Target-Driven Robot Navigation using Deep Reinforcement Learning for Mapless Navigation

Nithin Pingili

Siddhi Manoj Varma Rudraraju

Veda Nandan Gandi

Nikitha Ramisetty

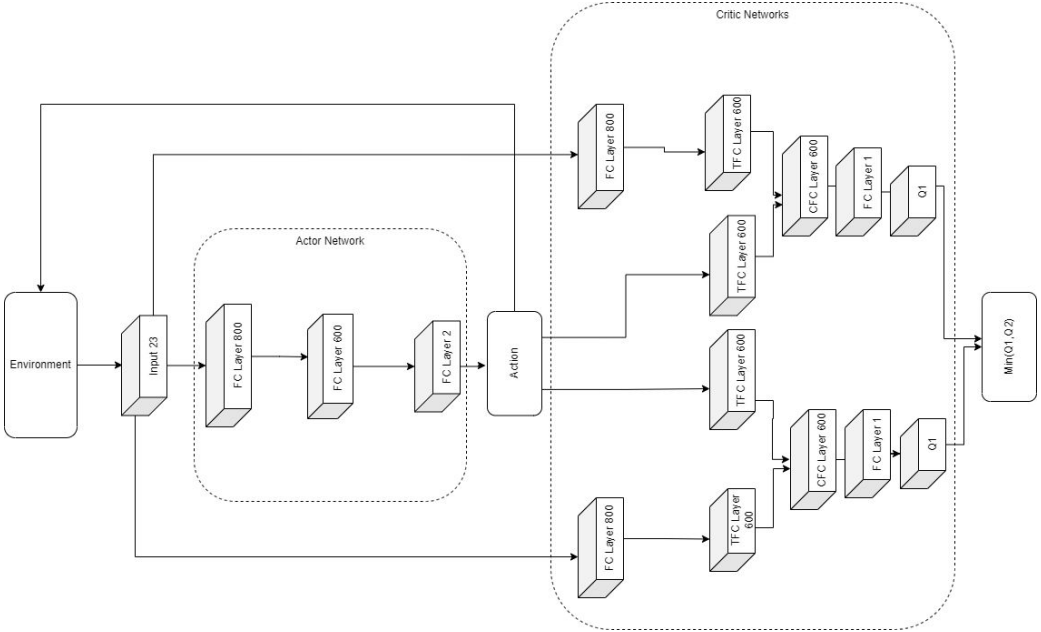
Introduction

- Deep Reinforcement Learning (DRL) has long been speculated to be able to solve all sorts of tasks in various fields.
- We will be implementing a twin-delayed deep deterministic policy gradient (TD3) architecture for learning mobile robot motion with Pytorch and ROS Noetic in python.
- The goal is to avoid collisions with static objects in environment and to find optimal path from source to target.

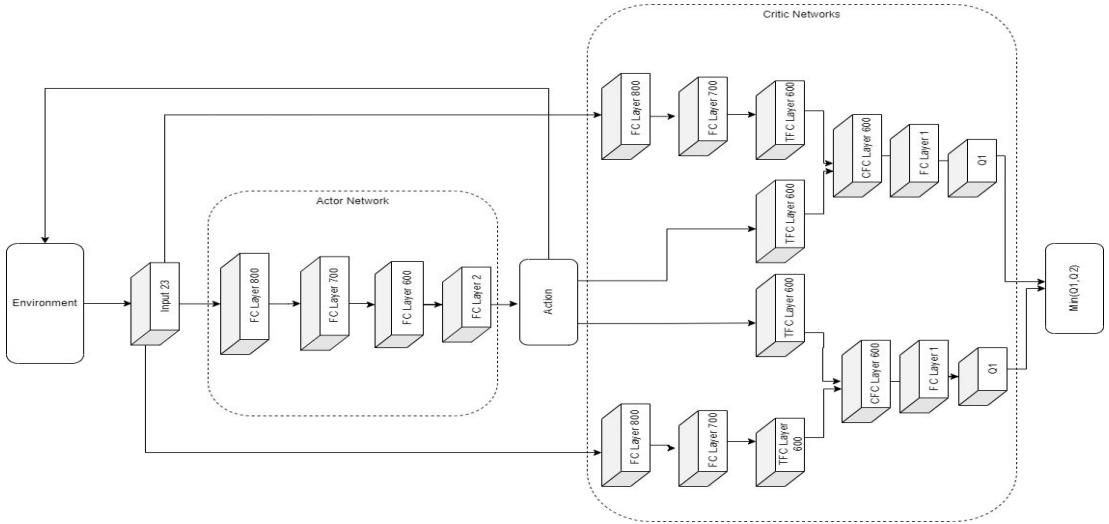
TD3 Architecture

- TD3 is extension of DDPG to solve the problem of overestimating the Q-value.
- It is a actor-critic type of network - “actor” network calculates what action should be taken and “critic” network that evaluates how good is this action.
- The “actor” network takes the environment state as input and outputs what action should be taken.
- The “critic” network which has two critic networks in a loop takes the environment state and the output from the “actor” network and will output the estimated value of the state-action pair.
- The tanh activation function is used to cap the output of the actor network so the output will be in the -1,1 range.
- The output from the “actor” network will be the basis of our robots linear and angular velocities.

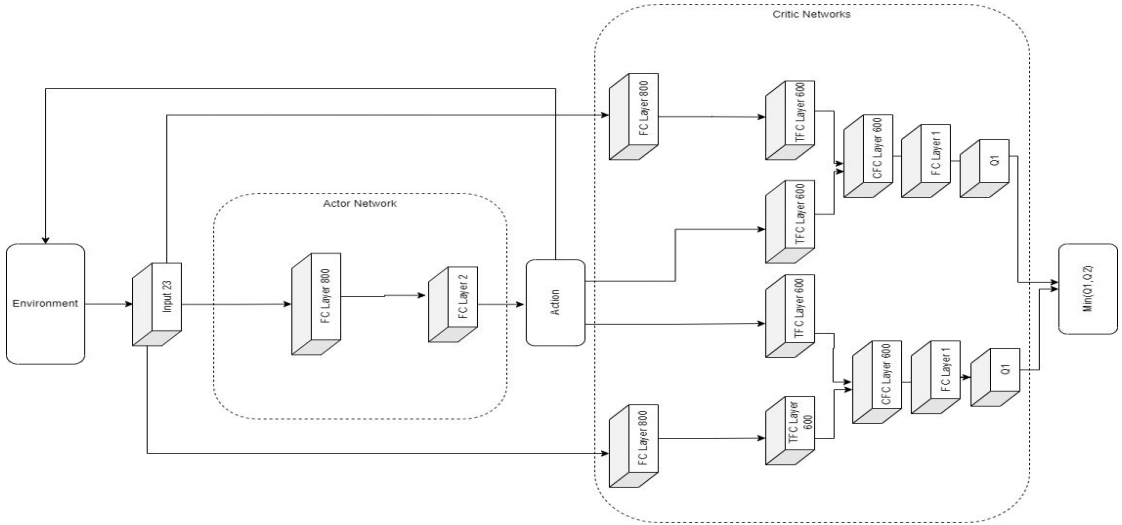
TD3 Neural Network Scheme



3 layer TD3 Neural Network Scheme



1 Layer TD3 Neural Network Scheme



Training

- We used the Pioneer P3DX simulated robot. This is a nonholonomic, differential-drive mobile robot that is capable of moving either forward and backward or rotating around its axis.
- The robot uses a laser sensor to detect the environment by using the numerical representation of the environment of the surroundings of the laser sensor.
- We express polar coordinates in terms of the distance to the target position and the angular difference between the robot's orientation and the target's orientation to guide the robot to the target.
- The action is tuple of linear velocity and angular velocity.
- The state is 24 feature 1D vector combining the laser readings(20), polar coordinates of goal(2) and the action taken at previous time step(2).

Environment

- The robot is trained in 10x10 Meter ROS Gazebo simulation environment with static obstacles.
- 4 cardboard boxes are placed in the simulation and the position of the boxes is changed randomly at the start of each episode.
- We give 100 reward if the robot arrives at the goal, a -100 reward if the robot collides with obstacle and an immediate reward otherwise which is expressed as $r = v - |\omega|$ where v is linear velocity and ω is angular velocity

Results(Average Reward, Number of Collisions)

Neural Network	No obstacles, Number of collisions	With obstacles, Number of collisions
1 Intermediate layer	98.37,2	-94.34,7
2 Intermediate layers	96.54,2	53.33, 2
3 Intermediate layers	-40.45,6	-140.43,0(Moving around the same point)

* Results varied for each run based on the random initial weights in the neural network

Demo

Future Work

- Place more obstacles randomly in the environment
- Speedup the training process using GPU
- Exploring on the relation between number of layers and the complexity of the scene.
- Place the robot in certain edges during training to deliberately let it collide with an obstacle.
- Find the shortest path by reducing the weight of the angular velocity of the robot.

References

- Cimurs, Reinis, Il Hong Suh, and Jin Han Lee. "Goal-Driven Autonomous Exploration Through Deep Reinforcement Learning." IEEE Robotics and Automation Letters 7.2 (2021): 730-737.
- “Twin Delayed DDPG¶.” Twin Delayed DDPG - Spinning Up Documentation, https://spinningup.openai.com/en/latest/algorithms/td3.html.