

# DeepRL Project

Robert Aleck | [github.com/mnbf9rca](https://github.com/mnbf9rca) | 16<sup>th</sup> August 2019

---

## Introduction

A Deep Q-Learning Network was implemented against a framework for a robotic arm with 3 DOF simulated within the Gazebo environment. The author had to:

- Subscribe to the relevant topics (camera imagery, collisions)
- Instantiate a DQN Agent
- Choose between velocity or position based arm joint manipulation, and implement in code
- Design an appropriate reward function, covering successful (collision between defined objects) and unsuccessful actions, including episode terminal states and interim rewards
- Tune hyperparameters and reward function for two scenarios:
  - At least 90% collision accuracy between any part of the arm and the collision object over at least 100 episodes, and
  - At least 80% accuracy in collisions between the gripper base of the robot arm and the collision object over at least 100 episodes

## Reward functions

The robot arm control was selected as “position”, using default values.

At first, a large (>100 or even >500) REWARD\_WIN and REWARD\_LOSS value was used, with smaller/fractional rewards issued for interim states, however, the model failed to train effectively. Smaller values were then chosen, with additional multipliers providing greater weighting to episode-terminal (collision) events, as follows:

```
#define REWARD_WIN 0.15f
#define REWARD_LOSS -0.15f
#define REWARD_COLLISION_GROUND 10           // multiplier - hit the ground
#define REWARD_COLLISION_CORRECT_PART 20      // multiplier - hit the correct item
#define REWARD_COLLISION_WRONG_PART 10        // multiplier - hit the wrong item
#define MIN_DISTANCE_TO_MOVE_WITHOUT_PENALTY 0.05f // how far must the gripper have moved compared to last frame
to avoid a penalty
```

The reward function was unchanged when training to hit the gripper base. The selection of collision target is changed with the REWARD\_ANY\_COLLISION parameter – when set to false, only collisions between COLLISION\_ITEM and COLLISION\_POINT were rewarded, otherwise all collisions with COLLISION\_ITEM resulted in a reward.

```
#define REWARD_ANY_COLLISION false           // reward for hitting any part of the arm on the tube
```

The following rewards were issued for episode-terminal states:

Ground contact	REWARD_LOSS * REWARD_COLLISION_GROUND
Collision between COLLISION_ITEM and either any other part, or with COLLISION_POINT	REWARD_WIN * REWARD_COLLISION_CORRECT_PART
Collision between COLLISION_ITEM and any part other than COLLISION_POINT	REWARD_LOSS * REWARD_COLLISION_WRONG_PART

To encourage the arm to move towards the target, interim rewards were issued. A positive movement towards the goal received  $\text{REWARD\_WIN} * \text{timePenalty}$  (where  $\text{timePenalty}$  is a value trending from 1 to 0 proportional to the number of frames completed), whereas movement away received  $\text{REWARD\_LOSS}$ . Additionally, as it was noticed that the arm spent a long time in a relatively stable position,  $\text{REWARD\_LOSS}$  is added to  $\text{rewardHistory}$  at each frame that the gripper failed to move, in a smoothed, moving average, at least  $\text{MIN\_DISTANCE\_TO\_MOVE\_WITHOUT\_PENALTY}$  since the previous frame.

## Hyperparameters

The following hyperparameters were chosen and used for both collision models:

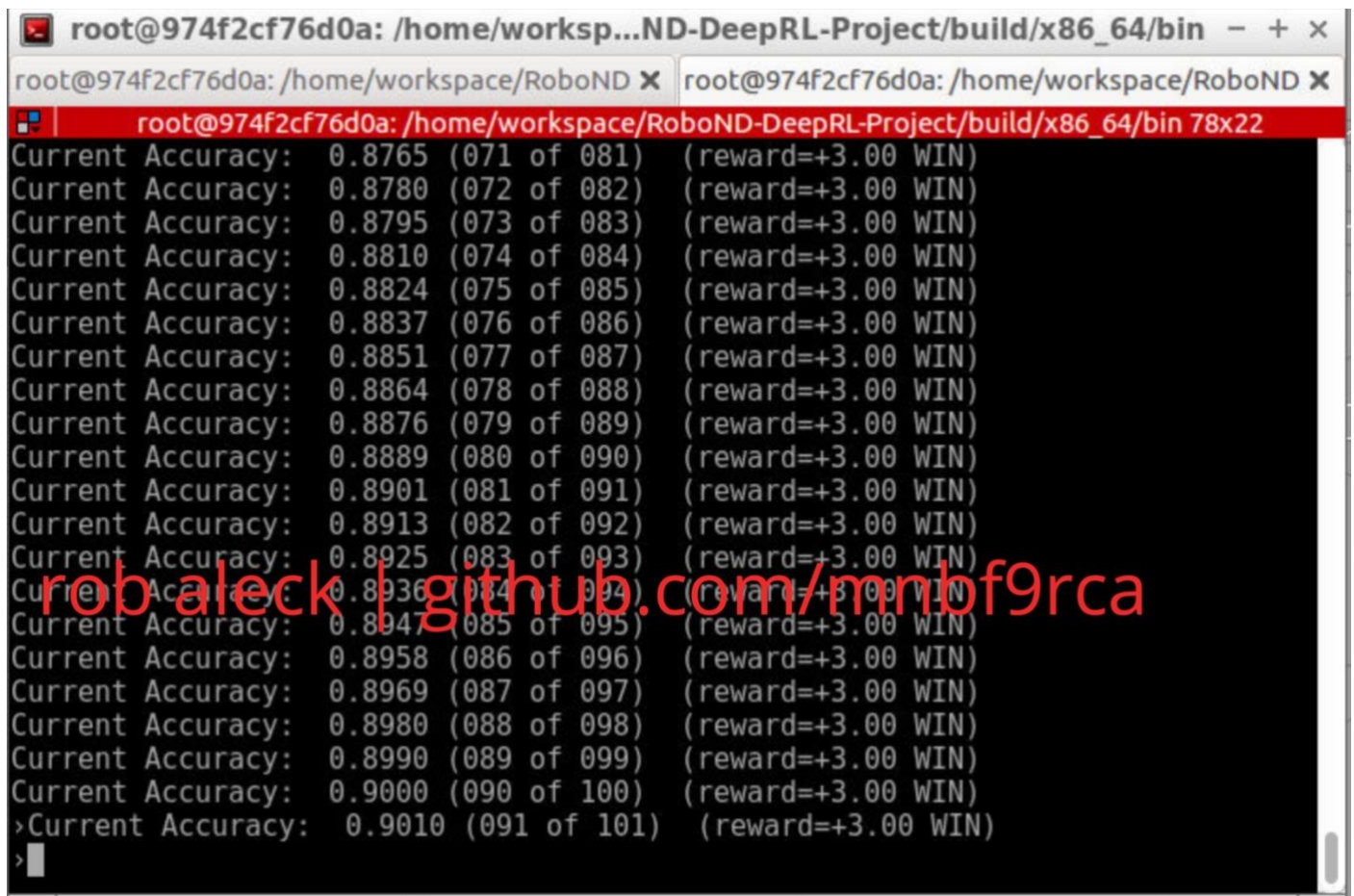
```
#define INPUT_WIDTH 64
#define INPUT_HEIGHT 64
#define OPTIMIZER "Adam"
#define LEARNING_RATE 0.1f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 512
#define USE_LSTM true
#define LSTM_SIZE 256
```

The input image size was observed to be 64x64, so this was selected for the model. Adam was selected for the optimizer based on previous experience with reinforcement learning, where the author has found it to converge quicker for relatively simple models. The learning rate was initially 0.01, but was increased to 0.1 to speed up learning – no evidence of overfitting was found. Replay memory was left as default. Batch size was increased to take advantage of the large memory available on the server, and the LSTM size increased experimentally – higher than 256 failed to fit, lower than 128 slowed learning

## Results

In general, both objectives were met relatively quickly. If the arm initially trained away from the collision item, learning took longer, but it almost always achieved at least one [WIN](#) within the first 10 episodes. In some cases, after apparently operating correctly for 20-30 episodes, the arm would “overstretch” and hit the floor – this can be seen in one of the examples below at episode 82.

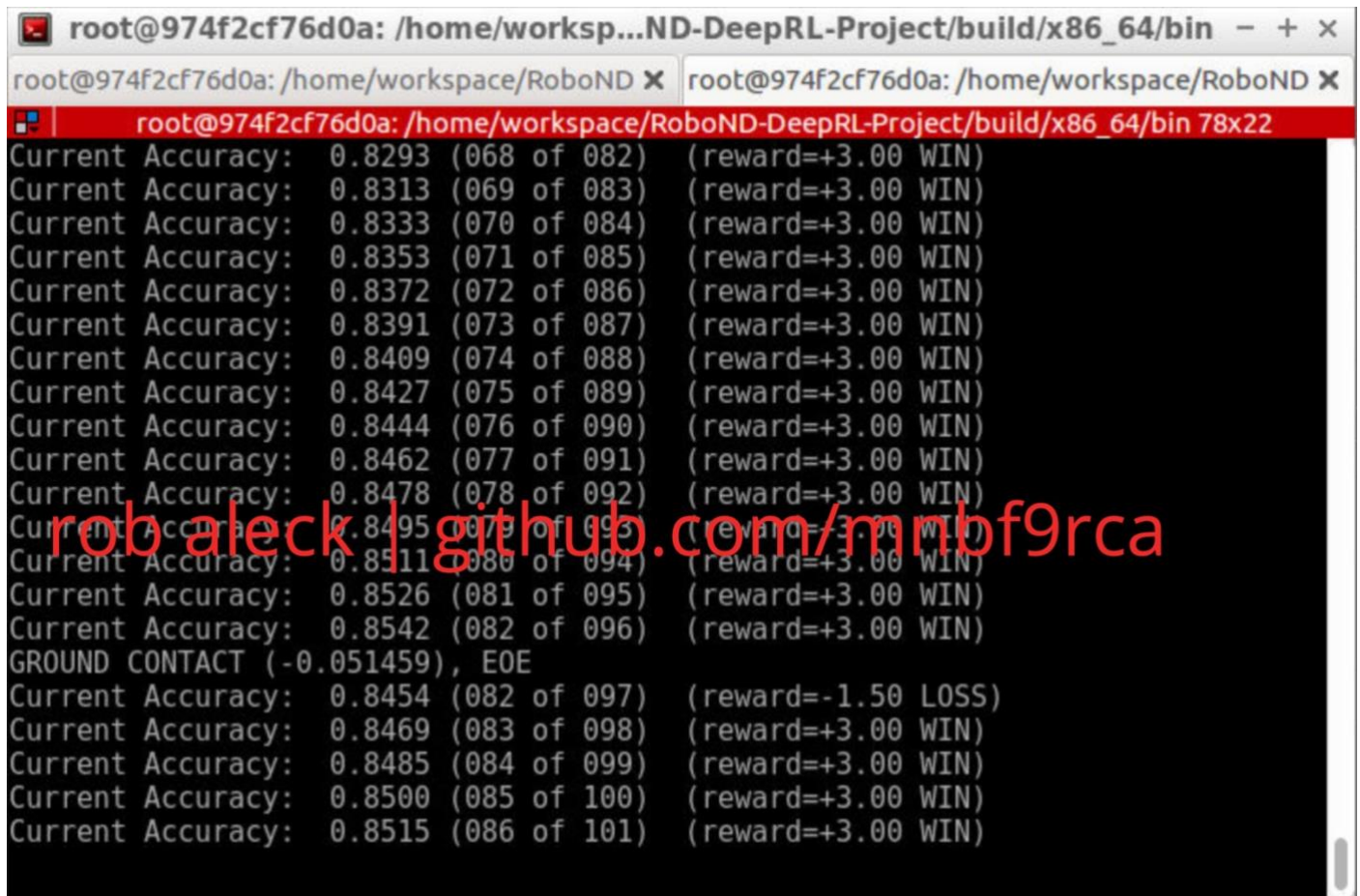
Have any part of the robot arm touch the object of interest, with at least a 90% accuracy for a minimum of 100 runs.



```
root@974f2cf76d0a: /home/worksp...ND-DeepRL-Project/build/x86_64/bin - + x
root@974f2cf76d0a: /home/workspace/RoboND x root@974f2cf76d0a: /home/workspace/RoboND x
root@974f2cf76d0a: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 78x22
Current Accuracy: 0.8765 (071 of 081) (reward=+3.00 WIN)
Current Accuracy: 0.8780 (072 of 082) (reward=+3.00 WIN)
Current Accuracy: 0.8795 (073 of 083) (reward=+3.00 WIN)
Current Accuracy: 0.8810 (074 of 084) (reward=+3.00 WIN)
Current Accuracy: 0.8824 (075 of 085) (reward=+3.00 WIN)
Current Accuracy: 0.8837 (076 of 086) (reward=+3.00 WIN)
Current Accuracy: 0.8851 (077 of 087) (reward=+3.00 WIN)
Current Accuracy: 0.8864 (078 of 088) (reward=+3.00 WIN)
Current Accuracy: 0.8876 (079 of 089) (reward=+3.00 WIN)
Current Accuracy: 0.8889 (080 of 090) (reward=+3.00 WIN)
Current Accuracy: 0.8901 (081 of 091) (reward=+3.00 WIN)
Current Accuracy: 0.8913 (082 of 092) (reward=+3.00 WIN)
Current Accuracy: 0.8925 (083 of 093) (reward=+3.00 WIN)
Current Accuracy: 0.8936 (084 of 094) (reward=+3.00 WIN)
Current Accuracy: 0.8947 (085 of 095) (reward=+3.00 WIN)
Current Accuracy: 0.8958 (086 of 096) (reward=+3.00 WIN)
Current Accuracy: 0.8969 (087 of 097) (reward=+3.00 WIN)
Current Accuracy: 0.8980 (088 of 098) (reward=+3.00 WIN)
Current Accuracy: 0.8990 (089 of 099) (reward=+3.00 WIN)
Current Accuracy: 0.9000 (090 of 100) (reward=+3.00 WIN)
>Current Accuracy: 0.9010 (091 of 101) (reward=+3.00 WIN)
>
```

rob aleck | [github.com/mnbf9rca](https://github.com/mnbf9rca)

Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy for a minimum of 100 runs.



The image shows a terminal window with a red title bar. The terminal displays a series of log entries for a robot gripper task. Each entry shows the current accuracy, the number of runs completed out of a total, and the reward for that run. The accuracy starts at 0.8293 and increases to 0.8515 over 101 runs. Most runs result in a reward of +3.00 (WIN), but run 082 results in a reward of -1.50 (LOSS) due to a ground contact. A large red watermark is overlaid on the terminal text.

```
root@974f2cf76d0a: /home/worksp...ND-DeepRL-Project/build/x86_64/bin - + x
root@974f2cf76d0a: /home/workspace/RoboND x root@974f2cf76d0a: /home/workspace/RoboND x
root@974f2cf76d0a: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 78x22
Current Accuracy: 0.8293 (068 of 082) (reward=+3.00 WIN)
Current Accuracy: 0.8313 (069 of 083) (reward=+3.00 WIN)
Current Accuracy: 0.8333 (070 of 084) (reward=+3.00 WIN)
Current Accuracy: 0.8353 (071 of 085) (reward=+3.00 WIN)
Current Accuracy: 0.8372 (072 of 086) (reward=+3.00 WIN)
Current Accuracy: 0.8391 (073 of 087) (reward=+3.00 WIN)
Current Accuracy: 0.8409 (074 of 088) (reward=+3.00 WIN)
Current Accuracy: 0.8427 (075 of 089) (reward=+3.00 WIN)
Current Accuracy: 0.8444 (076 of 090) (reward=+3.00 WIN)
Current Accuracy: 0.8462 (077 of 091) (reward=+3.00 WIN)
Current Accuracy: 0.8478 (078 of 092) (reward=+3.00 WIN)
Current Accuracy: 0.8495 (079 of 093) (reward=+3.00 WIN)
Current Accuracy: 0.8511 (080 of 094) (reward=+3.00 WIN)
Current Accuracy: 0.8526 (081 of 095) (reward=+3.00 WIN)
Current Accuracy: 0.8542 (082 of 096) (reward=+3.00 WIN)
GROUND CONTACT (-0.051459), EOE
Current Accuracy: 0.8454 (082 of 097) (reward=-1.50 LOSS)
Current Accuracy: 0.8469 (083 of 098) (reward=+3.00 WIN)
Current Accuracy: 0.8485 (084 of 099) (reward=+3.00 WIN)
Current Accuracy: 0.8500 (085 of 100) (reward=+3.00 WIN)
Current Accuracy: 0.8515 (086 of 101) (reward=+3.00 WIN)
```

## Future work

Future work could include:

- Training the model to locate the object between the gripper arms, close the gripper when in place, and move the object
- Randomising the location of the object
- The LSTM has not been optimised (using only defaults the author has used successfully in the past).
- Adjusting other hyperparameters, such as the `DISTANCE_DECAY_FACTOR`, which was left at 0.90 throughout.