Deep Reinforcement Learning for Autonomous Systems

Piero Macaluso - s252894
Candidate
Politecnico di Torino

Prof. Pietro Michiardi
Supervisor
EURECOM

Prof. Elena Baralis
Supervisor
Politecnico di Torino

This document represents the summary of the master thesis project. The source code of this work is publicly available at `https://github.com/pieromacaluso/Deep-RL-Autonomous-Systems`

1. Introduction

Because of its potential to thoroughly change mobility and transport, autonomous systems and self-driving vehicles are attracting much attention from both the research community and industry. Recent work has demonstrated that it is possible to rely on a comprehensive understanding of the immediate environment while following simple high-level directions, to obtain a more scalable approach that can make autonomous driving a ubiquitous technology. However, to date, the majority of the methods concentrates on deterministic control optimisation algorithms to select the right action, while the usage of deep learning and machine learning is entirely dedicated to object detection and recognition.

Recently, we have witnessed a remarkable increase in interest in Reinforcement Learning (RL). It is a machine learning field focused on solving Markov Decision Processes (MDP), where an agent learns to make decisions by mapping situations and actions according to the information it gathers from the surrounding environment and from the reward it receives, trying to maximise it. As researchers discovered, it can be surprisingly useful to solve tasks in simulated environments like games and computer games, and it showed encouraging performance in tasks with robotic manipulators. Furthermore, the great fervour produced by the widespread exploitation of deep learning opened the doors to function approximation with convolutional neural networks, developing what is nowadays known as deep reinforcement learning.

1.1. Objective

In this Thesis, we argue that the generality of reinforcement learning makes it a useful framework where to apply autonomous driving to inject artificial intelligence not only in the detection component but also in the decision-making one. The focus of the majority of reinforcement learning projects is on a simulated environment. However, a more challenging approach of reinforcement learning consists of the application of this type of algorithms in the real world.

After an initial phase where we studied the state-of-the-art literature about reinforcement learning and analysed the set of possible alternatives about technologies to use, we started our project starting from the ideas contained in [2], where the authors were able to train a self-driving vehicle by using Deep Deterministic Policy Gradient (DDPG) [3] by tuning hyper-parameters in simulation. We decided to not use simulators in our approach, therefore we researched an algorithm suitable for real-world experiments and capable of work fine without an expensive real-world hyper-parameter tuning. We found in Soft Actor-Critic (SAC) [1] the algorithm we needed.

Therefore, our thesis consisted of two main contribution:

1. Designing of the Control System to let all components and technologies involved interact;
2. Setting up the experimental framework with SAC algorithm and carrying out an experiment to analyse strengths and weaknesses.

2. Design of the control system

We based our project on Cozmo, a little toy robot produced by Anki, whose developers offered a granular and fully-featured Python SDK with many interfaces to allow a direct control of the robot. We found it to be suitable for our experimental needs: it mounts a grayscale camera with 60 FOV and has two tracks to steer and drive. Our aim was to apply deep reinforcement learning algorithms, so we decided to use PyTorch as deep learning framework and the standardisation approach provided by OpenAI Gym to build up the reinforcement learning environment for the experiment.

Our idea was to build up a system where the human could directly teach to the robot how to drive. The intention was to give the human the total control on the flow of experiments: he is able to start the episode and to stop it when the robot

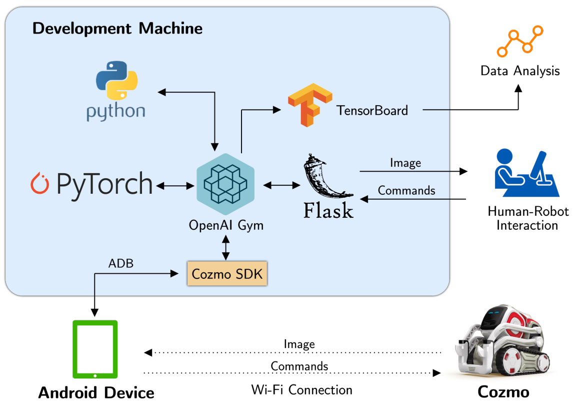

Figure 1. Outline of the control system that shows the most crucial technologies and component involved.

reaches a pernicious situation, so the human can reposition the robot in the closest safe situation and restart the loop. In this scenario, the human has total responsibility for how the robot is trained: he is the one who decides when an action is dangerous or not by disengaging reinforcement learning algorithms decision.

To obtain this configuration, we managed to design a simple and intuitive user interface that prompt the user when he started an experiment. This interface provides a live stream from Cozmo on-board camera and a set of key with the related function. It works through Javascript to communicate to a Flask server that communicates directly to Cozmo SDK and the OpenAI Gym environment we designed to provide information for the user (e.g. images, learning information) and the robot (e.g. commands). We used TensorBoard to gather data for posterior analysis and representations. The system we obtained is outlined in figure 1.

In order to start the experiment, we formalised the problem as a MDP obtaining the following setup:

State Space The representation of the current state consists of two subsequent 64×64 grayscale images. They represent the situation before and after the action respectively. We opted for resizing the images provided by Cozmo SDK for memory consumption reason due to the large experience replay memory needed for off-policy reinforcement learning algorithms.

Action Space The representation of how the robot can interact with the surrouding environment is given by a vector of two real value. The first value represents the desired speed and its range is between 0 and 1, while the second one represent the steering of the vehicle and its range is between -1 and 1. Both values are the result of a simplification useful in the learning phase. In practise, the system designed converts these values to properly interact with the robot.

Reward Function

3. Experiments

TODO:

- Experiment Setup
- Cite preliminary experiment
- Showcase and comment about results with Cozmo Driver

3.1. Experiments with Pendulum-v0

3.2. Experiments with CozmoDriver-v0

4. Conclusions

TODO:

- Conclusion
- Future Work

4.1. Future Work

References

[1] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[2] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[3] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.