

# Deep RL Arm Manipulation

## Introduction

RL emulates the process that human learns a skill - making an action and adjust next action based on feedback/reward from the world. The goal of this project is to teach robotic arm to touch an object (cylinder) using DQN.

``

Two objectives are specified for this project.

- 1. Robot arm touches the object without the gripper touching the ground at greater than 90% accuracy for at least 100 attempts.
- 2. Only the gripper is allow to touch the object without touching the ground with at least 80% accuracy for at least 100 attempts.

## Reward Functions

ArmPlugin.cpp contains the rewards and penalties used to train the Deep RL Arm DQN agent.

More specifically, a reward of +25 is given if:

- robot arm touches the cylinder in objective 1
- gripper touches the cylinder in objective 2

A penalty of -25 is given if:

- robot touched the ground before completing its objective
- robot is unable to complete the objective within 100 frames

An interim reward is given based on the position of the robot arm while moving, where the Smoothed Moving Average (SMMA) of the delta of the distance to the goal is used.

As in defined in ArmPlugin.cpp:

- avgGoalDelta is the resulting reward function to determine if robot is moving closer to the object
- distDelta is (last distance to the goal - current distance to the goal)
- ALPHA is a constant between 0 and 1 to allow the most current delta to have a heavier weight than previous delta's.

The SMMA eventually helps reducing fluctuations and also assists the agent to maximize its rewards. The agent rewards are based on distance and position of the robot/gripper. Joint control is used to drive the robot arm.

An overview of all parameters is given below.

```
// Define DQN API Settings
#define INPUT_CHANNELS 3
#define ALLOW_RANDOM true
#define DEBUG_DQN false
#define GAMMA 0.9f
#define EPS_START 0.7f
#define EPS_END 0.02f
#define EPS_DECAY 200

/*
    Tune the following hyperparameters
*/
#define INPUT_WIDTH 64
#define INPUT_HEIGHT 64
#define OPTIMIZER "Adam"
#define LEARNING_RATE 0.1f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 256
#define USE_LSTM true
#define LSTM_SIZE 256

/*
    Define Reward Parameters
*/
#define REWARD_WIN 25.0f
#define REWARD_LOSS -25.0f
#define INTERIM_REWARD 4.0f
#define INTERIM_OFFSET 0.3f
#define ALPHA 0.4f
```

## Hyperparameters

Careful selection of hyperparameters are the key to train a good DQN agent in terms of speed and accuracy in achieving the objectives. Hyperparameter tuning is more or less a trial and error process.

I first reduced the input size to 64x64 to speed up the processing. As mentioned in the course, a square image accelerates matrix operations and makes it run faster on GPU.

Adam optimizer run faster than RMSprop in converging hence was selected after a few tests. I also had good experience with Adam optimizer in other ML applications.

I then tested learning rate between 0.01 and 0.1 and 0.1 seemed good for both objectives.

Accordingly, replay\_memory was set to 10000 to hold sufficient memory to help training.

LSTM with a LSTM\_SIZE of 256 was used to keep track of both long and short term memory. It allowed the agent to use both past and present camera frames in training.

A screenshot of hyperparameters is given below.

```
ArmPlugin::ArmPlugin()
ArmPlugin::Load('arm')
PropPlugin::Load('tube')
[deepRL] use_cuda: True
[deepRL] use_lstm: 1
[deepRL] lstm_size: 256
[deepRL] input_width: 64
[deepRL] input_height: 64
[deepRL] input_channels: 3
[deepRL] num_actions: 6
[deepRL] optimizer: Adam
[deepRL] learning rate: 0.1
[deepRL] replay_memory: 10000
[deepRL] batch_size: 256
[deepRL] gamma: 0.9
[deepRL] epsilon_start: 0.7
[deepRL] epsilon_end: 0.02
[deepRL] epsilon_decay: 200.0
[deepRL] allow_random: 1
[deepRL] debug_mode: 0
[deepRL] creating DQN model instance
[deepRL] DRQN::__init__()
[deepRL] LSTM (hx, cx) size = 256
[deepRL] DQN model instance created
[deepRL] DQN script done init
```

## Results

Objective 1 was achieved with at least 90% accuracy for a minimum of 100 runs with robot not touching the ground.

Objective 2 was achieved with the gripper touching the object without touching the ground at  $\geq 80\%$  accuracy for a minimum of 100 runs.

```
root@69dfbab60fb8: /home/workspace/RoboND-DeepRL-
Current Accuracy: 0.6952 (130 of 187) (reward=+25.00 WIN)
Current Accuracy: 0.6968 (131 of 188) (reward=+25.00 WIN)
Current Accuracy: 0.6931 (131 of 189) (reward=-25.00 LOSS)
Current Accuracy: 0.6947 (132 of 190) (reward=+25.00 WIN)
Current Accuracy: 0.6963 (133 of 191) (reward=+25.00 WIN)
Current Accuracy: 0.6979 (134 of 192) (reward=+25.00 WIN)
Current Accuracy: 0.6995 (135 of 193) (reward=+25.00 WIN)
Current Accuracy: 0.6959 (135 of 194) (reward=-25.00 LOSS)
Current Accuracy: 0.6974 (136 of 195) (reward=+25.00 WIN)
Current Accuracy: 0.6990 (137 of 196) (reward=+25.00 WIN)
Current Accuracy: 0.7005 (138 of 197) (reward=+25.00 WIN)
Current Accuracy: 0.7020 (139 of 198) (reward=+25.00 WIN)
Current Accuracy: 0.7035 (140 of 199) (reward=+25.00 WIN)
Current Accuracy: 0.7050 (141 of 200) (reward=+25.00 WIN)
Current Accuracy: 0.7065 (142 of 201) (reward=+25.00 WIN)
Current Accuracy: 0.7079 (143 of 202) (reward=+25.00 WIN)
Current Accuracy: 0.7094 (144 of 203) (reward=+25.00 WIN)
Current Accuracy: 0.7108 (145 of 204) (reward=+25.00 WIN)
Current Accuracy: 0.7073 (145 of 205) (reward=-25.00 LOSS)
Current Accuracy: 0.7087 (146 of 206) (reward=+25.00 WIN)
Current Accuracy: 0.7101 (147 of 207) (reward=+25.00 WIN)
Current Accuracy: 0.7115 (148 of 208) (reward=+25.00 WIN)
Current Accuracy: 0.7081 (148 of 209) (reward=-25.00 LOSS)
Current Accuracy: 0.7095 (149 of 210) (reward=+25.00 WIN)
Current Accuracy: 0.7109 (150 of 211) (reward=+25.00 WIN)
Current Accuracy: 0.7075 (150 of 212) (reward=-25.00 LOSS)
Current Accuracy: 0.7042 (150 of 213) (reward=-25.00 LOSS)
Current Accuracy: 0.7056 (151 of 214) (reward=+25.00 WIN)
Current Accuracy: 0.7070 (152 of 215) (reward=+25.00 WIN)
Current Accuracy: 0.7037 (152 of 216) (reward=-25.00 LOSS)
Current Accuracy: 0.7051 (153 of 217) (reward=+25.00 WIN)
Current Accuracy: 0.7064 (154 of 218) (reward=+25.00 WIN)
Current Accuracy: 0.7078 (155 of 219) (reward=+25.00 WIN)
Current Accuracy: 0.7091 (156 of 220) (reward=+25.00 WIN)
Current Accuracy: 0.7104 (157 of 221) (reward=+25.00 WIN)
Current Accuracy: 0.7117 (158 of 222) (reward=+25.00 WIN)
Current Accuracy: 0.7130 (159 of 223) (reward=+25.00 WIN)
Current Accuracy: 0.7143 (160 of 224) (reward=+25.00 WIN)
Current Accuracy: 0.7156 (161 of 225) (reward=+25.00 WIN)
Current Accuracy: 0.7168 (162 of 226) (reward=+25.00 WIN)
Current Accuracy: 0.7181 (163 of 227) (reward=+25.00 WIN)
Current Accuracy: 0.7193 (164 of 228) (reward=+25.00 WIN)
Current Accuracy: 0.7205 (165 of 229) (reward=+25.00 WIN)
Current Accuracy: 0.7217 (166 of 230) (reward=+25.00 WIN)
Current Accuracy: 0.7229 (167 of 231) (reward=+25.00 WIN)
Current Accuracy: 0.7241 (168 of 232) (reward=+25.00 WIN)
Current Accuracy: 0.7253 (169 of 233) (reward=+25.00 WIN)
Current Accuracy: 0.7265 (170 of 234) (reward=+25.00 WIN)
Current Accuracy: 0.7277 (171 of 235) (reward=+25.00 WIN)
Current Accuracy: 0.7288 (172 of 236) (reward=+25.00 WIN)
Current Accuracy: 0.7300 (173 of 237) (reward=+25.00 WIN)
Current Accuracy: 0.7311 (174 of 238) (reward=+25.00 WIN)
Current Accuracy: 0.7322 (175 of 239) (reward=+25.00 WIN)
Current Accuracy: 0.7333 (176 of 240) (reward=+25.00 WIN)
```



```
Current Accuracy: 0.7344 (177 of 241) (reward=+25.00 WIN)
Current Accuracy: 0.7355 (178 of 242) (reward=+25.00 WIN)
Current Accuracy: 0.7366 (179 of 243) (reward=+25.00 WIN)
Current Accuracy: 0.7336 (179 of 244) (reward=-25.00 LOSS)
Current Accuracy: 0.7347 (180 of 245) (reward=+25.00 WIN)
Current Accuracy: 0.7358 (181 of 246) (reward=+25.00 WIN)
Current Accuracy: 0.7368 (182 of 247) (reward=+25.00 WIN)
Current Accuracy: 0.7379 (183 of 248) (reward=+25.00 WIN)
Current Accuracy: 0.7390 (184 of 249) (reward=+25.00 WIN)
Current Accuracy: 0.7360 (184 of 250) (reward=-25.00 LOSS)
Current Accuracy: 0.7331 (184 of 251) (reward=-25.00 LOSS)
Current Accuracy: 0.7341 (185 of 252) (reward=+25.00 WIN)
Current Accuracy: 0.7352 (186 of 253) (reward=+25.00 WIN)
```

## Conclusions and Future Work

The project is quite interesting and useful. Eventhough it is a greatly simplified problem (e.g. 2D as opposed to 3D, touching as opposed to picking up), the experience I acquired throughout the learning process was very valuable. If given more time, I would like do more hyperparameter tuning to achieve even better accuracy. I may also try a different implementation with Keras/Tensorflow, or even make the problem harder, e.g. touching two objects one after another.