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 — EURECOM, France

Prof. Elena Baralis — Politecnico di Torino, Italy

March 17, 2020

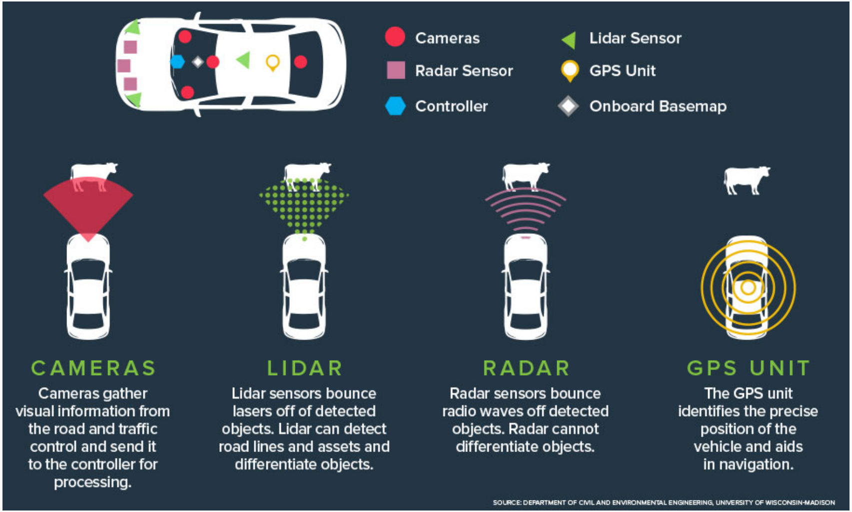
This work of this thesis was developed at EURECOM (Sophia Antipolis, France)
in collaboration with

Prof. Pietro Michiardi (EURECOM)
Prof. Elena Baralis (Politecnico di Torino)

Table of contents

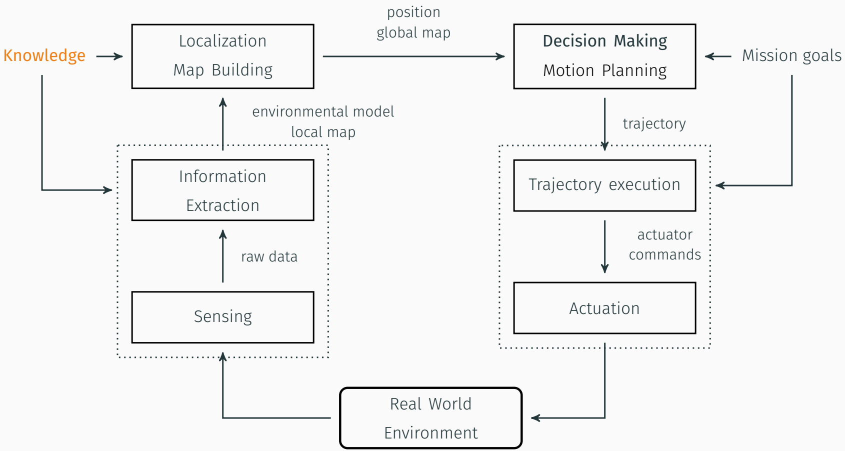
Background

State-of-the-art Autonomous Driving Systems



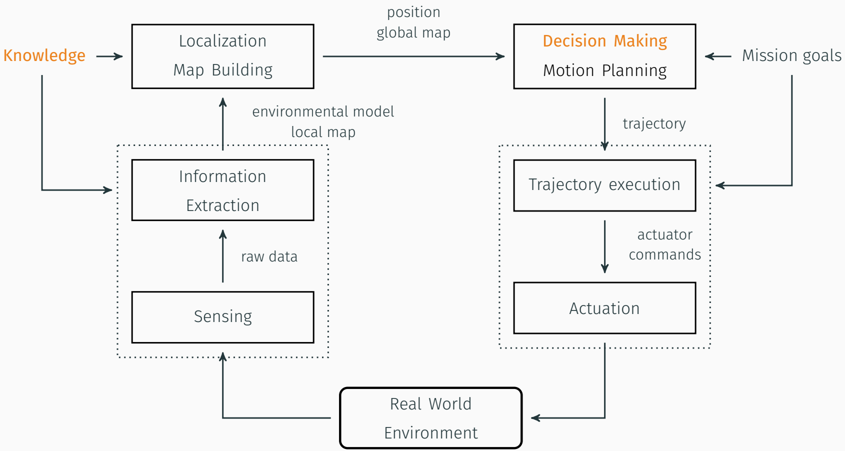
GovTech: Government Technology, *Autonomous Vehicles: Coming to a Road Near You*.

State-of-the-art Autonomous Driving Systems



Pavone, *Veicoli a guida autonoma: a che punto siamo e cosa ci aspetta?*

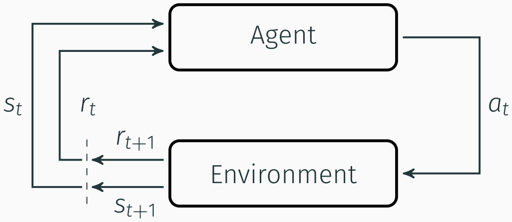
State-of-the-art Autonomous Driving Systems



Pavone, *Veicoli a guida autonoma: a che punto siamo e cosa ci aspetta?*

Reinforcement Learning

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

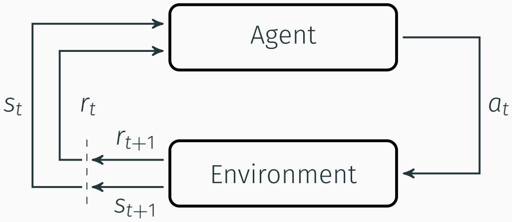


Sutton and Barto, *Reinforcement learning: An introduction*.

Reinforcement Learning

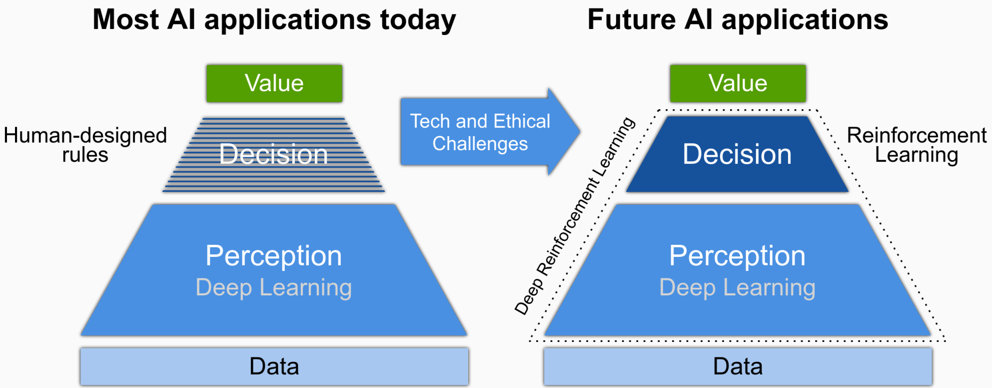
Problems involving an **agent** interacting with an **environment** , which provides numeric **reward signals**.

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



Sutton and Barto, *Reinforcement learning: An introduction*.

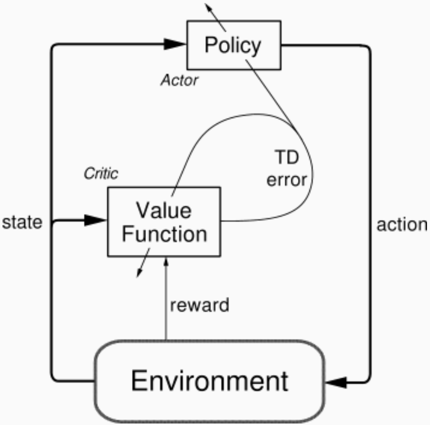
From Data to Value



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

Algorithms implemented

- **Model-Free**
- **Off-Policy** with **Experience Replay Memory** of $(s_t, a_t, r_t, s_{t+1}, d_t)$ tuples
- **Continuous** Action space
- **Actor-Critic** approach



Sutton and Barto, *Reinforcement learning: An introduction*.

Deep Deterministic Policy Gradient (DDPG)

DDPG learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

- **Policy**: Deterministic

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

Deep Deterministic Policy Gradient (DDPG)

DDPG learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

- **Policy**: Deterministic
- **Exploration**:

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

Deep Deterministic Policy Gradient (DDPG)

DDPG learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

- **Policy**: Deterministic
- **Exploration**:
  - **Ornstein–Uhlenbeck** process noise

Lillicrap et al., "Continuous control with deep reinforcement learning".

Deep Deterministic Policy Gradient (DDPG)

DDPG learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

- **Policy**: Deterministic
- **Exploration**:
  - **Ornstein–Uhlenbeck** process noise
  - Noise regulation with ϵ-**decay function**

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

Deep Deterministic Policy Gradient (DDPG)

DDPG learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

- **Policy**: Deterministic
- **Exploration**:
  - **Ornstein–Uhlenbeck** process noise
  - Noise regulation with ϵ-**decay function**

Needs accurate hyper-parameters fine-tuning

Lillicrap et al., "Continuous control with deep reinforcement learning".

Soft Actor-Critic (SAC)

SAC learns a policy and two Q-functions. It exploits **entropy regularization**.

- **Policy**: Stochastic

Haarnoja et al., “Soft actor-critic algorithms and applications”.

Soft Actor-Critic (SAC)

SAC learns a policy and two Q-functions. It exploits **entropy regularization**.

- **Policy**: Stochastic
- **Exploration**:

Haarnoja et al., "Soft actor-critic algorithms and applications".

Soft Actor-Critic (SAC)

SAC learns a policy and two Q-functions. It exploits **entropy regularization**.

- **Policy**: Stochastic
- **Exploration**:
  - **Temperature** Parameter

Haarnoja et al., "Soft actor-critic algorithms and applications".

Soft Actor-Critic (SAC)

SAC learns a policy and two Q-functions. It exploits **entropy regularization**.

- **Policy**: Stochastic
- **Exploration**:
    - **Temperature** Parameter
    - **Auto-tuning**

Haarnoja et al., “Soft actor-critic algorithms and applications”.

Soft Actor-Critic (SAC)

SAC learns a policy and two Q-functions. It exploits **entropy regularization**.

- **Policy**: Stochastic
- **Exploration**:
  - **Temperature** Parameter
  - **Auto-tuning**

Suitable for Real-World Experiments

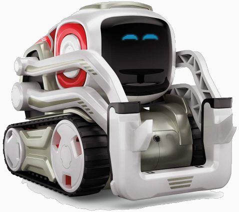
Haarnoja et al., "Soft actor-critic algorithms and applications".

Main Goals

1. Implementation of Reinforcement Learning algorithms
   - Preliminary experiments on a simplified environment
2. Building a **control system** binding every technology used.
   - Formalise the problem as MDP
3. **Real World** Reinforcement Learning experiments analysis.
   - **No model** of the environment.
   - **No preliminary simulation** to tune hyper-parameters

Design of the control system

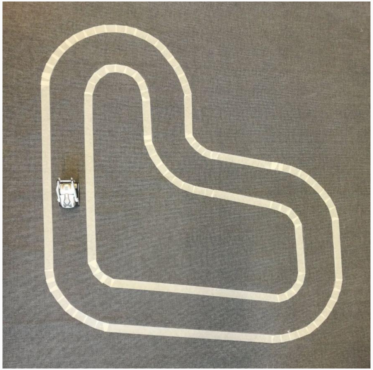
Anki Cozmo - Not just a toy robot



Why Anki Cozmo?

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

Track Design



Features:

- Low-reflections
- Scaled Reality
- Reproducible

MDP Formalisation - Observation

Configuration:

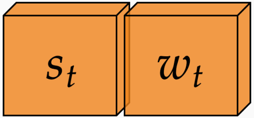
- Gray-scale image
- Frame Rate: $\sim 15fps$
- Raw image size: $64{\times}64$ pixels

MDP Formalisation - Observation

Configuration:

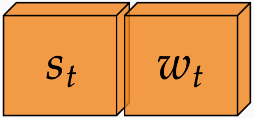
- **Gray-scale image**
- **Frame Rate**: $\sim 15fps$
- **Raw image size**: 64×64 pixels
- **Stack size**: 2

MDP Formalisation - Actions



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

MDP Formalisation - Actions

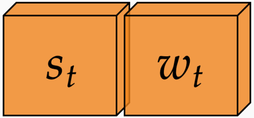


$s_t \in \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$ $\quad$ $w_t \in \{x \in \mathbb{R} \mid -1 \leq x \leq 1\}$

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



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

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

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

Desired Speed: s_t [mm/s]

MDP Formalisation - Reward

Distance Covered

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

Desired Speed: s_t [mm/s]

$$R_t = s_t \cdot T_t$$

if t is terminal $\rightarrow R_t = 0$

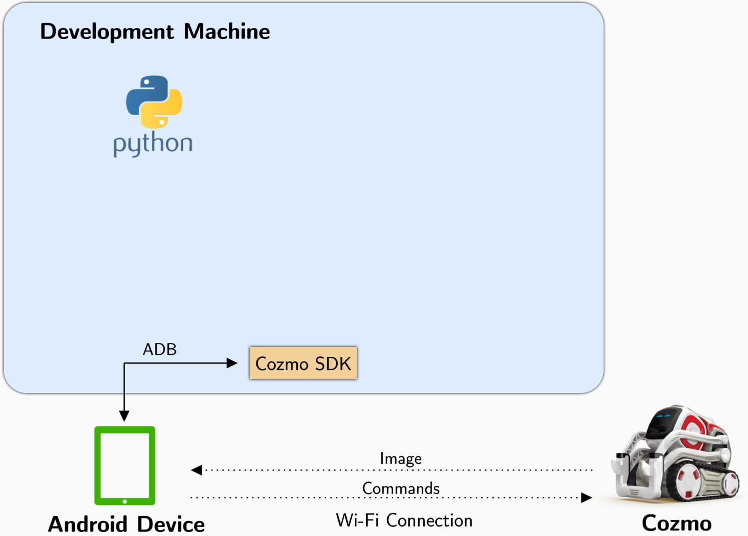
Outline of the control system

Development Machine

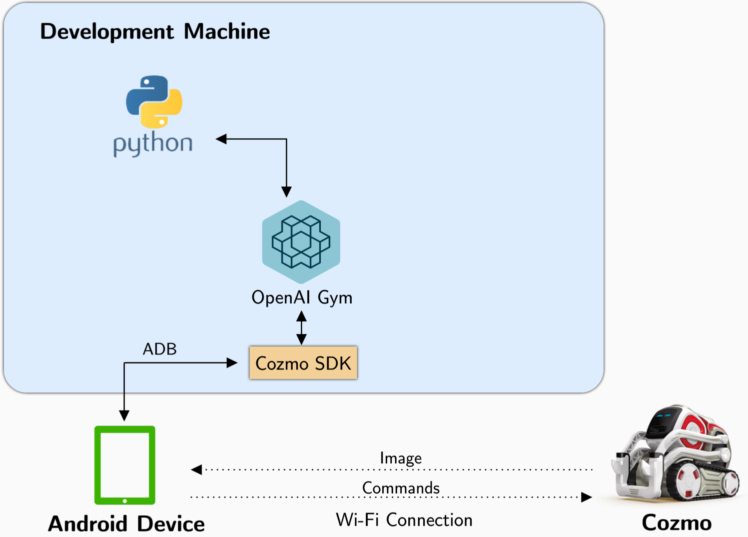
python

Cozmo

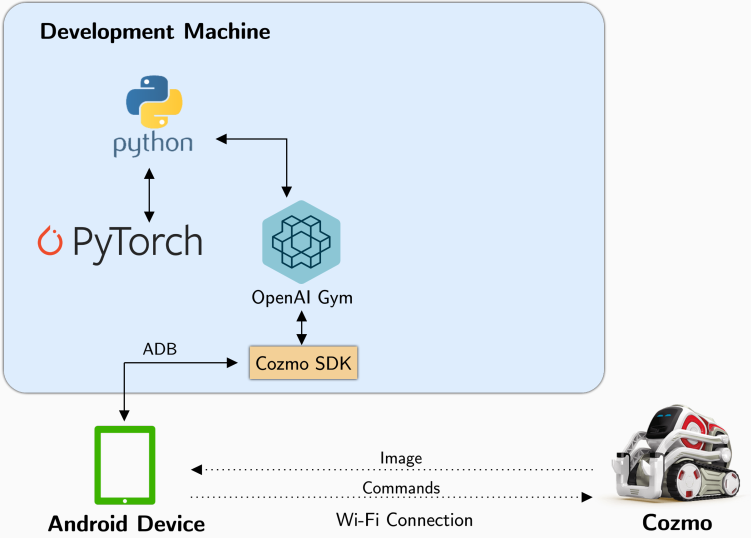
Outline of the control system



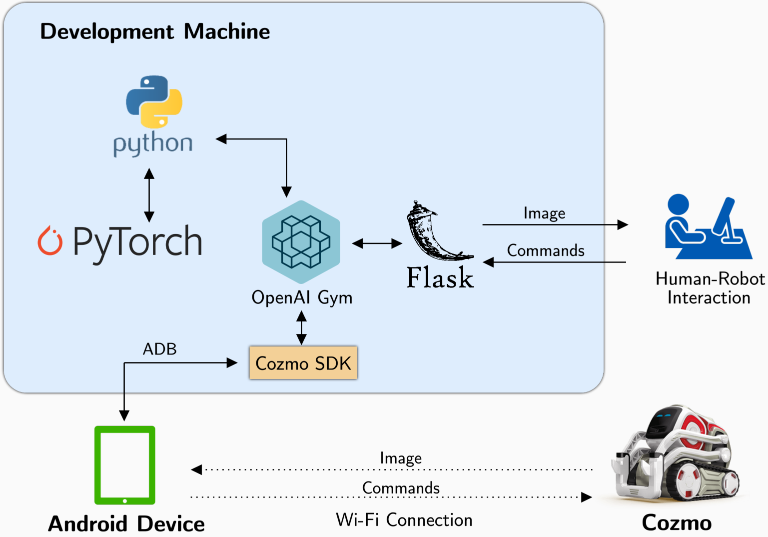
Outline of the control system



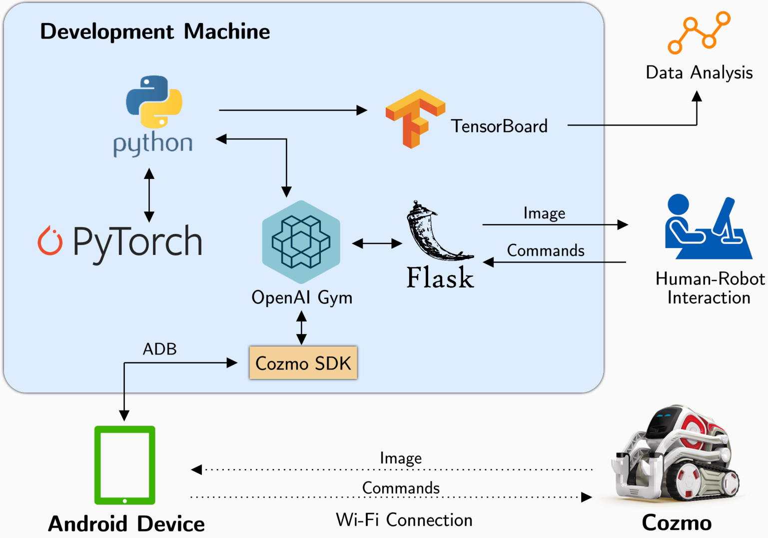
Outline of the control system



Outline of the control system



Outline of the control system



System Features

- Backup/Restore feature:

System Features

- Backup/Restore feature:
  - Episode restore

System Features

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

System Features

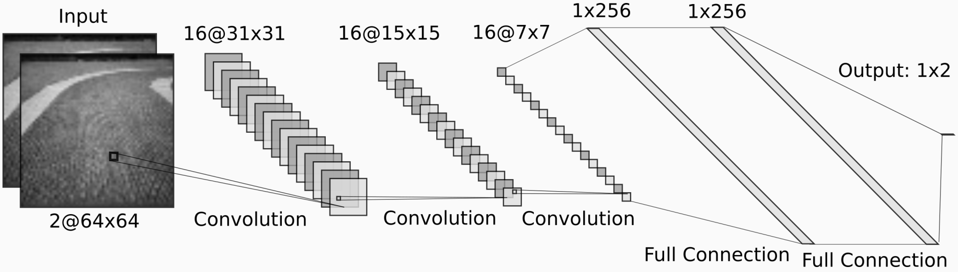
- Backup/Restore feature:
  - Episode restore
  - Checkpoint restore
- Playground Recording

Experimental methodology and results

Hyper-parameters used

TODO

Convolutional Neural Network Architecture



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

Pendulum-v0 DDPG Results

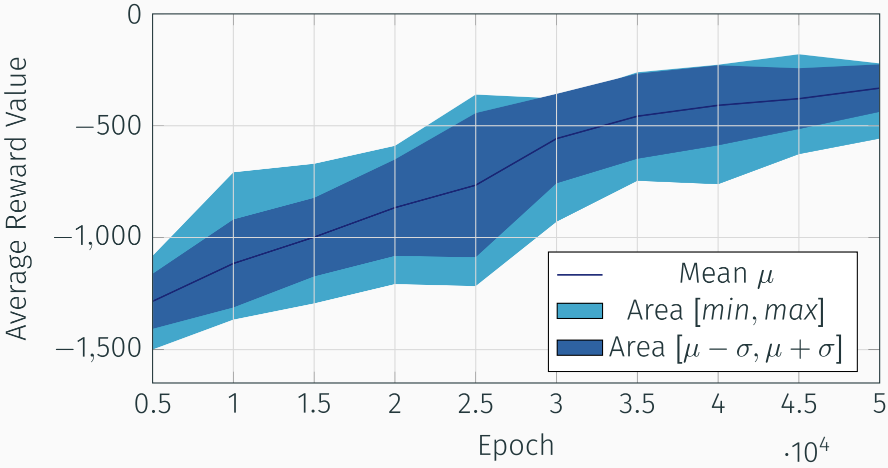


Figure 1: DDPG Pendulum-v0 Test Average Reward Plot.

Pendulum-v0 SAC Results

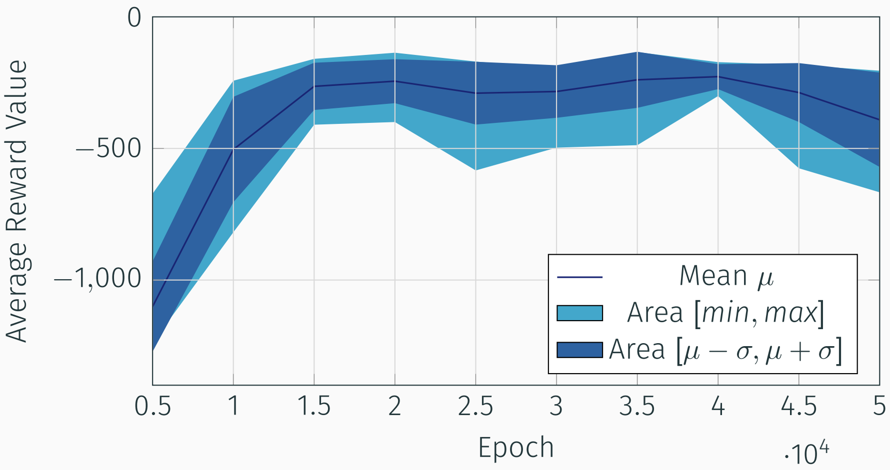


Figure 2: SAC Pendulum-v0 Test Average Reward Plot.

CozmoDriver-v0 SAC Training

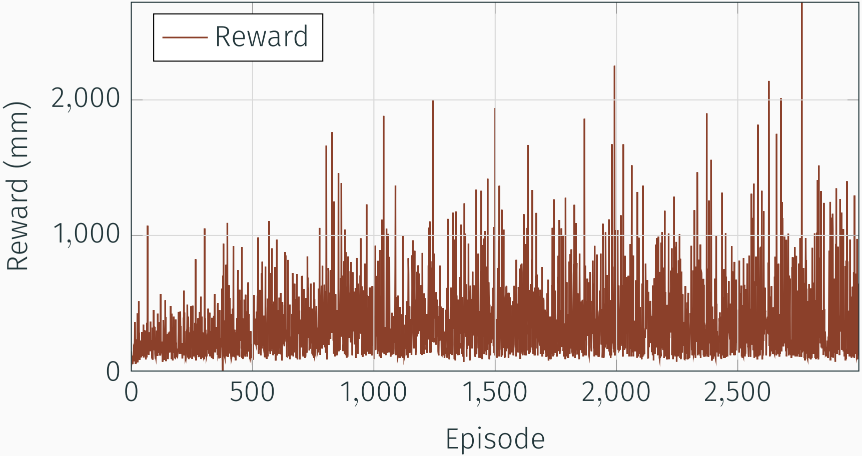


Figure 3: SAC CozmoDriver-v0 Reward Plot.

CozmoDriver-v0 SAC Training 100 average

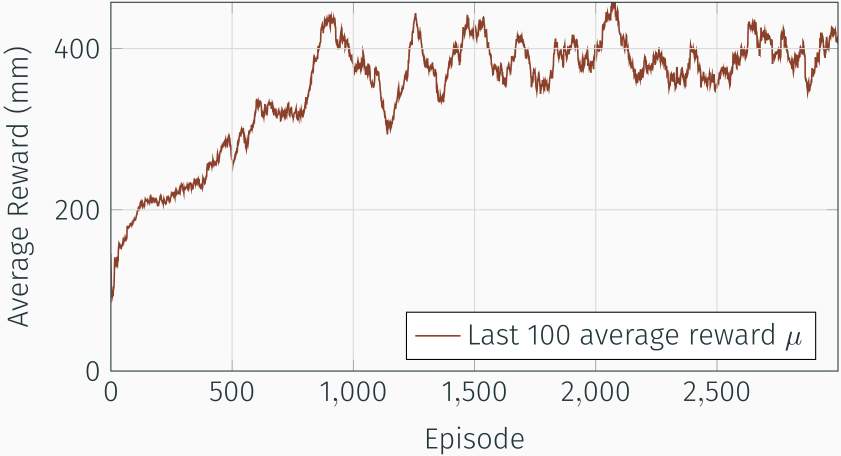


Figure 4: SAC CozmoDriver-v0 Last 100 Episode Average Reward Plot.

CozmoDriver-v0 SAC Temperature

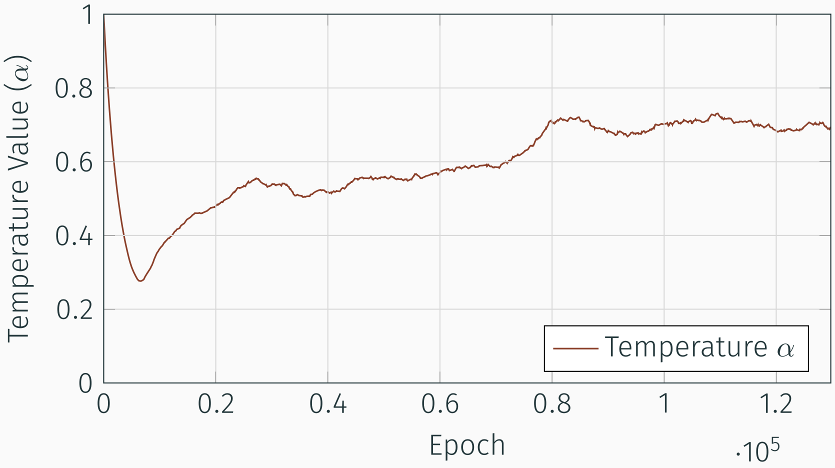


Figure 5: SAC Pendulum-v0 auto-tuned temperature.

CozmoDriver-v0 SAC Test

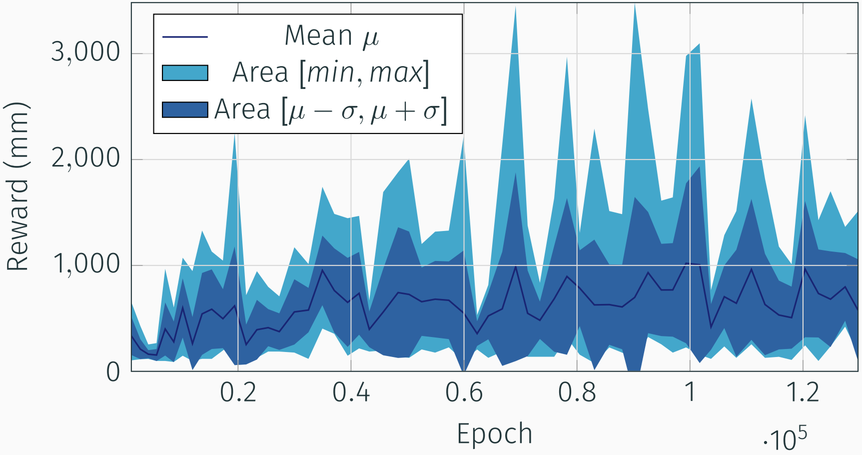


Figure 6: SAC CozmoDriver-v0 Test Reward Plot.

CozmoDriver-v0 SAC Test Mean

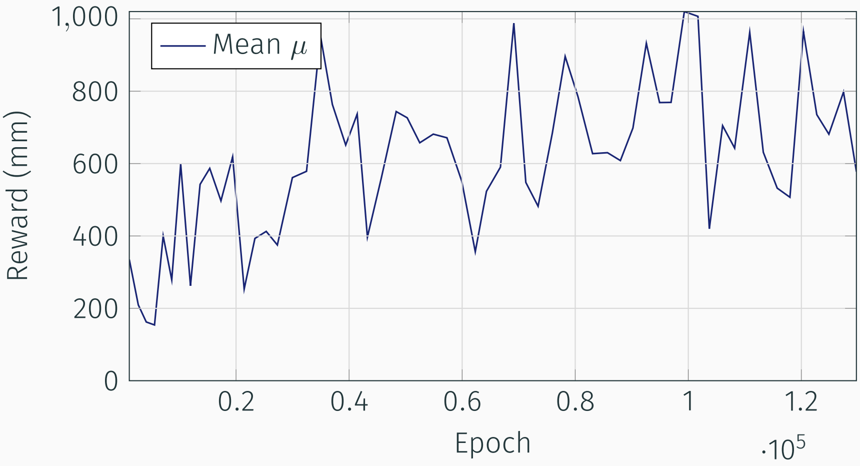


Figure 7: SAC CozmoDriver-v0 Test Average Reward Plot.

Episode Showcase - 1

Episode Showcase - 2

Conclusions and future work

Conclusions

- Promising approach:

Conclusions

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

Conclusions

- Promising approach:
  - **Maximum reward reached**: $\sim$ 3.5 meters
  - Visible improvements during experiments

Conclusions

- Promising approach:
  - **Maximum reward reached**: $\sim$ 3.5 meters
  - Visible improvements during experiments
- Unstable for concrete application:

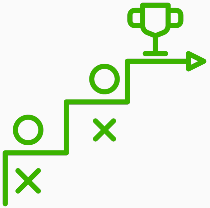
Conclusions

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

Conclusions

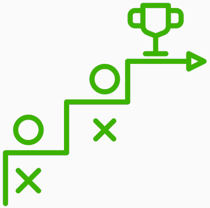
- Promising approach:
  - **Maximum reward reached**: $\sim$ 3.5 meters
  - Visible improvements during experiments
- Unstable for concrete application:
  - **Average reward reached**: $\sim$ 1 meter
  - It needs time to improve

Future Work



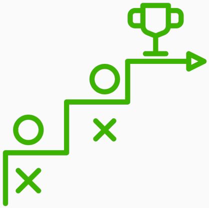
- Alternative Reward function analysis

Future Work



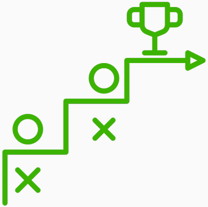
- Alternative Reward function analysis
  - Penalise terminal high speed

Future Work



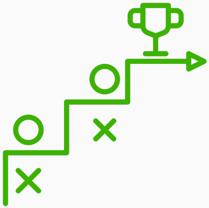
- Alternative Reward function analysis
  - Penalise terminal high speed
- Improving Sensors

Future Work



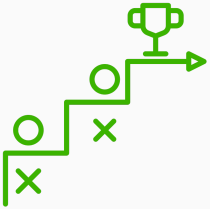
- Alternative Reward function analysis
  - Penalise terminal high speed
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)

Future Work



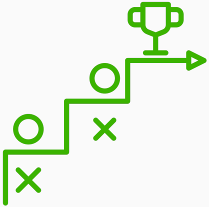
- Alternative Reward function analysis
  - Penalise terminal high speed
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector

Future Work



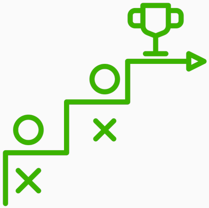
- Alternative Reward function analysis
  - Penalise terminal high speed
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector
- Feature Extraction

Future Work



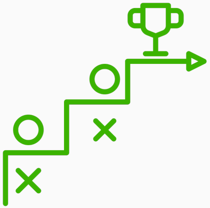
- Alternative Reward function analysis
  - Penalise terminal high speed
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector
- Feature Extraction
  - Variational Auto-Encoder (VAE)

Future Work



- Alternative Reward function analysis
    - Penalise terminal high speed
- Improving Sensors
    - Custom RC car (e.g. Donkey Car)
    - Anki Vector
- Feature Extraction
    - Variational Auto-Encoder (VAE)
- Data fusion

Future Work



- Alternative Reward function analysis
  - Penalise terminal high speed
- Improving Sensors
  - Custom RC car (e.g. Donkey Car)
  - Anki Vector
- Feature Extraction
  - Variational Auto-Encoder (VAE)
- Data fusion
- Model-based approach

Questions?

Thank you!

References

- Charafeddine, Mohamad. *Reinforcement Learning in the Wild and Lessons Learned*. 2018. URL: `https://link.medium.com/SRUZ24Itx4` (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

$$\textbf{Deterministic}: \pi(s) = a$$

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

- **Value Function**: policy evaluation function

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

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

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

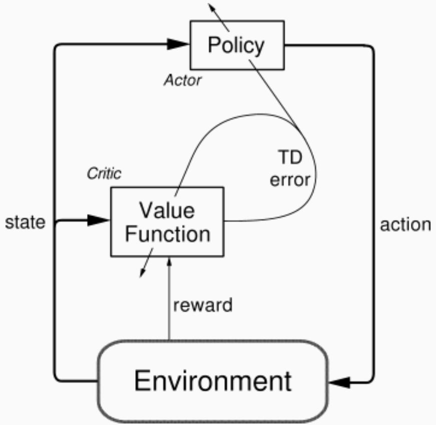
Model-Free Actor Critic methods

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).



Sutton and Barto, *Reinforcement learning: An introduction*.

Model-Free Actor Critic methods

$$V(s_t) \leftarrow V(s_t) + \alpha\Big(\underbrace{\overbrace{r_{t+1} + \gamma V(s_{t+1})}^{\text{TD target}} - V(s_t)}_{\text{TD error }(\delta_t)}\Big) \tag{1}$$

Sutton and Barto, *Reinforcement learning: An introduction*.

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

2 Target Neural Networks:

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

Deep Deterministic Policy Gradient (DDPG) - Learning Equations

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

Lillicrap et al., "Continuous control with deep reinforcement learning".