Deep Reinforcement Learning

Forrest Edwards

August 7, 2018

1 Description

The work presented here is an investigation of the use of deep Q-Learning with a recurrent neural network(RNN). Since a physical test environment is not readily accessible a virtual environment is leveraged consisting of a 3DOF robot built in the Gazebo physics simulator. A screenshot of this environment is shown in Figure 1. To simplify the training task, the robot was limited to only 2 DOF and the base joint was locked in the position indicated. The input into the RNN consists of

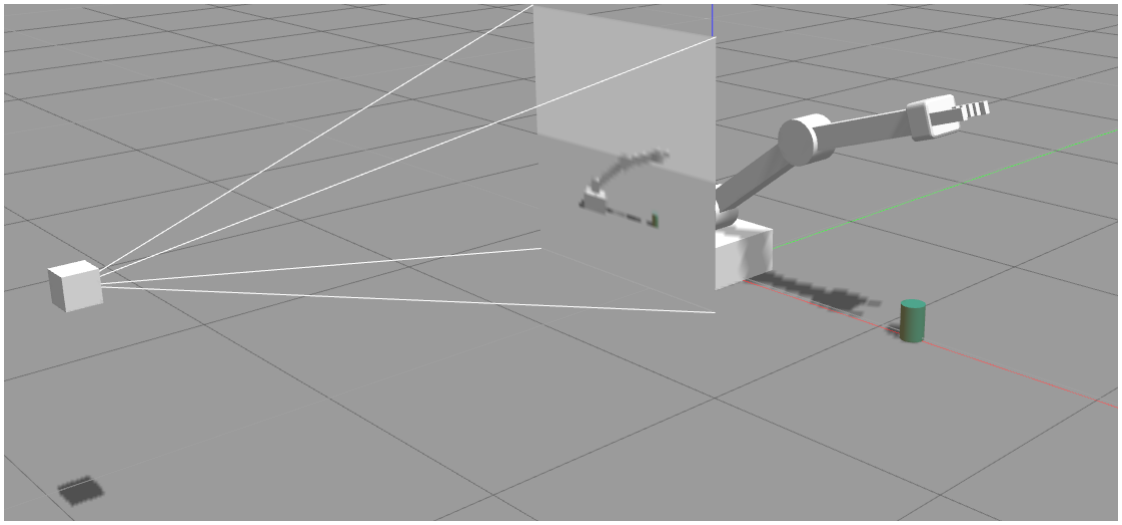

Figure 1: Gazebo physics environment showing robot, camera, and target tube.

a 64x64x3 virtual camera positioned to observe the robot, and its target, a cylindrical tube. To complete the work successfully, two goals must be achieved and are listed below:

1. **Part 1 Goal:** Train the RNN to direct the arm such that it touches the tube without colliding with the ground at a rate of 90% on any 100 consecutive attempts.
2. **Part 2 Goal:** Train the RNN to successfully touch the tube with only the gripper base of the robot's end effector at a rate of 80% on any 100 consecutive attempts.

2 Reward Functions

The architecture of reward structure implemented to address the second challenge (touching the gripper base to the tube) is shown in 3. The algorithm was rewarded or penalized based on the following events: movement, collision, and exceeding the max time limit.

For the first challenge, the only modification to the reward structure is the in the colliding section such that a reward is assigned when any portion of the arm impacts the gripper. The changed collision reward structure is shown in Figure 4

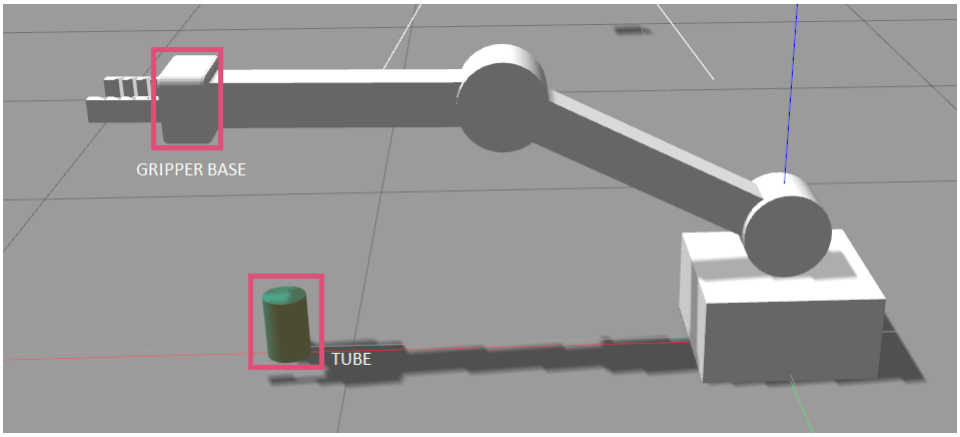

Figure 2: For goal 2, only the gripper (indicated) is allowed to touch the tube to qualify as a successful attempt.

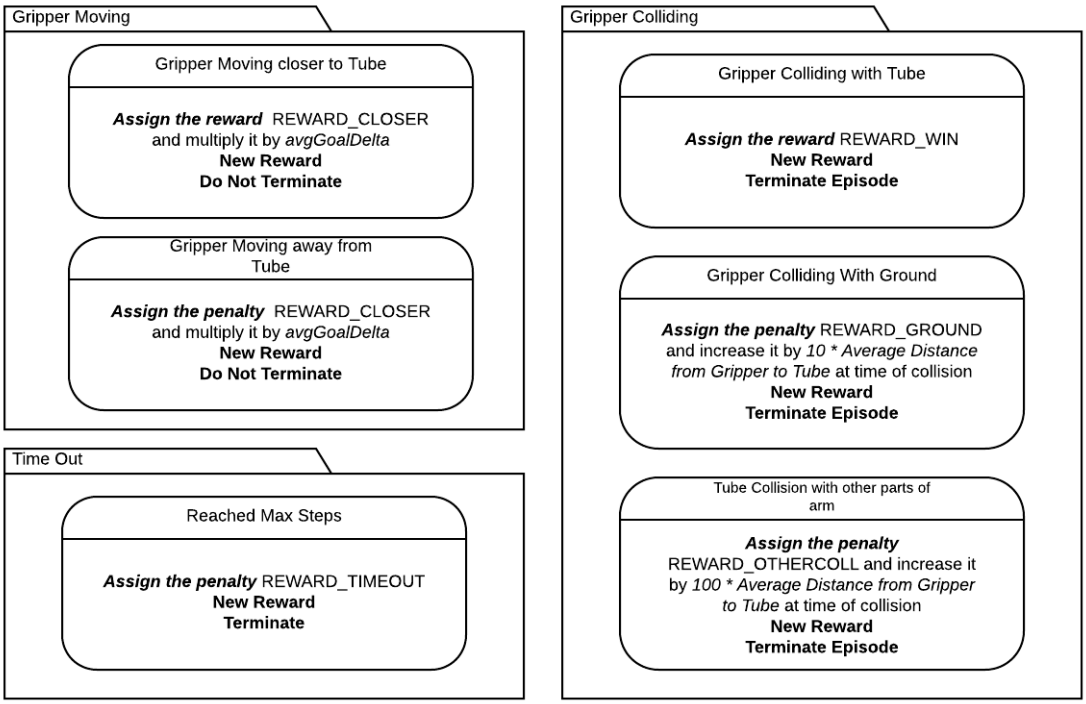

Figure 3: Overview of robot reward system for part 2.

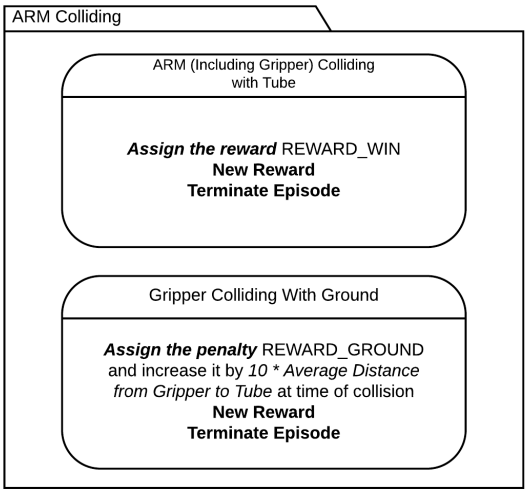

Figure 4: Relaxation of reward structure to accomplish part 1 goal.

2.1 Movements

Several different approaches to rewarding movement were attempted, including a logarithmic approach that rewarded the algorithm at higher rates as it moved closer to the tube target. However, the logarithmic reward function did not produce good results. It was found that this approach, as well as any reward function that was inversely related to the distance to the tube, would cause the algorithm to converge on solutions where the robot would oscillate in place. These solutions had many local minima where the gradient encouraged the robot to exploit the intermediate reward as one step forward and one step back resulted in a net positive reward.

Ultimately, the best results observed (for both part 1 and part 2) were achieved using the moving average of the change in distance from the gripper to the tube. By using this distance delta between images, a single reward could address both cases of retreating or advancing on the target since retreating movements result in negative delta values. By taking the moving average of the delta, $\overline{\Delta}_{gt}$ oscillations and step changes in rewards were smoothed prior to being fed into DQN agent.

$$R_{move} = r_c * \overline{\Delta}_{gt}$$

A weighting factor r_c is used as shown so that movement rewards are scaled properly in relation to the overall reward strategy,

2.2 Timeouts

To handle the case where, for example, the network converges on local exploitation, or is taking too long to win or lose through collisions, a penalty R_{to} for reaching the maximum number of frames is assigned. This penalty was configured to be significantly larger than cumulative interim rewards, but less than the penalty for ground collision.

$$R_{to} = r_{to}$$

2.3 Collisions

Only three collision cases were considered. A fourth case, robot collision with itself, was not possible in the environment due to constraints placed on the rotation limits of the joints. The three cases are shown in Figure 3. Slightly different approaches were taken to collisions between part 1 and part 2, and are noted below.

Gripper base colliding with tube represents the sole win condition for the part 2 goal and results in a reward R_{win} being issued. This reward was set to be significantly higher than cumulative interim rewards issued. No additional adjustments were made to account for distance, or frames

elapsed. Note that there is a hidden elapsed time reward in the interim movement reward for higher positive values of $\overline{\Delta}_{gt}$.

$$R_{win} = r_w$$

For the case where the gripper was detected colliding with the ground a a penalty was issued, R_{gnd}. Both part 1 and part 2 challenges shared identical approaches to ground collisions. The ground penalty was scaled similarly to R_w so that it would be significantly greater than any interim rewards received. In addition to the constant penalty r_g, the average distance from gripper to tube, $\overline{d}_{gt}$ scaled by a factor of 10 was added to the penalty. This made ground collisions near the tube preferable to the agent.

$$R_{gnd} = r_g + 10 * \overline{d}_{gt}$$

For the part 1 challenge, collisions between the arm and the tube were assigned the reward $R_{oc} = r_w$. However, for the part 2 goal, collisions between the tube and any arm component except the gripper base were assigned a penalty.. A constant penalty r_{oc} was selected to be less than either r_g or r_{to} to make arm collisions preferable to either time out events or ground collisions. As it turns out, it was critical to also assess an additional penalty, as was done for ground collisions based off of $\overline{d}_{gt}$ at the time of collision. Without this additional penalty the agent did not show consistent progress in moving the collision point (typically the grippers or arm link) toward the desired point on the gripper base. Since the distances involved in these collisions are much smaller than those in ground collisions, a scaling factor of 100 was used to make them comparable to interim rewards.

$$R_{oc} = r_{oc} + 100 * \overline{d}_{gt}$$

3 Hyperparameters

With the exception of the dimensions for the input layer of the DQN which should match the dimensions of the image feed, the tuning of the remaining hyperparameters required significant trial and error to optimize performance. The random initialization of the DNN weights and biases, as well as the epsilon driven random sampling that occurs to expand exploration the solution space guarantees that no two models run the same, even with identical parameters. As such, small variations in model performance with hyperparameter changes are not significant and drawing conclusions would be reckless. That being said, for some hyperparameters, adjustments resulted in wide swings in performance. In Table 1 a list of the hyperparameters, their purpose, and observations are listed.

The bulk of analysis and tuning was performed while attempting to solve the part 2 goal, touching the gripper to the tube. As a result, a separate set of parameters was not developed (or needed) for part 1, since that is a less demanding task. The only change between the two was a change to the reward parameters discussed previously.

4 Results

Results were obtained in 2 stages. In the first stage, 14 separate runs of 200 episodes each were conducted, with each run varying one of 6 parameters. Based on those results, the parameters listed in Table 1 were selected. Following the first stage, three additional pairs of runs were made, each pair using the same parameters to observe the magnitude of stochastic effects on the outcome. Each of these was allowed to run for 500 episodes. The 3 sets of parameters chosen to vary for this last set were:

1. **Runs 1a & 1b**: LSTM: false
2. **Runs 2a & 2b**: LSTM true
3. **Runs 3a & 3B**: LSTM true, REWARD_CLOSER = 1 (vs. 0.75)

As can be seen in Figure 5 there were multiple centuries (10 in total) that met the minimum 80% goal. Additionally, Run 2A met the goal for the entire run, inclusive of losses early on in the training. Lastly, 5 centuries were able to break the 90% threshold.

Results for part 1 of the project were trivial following completion of goal 2. The parameters were sufficiently tuned, to perform well with no other changes other than to the reward architecture.

Table 1: Summary of hyperparameters and observations

Hyperparameter	Purpose	Default Value	Tuned Value	Comment
INPUT_WIDTH INPUT_HEIGHT	Size of Input filter for RNN, should match image dims	512 512	64 64	None, straightforward
OPTIMIZER	Sets optimization algorithm used to minimize RNN error	None	Adam	Adam integrates advantages of AdaGrad and RMSProp
LEARNING_RATE	Sets rate at which optimizer descends error gradient	0.0	0.1	Higher Learning rates returned clear improvements
REPLAY_MEMORY	Sets size of replay buffer	10000	1000	High values resulted in poor performance
BATCH_SIZE	Sets number of tensors passed through the RNN at a time	0.0	0.1	Allows previous experiences to be recalled randomly and avoid excessive consecutivbe reinforcement
USE_LSTM	Enables LSTM: Long Term Short Term Memory	false	true	Significant improvements seen with LSTM enabled
LSTM_SIZE	LTSM Layer depth	32	256	Sharp reduction in performance above 256

Graphical results of training progress are shown in Figure 6. Overall Win % for the goal 1 500 episode run was 97.2%.

Clearly stochastic effects have a significant impact on the results. This is especially evident in runs 1a/1b and 2a/2b where identical parameters produce significantly different outcomes. In runs 1A and 1B, disabling/enabling LTSM appears to have had the greatest impact on how quickly the model can begin to start winning with consistency. Run 3a/3b left the tuning parameters largely untouched, instead making a small change to the REWARD_CLOSER parameter.

The impact of a properly designed reward architecture was just as, if not more, important than the tuning of the parameters. Model performance was effectively 0% until the right reward structure was in place. Two key points surfaced in this work as it pertains to rewards. The first was to ensuring rewards are properly scaled, not in absolute terms, but relative to other rewards. Especially where interim rewards compete with terminal rewards, the terminal reward should be greater than accumulated interim reward. This appears to be, at least to some degree, successful in preventing the algorithm from short term exploitation of the reward system. The second point that surfaced in completing the work was that a reward gradient of some sort should exist in most losing states. This can take the form of reducing the penalty for being close to the goal state, or increasing it for being further away, but without this gradient, the feedback to the algorithm is binary and no additional information is propagated from losses.

5 Future Work

There are multiple different avenues for improving and building on this work. Firstly, the results shown here could be improved on with additional tuning of the reward system and the DQN parameters. Additionally, the idea of an iterative tuning algorithm is interesting. An algorithm

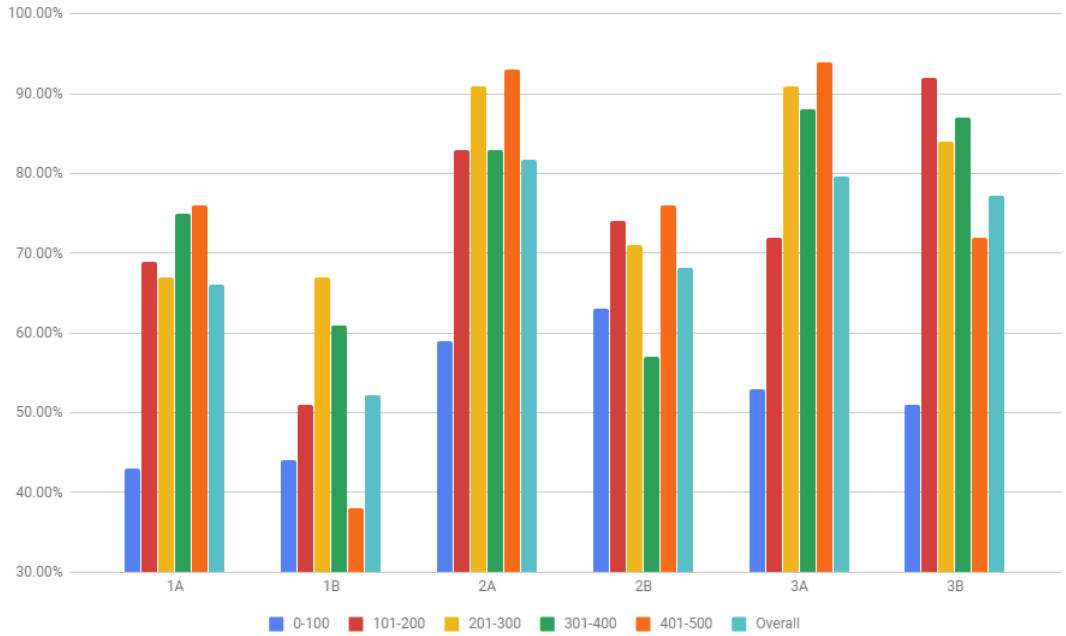

Figure 5: Summary of results for goal 2. Individual bars represent the win % for each century of the 500 episode run. Overall run win % is indicated by the last bar for each run.

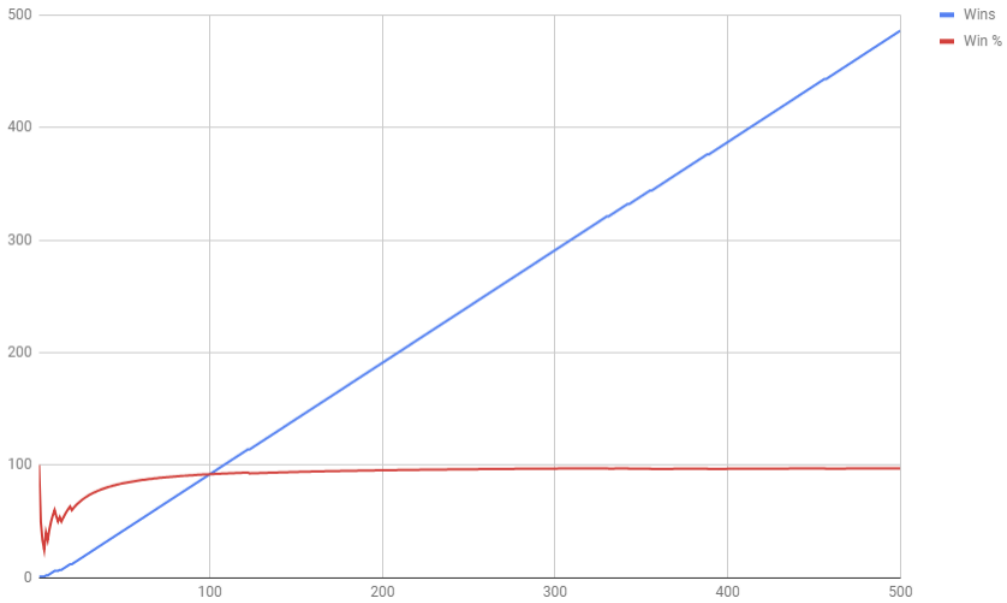

Figure 6: Results of 1 run of 500 episodes for goal 1 of touching robot arm to the tube with a win % greater than 90%

similar to that used for the iterative process used for Maximum Likelihood Estimation could be adopted to tune parameters more effectively.

6 Appendix

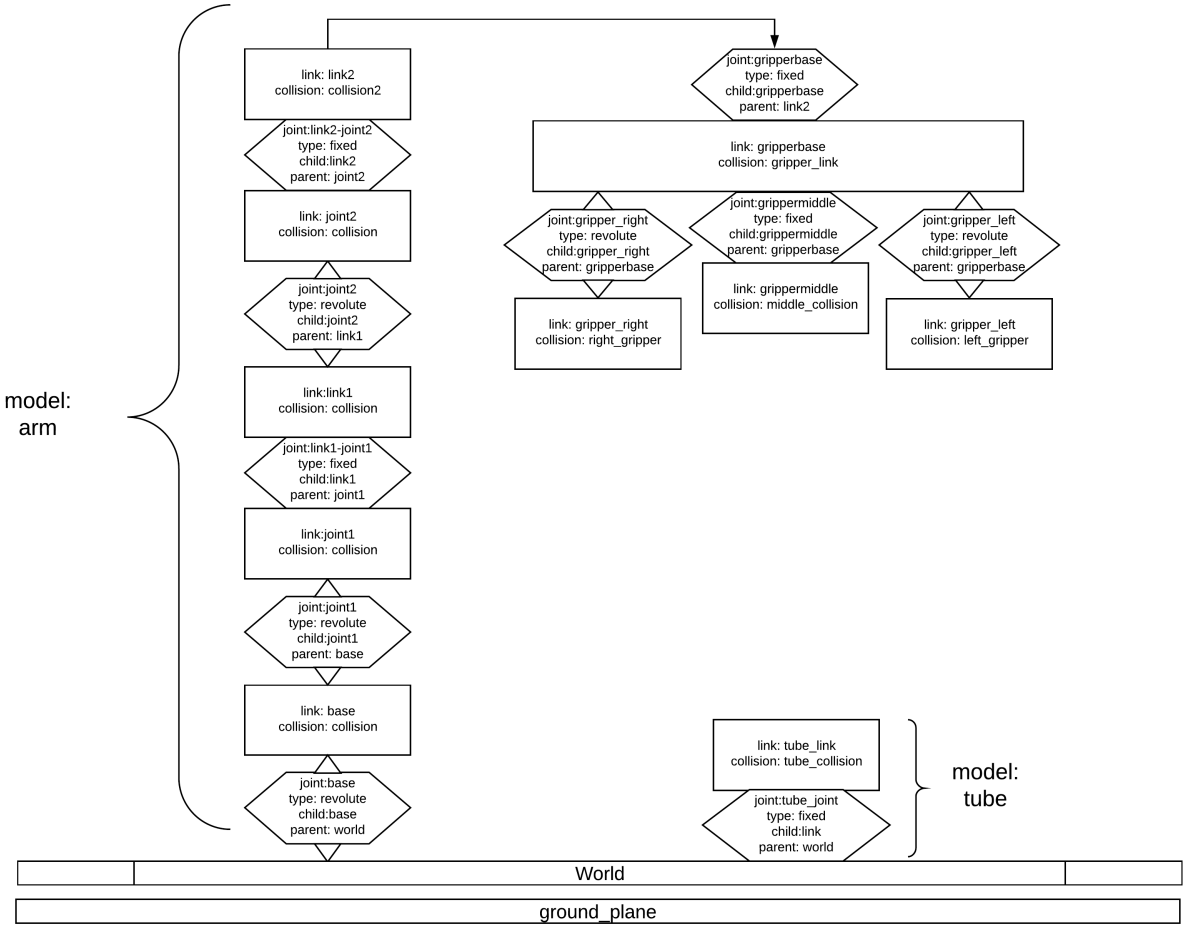

Figure 7: Logical diagram of robot structure and joint/link relationships.