

# Deep Reinforcement Learning for Autonomous Systems

---

Designing a control system to exploit model-free deep reinforcement learning algorithms to solve a real-world autonomous driving task of a small robot.

**Candidate:** Piero Macaluso

**Supervisors:** Prof. Pietro Michiardi  
Prof. Elena Baralis

EURECOM, France  
Politecnico di Torino, Italy

March 14, 2020



**POLITECNICO  
DI TORINO**

This work of this thesis was developed at EURECOM (Sophia Antipolis, France)  
in collaboration with

Prof. Pietro Michiardi (EURECOM)

Prof. Elena Baralis (Politecnico di Torino)

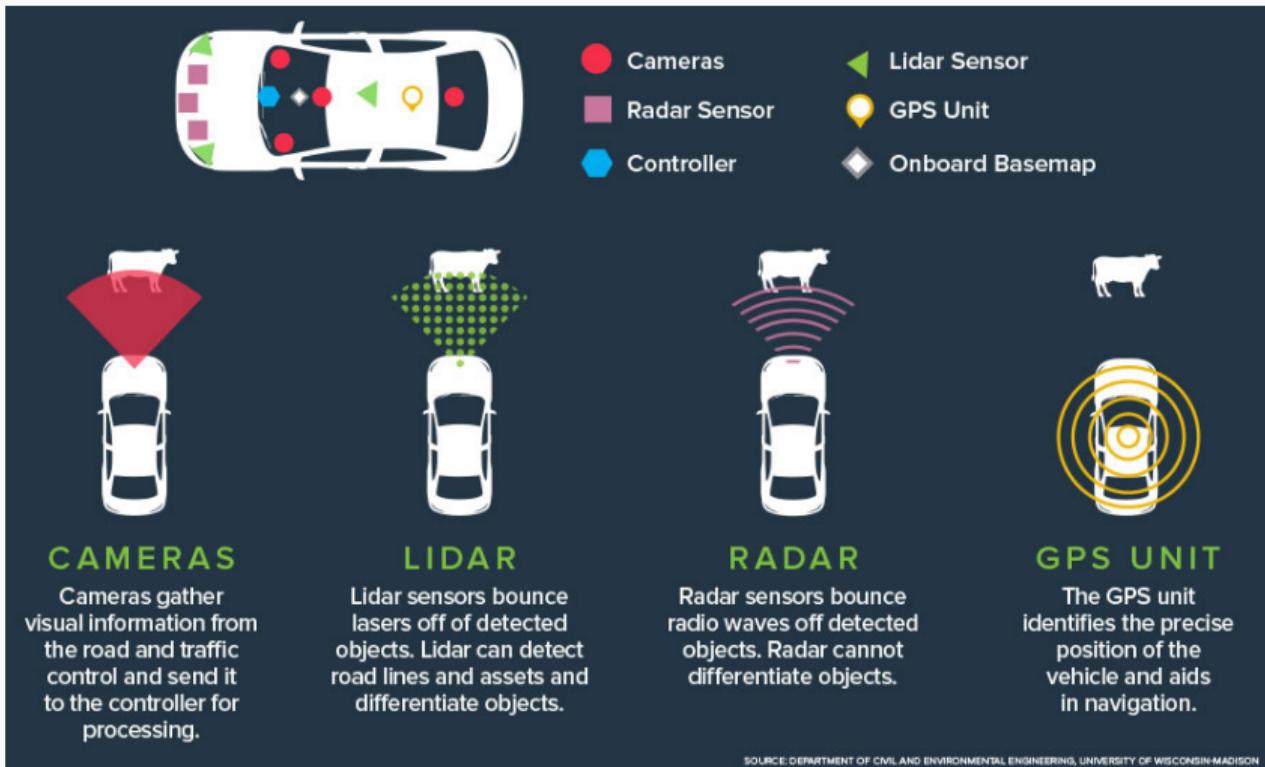
## Table of contents

1. Background
2. Design of the control system
3. Experimental methodology and results
4. Conclusions and future work

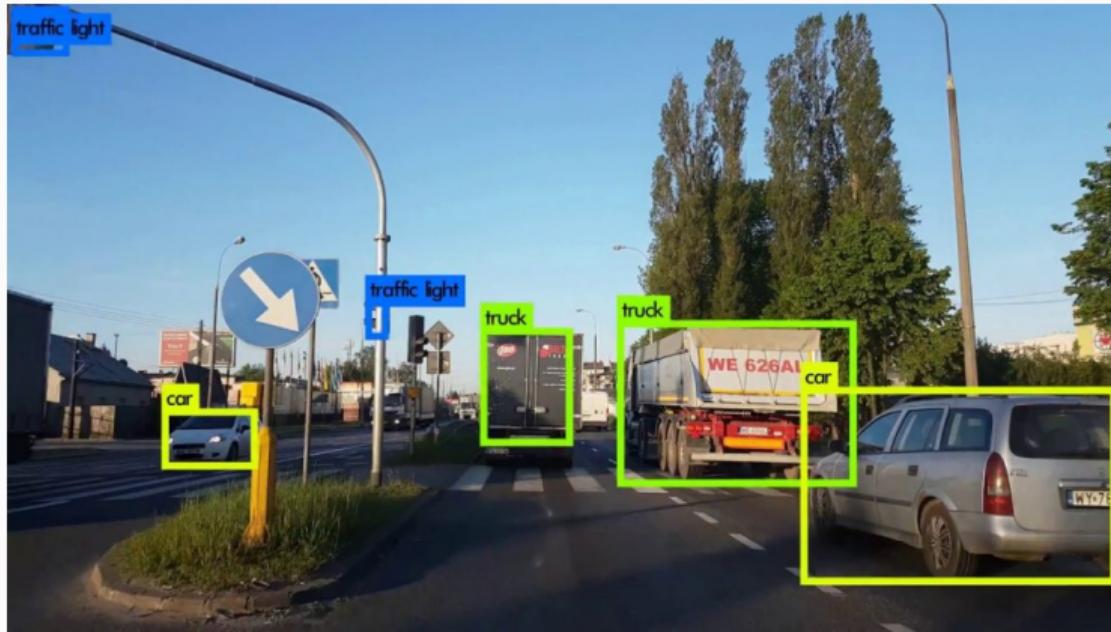
# Background

---

# State-of-the-art Autonomous Driving Systems

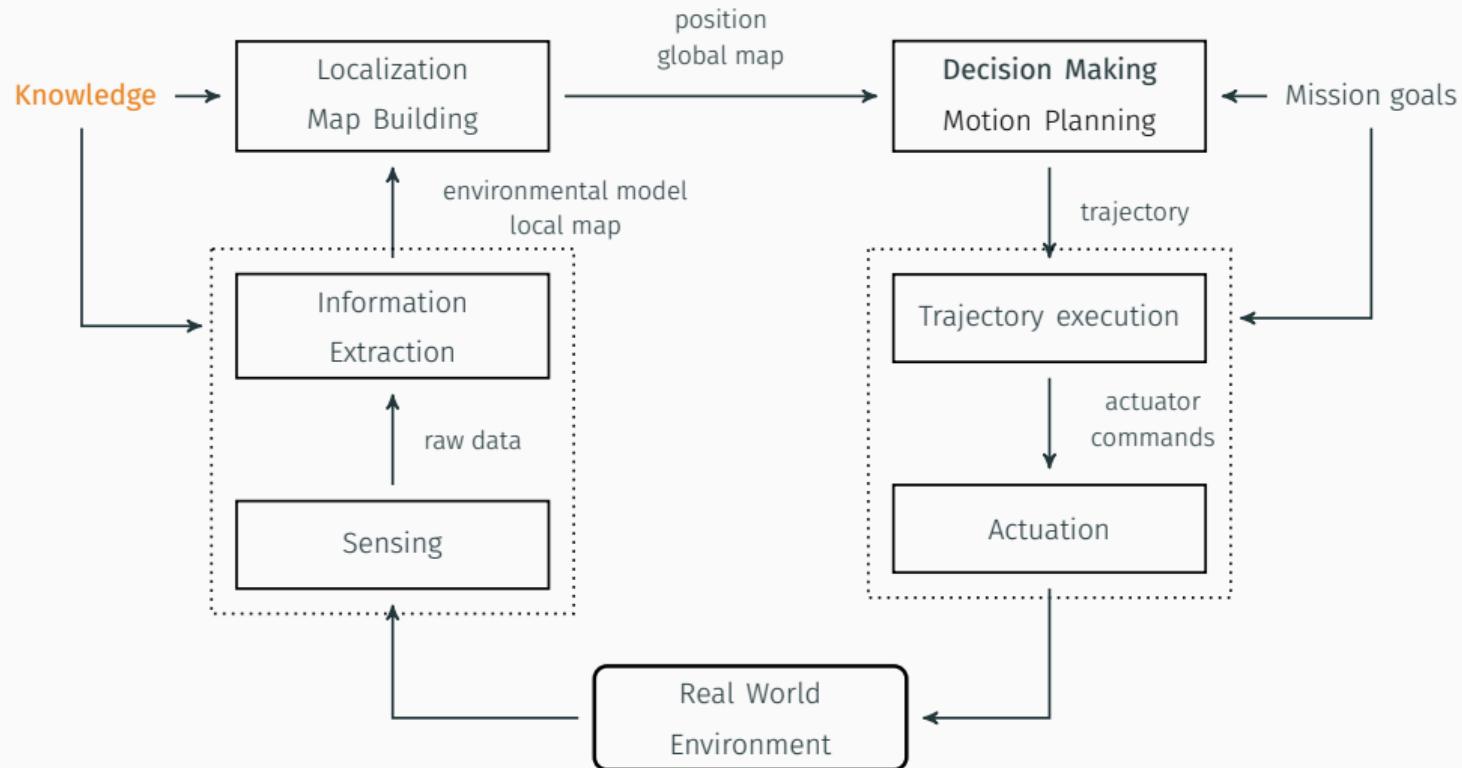


# State-of-the-art Autonomous Driving Systems

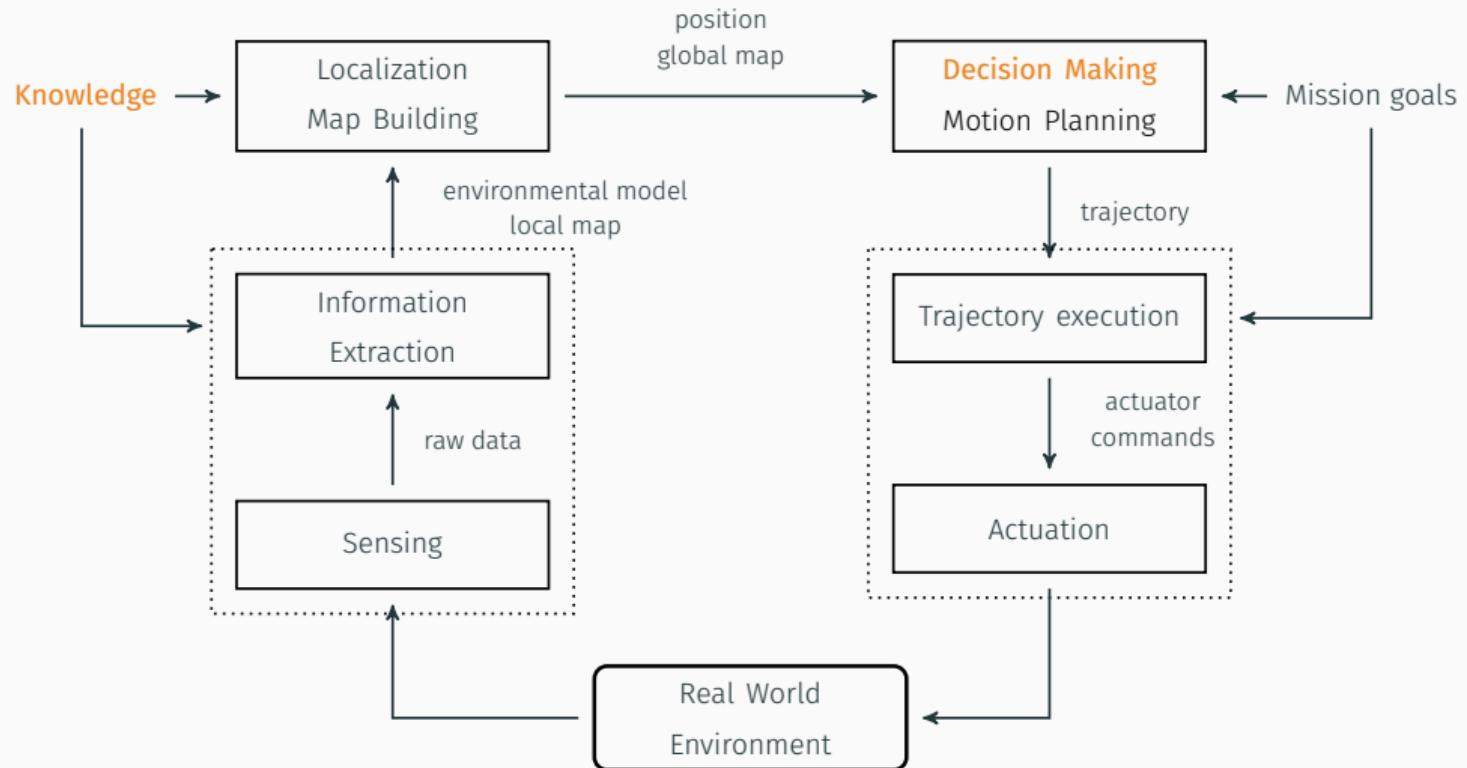


Deep Learning and Machine Learning are mainly exploited in  
**object detection** and **recognition**.

# State-of-the-art Autonomous Driving Systems



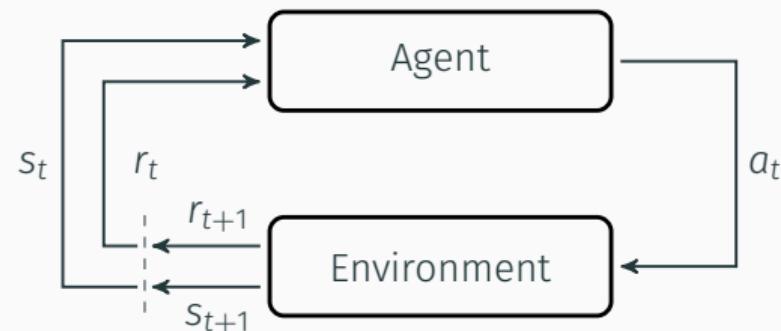
# State-of-the-art Autonomous Driving Systems



# Reinforcement Learning

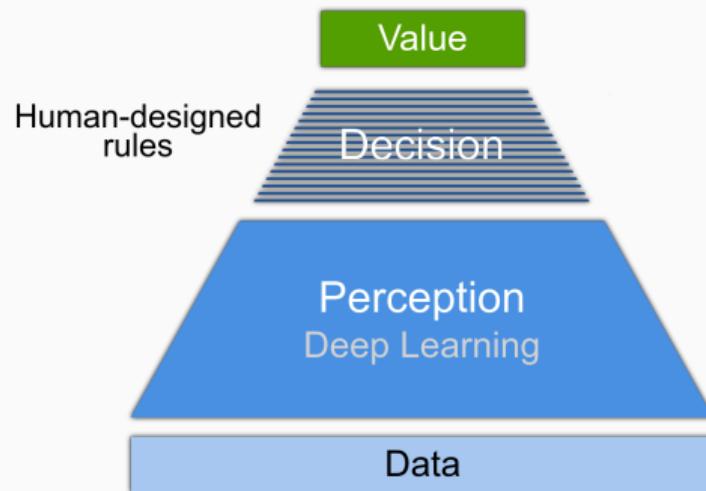
Problems involving an **agent** interacting with an **environment**, which provides numeric **reward signals**.

**Goal:** Learn how to take actions in order to maximize a reward function.



# From Data to Value

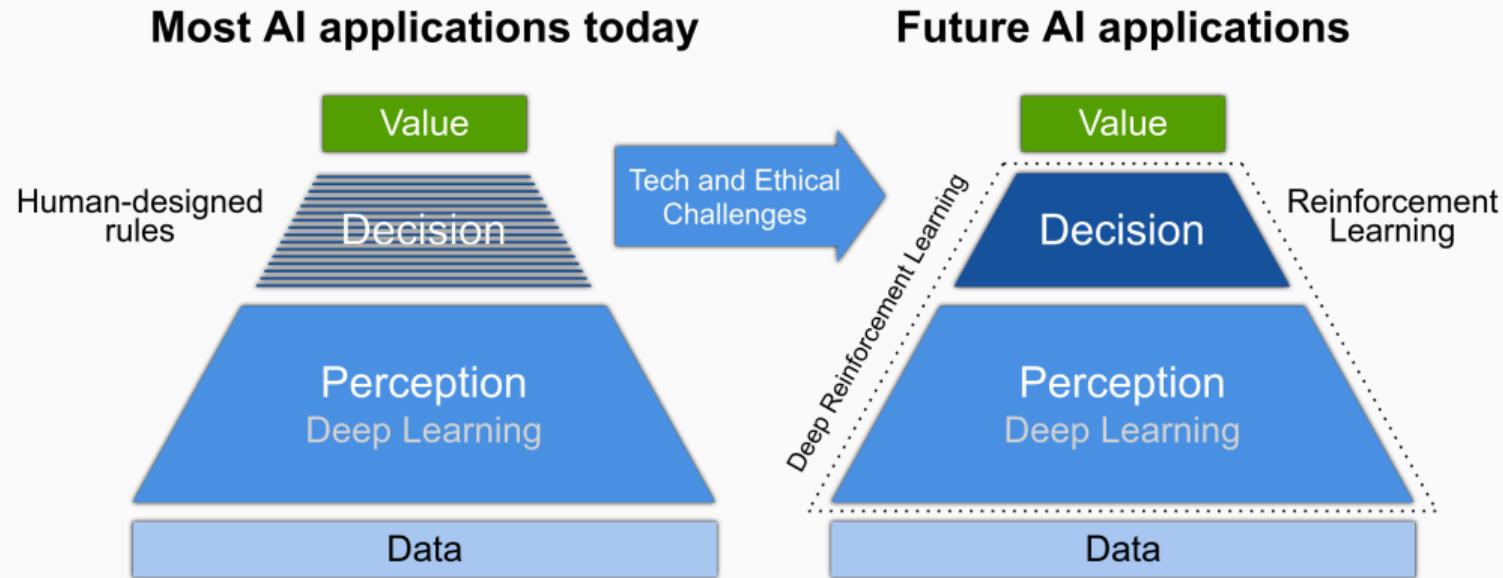
**Most AI applications today**



---

Charafeddine, *Reinforcement Learning in the Wild and Lessons Learned*.

# From Data to Value



---

Charafeddine, *Reinforcement Learning in the Wild and Lessons Learned.*

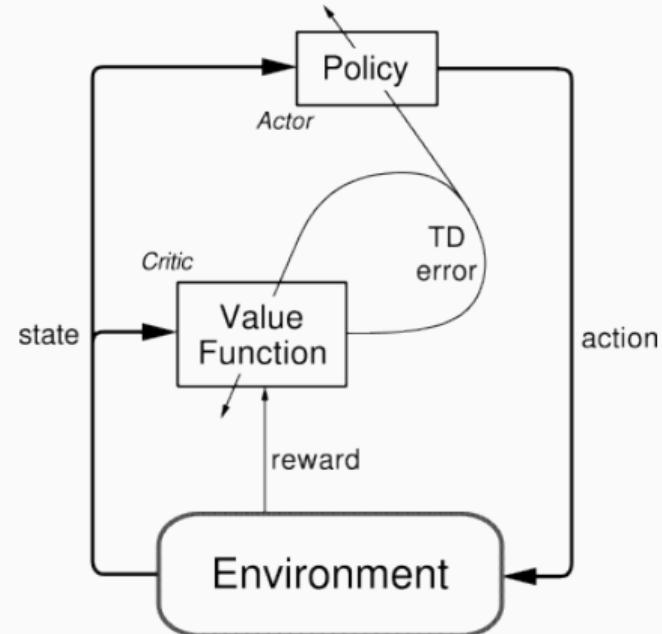
# Model-Free Actor Critic methods

## Critic Network

Estimates the value function. This could be the action value  $Q$  or state value  $V$ .

## Actor Network

Updates the policy distribution in the direction suggested by the Critic (such as with policy gradients).



# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Deterministic

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Deterministic
- Exploration:

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Deterministic
- Exploration:
  - Ornstein–Uhlenbeck process noise

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Deterministic
- Exploration:
  - Ornstein–Uhlenbeck process noise
  - Noise regulation with  $\epsilon$ -decay function

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

# Deep Deterministic Policy Gradient (DDPG)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Deterministic
- Exploration:
  - Ornstein–Uhlenbeck process noise
  - Noise regulation with  $\epsilon$ -decay function

Needs accurate hyper-parameters fine-tuning

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.

## Soft Actor-Critic (SAC)

- Off-Policy:

## Soft Actor-Critic (SAC)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples

## Soft Actor-Critic (SAC)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous

## Soft Actor-Critic (SAC)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Stochastic

## Soft Actor-Critic (SAC)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Stochastic
- Exploration:

## Soft Actor-Critic (SAC)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Stochastic
- Exploration:
  - Temperature Parameter

## Soft Actor-Critic (SAC)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Stochastic
- Exploration:
  - Temperature Parameter
  - Auto-tuning

## Soft Actor-Critic (SAC)

- Off-Policy:
  - Experience Replay Memory of  $(s_t, a_t, r_t, s_{t+1}, d_t)$  tuples
- Action space: Continuous
- Policy: Stochastic
- Exploration:
  - Temperature Parameter
  - Auto-tuning

Suitable for Real-World Experiments

## Design of the control system

---

# Anki Cozmo - Not just a toy robot



Why Anki Cozmo?

- Small and portable

# Anki Cozmo - Not just a toy robot



## Why Anki Cozmo?

- Small and portable
- 30fps VGA Camera

# Anki Cozmo - Not just a toy robot



## Why Anki Cozmo?

- Small and portable
- 30fps VGA Camera
- Powerful mechanics

# Anki Cozmo - Not just a toy robot



## Why Anki Cozmo?

- Small and portable
- 30fps VGA Camera
- Powerful mechanics
- **Python SDK and interfaces**

# Track Design



## Features:

- Portable

# Track Design



## Features:

- Portable
- Low-reflections

# Track Design



## Features:

- Portable
- Low-reflections
- Scaled Reality

# Track Design



## Features:

- Portable
- Low-reflections
- Scaled Reality
- Reproducible

## Main Goals



- Building a **control system** and an **interface** between Cozmo robot and algorithms using OpenAI Gym.

## Main Goals



- Building a **control system** and an **interface** between Cozmo robot and algorithms using OpenAI Gym.
- **Real World Reinforcement Learning experiments analysis.**

# Main Goals



- Building a **control system** and an **interface** between Cozmo robot and algorithms using OpenAI Gym.
- **Real World Reinforcement Learning** experiments analysis.
  - **No model** of the environment.

# Main Goals



- Building a **control system** and an **interface** between Cozmo robot and algorithms using OpenAI Gym.
- **Real World Reinforcement Learning** experiments analysis.
  - **No model** of the environment.
  - **No preliminary simulation** to tune hyper-parameters

## MDP Formalisation - Observation



Configuration:

- Gray-scale image

## MDP Formalisation - Observation



Configuration:

- Gray-scale image
- Frame Rate:  $\sim 15fps$

## MDP Formalisation - Observation



Configuration:

- Gray-scale image
- Frame Rate:  $\sim 15fps$
- Raw image size:  $64 \times 64$  pixels

## MDP Formalisation - Observation



Configuration:

- Gray-scale image
- Frame Rate:  $\sim 15fps$
- Raw image size:  $64 \times 64$  pixels
- Stack size: 2

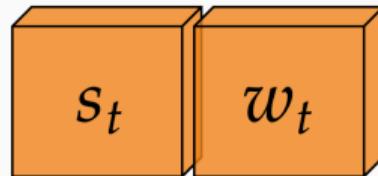
## MDP Formalisation - Observation



Configuration:

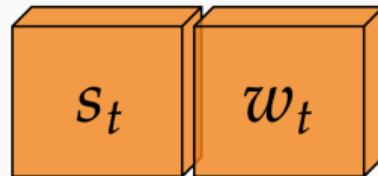
- Gray-scale image
- Frame Rate:  $\sim 15fps$
- Raw image size:  $64 \times 64$  pixels
- Stack size: 2

## MDP Formalisation - Actions



$$s_t \in \{x \in \mathbb{R} \mid 0 \leq x \leq 1\} \quad w_t \in \{x \in \mathbb{R} \mid -1 \leq x \leq 1\}$$

## MDP Formalisation - Actions

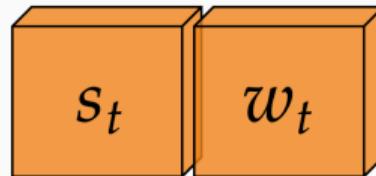


$$s_t \in \{x \in \mathbb{R} \mid 0 \leq x \leq 1\} \quad w_t \in \{x \in \mathbb{R} \mid -1 \leq x \leq 1\}$$

Maximum forward speed  $\rightarrow s_{\text{forward\_max}} = 150\text{mm/s}$

Maximum turning speed  $\rightarrow s_{\text{turning\_max}} = 100\text{mm/s}$

## MDP Formalisation - Actions



$$s_t \in \{x \in \mathbb{R} \mid 0 \leq x \leq 1\} \quad w_t \in \{x \in \mathbb{R} \mid -1 \leq x \leq 1\}$$

Maximum forward speed  $\rightarrow s_{\text{forward\_max}} = 150\text{mm/s}$

Maximum turning speed  $\rightarrow s_{\text{turning\_max}} = 100\text{mm/s}$

Left tread speed  $\leftarrow s_t \cdot s_{\text{forward\_max}} + w_t \cdot s_{\text{turning\_max}}$

Right tread speed  $\leftarrow s_t \cdot s_{\text{forward\_max}} - w_t \cdot s_{\text{turning\_max}}$

## MDP Formalisation - Reward

Distance Covered



## MDP Formalisation - Reward



Distance Covered

Fixed timing between actions:  $T_t [s] \leftarrow \frac{1}{15 \text{ fps}}$

## MDP Formalisation - Reward



Distance Covered

Fixed timing between actions:  $T_t [s] \leftarrow \frac{1}{15 \text{ fps}}$   
Desired Speed:  $s_t [\text{mm/s}]$

## MDP Formalisation - Reward



Distance Covered

Fixed timing between actions:  $T_t$  [s]  $\leftarrow \frac{1}{15}$  fps  
Desired Speed:  $s_t$  [mm/s]

$$R_t = s_t \cdot T_t$$

## MDP Formalisation - Reward



Distance Covered

Fixed timing between actions:  $T_t$  [s]  $\leftarrow \frac{1}{15}$  fps  
Desired Speed:  $s_t$  [mm/s]

$$R_t = s_t \cdot T_t$$

if  $t$  is terminal  $\rightarrow R_t = 0$

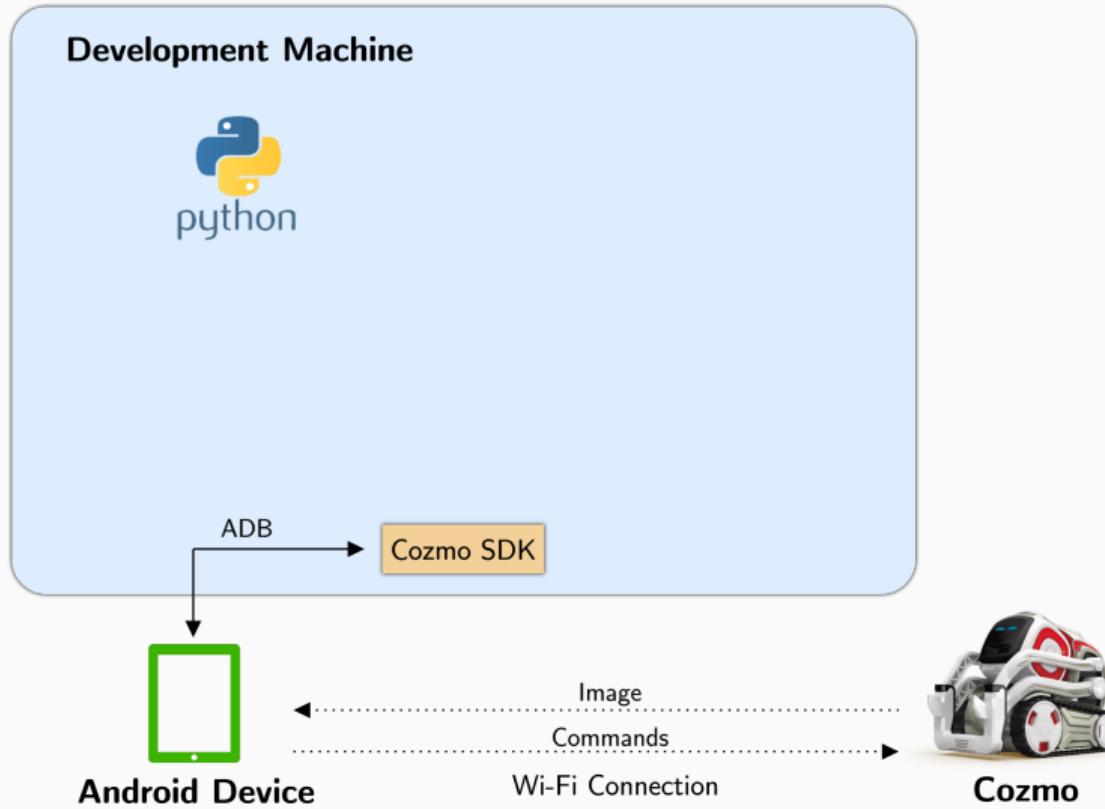
# Outline of the control system

**Development Machine**

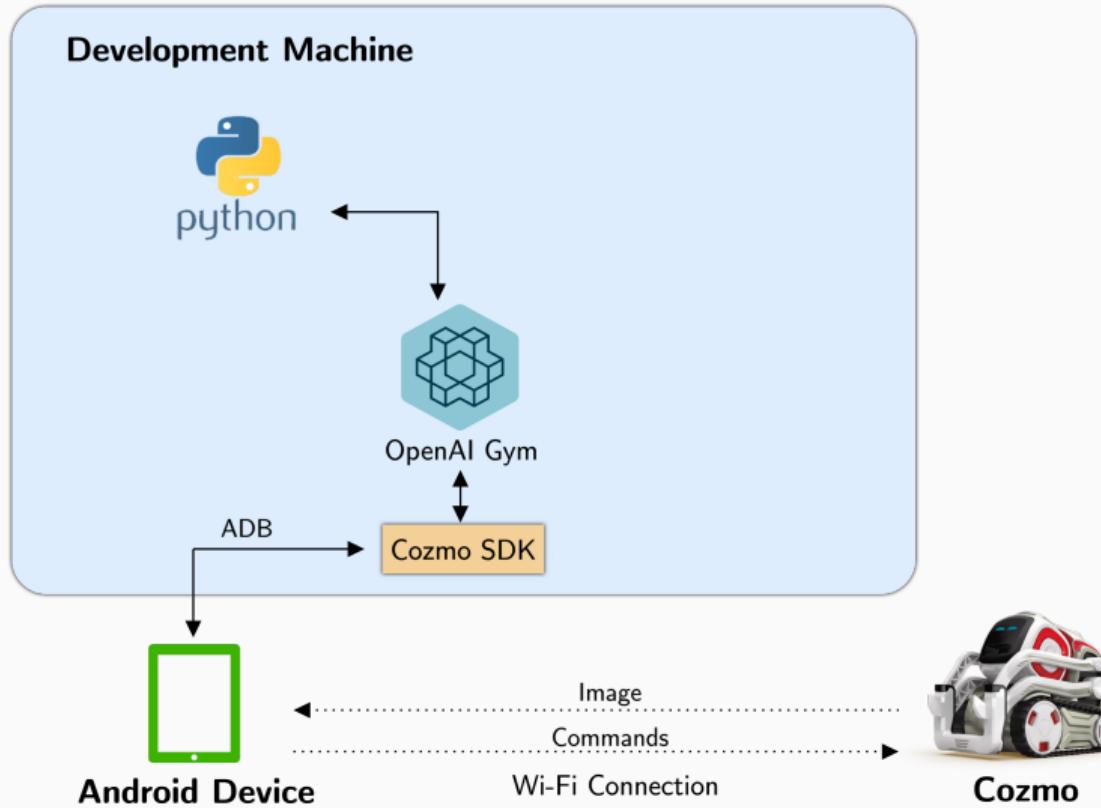


**Cozmo**

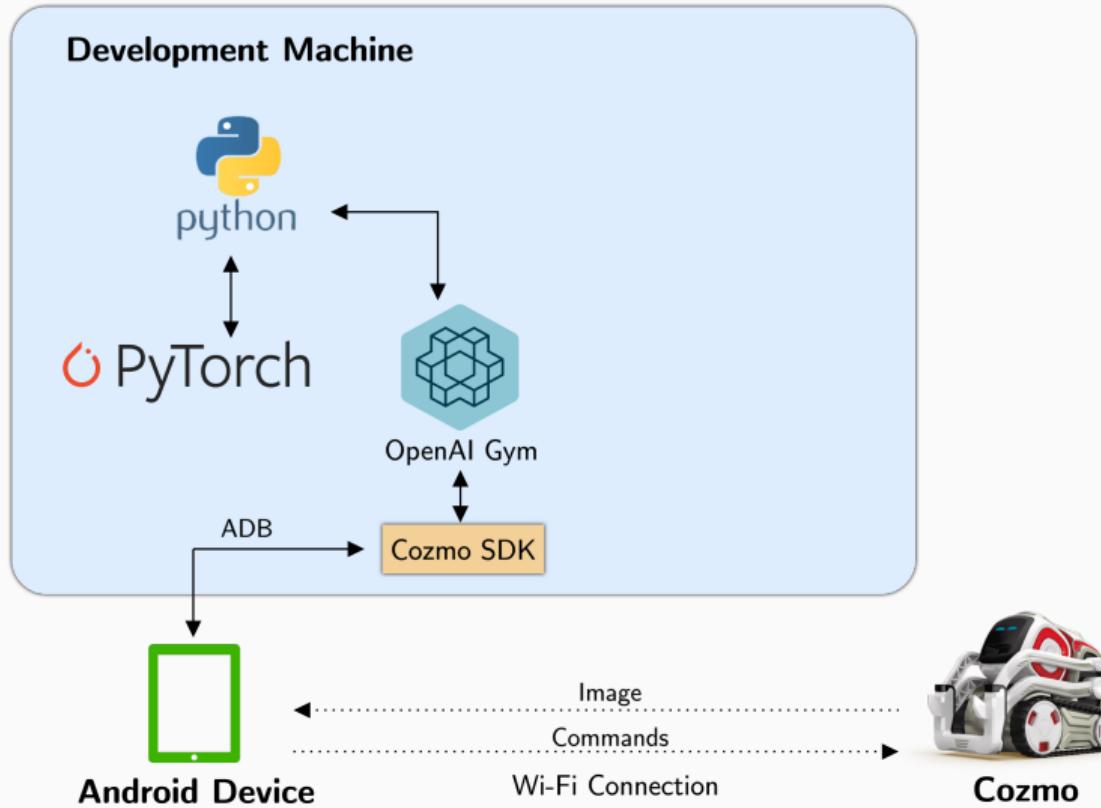
# Outline of the control system



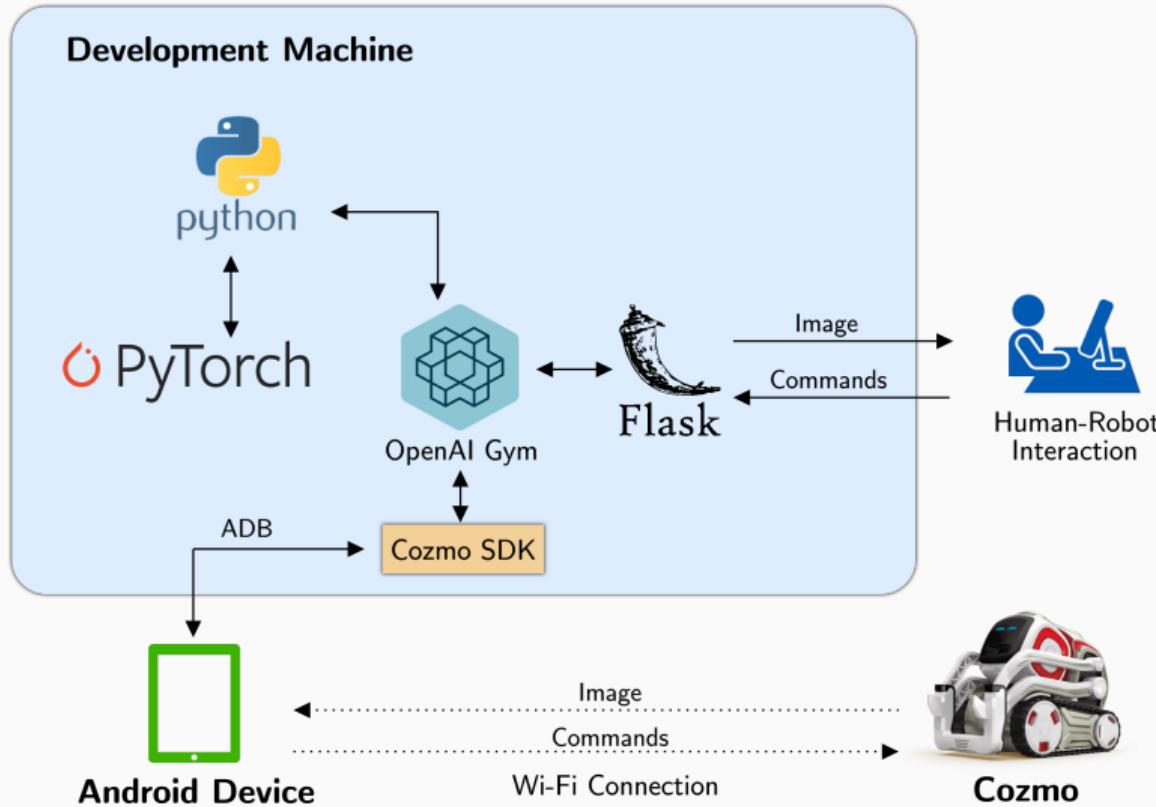
# Outline of the control system



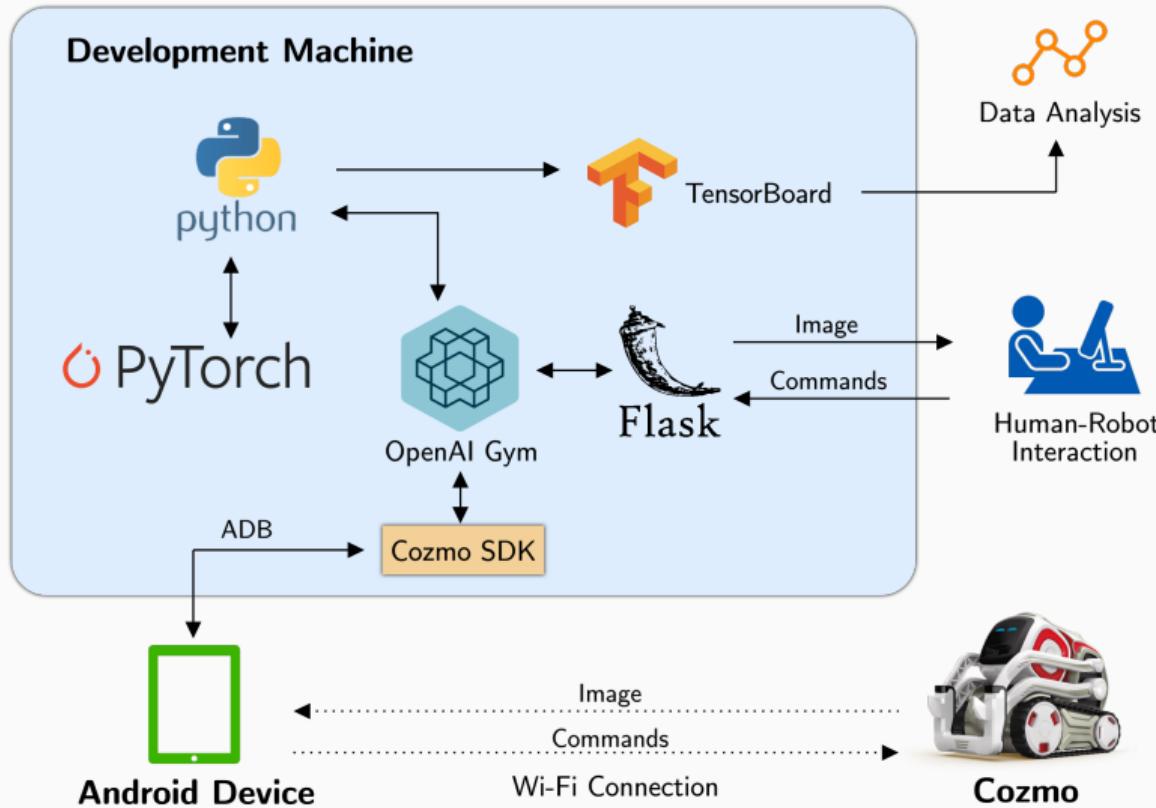
# Outline of the control system



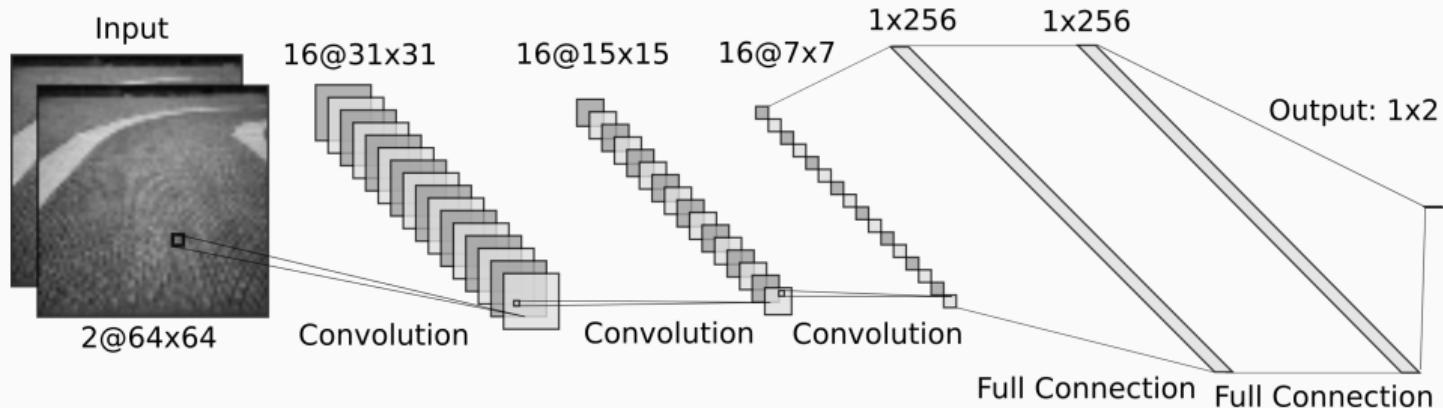
# Outline of the control system



# Outline of the control system

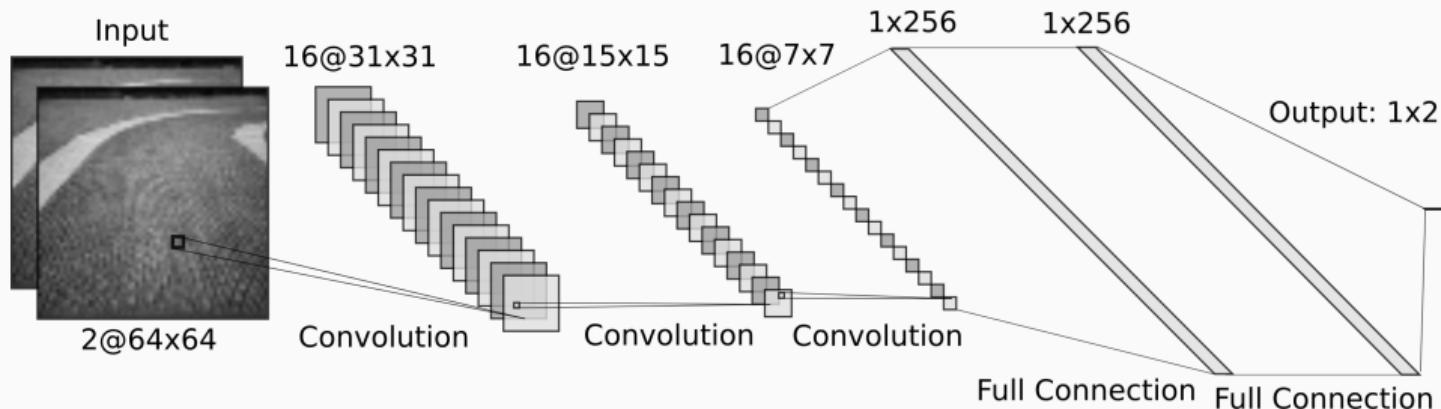


# Convolutional Neural Network Architecture



- 3 Convolutional Layers: 16 features ( $3 \times 3$ ), Stride 2, Padding 0

# Convolutional Neural Network Architecture



- 3 Convolutional Layers: 16 features ( $3 \times 3$ ), Stride 2, Padding 0
- 2 Fully Connected Layer with hidden size = 256

## System Features



- Backup/Restore feature:

## System Features



- Backup/Restore feature:
  - Episode restore

## System Features



- Backup/Restore feature:
  - Episode restore
  - Checkpoint restore

## System Features



- Backup/Restore feature:
  - Episode restore
  - Checkpoint restore
- [Playground Recording](#)

## Experimental methodology and results

---

# Pendulum-v0 Results

TODO

# CozmoDriver-v0 Results

TODO

## Conclusions and future work

---

## Conclusions



- Promising approach:

# Conclusions



- Promising approach:
  - Maximum reward reached: ~ 3.5 meters

# Conclusions



- Promising approach:
  - Maximum reward reached: ~ 3.5 meters
  - Visible improvements during experiments

# Conclusions



- Promising approach:
  - Maximum reward reached: ~ 3.5 meters
  - Visible improvements during experiments
- Unstable for concrete application:

# Conclusions



- Promising approach:
  - Maximum reward reached: ~ 3.5 meters
  - Visible improvements during experiments
- Unstable for concrete application:
  - Average reward reached: ~ 1 meter

# Conclusions



- Promising approach:
  - Maximum reward reached: ~ 3.5 meters
  - Visible improvements during experiments
- Unstable for concrete application:
  - Average reward reached: ~ 1 meter
  - Visible improvements during experiments

# Conclusions



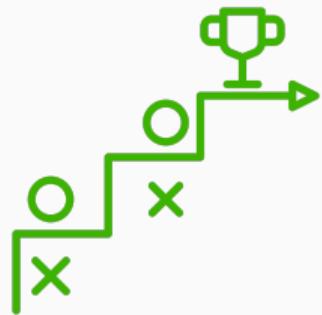
- Promising approach:
  - Maximum reward reached: ~ 3.5 meters
  - Visible improvements during experiments
- Unstable for concrete application:
  - Average reward reached: ~ 1 meter
  - Visible improvements during experiments
  - It needs time to improve

## Future Work

- Improving Sensors

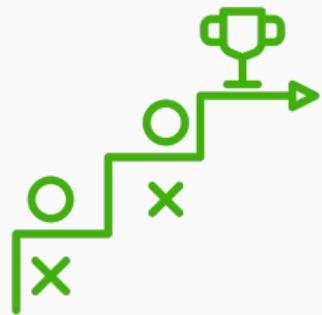


## Future Work



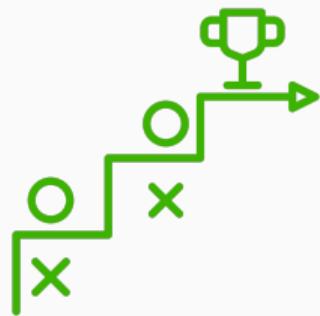
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)

## Future Work



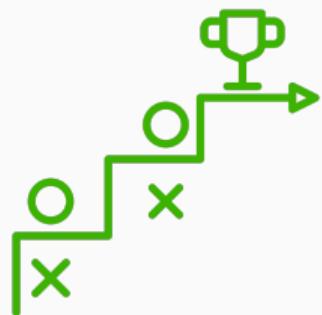
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector

# Future Work



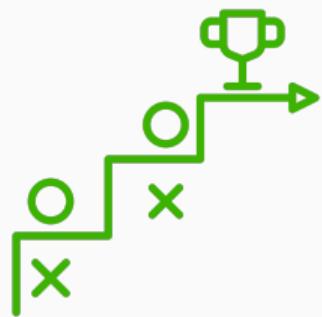
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector
- Feature Extraction

## Future Work



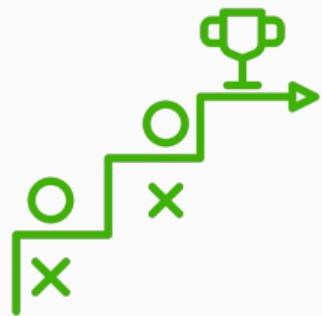
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector
- Feature Extraction
  - Variational Auto-Encoder (VAE)

## Future Work



- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector
- Feature Extraction
  - Variational Auto-Encoder (VAE)
- Data fusion

## Future Work



- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector
- Feature Extraction
  - Variational Auto-Encoder (VAE)
- Data fusion
- Model-based approach



Thank you!

## References

---

-  Charafeddine, Mohamad. *Reinforcement Learning in the Wild and Lessons Learned*. 2018. URL: <https://link.medium.com/SRUZZ24ITx4> (visited on 10/26/2018).
-  GovTech: Government Technology. *Autonomous Vehicles: Coming to a Road Near You*. 2018. URL:  
<https://www.govtech.com/transportation/Autonomous-Vehicles-Coming-to-a-Road-Near-You.html>.
-  Haarnoja, Tuomas et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).

## References ii

-  Lillicrap, Timothy P et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).
-  Pavone, Marco. *Veicoli a guida autonoma: a che punto siamo e cosa ci aspetta?* Festival della Tecnologia Conference. 2019.
-  Sutton, Richard S and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

## Appendix - Background

---

# Components of the Agent

- **Policy:** agent's behaviour function

Deterministic:  $\pi(s) = a$

Stochastic:  $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

- **Value Function:** policy evaluation function

State Value:  $V^\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^k r_t | s_0 = s, \pi \right]$

Action Value:  $Q^\pi(s, a) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^k r_t | s_0 = s, a_0 = a, \pi \right]$

- **Model:** agent's representation of the environment

# Categorizing Reinforcement Learning agents

- Value Based
  - No Policy (implicit)
  - Value Function
- Policy Based
  - Policy
  - No value function
- Actor Critic
  - Policy
  - Value function
- Model Free
  - Policy and/or value function
  - No Model
- Model Based
  - Policy and/or value function
  - Model

# Deep Deterministic Policy Gradient (DDPG) - Neural Networks

It uses **Target Networks** to minimise the instability MSBE loss

2 Local Neural Networks:

- Actor  $\pi(s | \theta)$
- Critic  $Q(s, a | \phi)$

2 Target Neural Networks:

- Actor  $\pi'(s | \bar{\theta})$
- Critic  $Q'(s, a | \bar{\phi})$

## Deep Deterministic Policy Gradient (DDPG) - Learning Equations

$$\begin{aligned} L(\phi) &= \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E} [(Q(s_t, a_t | \phi) - y_t)^2] \\ y_t &= r(s_t, a_t) + \gamma(1 - d_t) Q'(s_{t+1}, \pi'(s_t + 1 | \bar{\theta}) | \bar{\phi}) \end{aligned} \tag{1}$$

---

Lillicrap et al., “Continuous control with deep reinforcement learning”.