Deep RL Arm Manipulation

Stephan Becker

Abstract—This paper aims to give the results for a Deep Reinforcement Learning project, where a robotic arm is trained to touch a small tube. The objectives of the project where to

- Have any part of the robot arm touch the object of interest, with at least a 90% accuracy for a minimum of 100 runs.
- Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy for a minimum of 100 runs.

Both objectives could be achieved by tuning the rewards to the respective tasks.

Index Terms—Robot, IEEEtran, Mobile Robotics, DeepRL.

1 Reward Functions

The robotic arm was controlled by updating the joint positions. Each joint's position could be either increased or decreased by the agent.

The reward for winning an episode (REWARD_WIN) was set to +10, and to -10 for losing (REWARD_LOSS).

The full winning reward was issued when the arm successfully touched the tube for task 1 or when the gripper based touched the tube for task 2. This was implemented by checking if the COLLISION_ITEM (the tube) was colliding with one of the arm collision elements; for task 2 it was furthermore checked if the name of the colliding arm element was the same as the gripper base link (COLLISION_POINT).

Hitting the ground was penalized by REWARD_LOSS, as was running out of time (exceeding 100 frames). To check whether the arm had contacted the ground or not, it was checked if either the minimum or the maximum z-value of the gripper bounding box was below a certain threshold (0.05 in this case). The end of episode timeout was simply triggered by counting the number of frames and terminating the episode when the maximum episode length was exceeded.

In all of the above cases the episode was terminated.

To guide the arm toward the goal intermediate rewards were used. These intermediate rewards were based on the (smoothed) progress the gripper was making towards the goal. Failing to make sufficient progress or moving in the wrong direction resulted in a small penalty, while progress towards the goal resulted in a small (based on the magnitude of the progress) reward. To compute the intermediate reward, first the distance between the gripper bounding box and the tube bounding box was calculated; then this distance was subtracted from the previous distance, to get a value for the progress that the gripper was making towards the goal. A smoothed moving average of the distance delta was then used as the reward value. If the average distance delta was below a certain level, a small penalty was incurred, to penalize very slow progress towards the goal and encourage the agent to converge faster on the goal position.

2 Hyperparameters

Trainig was completed on a AWS p2 instance. The input for the agent was a 128x128x3 image.

The agent used a LSTM network of size 256 and a replay memory of 20000. The network was trained with RMSprop, a learning rate of 0.1 and a batch size of 256. The choice of parameters was mainly based on the default parameters used in the original implementation by Dustin Franklin, with some increases to input size, replay memory and batch size to better utilize the capabilities of the p2 instance.

3 Results

The reinforcement learning agent was using a deep neural network (LSTM) to learn the Q-function for the given tasks. With the learned Q-function the agent can estimate the expected reward for a given state-action pair, and make decision by maximizing the expected reward (choosing the action which gives the maximum reward for the current state). At the start of the simulation the DQN-agent is initialized with the input sizes, the optimizer it's supposed to train the network, whether to use a LSTM or not and some other hyperparameters. For each frame of the simulation (a new camera message received from gazebo is ready for processing), the agent choses an action for the received input image. At the end of each frame in the simulation, the agent receives the current reward from the environment and a flag indicating whether it is the end of the episode. With these values the agent can update the network.

The RL agent was able to quickly (within 50 episodes) achieve the goal accuracy of 90% on the first task of being able to touch the tube with any part of the arm. The agent would then just repeat the winning trajectory over and over again, finishing each episode quickly and with high accuracy.

For the second task the agent needed considerably more training time to converge on the 80% overall accuracy; eventually the agent was able to consistently touch the tube with the gripper's base, hitting the ground very rarely (approx. 1 in 10 approaches). The agent had more trouble finding an optimal trajectory than in the first task. Even with many episodes of training the arm would sometimes just

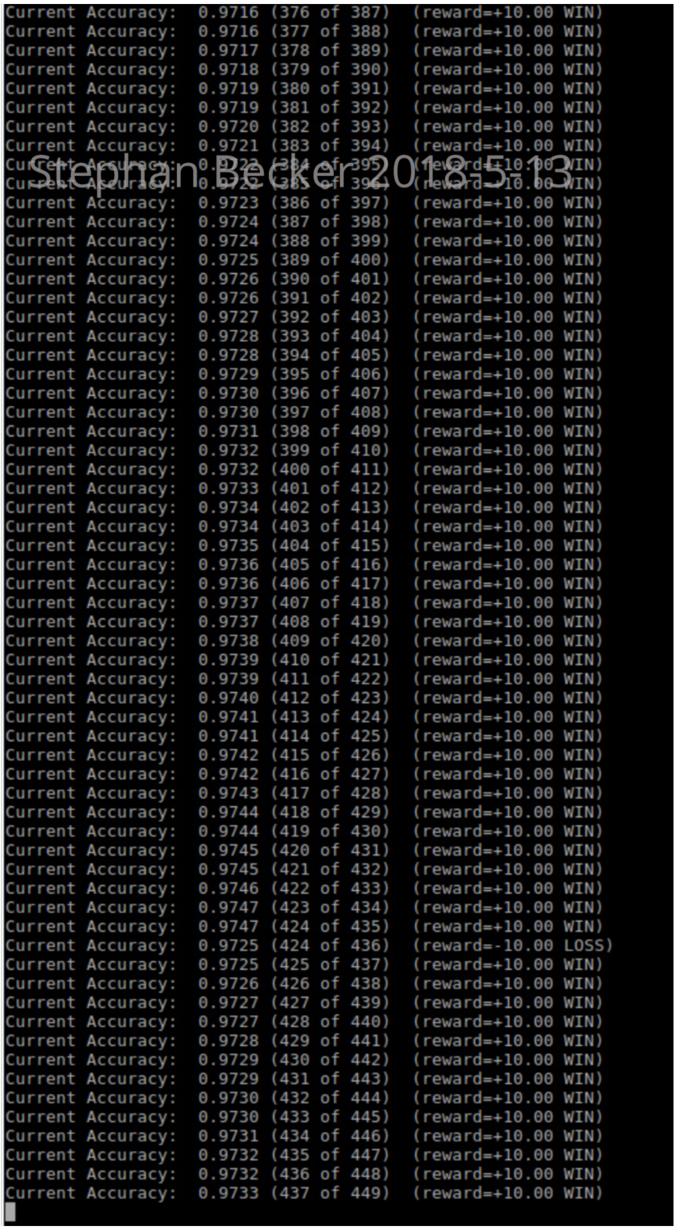

Fig. 1. Task 1 accuracy

move a small distance forward and back again repeatedly for multiple frames.

4 Future Work

Especially for the second task the convergence too a long time; it would be worthwhile to further experiment with the reward system to get the agent to learn more quickly. Building on that the additional challenges should be a future project to be investigated. One of the interesting challenges would be experimenting with the random starting placement of the tube; the agent could not simply learn a single winning trajectory, but would need to determine a different appropriate trajectory for each episode, based on the location of the tube. It would likely need to learn how to localize the tube within the image to accomplish this, a much harder challenge then the given ones with a fixed goal location. Another way to make the task more difficult

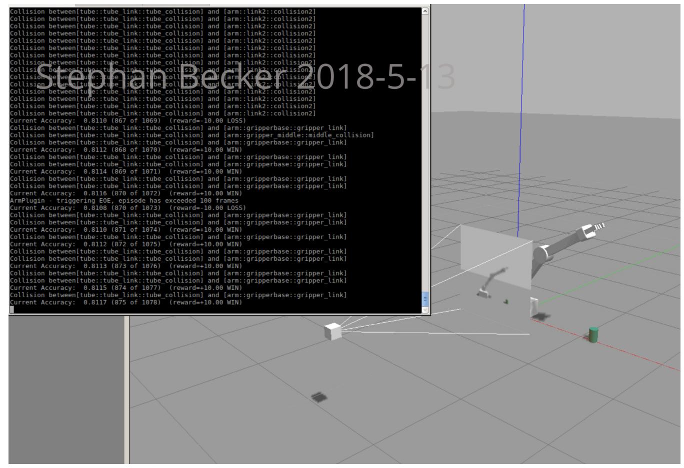

Fig. 2. Task 2 accuracy

would be to give the arm more degrees of freedom, allowing the agent to choose between more action parameters.