Report: Deep RL Arm Manipulation

May 20, 2018

Objectives

Objectives of the project was to creare DQN Agent and tune hyperparameters to achieve the two following tasks:

1. Any part of the robot arm should touch the object with atleast an accuracy of 90%.
2. Only the gripper base of the robot arm should touch the object with at least an accuracy of 80%.

Reward Functions

Both task are using the same reward function and params except the episode end condition.

Win and loss reward values was selected as following:

```
#define REWARD_WIN  20.0f
#define REWARD_LOSS -20.0f

#define REWARD_ALPHA 0.5f
```

`REWARD_ALPHA` is a smoothing parameter that determine the contribution of a running average delta distance to the goal `avgGoalDelta`. The value of `0.5` was determined experimentally and works great for both tasks.

```
// are we decreasing distance to the goal?
const float distDelta  = lastGoalDistance - distGoal;

// compute the smoothed moving average of the delta of the distance to the
avgGoalDelta  = avgGoalDelta * REWARD_ALPHA + distDelta * (1 - REWARD_ALPH

rewardHistory = 4 * avgGoalDelta - 0.2; // Task #2 WINNING! And works for '
```

Transitional reward consists of `avgGoalDelta` with a multiplier of `4` - selected experimentally to balance `REWARD_WIN` / `REWWARD_LOSS` values.

And the second term `-0.2` is a penalty for each move that prevents the agent to just hold the position and stop moving at all.

The same reward function and values are used for both tasks.

Joint control was selected as positional for both tasks. Also quick experiments show that velocity control works good as well. However the final params wasn’t tested with the velocity control and may require further tuning of reward function values and DQN parameters.

Hyperparameters

Hyperparameters for the DQN Agent was selected as following:

```
#define INPUT_WIDTH   64
#define INPUT_HEIGHT  64
#define OPTIMIZER "RMSprop"
#define LEARNING_RATE 0.01f // Task #1 = 0.01f, Task #2 = 0.1f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 32
#define USE_LSTM true
#define LSTM_SIZE 256
```

Above parameters was mainly selected during the tuning of Task #2, which is harder to learn for the agent, and tested again for Task #1 later. During such backward tests were determined that lower `LEARNING_RATE` of `0.01f` works better for Task #1. And higher `LEARNING_RATE` of `0.1f` is more important for Task #1.

During parameters tuning higher `LSTM_SIZE` showed better results thus it’s `256` this means that agent can learn more complex feature in it’s state and track them during the subsequent actions.

`BATCH_SIZE` higher than `32` showed worse results and wasn’t been able to learn past `56%` of accuracy for Task #2.

`Adam` optimizer showed worse results than `RMSProp`.

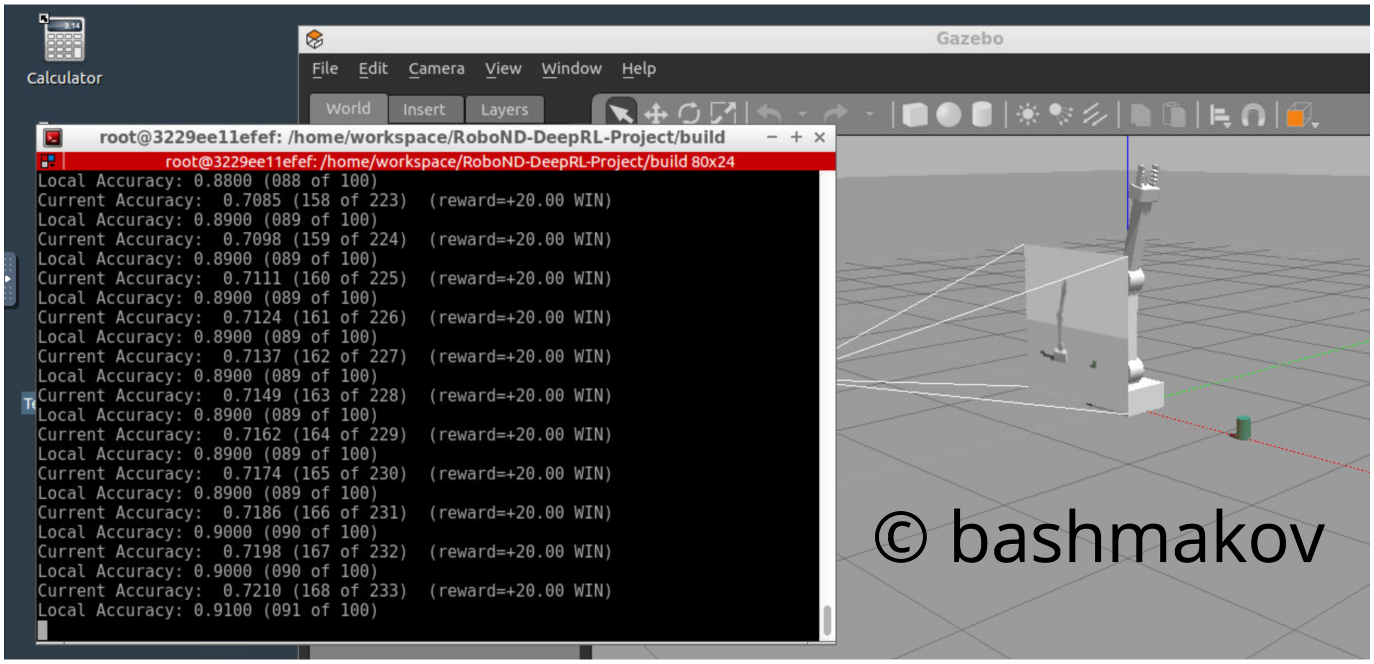
As a result we have one set of hyperparameters that works for both objectives with the only change in `LEARNING_RATE` between them.

Results

Training environment - Udacity’s Project Workspace with enabled GPU.

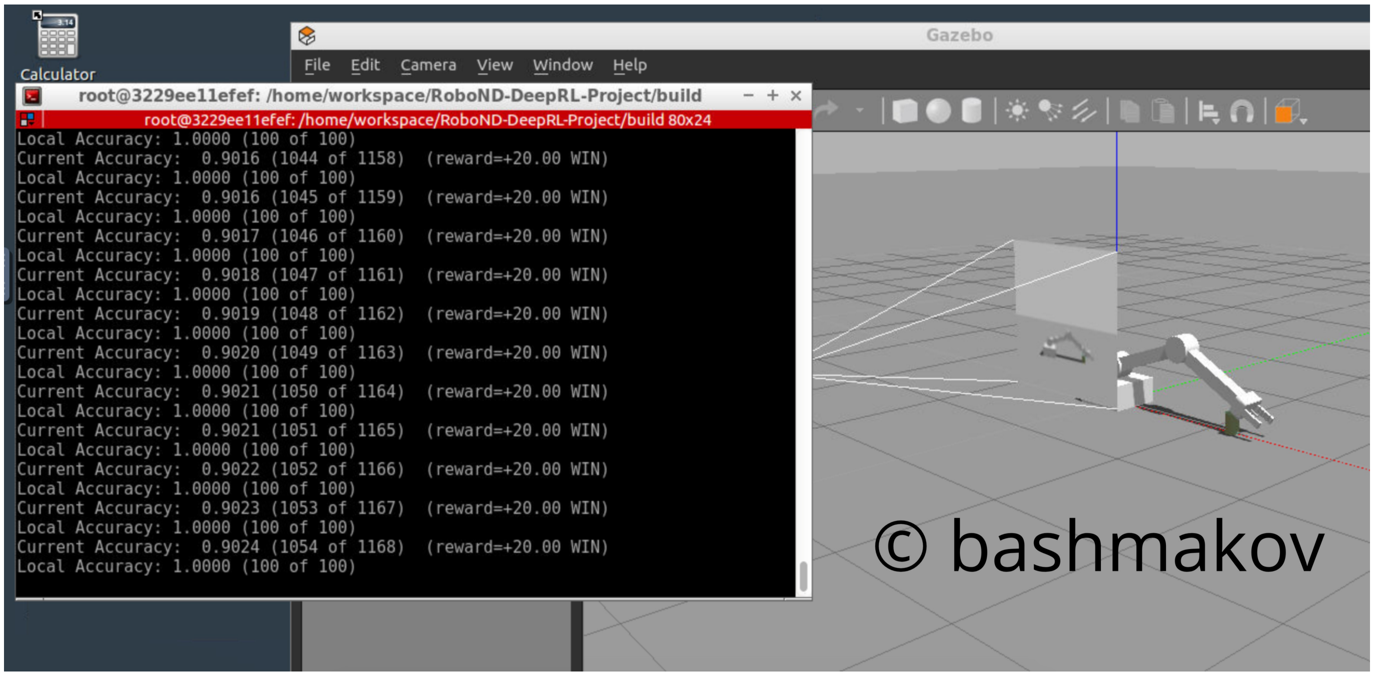
Task 1

Tests of the selected parameters on Task #1 showed good results around episode `~240` with the achieved accuracy `91%` for the last `100` episodes.



```
root@3229ee11efef: /home/workspace/RoboND-DeepRL-Project/build
root@3229ee11efef: /home/workspace/RoboND-DeepRL-Project/build 80x24
Local Accuracy: 0.8800 (088 of 100)
Current Accuracy:  0.7085 (158 of 223)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7098 (159 of 224)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7111 (160 of 225)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7124 (161 of 226)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7137 (162 of 227)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7149 (163 of 228)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7162 (164 of 229)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7174 (165 of 230)  (reward=+20.00 WIN)
Local Accuracy: 0.8900 (089 of 100)
Current Accuracy:  0.7186 (166 of 231)  (reward=+20.00 WIN)
Local Accuracy: 0.9000 (090 of 100)
Current Accuracy:  0.7198 (167 of 232)  (reward=+20.00 WIN)
Local Accuracy: 0.9000 (090 of 100)
Current Accuracy:  0.7210 (168 of 233)  (reward=+20.00 WIN)
Local Accuracy: 0.9100 (091 of 100)
```

Continue training for the next `1000` episodes showed the further improvements up to the accuracy `100%` of the last `100` episodes.



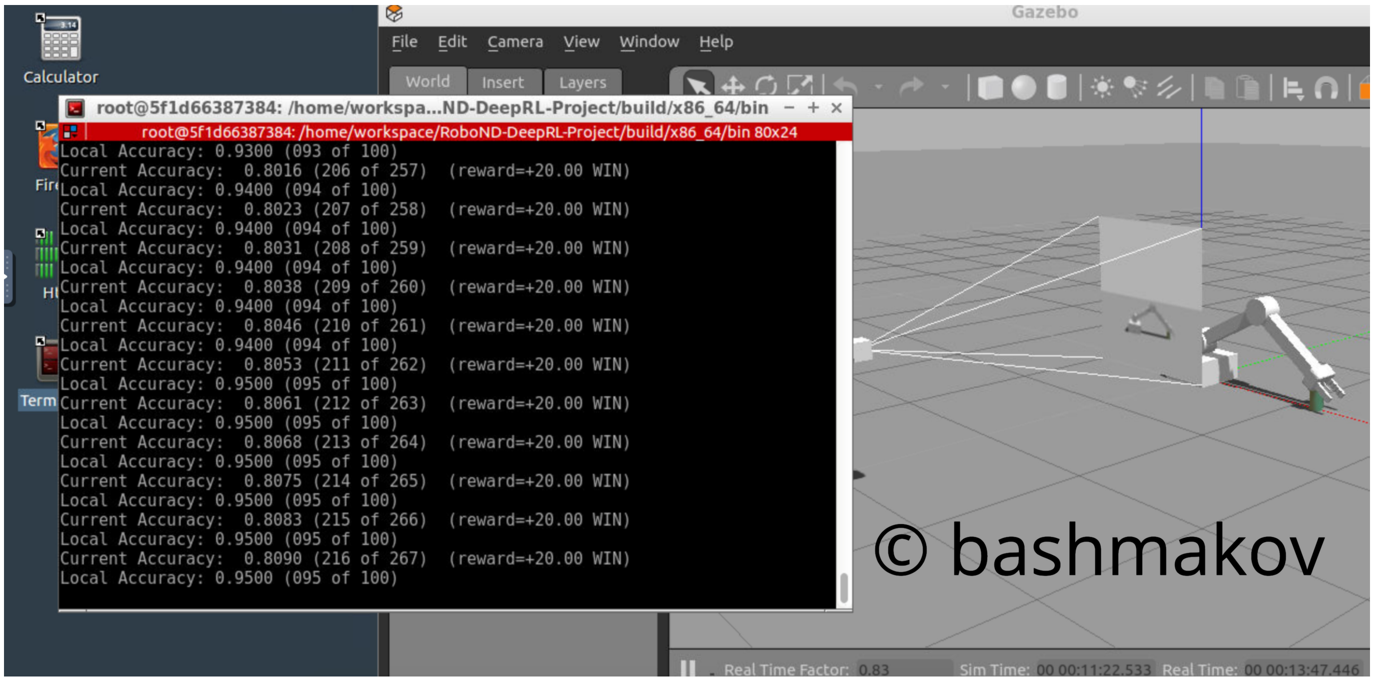
```
root@3229ee11efef: /home/workspace/RoboND-DeepRL-Project/build
root@3229ee11efef: /home/workspace/RoboND-DeepRL-Project/build 80x24
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9016 (1044 of 1158)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9016 (1045 of 1159)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9017 (1046 of 1160)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9018 (1047 of 1161)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9019 (1048 of 1162)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9020 (1049 of 1163)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9021 (1050 of 1164)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9021 (1051 of 1165)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9022 (1052 of 1166)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9023 (1053 of 1167)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
Current Accuracy:  0.9024 (1054 of 1168)  (reward=+20.00 WIN)
Local Accuracy: 1.0000 (100 of 100)
```

Video result for the Task #1 after `>1000` episodes.

Task 2

Second task was much harder to find the right balance of the hyperparameters but after `~20 hours` of tests on GPU it converges at start showing good enough results.

In just `267` episodes local accuracy of `95%` on the last `100` episodes were achieved for Task #2.



```
root@5f1d66387384: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Local Accuracy: 0.9300 (093 of 100)
Current Accuracy:  0.8016 (206 of 257)  (reward=+20.00 WIN)
Local Accuracy: 0.9400 (094 of 100)
Current Accuracy:  0.8023 (207 of 258)  (reward=+20.00 WIN)
Local Accuracy: 0.9400 (094 of 100)
Current Accuracy:  0.8031 (208 of 259)  (reward=+20.00 WIN)
Local Accuracy: 0.9400 (094 of 100)
Current Accuracy:  0.8038 (209 of 260)  (reward=+20.00 WIN)
Local Accuracy: 0.9400 (094 of 100)
Current Accuracy:  0.8046 (210 of 261)  (reward=+20.00 WIN)
Local Accuracy: 0.9400 (094 of 100)
Current Accuracy:  0.8053 (211 of 262)  (reward=+20.00 WIN)
Local Accuracy: 0.9500 (095 of 100)
Current Accuracy:  0.8061 (212 of 263)  (reward=+20.00 WIN)
Local Accuracy: 0.9500 (095 of 100)
Current Accuracy:  0.8068 (213 of 264)  (reward=+20.00 WIN)
Local Accuracy: 0.9500 (095 of 100)
Current Accuracy:  0.8075 (214 of 265)  (reward=+20.00 WIN)
Local Accuracy: 0.9500 (095 of 100)
Current Accuracy:  0.8083 (215 of 266)  (reward=+20.00 WIN)
Local Accuracy: 0.9500 (095 of 100)
Current Accuracy:  0.8090 (216 of 267)  (reward=+20.00 WIN)
Local Accuracy: 0.9500 (095 of 100)
```

Calculator
Gazebo
File Edit Camera View Window Help
World Insert Layers
root@5f1d66387384: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
Term
© bashmakov
Real Time Factor: 0.83 Sim Time: 00 00:11:22.533 Real Time: 00 00:13:47.446

Future Work

Training for Task #1 on the parameters tuned for the Task #2 is much longer than was achieved without Task #2 in mind. Interesting question is 'How we can achieve the best result in the shortest amount of episodes with the same parameters of DQN Agent and reward function for both tasks simultaneously?'

Perhaps joint training of the agent on both task simultaneously can help with it.

Other interesting task is to try the trained agent on different arm configurations (link length, joint shapes, etc) and see how transferable our agent between different arms.

Capsules Bot

Pavlo Bashmakov

bexcite

bashmakov

Exploring autonomous systems, robotics and mapping world around us.