Robot Drivers: Learning to Drive by Trial & Error

Michael Bosello[*] Rita Tse[**] Giovanni Pau[*‡†]

[*] Università di Bologna – Department of Computer Science and Engineering, Cesena, Italy

[**] Macao Polytechnic Institute – School of Applied Sciences, Macao, SAR

[‡] UCLA Computer Science Department – Los Angeles, USA

[†] Sorbonne Universitè – LIP 6, Paris, France

MSN 2019

Outline

1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. Conclusions
7. Future Works
8. Demo
9. Related Works

Next in line...

1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. Conclusions
7. Future Works
8. Demo
9. Related Works

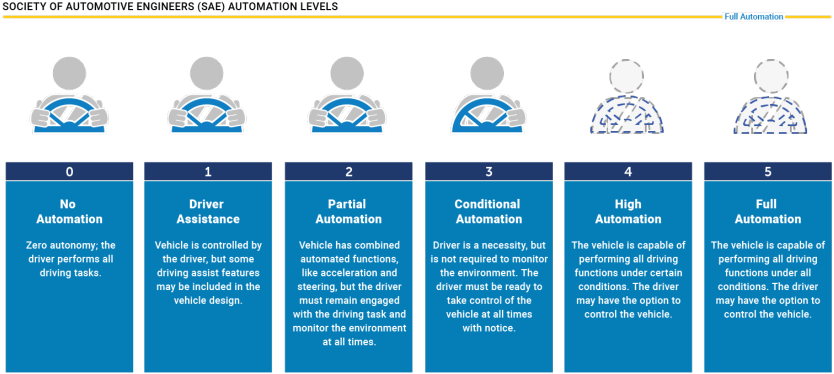
Autonomous cars

- Driving is still a complex task for artificial agents

Three problems

Recognition	Identify environment's component
Prediction	Predict the evolution of the surrounding
Planning	Take decisions and act to pursue the goal

Current status in vehicle automation (credits to SAE & NHTSA)



Level 5

- Full autonomous
- Predicting all possible scenarios
  - known
  - unknown

Current approaches by major players

- Wide range of sensors
- Very high definition multi-layer maps
- Supervised learning
- **Robotic drivers are expected to be perfect**

Drawbacks

Manufacturing costs Expensive sensors

Operational costs Constant need of data-updates

Training data Huge amounts of labeled data or human effort

Covering all possible driving scenarios is very hard

Reinforcement learning for autonomous driving

- Unsupervised learning
- Learns behaviors by trial-and-error
  - like a young human driving students

- Does not require explicit supervision from humans.

- Mainly used on simulated environment
  - to avoid consequences in real-life
- An agent trained in a virtual environment will not perform well in the real setting
  - different visual appearance, especially for textures

Next in line...

1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. Conclusions
7. Future Works
8. Demo
9. Related Works

Reinforcement learning in the physical world

- Depart from the assumption of infallible self-driving vehicles
  - Granting some time to learn how to drive in certain scenarios
- Use of realistic small-scale cars models
  - the agent still faces challenges of a real driving scene
  - inexpensive
  - safe
- We used Deep Q-Network (DQN) to understand the feasibility of the approach

Next in line...

1 Introduction

2 idea

3 Reinforcement Learning Basics

4 Approach

5 Driver Agent Software

6 Conclusions

7 Future Works

8 Demo

9 Related Works

Agent-environment interaction

- In Reinforcement Learning (RL), an agent learns how to fulfill a task by interacting with its environment
- The interaction between the agent and the environment can be reduced to three signals

Signals

State Every information about the environment useful to predict the future

Action What the agent do

Reward A real number that Indicates how well the agent is doing

Policy

- A map from states to actions used by the agent to choose next action

Episode

- A complete run from one of the initial states to a final state

Learning a policy

- The agent wants to maximize the cumulative reward

Cumulative reward

- The sum of rewards over time

- A way to produce a policy is to estimate the action-value function
  - given the function, the agent needs only to perform the action with the greatest value

Action-value function

- (state, action) $\rightarrow$ expected return (the expectation of future rewards)

The exploration/exploitation dilemma

- The agent has to behave optimally but it needs to explore the environment to improve
- ϵ-greedy policy: the agent behave greedily but there is a (small) ϵ probability to select a random action.

Deep Q-Network

Q-learning

- Is an algorithm that approximate the action-value function
- At every iteration the estimation is refined thanks to the new experience.
- Every time the evaluation becomes more precise, the policy gets closer to the optimal one.

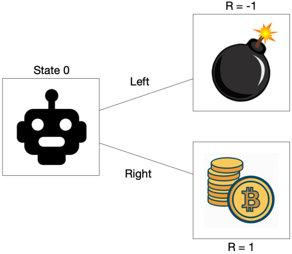
DQN

- When the number of states increases, it become impossible to use a table to store the Q-function
- A neural network can approximate the Q-function

Q-learning example

Formula

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$



State	Action	Expected Reward
State 0	left	0.6
State 0	right	0.5

$\downarrow$ left

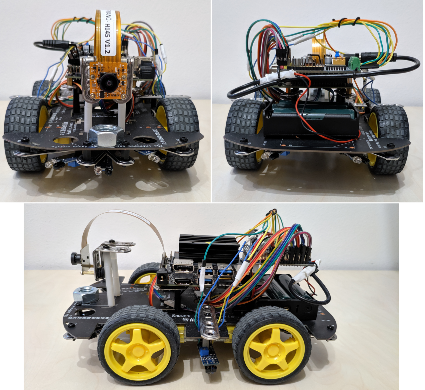
State	Action	Expected Reward
State 0	left	0.4
State 0	right	0.5

$Q(S0, left) \leftarrow 0.6 + 0.2[-1 + 1 * 0.6 - 0.6]$

Next in line...

1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. Conclusions
7. Future Works
8. Demo
9. Related Works

Hardware setup



The cars

- Mainboard: Jetson Nano
  - designed to enable embedded AI
  - train neural networks in real-time.
- Front-facing wide-angle camera.
  - to obtain an ample field of view
- Three frontal IR distance sensors
- Two rear IR distance sensors
- Two line sensors to the right and left.

Circuit

RL environment I

State

- Two subsequent camera frames form the state
  - decisions are based solely on the raw pixel
  - two frames allow to detect movements

Action

- Go forward
- Turn right
- Turn left
- Brake

Episode

- An episode ends when the car crashes
- The car goes backward until
  - all the front sensors turn off
  - the back sensors reveal an obstacle
- Then, a new episode of training starts

RL environment II

Reward

- Based on agent's actions and sensors value
  - After training, the agent can foresee the sensors values
- Clipped between [-1, 1] because of learning efficiency [Mnih et al., 2013]

- Forward movement = 0.9
  - Incentive to run along the circuit
- Turning = 0.2
- Brake = 0
- Collision = -1
  - frontal sensor activation
- Line sensor activation = -0.3
  - to keep a smoother path
- Turning to a direction and just after turning to the opposite one = -0.2
  - to reduce oscillations

Next in line...

1 Introduction

2 idea

3 Reinforcement Learning Basics

4 Approach

5 Driver Agent Software

6 Conclusions

7 Future Works

8 Demo

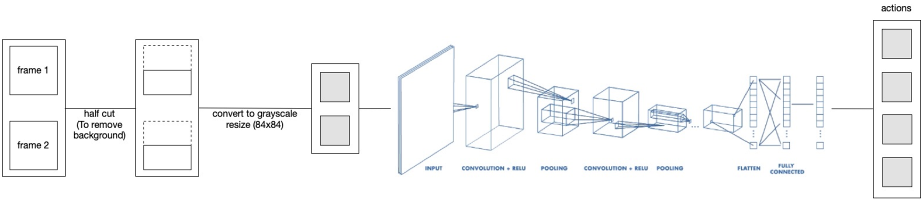
9 Related Works

Driver agent software I

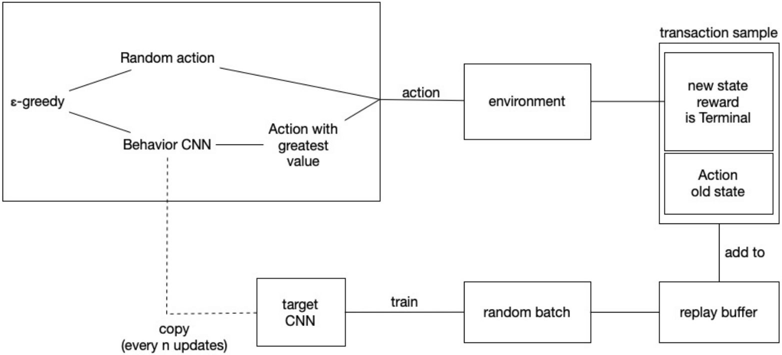
Two parts

- Neural network
- Training cycle

Driver agent software II



Driver agent software III



- Experience replay is needed to break temporal relation
- Transaction samples are used to compute the expected reward which is needed to do training

Software

Implementation available on GitHub

- `https://github.com/MichaelBosello/Self-Driving-Car`
- Includes trained models

Dataset

- Produced as a result of the experiment
- Contains the videos of the car's camera with timestamped event logs
- Available in the repo

Next in line...

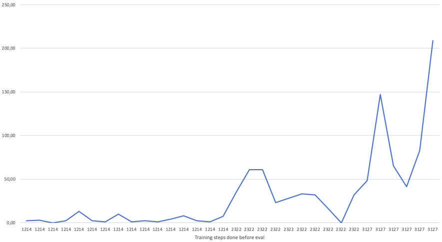
1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. **Conclusions**
7. Future Works
8. Demo
9. Related Works

Conclusions

Results

- The driver-agent successfully learned a control policy
  - to drive in two circuits
  - only using raw pixels

$\Rightarrow$ DQN can operate in physical environments



Evaluation

- The plot shows the sum of rewards for ten episodes in the evaluation phase
- The evaluation phase is interleaved with training
- The X-axis indicates the number of training episodes before the considered evaluation run

Results demo

[https://youtu.be/pAwzMXldfss]

Next in line...

1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. Conclusions
7. **Future Works**
8. Demo
9. Related Works

Future works

- Instrumenting the small-scale vehicles with LiDAR
- Training the agent in realistic small-scale urban scenarios
- Adding more sensory data
- Transfer learning from small-scale vehicles to actual vehicles in a controlled environment
- Cooperative learning
  - exchange of experience between cars with efficient communication [Kamp et al., 2018]
- Reinforcement learning with bounded risk [Geibel, 2001]
  - find the optimal policy with bounded risk
  - risk as the probability to enter in a fatal state
- Safe reinforcement learning [Shalev-Shwartz et al., 2016]
  - ensure functional safety through hard constrains

Next in line...

1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. Conclusions
7. Future Works
8. Demo
9. Related Works

Results demo

youtu.be/pAwzMXldfss

Next in line...

1. Introduction
2. idea
3. Reinforcement Learning Basics
4. Approach
5. Driver Agent Software
6. Conclusions
7. Future Works
8. Demo
9. Related Works

Related works I

End-to-end framework [Sallab et al., 2017]

- The input is the environment's state
- The output is the driving action

Splitting the task into two modules [Li et al., 2019]

- The perception module uses DL to extract the track features
- The control module uses RL to make decisions

- Both the works used TORCS, an open-source car racing simulator

Use of DQN as is [Yu et al., 2016]

- In a racing game.

Transfer learning from a virtual world to the real one [You et al., 2017]

- The framework converts the images rendered by the simulator to realistic ones
- The agent is trained on those synthesized scene

Related Works II

Donkey Car [don, 2019]

- Self-driving small robotic car
- Uses deep learning to mimic the trajectories provided by the user

Feedback and cooperation

- Anyone who is interested in this research line that wants to
  - discuss
  - cooperate with us
  - give feedback

  can contact

  Michael michael.bosello@studio.unibo.it
  Me giovanni.pau@unibo.it

- Any questions?

References I

(2019).
Donkey Car.
[Online; accessed 25. Sep. 2019].

Geibel, P. (2001).
Reinforcement learning with bounded risk.
In *ICML*, pages 162–169.

Kamp, M., Adilova, L., Sicking, J., Hüger, F., Schlicht, P., Wirtz, T., and Wrobel, S. (2018).
Efficient decentralized deep learning by dynamic model averaging.

Li, D., Zhao, D., Zhang, Q., and Chen, Y. (2019).
Reinforcement learning and deep learning based lateral control for autonomous driving [application notes].
IEEE Computational Intelligence Magazine, 14(2):83–98.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013).
Playing atari with deep reinforcement learning.

References II

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017).
Deep reinforcement learning framework for autonomous driving.
Electronic Imaging, 2017(19):70–76.

Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016).
Safe, multi-agent, reinforcement learning for autonomous driving.

You, Y., Pan, X., Wang, Z., and Lu, C. (2017).
Virtual to real reinforcement learning for autonomous driving.
CoRR, abs/1704.03952.

Yu, A., Palefsky-Smith, R., and Bedi, R. (2016).
Deep reinforcement learning for simulated autonomous vehicle control.
Studocu, Stanford University.

Robot Drivers: Learning to Drive by Trial & Error

Michael Bosello[*] Rita Tse[**] Giovanni Pau[*‡†]

[*] Università di Bologna – Department of Computer Science and Engineering, Cesena, Italy

[**] Macao Polytechnic Institute – School of Applied Sciences, Macao, SAR

[‡] UCLA Computer Science Department – Los Angeles, USA

[†] Sorbonne Universitè – LIP 6, Paris, France

MSN 2019