Deep Reinforcement Learning

Kevin Fructuoso

Abstract—The purpose of this report is to apply Deep Reinforcement Learning principles in order to train a robotic arm to make contact with an object. Deep Reinforcement Learning is a very heavily researched field for its many different applications. The robotics domain is among those at the forefront of these remarkable advances in technological capability. In this project, a simulated robot successfully learned a set of behaviors in order to achieve an unknown goal with high accuracy and repeatability.

1 Introduction

Deep Reinforcement Learning (DRL) is a cross-discipline term referring to the application psychological reinforcement principles to train a neural network. In reinforcement learning, an agent is placed in an environment and allowed to perform an action onto the environment with an unknown goal to achieve. The agent observes the reaction of the environment and collects a "reward" or intuition about whether the action positively or negatively contributed to the agent's goal within the environment. DRL is when a neural network performs the function of the agent in order to learn how to achieve the unknown goal on it's own.

When applied to the field of robotics, it can be used in order to train a robot to perform a specific task autonomously. There are a wide array of simple tasks that this may include such as catching a ball or picking up an object. Taking that a step further, DRL can be used to train robots to even play the old arcade game "Brick!"

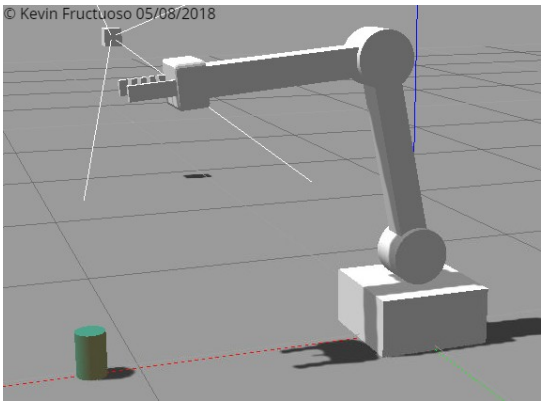

Fig. 1. Gazebo arm and tube target

1.1 Project Goal

This project focuses on using DRL to train a simulated robotic arm in Gazebo to make contact and a target object, shown in Figure 1, at a specified accuracy rate. There are two objectives in this project which must be met with a minimum of 100 episodes:

1) Contact the target object with any part of the arm with 90% accuracy
2) Contact the target object with only the gripper base of the arm with 80% accuracy

2 Project Set Up

2.1 Environment

The project environment is shown in Figure 1. Within it, there is a model robotic arm simulated in Gazebo. A Deep Q-Learning Network (DQN) agent is set up to perform the task of the agent in DRL. This project implements Pytorch, a Python library widely used for machine learning with DRL principles. [1] The Pytorch set up is used to provide reinforcement learning feedback to the arm. The robotic arm employed position control to actuate the joints and make contact with the target object.

2.2 Reward Functions

In DRL, it is critical to properly create reward functions that accurately inform the agent of the effectiveness of the actions it chose to take. This involves positively affirming the agent if a good action was performed and negatively penalize the agent if a poor action was performed. These rewards are given as a numerical value - positive for good actions and negative for bad actions.

Both objectives were met using mostly the same set of reward functions with two additional functions for the goal of contacting the target with only the gripper base.

Reward Function	Value
Episode Objective Met	+100.0
Ground Contact	-5.0
Episode Timeout	-20.0
Non-Gripper Base or Middle Link Collision[1]	-5.0
Gripper Middle Link Collision[1]	+10.0

[1] Only applied in objective to contact the target object with the gripper base only

Additionally, a negative reward is given every frame of an episode based on the below equation for both objectives of this project. This equation subtracts the current frames distance to the target object from the average episode distance from the target, and normalizes it with respect to the average episode target distance. From there, it is scaled down in magnitude in order to prevent a large negative overflow. This reward function is implemented to draw the arm closer to the target object.

$$-0.1*(abs((avgGoalDelta - distDelta)/avgGoalDelta)$$

The reward for completing the goal is set to a significantly larger value in order to give large incentive to repeat these actions. Both ground contact and reaching the frame count limit are met with negative rewards to discourage those behaviors as they are not conducive to achieving the goal. The timeout reward is larger in magnitude to add significant discouragement from taking a long time to act.

For the objective of contacting the object with the gripper base only, two additional reward functions were created. First, a negative reward is given when a collision occurs that does not involve the gripper base (goal) and the gripper middle link located within the grippers claws. This was set to prevent this collision similar to the ground contact reward. Secondly, a collision with the gripper's middle link is given a small reward compared to the episode objective but still results in a failed episode. This is set as the goal requires great precision and this collision represents "nearly" the appropriate behavior. Also, it could possibly occur due to a randomly chosen motion.

2.3 Hyper-parameters

The same hyper-parameters were used for both objectives in the project and are listed in the table below.

Hyper-Parameter	Value
INPUT_WIDTH	128
INPUT_HEIGHT	128
OPTIMIZER	RMSprop
LEARNING_RATE	0.001
REPLAY_MEMORY	10000
BATCH_SIZE	32
USE_LSTM	False

These values play important parts of the training process for the neural network behind the DQN agent. The input width and height were decreased to 128 to lower computational overhead. The learning rate is set to a very low value to prevent early learning "saturation." While larger learning rates should usually end up with better models and faster learning in theory, it can be observed that networks with larger learning rates can plateau earlier than those with smaller learning rates. This results in more loss during training.

Some parameters were chosen to improve learning through optimizing gradient descent. The "RMSprop" optimizer was selected to speed up learning by improving gradient descent in the direction of intended change (towards the end goal) and minimizing gradient descent oscillations in different directions. The batch size was set to 32 as larger sizes provide better estimates of gradient descent in neural network training. However, the value was kept fairly low to reduce computational overhead.

Lastly, the neural network was able to benefit and learn from it's past episode experiences. This project focused on the use of replay memory and did not use long short term memory. Long short term memory allows a network to use a set of episodes long ago from the past and a set of recent episodes in order to inform it's decisions. Similarly, replay memory provides a "knowledge vault" of a defined size. The network agent samples from it randomly in order to help inform it of appropriate actions to take. It has been shown that this greatly stabilizes and improves the DQN training procedure. [2] Therefore, the replay memory size was set to a large value to keep information of all of the episodes it will perform. While a large number is usually better, it should be noted that the system may eventually meet system memory limits if set too high.

3 Results

3.1 Any Arm Contact

For this objective, the arm was capable of making any type of contact with the target object above 90% accuracy as early as 150 episodes (135 passing). The arm was allowed to continue further episodes to ensure sustained success and finished with a final accuracy of 91.5% over 200 episodes total (183 passing).

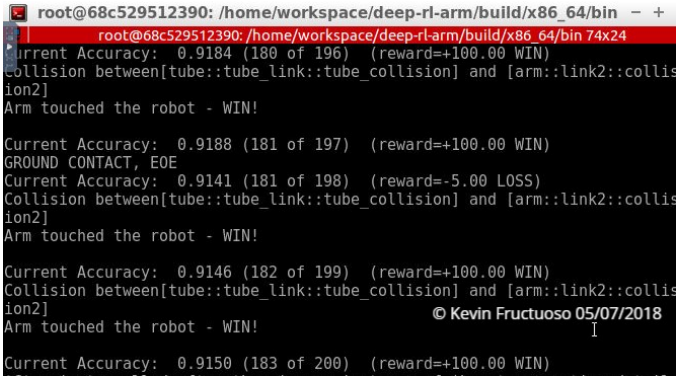

Fig. 2. Results making contact with any part of the arm

3.2 Gripper Base Only

For this objective, the arm was capable of making contact with the target object using only the gripper base above 80% accuracy as early as 350 episodes (280 passing). The arm was allowed to continue further episodes to ensure sustained success and finished with a final accuracy of 80.75% over 400 episodes total (323 passing).

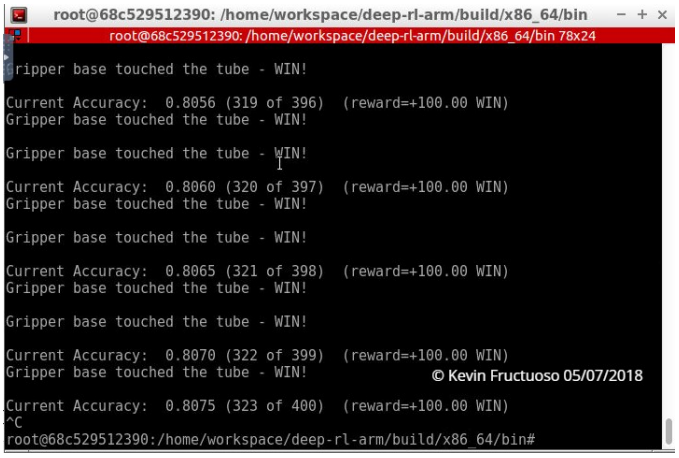

Fig. 3. Results making contact with only the gripper base

4 Discussion

As noted in the Results section, the arm was capable of achieving both of the intended goals. Although there were some bright spots in the arm's performance, there is also some room for improvement.

While the hyper-parameter tuning and reward functions were able to achieve consistency, the agent always started with a rough learning curve. At the beginning of each instantiation of the training, the agent had to work through dozens of episodes of mostly failures before it could meet the goal. Once the goal was met a few times, the reward functions were able to inform the network and begin consistently repeating successful episodes. The speed at which this learning curve is completed correlates with the random motions that are selected in the early episodes of training. Additionally, the arm learned enough such that episode failures due to ground contact or episode timeout were very few and far between.

Another challenge during the training process was that there was a small collision object on the arm within the gripper blades shown in Figure 4. Due to the requirement, colliding with this link results in a failure for the objective to contact the target object with the gripper base. This resulted in the arm tending to touch the object with the bottom of the gripper base. This behavior would not be desired in reality as the arm would not be able to pick up the object in that way. It is recommended to use the gripper's middle link as the sole point of collision desired for a scalable model moving forward.

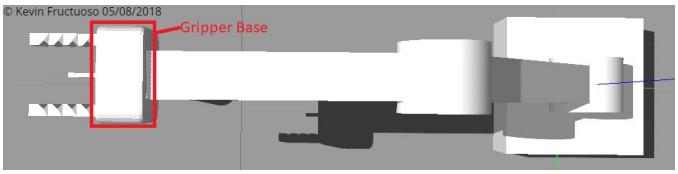

Fig. 4. Closeup of arm gripper

5 Conclusion / Future work

Deep Reinforcement Learning paves the way for countless robotics applications. With these principles, robots can be trained to perform tasks both simple and complicated alike. In this project, the robotic arm was successfully trained to meet the objectives listed in the earlier secionts. The robotic arm in this project could even be trained to pick up and move the object.

The control methods could be fine tuned even further. The position control employed was fairly choppy in the commanded motion. The arm control could be tuned with finer steps in motion. This would need to be proportionately accommodated with increased length of episodes as the finer motion requires more actions for the same distance of motion. Additionally, the control method could be switched to velocity mode to smooth out the executed motions.

Additionally, the accuracy and "realism" of the objective can be increased as well. The hyper-parameters and reward functions could be tuned further to increase the overall speed at which the arm learns how to meet the goal. This could drastically decrease the number of necessary episodes to reach the desired objective accuracy. While the project required the gripper's base to make contact with the target object, a more realistic approach would have been to require only the gripper middle link to contact the object. This would provide a better learned approach with the future goal of actually picking up the object in mind. As discussed in the earlier sections, the gripper base learned to make contact in a way that would not scale well for that future goal.

References

[1] Kevin Fructuoso, "deep-rl-arm mapping project repository," 2018.
[2] PyTorch, "Reinforcement learning (dqn) tutorial," 2017.