Deep Reinforcement Learning in Duckietown Environment

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

In this project we trained autonomous driving agents in simulation, that are to be tested later on real hardware as well. For the simulation we used the Gym-Duckietown Simulator -provided by the official Duckietown project. Several navigation challenges exist, this time we focused on LF, hence we only had to follow curves, no intersections or extra obstacles were present.

1 INTRODUCTION

Duckietown is a modular robotics and AI ecosystem. It provides hardware, software, simulation, datasets and learning materials to encourage both teaching and learning.

1.1 Simulator

One of the most important parts of the ecosystem is the Gym-Duckietown Simulator, which provides a physically realistic environment. It is compatible with real world, so the algorithms developed in simulation can be tested on real hardware and maps as well. In the simulation you can easily generate agents (Duckiebots), cities, streets, obstacles, intersections, etc. The fact that this simulator is also integrated with the OpenAI Gym environment, makes reinforcement learning much easier.

2 TRAINING

2.1 Reinforcement Learning

...TODO...

2.2 RAY

Ray provides a simple, universal API for building distributed applications. It is packaged with other libraries for accelerating machine learning workloads, of which we used RLlib and Tune. ...TODO...

2.2.1 RLlib

RLlib is the industry-standard reinforcement learning Python framework built on Ray. It offers both high scalability and a unified API for a variety of applications. ...TODO...

2.2.2 Tune

Tune is a Python library for experiment execution and hyperparameter tuning at any scale. ...TODO...

2.2.3 PPO

RLlib comes with built-in NN models (that can be customized), but an algorithm for the training had to be chosen. Based on comparisions, we opted for Proximal Policy Optimization (PPO). ...TODO...

2.3 Hyperparam optimization

...TODO...

2.4 Tensorboard logs

...TODO...

3 TESTING

The trained model can be easily tested with the help of the Simulator. Given any observation (image), the trained agent can compute the best action from the action space. In our case we need 2 values - velocity of left and right motors. Then the environment's step function can be called with the computed action passed as an argument. As the action is executed, the environment returns the next observation and this process continues.. The environment's render function makes the self-driving agent visible.

4 WRAPPERS

Wrappers are quite important, since they allow us to add functionality to environments - modify rewards, observations etc. We used the original gym-duckietown wrappers with slight modifications for basic observation processing, like image resizing or normalizing. In addition, we prepared 2 custom wrappers.

4.1 CropWrapper

This wrapper is used during the observation phase in order to mitigate the processing of the unnecessary data parts such as the sky and the two sides of the camera view. In this project, we used a custom script, that makes it easy to manipulate the crop ratio for the processed image. This is done by using an additional argument, the crop parameter, which basically means the proportion of the cropped size relative to the whole size of that specific side.

4.2 DtRewardWrapper

The rewarding concept is one of the most important parts of the Reinforcement Learning framework, as the agents' performance is highly relied on it. The reason for this, is that it is the only (scalar) feedback from the environment which the agent receives and later uses to understand the inner dynamics, which is basically the core of the learning process. As a result, the right choice of rewarding concept can improve the success of the training and thus the performance of the trained agent (model).
One of the concepts most commonly used in literature is based on the distance between the lane center line and the agent, but due to the fact, that in our case through the actions we not only manipulate the position of the agent, but also influence the speed. Therefore we had to formulate the reward to be able to control the agent's speed and the position through the rewarding concept.

5 DEVELOPMENT ENVIRONMENT

5.1 Linux VM

...TODO...

5.2 Docker

...TODO...

6 CONCLUSION

...TODO...