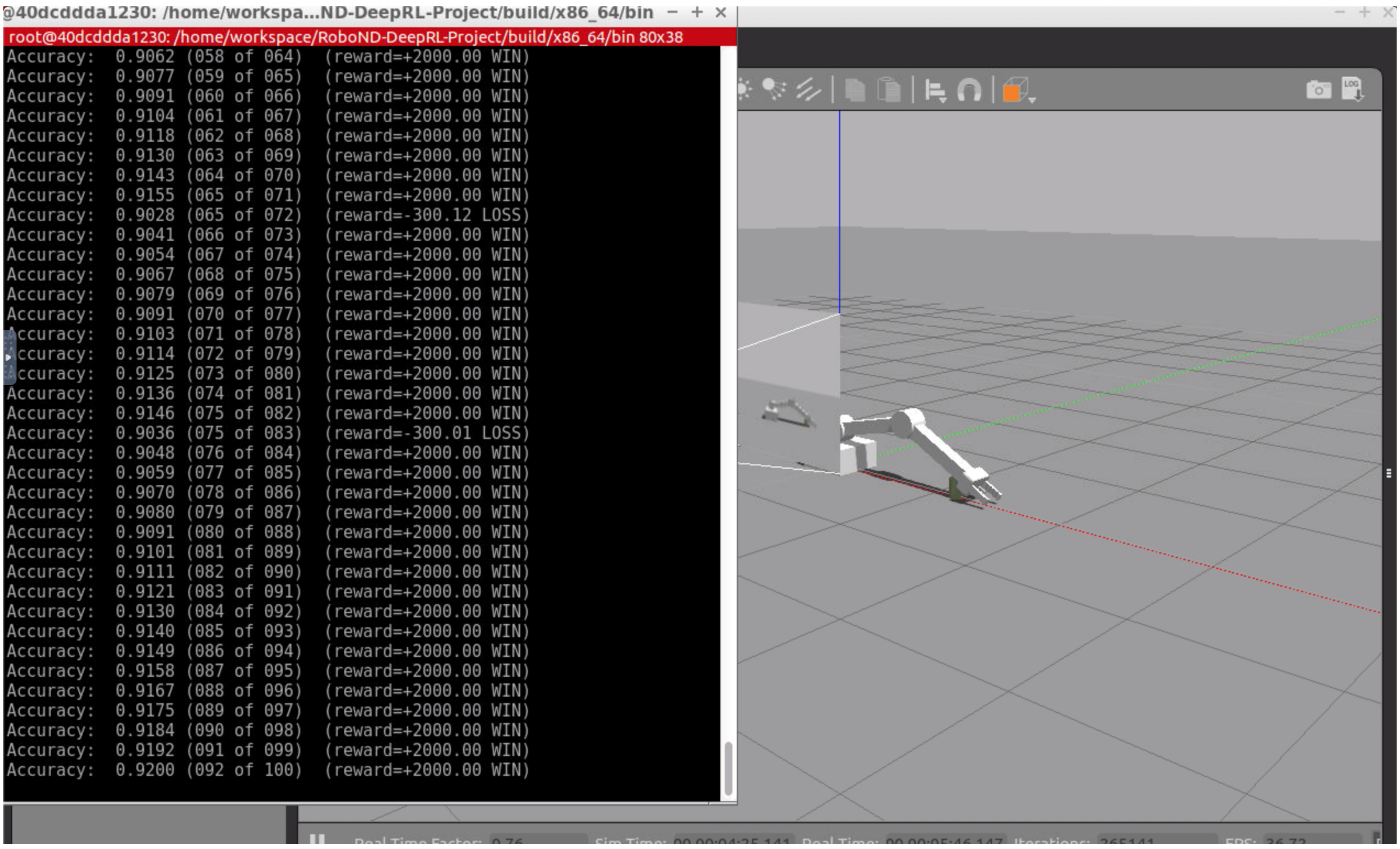
To utilize LSTM, `USE_LSTM` is set to `true` and `LSTM_SIZE` is set to 256.

Results

With some iterations, I could get successful results which satisfy both conditions. But how fast it converges depends on starting a few episodes. If it starts with successful episodes, it seldom tries other things and successfully converges over 90 percent accuracy. But if it starts with some errornous episodes, it takes longer to achieve 90 percent accuracy. Sometimes, it takes more than 300 episodes.

Collision between the arm and the object

This is one of the most successful runs. You can see the arm tries various angles at first few episodes. Also, there's a lot of hesitations. Afterward, the angle is constant and no hesitation at all. The accuracy 92 percent.

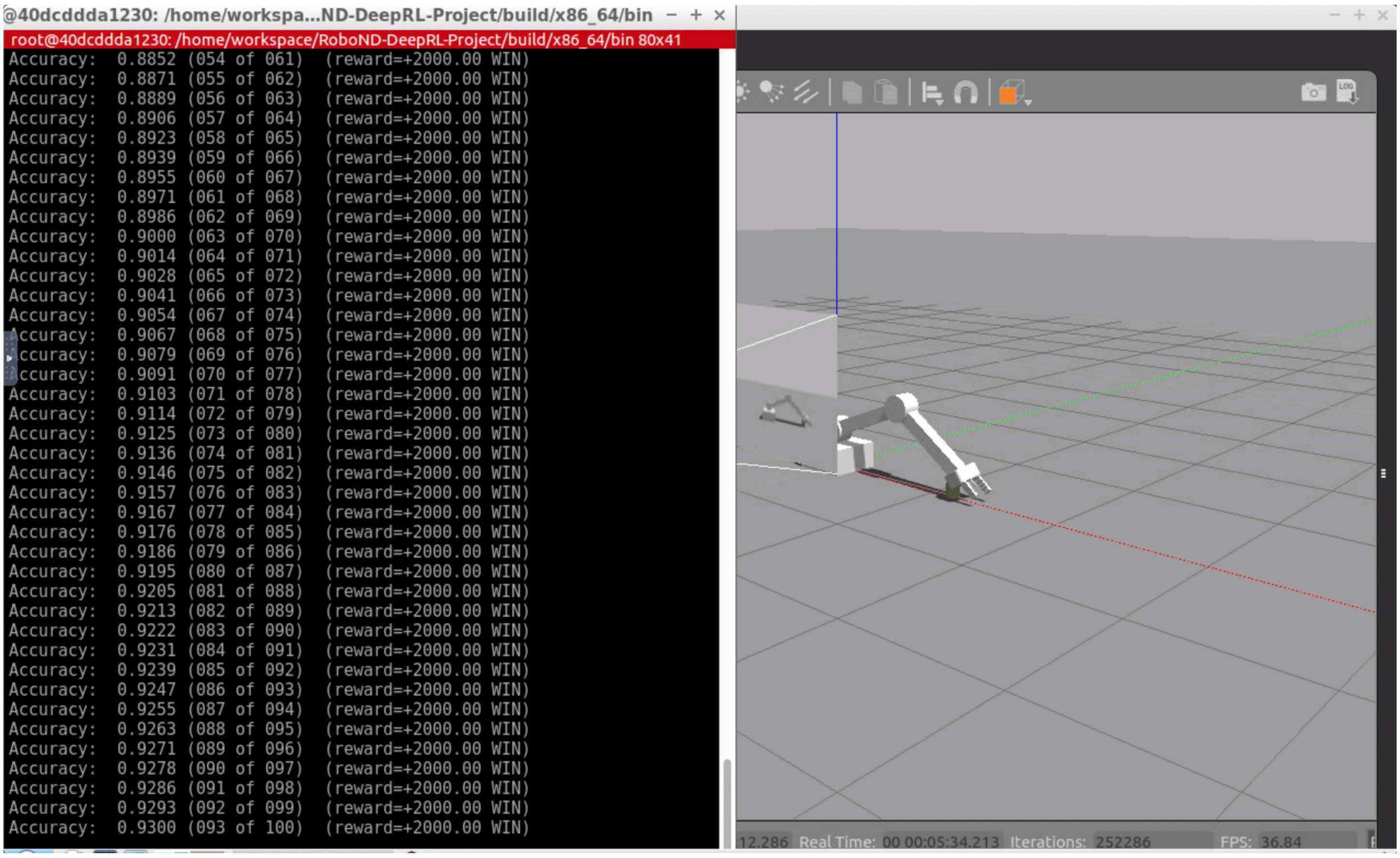


```
@40dcddda1230: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@40dcddda1230: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x38
Accuracy:  0.9062 (058 of 064)  (reward=+2000.00 WIN)
Accuracy:  0.9077 (059 of 065)  (reward=+2000.00 WIN)
Accuracy:  0.9091 (060 of 066)  (reward=+2000.00 WIN)
Accuracy:  0.9104 (061 of 067)  (reward=+2000.00 WIN)
Accuracy:  0.9118 (062 of 068)  (reward=+2000.00 WIN)
Accuracy:  0.9130 (063 of 069)  (reward=+2000.00 WIN)
Accuracy:  0.9143 (064 of 070)  (reward=+2000.00 WIN)
Accuracy:  0.9155 (065 of 071)  (reward=+2000.00 WIN)
Accuracy:  0.9028 (065 of 072)  (reward=-300.12 LOSS)
Accuracy:  0.9041 (066 of 073)  (reward=+2000.00 WIN)
Accuracy:  0.9054 (067 of 074)  (reward=+2000.00 WIN)
Accuracy:  0.9067 (068 of 075)  (reward=+2000.00 WIN)
Accuracy:  0.9079 (069 of 076)  (reward=+2000.00 WIN)
Accuracy:  0.9091 (070 of 077)  (reward=+2000.00 WIN)
Accuracy:  0.9103 (071 of 078)  (reward=+2000.00 WIN)
Accuracy:  0.9114 (072 of 079)  (reward=+2000.00 WIN)
Accuracy:  0.9125 (073 of 080)  (reward=+2000.00 WIN)
Accuracy:  0.9136 (074 of 081)  (reward=+2000.00 WIN)
Accuracy:  0.9146 (075 of 082)  (reward=+2000.00 WIN)
Accuracy:  0.9036 (075 of 083)  (reward=-300.01 LOSS)
Accuracy:  0.9048 (076 of 084)  (reward=+2000.00 WIN)
Accuracy:  0.9059 (077 of 085)  (reward=+2000.00 WIN)
Accuracy:  0.9070 (078 of 086)  (reward=+2000.00 WIN)
Accuracy:  0.9080 (079 of 087)  (reward=+2000.00 WIN)
Accuracy:  0.9091 (080 of 088)  (reward=+2000.00 WIN)
Accuracy:  0.9101 (081 of 089)  (reward=+2000.00 WIN)
Accuracy:  0.9111 (082 of 090)  (reward=+2000.00 WIN)
Accuracy:  0.9121 (083 of 091)  (reward=+2000.00 WIN)
Accuracy:  0.9130 (084 of 092)  (reward=+2000.00 WIN)
Accuracy:  0.9140 (085 of 093)  (reward=+2000.00 WIN)
Accuracy:  0.9149 (086 of 094)  (reward=+2000.00 WIN)
Accuracy:  0.9158 (087 of 095)  (reward=+2000.00 WIN)
Accuracy:  0.9167 (088 of 096)  (reward=+2000.00 WIN)
Accuracy:  0.9175 (089 of 097)  (reward=+2000.00 WIN)
Accuracy:  0.9184 (090 of 098)  (reward=+2000.00 WIN)
Accuracy:  0.9192 (091 of 099)  (reward=+2000.00 WIN)
Accuracy:  0.9200 (092 of 100)  (reward=+2000.00 WIN)
```

https://youtu.be/7NeBbCoMLow

Collision between the gripper base and the object

One interesting thing I noticed with this run is, at around 90 episodes, if the arm reaches too far, it immediately corrects the angle. The accuracy is 93 percent.


@40dcddda1230: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@40dcddda1230: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x41
12.286 Real Time: 00 00:05:34.213 Iterations: 252286 FPS: 36.84

```
Accuracy:  0.8852 (054 of 061)  (reward=+2000.00 WIN)
Accuracy:  0.8871 (055 of 062)  (reward=+2000.00 WIN)
Accuracy:  0.8889 (056 of 063)  (reward=+2000.00 WIN)
Accuracy:  0.8906 (057 of 064)  (reward=+2000.00 WIN)
Accuracy:  0.8923 (058 of 065)  (reward=+2000.00 WIN)
Accuracy:  0.8939 (059 of 066)  (reward=+2000.00 WIN)
Accuracy:  0.8955 (060 of 067)  (reward=+2000.00 WIN)
Accuracy:  0.8971 (061 of 068)  (reward=+2000.00 WIN)
Accuracy:  0.8986 (062 of 069)  (reward=+2000.00 WIN)
Accuracy:  0.9000 (063 of 070)  (reward=+2000.00 WIN)
Accuracy:  0.9014 (064 of 071)  (reward=+2000.00 WIN)
Accuracy:  0.9028 (065 of 072)  (reward=+2000.00 WIN)
Accuracy:  0.9041 (066 of 073)  (reward=+2000.00 WIN)
Accuracy:  0.9054 (067 of 074)  (reward=+2000.00 WIN)
 ccuracy:  0.9067 (068 of 075)  (reward=+2000.00 WIN)
 ccuracy:  0.9079 (069 of 076)  (reward=+2000.00 WIN)
 ccuracy:  0.9091 (070 of 077)  (reward=+2000.00 WIN)
Accuracy:  0.9103 (071 of 078)  (reward=+2000.00 WIN)
Accuracy:  0.9114 (072 of 079)  (reward=+2000.00 WIN)
Accuracy:  0.9125 (073 of 080)  (reward=+2000.00 WIN)
Accuracy:  0.9136 (074 of 081)  (reward=+2000.00 WIN)
Accuracy:  0.9146 (075 of 082)  (reward=+2000.00 WIN)
Accuracy:  0.9157 (076 of 083)  (reward=+2000.00 WIN)
Accuracy:  0.9167 (077 of 084)  (reward=+2000.00 WIN)
Accuracy:  0.9176 (078 of 085)  (reward=+2000.00 WIN)
Accuracy:  0.9186 (079 of 086)  (reward=+2000.00 WIN)
Accuracy:  0.9195 (080 of 087)  (reward=+2000.00 WIN)
Accuracy:  0.9205 (081 of 088)  (reward=+2000.00 WIN)
Accuracy:  0.9213 (082 of 089)  (reward=+2000.00 WIN)
Accuracy:  0.9222 (083 of 090)  (reward=+2000.00 WIN)
Accuracy:  0.9231 (084 of 091)  (reward=+2000.00 WIN)
Accuracy:  0.9239 (085 of 092)  (reward=+2000.00 WIN)
Accuracy:  0.9247 (086 of 093)  (reward=+2000.00 WIN)
Accuracy:  0.9255 (087 of 094)  (reward=+2000.00 WIN)
Accuracy:  0.9263 (088 of 095)  (reward=+2000.00 WIN)
Accuracy:  0.9271 (089 of 096)  (reward=+2000.00 WIN)
Accuracy:  0.9278 (090 of 097)  (reward=+2000.00 WIN)
Accuracy:  0.9286 (091 of 098)  (reward=+2000.00 WIN)
Accuracy:  0.9293 (092 of 099)  (reward=+2000.00 WIN)
Accuracy:  0.9300 (093 of 100)  (reward=+2000.00 WIN)
```

https://youtu.be/YwkcPO7HAZE

Future Works

Velocity control

It would be great if I could successfully achieve the goal with velocity control. The movement itself is more natural than position control, that means it needs smaller torque and consumes less power. But it was difficult to achieve an accuracy of more than 70 percent.

Faster converge

If the first few episodes fail, it takes very long to converge. this could be resolved with manipulating parameters like **LTSM size** or **Interim reward**.