Deep Reinforcement Learning summary

P. Alexoudi, P.A. Tosidis, K. Triaridis, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 3.6.2

Artificial Intelligence & Information Analysis Lab

Reinforcement Learning (RL)

- **Introduction to RL**
- RL algorithms
- Deep RL. DQN
- Policy Gradient methods
- Actor Critic Methods
- Maze example
- Autonomous Driving and Piloting
- Current Research Challenges

Reinforcement Learning

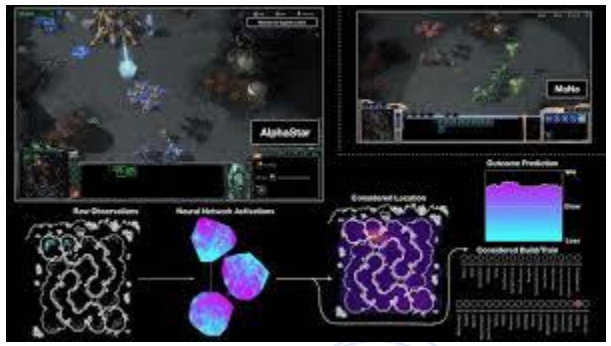
Reinforcement Learning (***RL***) is one of the main machine learning paradigms.

- It advocates ***learning through interaction***: An ***agent*** interacts with its environment, in order to maximize some form of ***cumulative rewards***.
- It maps situations to actions.

- ***Trial-and-error*** learning is its primary learning mechanism.

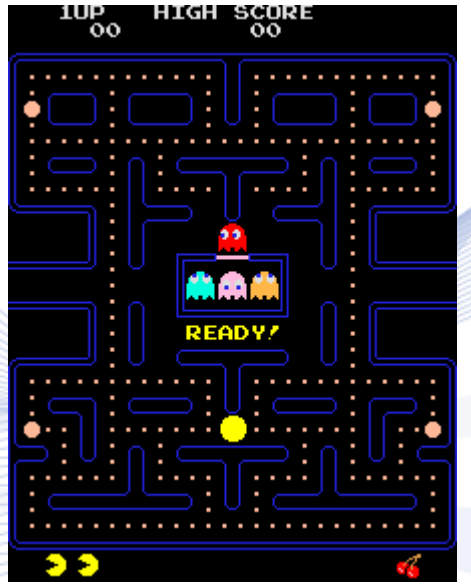
Examples

- Game playing agents.
- Self-driving vehicles.
- Robot cleaners.
- Stock exchange agents.



Reference: [DM]

Reference: [DM-GO]



Reference: [WIKI-PMAN]

Markov Decision Processes

- ***Finite Markov Decision Processes*** (***MDPs***) offer a method to frame the ***learning-from-interaction*** problem to achieve a goal.
- It employs actions to move from state to state, towards maximizing a reward function.

- The ***state*** $\mathcal{S}$, ***action*** $\mathcal{A}$ and ***rewards*** $\mathcal{R}$ sets have finite cardinality (a finite number of elements N_s, N_a, N_r, respectively).

Markov Decision Processes

- Markov processes are iterative, evolving over discrete time $n = 0,1,2, \ldots$.
- They satisfy the so-called ***Markov property***:

$$P(S_n = s_n | S_{n-1} = s_{n-1}, \ldots, S_0 = s_0) = P(S_n = s_n | S_{n-1} = s_{n-1}).$$

- S_n: ***random state variables***.
- $s_n \in \mathcal{S}$: states.

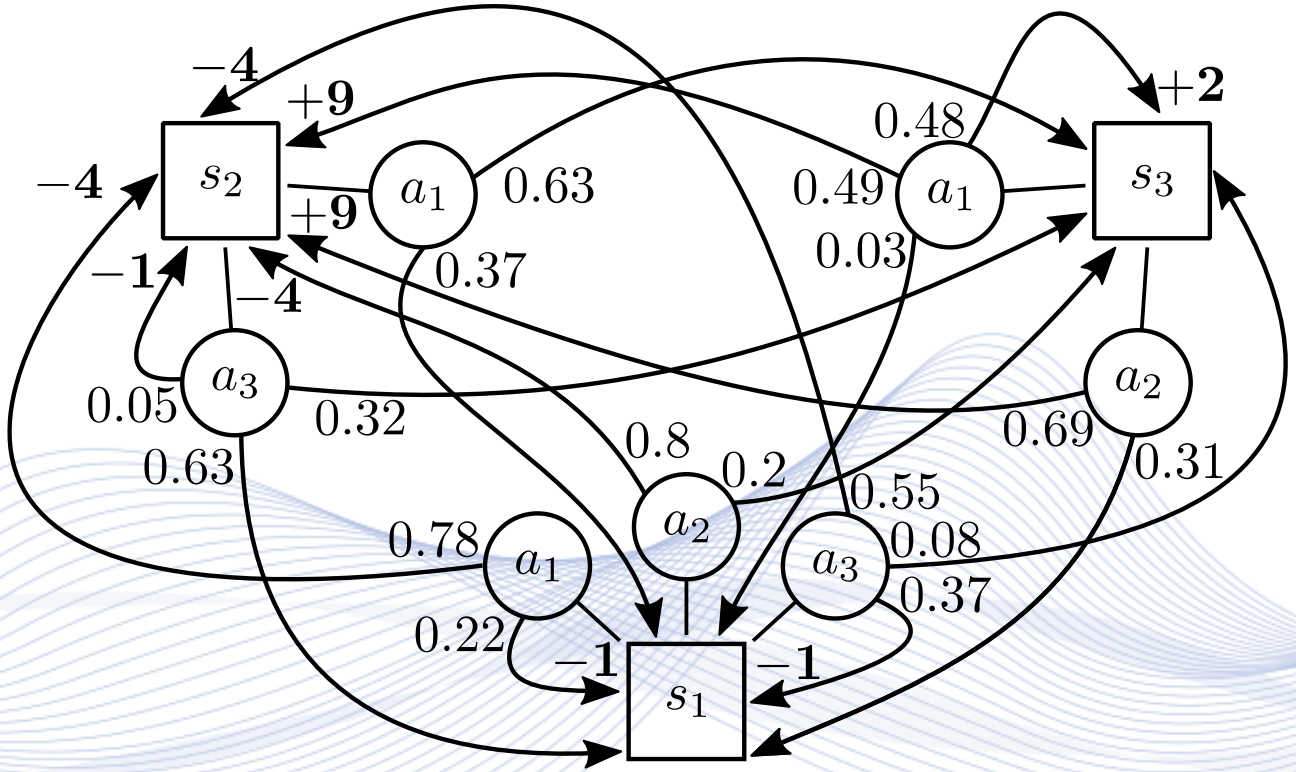
It basically means that a transition from one state to the next ***depends only on the current state*** and NOT on the past.

Finite Markov Decision Processes

Markov Processes are represented by a 4-tuple $(\mathcal{S}, \mathcal{A}, P_a, R_a)$:

- $\mathcal{S}$ is a finite set of states (***state space***).
- $\mathcal{A}$ is a finite set of actions (***action space***).
- $R_a(s, s')$ is the ***reward*** received after transitioning from state s to state s', due to action a.
- $P_a(s', r|s, a) = P(S_n = s', R_n = r|S_{n-1} = s, A_{n-1} = a)$ is the probability that taking action a in state s will lead to state s' in the next time instance, providing reward *r*.
- P_a is a ***probability function***: $\mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$.

Finite Markov Decision Processes



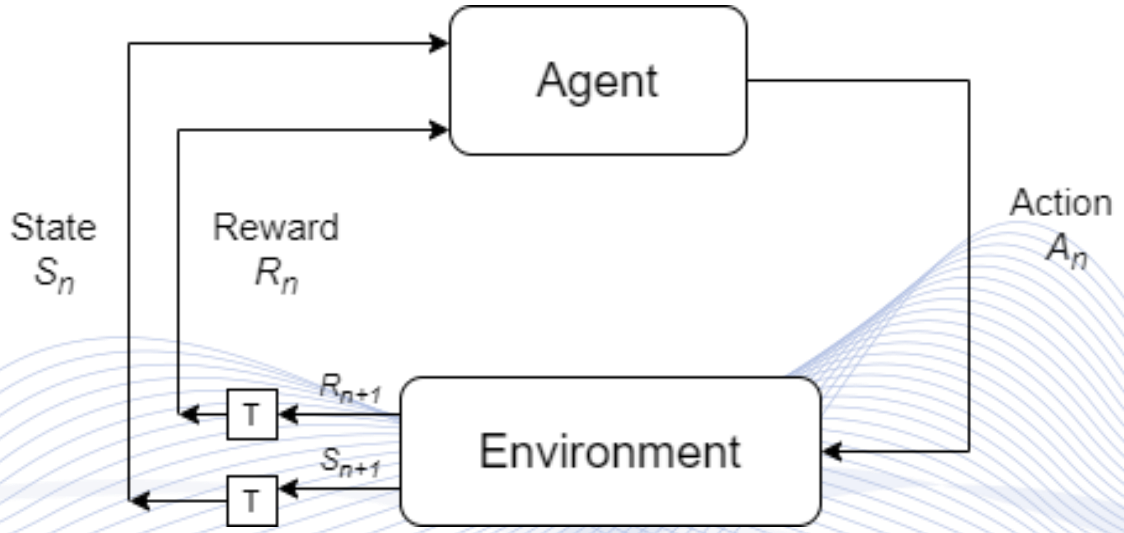
Markov process state transition diagram.

Elements of RL

The goal is to find a ***policy*** that ***maximizes a cumulative reward signal***.

- Reinforcement learning is a method to solve MDPs.
- An ***agent*** acts in a given environment. Examples:
  - A car driving agent acting on a car in a road.
- ***Environment*** is the world the agent interacts with.
- It provides the current state to the agent and a reward for her/his actions.
- The environment is typically stated in the form of a MDP.

Agent interaction

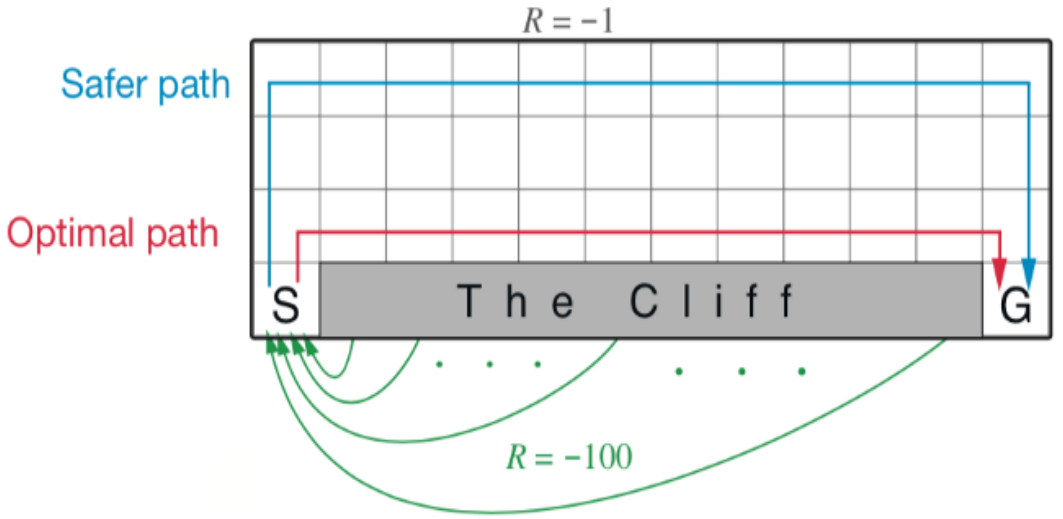


Environment-action interaction [SUT1998].

RL Example: Cliff Avoidance

Agent must move from point S to point G, while avoiding the cliff.

- State set: $4 \times 12 = 48$ blocks.
- Start/goal states: S, G.
- Actions: move right/left/up/down.
- Reward: $R = -1$ in every block, $R = -100$ in cliff blocks.

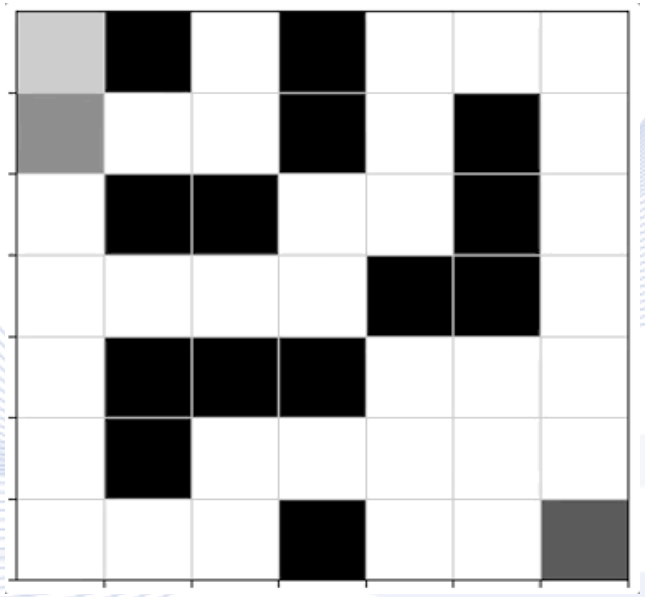


Reference: [SUT1998]

RL Example: Maze

A robot agent starts from a starting position and moves towards the maze exit:

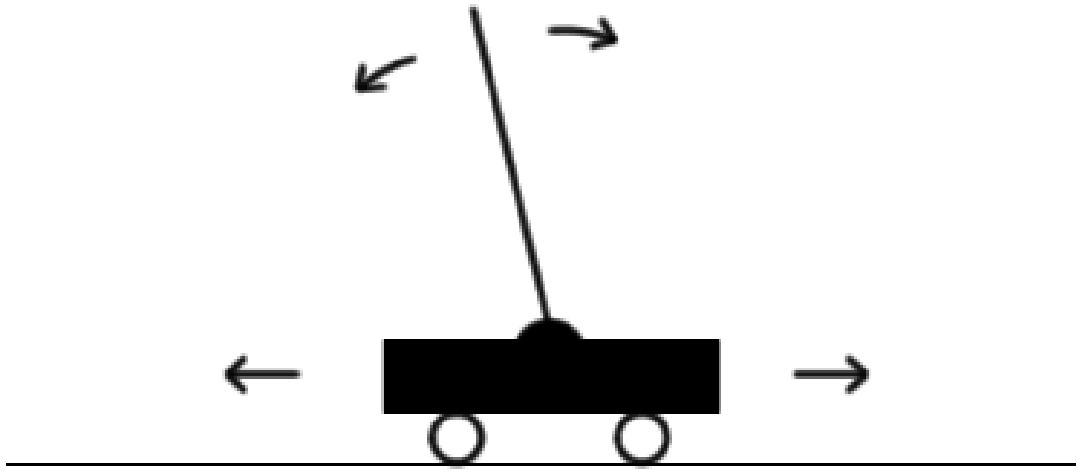
- Starting Point at (0, 0)
- Exit Point at (6, 6).



Trial and error

- ***Trial-and-error learning*** originates in Psychology and dates back to 1850s.
- It is a fundamental concept in reinforcement learning.
- The essential idea of trial and error learning is to reach a goal in an environment, ***by moving towards rewards and away from punishments***.
- ***Cart pole-balancing*** is one of the first successful applications of a trial-and-error learning task, without the use of complete knowledge.

Trial and error



Cart pole-balancing [POL].

Exploration vs Exploitation

- ***Exploration***: An agent explores, when, for every state it visits, it searches every action in $\mathcal{A}$.
- ***Exploitation***: An agent exploits, when it uses prior knowledge to search only a 'promising' subspace of $\mathcal{A}$.
- ***Exploration vs Exploitation*** is one of the biggest dilemmas an agent has to face:
  - An agent must look for the best actions. Thus, it has to explore.
  - In the meantime, it has to keep getting better. Thus, it has to exploit.

Reinforcement Learning (RL)

- Introduction to RL
- **RL algorithms**
- Deep RL. DQN
- Policy Gradient methods
- Actor Critic Methods
- Maze example
- Autonomous and Piloting
- Current Research Challenges

RL Algorithms

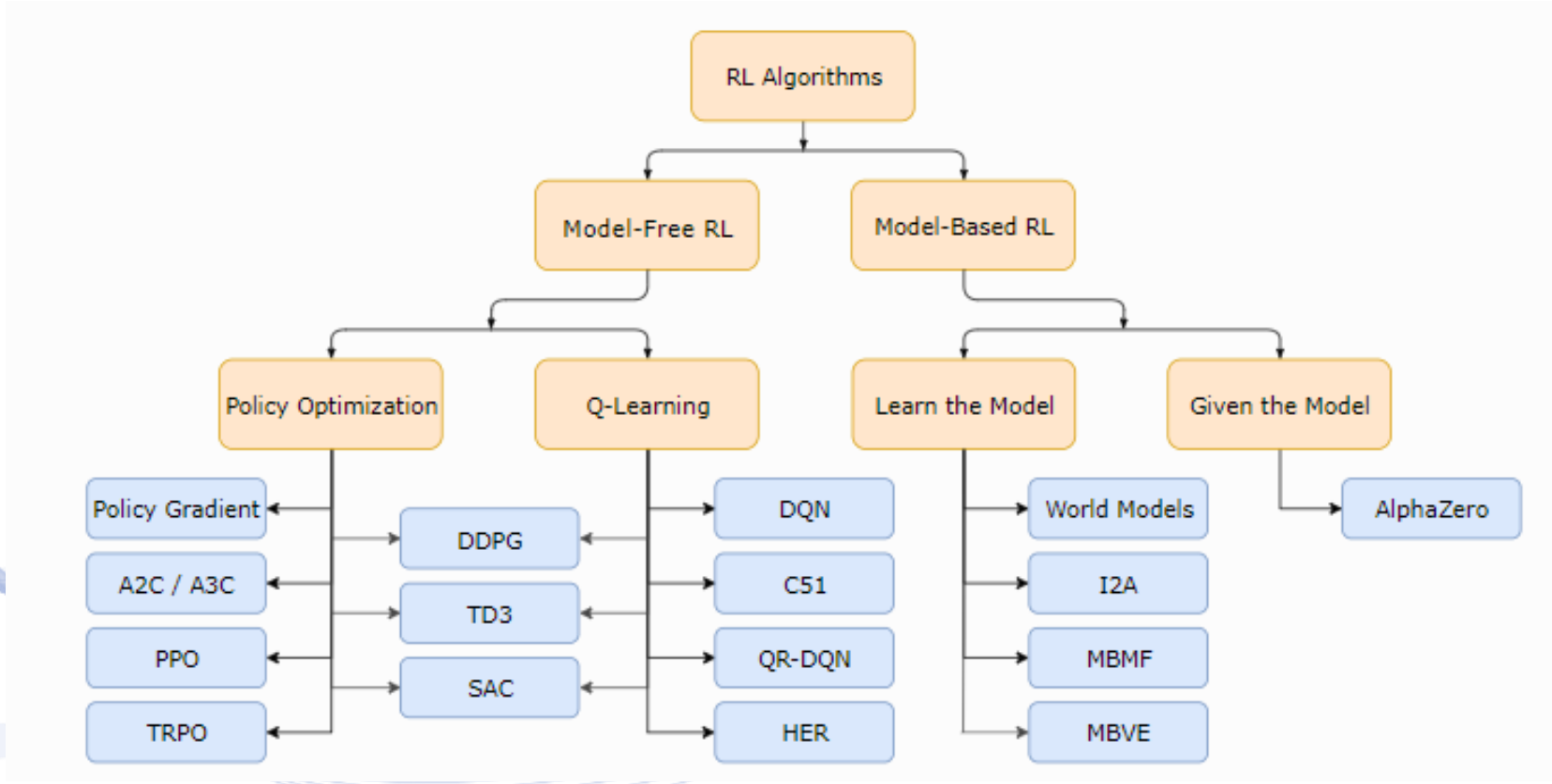
- ***Model-free RL algorithms***:
  - ***Action-value methods*** describe a value function that provides a prediction of how good a state, or a state-action pair, is.
  - ***Policy Gradient methods*** provide an immediate description of policy π.
- ***Model-based RL algorithms***:
  - Complete estimation of environment mechanics and a planning algorithm.

RL Algorithms

- ***On-policy methods*** aim to improve the same policy that is used to make decisions (the action-value function $Q_\pi(s, \alpha)$ is learned from actions taken using the current policy π).

- ***Off-policy methods*** improve a different policy, than the one used to generate the data (the action-value function $Q_\pi(s, \alpha)$ is learned from taking different actions than the ones the current policy π pointed, do not need the policy).

Taxonomy of RL Algorithms



Reference: [OPENAI-TAX]

RL Algorithms
Temporal-Difference

Temporal-Difference (***TD***) learning is a combination of MC and DP ideas.

- Unlike MC methods, environment dynamics model is not needed.
- TD, unlike MC methods, ***bootstraps*** (update of the estimates is part of other learned estimates):

$$V_\pi(S_n) \leftarrow V_\pi(S_n) + \alpha[R_{n+1} + \gamma V_\pi(S_{n+1}) - V_\pi(S_n)].$$

- It is not necessary to reach the end of an episode.

RL Algorithms
SARSA

State, Action, Reward, State, Action (***SARSA***) learning is on-policy TD Control. It employs Q-function:

$$Q_\pi(S_n, A_n) \leftarrow Q_\pi(S_n, A_n) + \alpha[R_{n+1} + \gamma Q_\pi(S_{n+1}, A_{n+1}) - Q_\pi(S_n, A_n)].$$

- If the agent visits all state–action pairs an infinite number of times, SARSA converges with probability 1 to an optimal policy and action-value function π_*, v_*.

RL Algorithms
Q-learning

Q-learning is Off-policy TD Control:

$$Q_\pi(S_n, A_n) \leftarrow Q_\pi(S_n, A_n) + \alpha \left[R_{n+1} + \gamma \max_a Q_\pi(S_{n+1}, a) - Q_\pi(S_n, A_n) \right].$$

- Q-learning is an off-policy algorithm:
  - ***only the best action is picked*** on S_{n+1} and not the one stated by the policy.

RL Algorithms
Q-learning

Double Q-learning is defined by:

$$Q_1(S_n, A_n) \leftarrow Q_1(S_n, A_n) + \alpha \left[R_{n+1} + \gamma Q_2(S_{n+1}, \arg\max_a Q_1(S_{n+1}, a)) - Q_1(S_n, A_n) \right].$$

- The expected reward is calculated from a ***different policy*** than the one being optimized.

Reinforcement Learning (RL)

- Introduction to RL
- RL algorithms
- **Deep RL. DQN**
- Policy Gradient methods
- Actor Critic Methods
- Maze example
- Autonomous and Piloting
- Current Research Challenges

Why Deep RL?

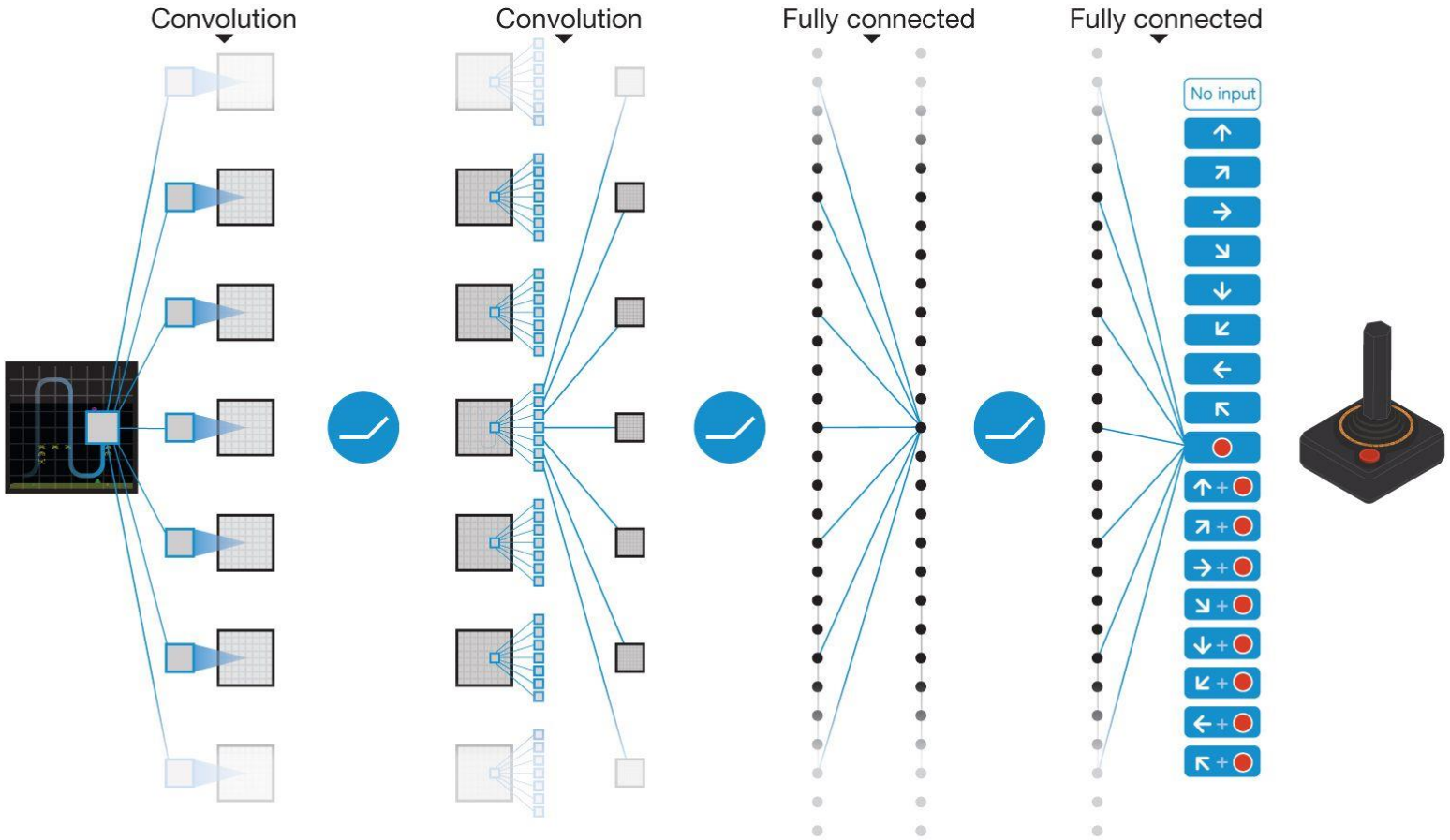
- In real life scenarios, state and action spaces $\mathcal{S}, \mathcal{A}$ are too large to be dealt with traditional methods.
- Look-up table storage, indexing and updating of all possible states, actions and their values is an unfeasible solution.
- ***Deep Learning*** RL approach:
  - ***Utilize a function approximator of the value function or Q-function***.
- ***Deep Neural Networks*** (***DNNs***) are the current SoA function approximators.

DQN

Deep reinforcement learning (***DRL***) combines Deep Learning (DNNs) and Reinforcement Learning principles (e.g., Q-learning) to produce efficient algorithms.

- ***Deep Q-Network*** (***DQN***) combines Reinforcement Learning and DNNs.

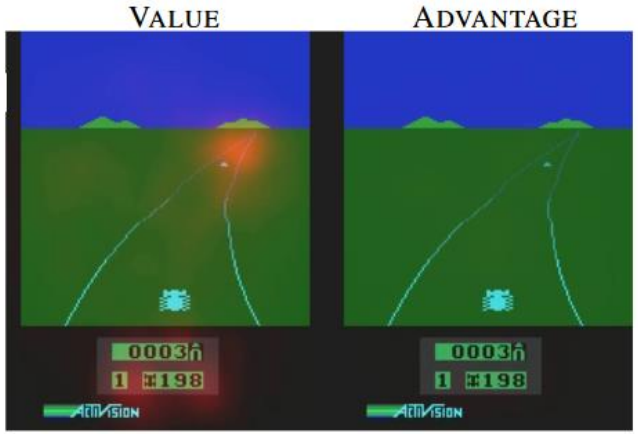
DQN



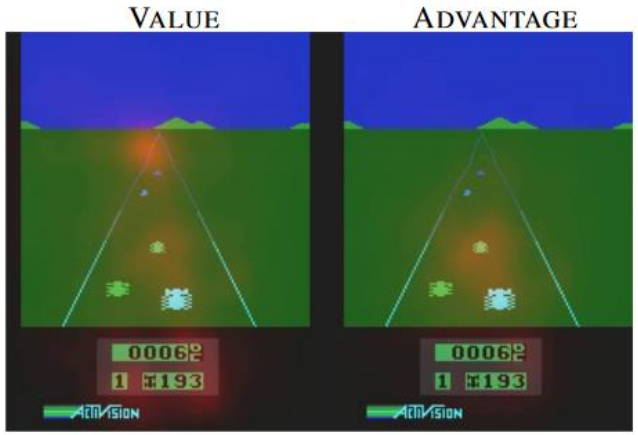
Graphic representation of CNN [MNI2015].

DQN Extensions

Focus on 2 things:
- The horizon where new cars appear
- On the score



No car in front, **does not pay much attention because action choice making is not relevant**



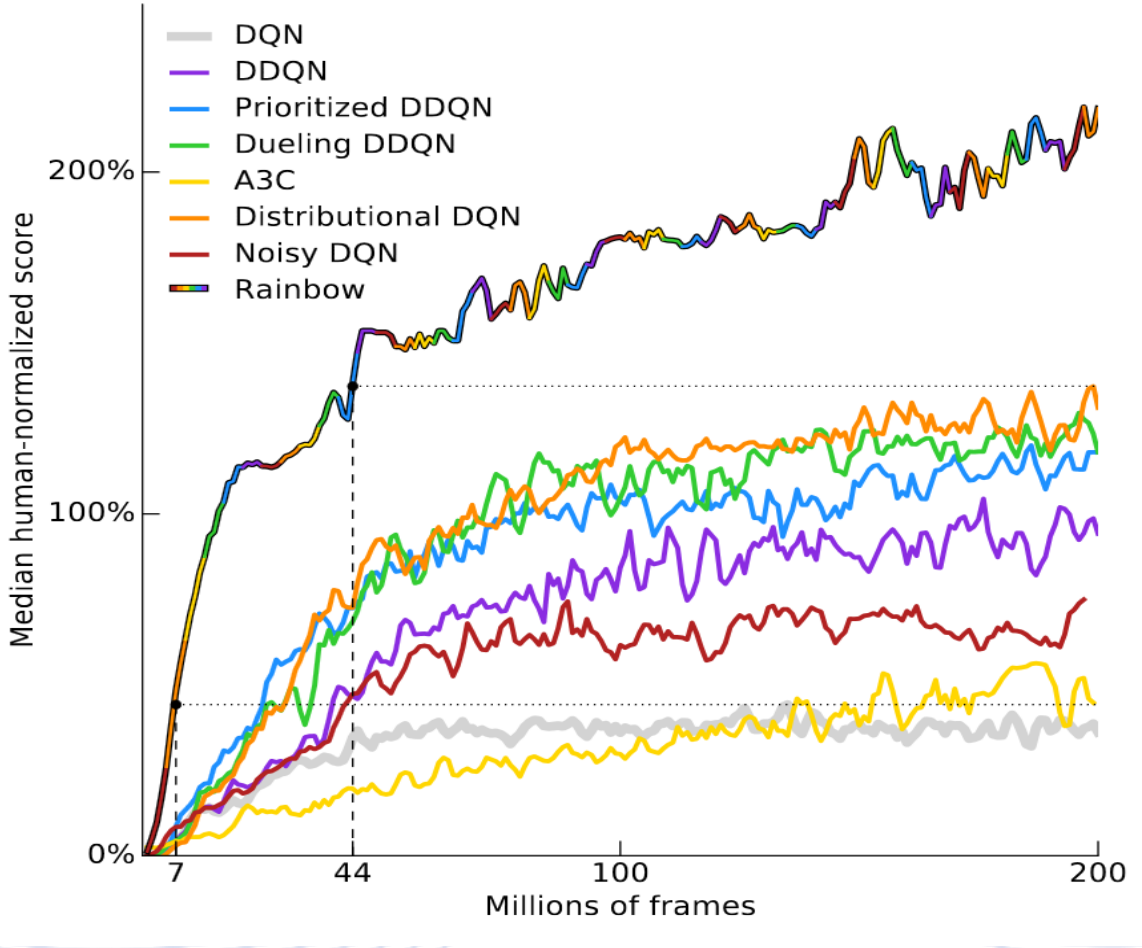
Pays attention to the front car, in this case **choice making is crucial to survive**

Car driving agent: Value and advantage saliency maps in two different time instances [WAN2016].

Rainbow, a combination of extensions

- Rainbow is a combination of all the aforementioned components.
- It has a remarkable performance, since each of its components address a specific RL problem.
- It showcases that different components can be used to create a new more powerful agent.

Rainbow



DQN variant performance across 57 Atari games [HES2018].

Reinforcement Learning (RL)

- Introduction to RL
- RL algorithms
- Deep RL. DQN
- **Policy Gradient methods**
- Actor Critic Methods
- Maze example
- Autonomous and Piloting
- Current Research Challenges

Reinforcement Learning (RL)

- Introduction to RL
- RL algorithms
- Deep RL. DQN
- Policy Gradient methods
- **Actor Critic Methods**
- Maze example
- Autonomous and Piloting
- Current Research Challenges

Policy Gradient - Actor Critic methods

Common values for this baseline:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{cases} E_{\pi_{\boldsymbol{\theta}}}\{\nabla_{\boldsymbol{\theta}} log\, \pi(a|s;\boldsymbol{\theta})\, G_n\}, & \text{REINFORCE} \\ E_{\pi_{\boldsymbol{\theta}}}\{\nabla_{\boldsymbol{\theta}} log\, \pi(a|s;\boldsymbol{\theta})\, Q_{\pi}(s,a)\}, & \text{Q Actor} - \text{Critic} \\ E_{\pi_{\boldsymbol{\theta}}}\{\nabla_{\boldsymbol{\theta}} log\, \pi(a|s;\boldsymbol{\theta})\, A_{\pi}(s,a)\}, & \text{Advantage Actor}-\text{Critic.} \end{cases}$$

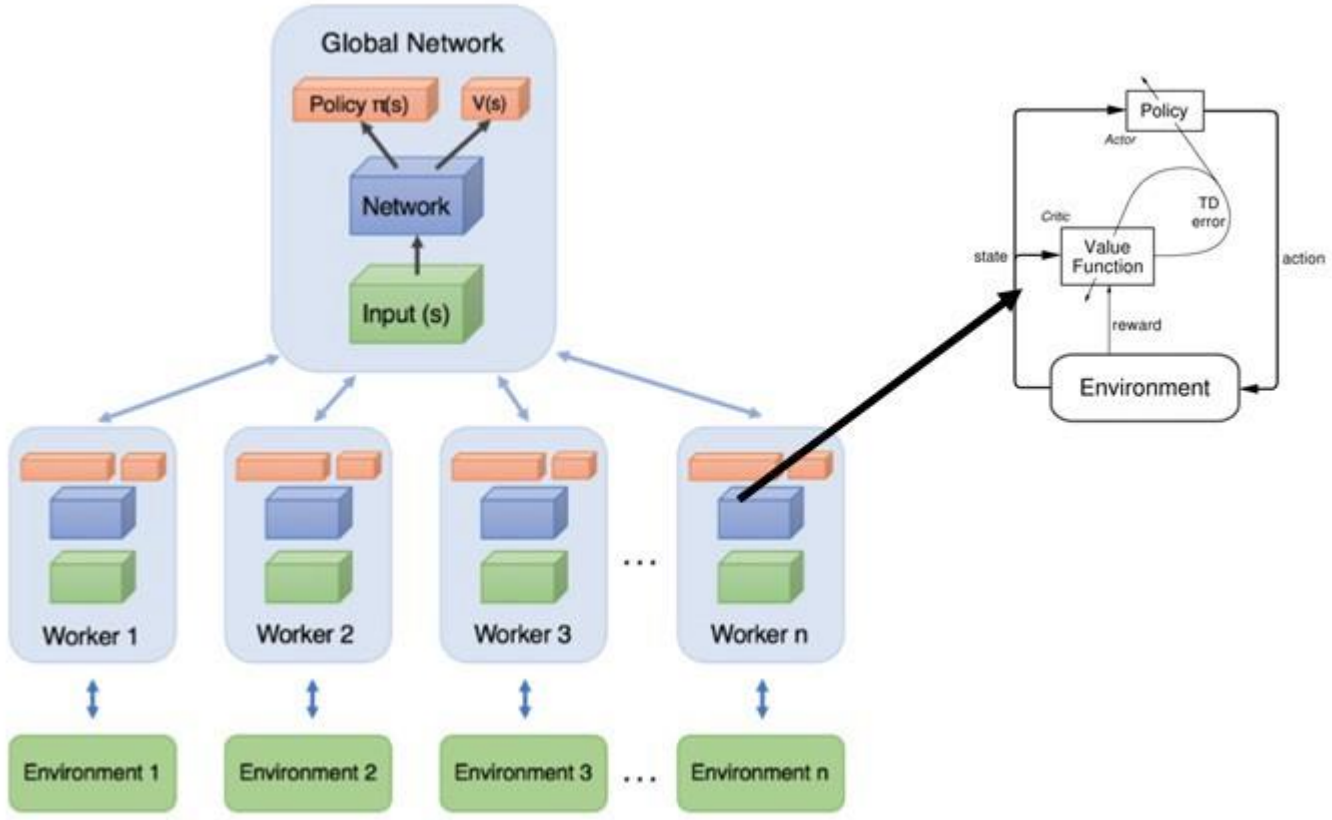
Actor Critic Methods

- The update equation can be written as:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E\{\sum_{n=0}^{N-1} \nabla_{\boldsymbol{\theta}} \log\pi(a|s;\boldsymbol{\theta}) Q_{\pi}(s_n, a_n)\}.$$

- If Q-function is parametrized by a DNN, then we can learn it.
- In this way, ***Actor Critic Methods*** are devised:
  - ***Critic*** evaluates the value (Q_{π} or V_{π}) function.
  - ***Actor*** updates the policy distribution, according to Critic directions.

Asynchronous Advantage Actor Critic



Asynchronous Advantage Actor Critic architecture [TDS-AC].

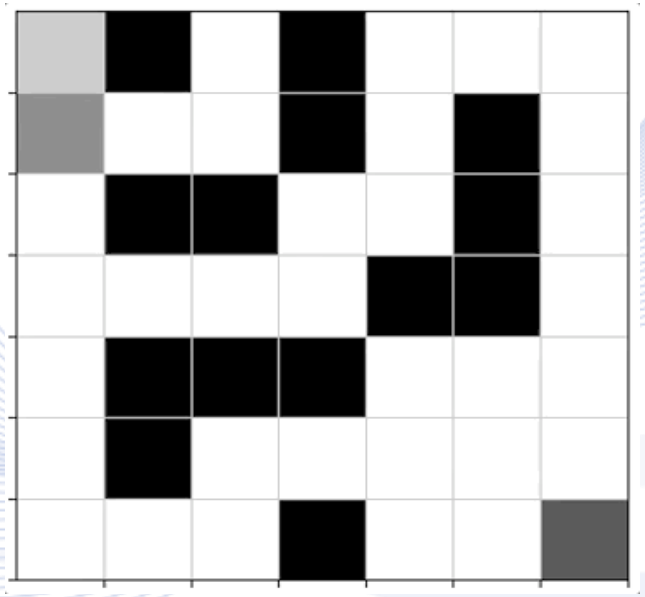
Reinforcement Learning (RL)

- Introduction to RL
- RL algorithms
- Deep RL. DQN
- Policy Gradient methods
- Actor Critic Methods
- **Maze example**
- Autonomous and Piloting
- Current Research Challenges

Deep RL Example: Maze

A robot agent starts from a starting position and moves towards the maze exit:

- Starting Point at (0, 0)
- Exit Point at (6, 6).



- Use of Q-Learning Algorithm, with temporal experience of past actions within each epoch.
- Example implemented in Python 2.7 using Keras Deep Learning Python library.

Maze: DNN model and inputs

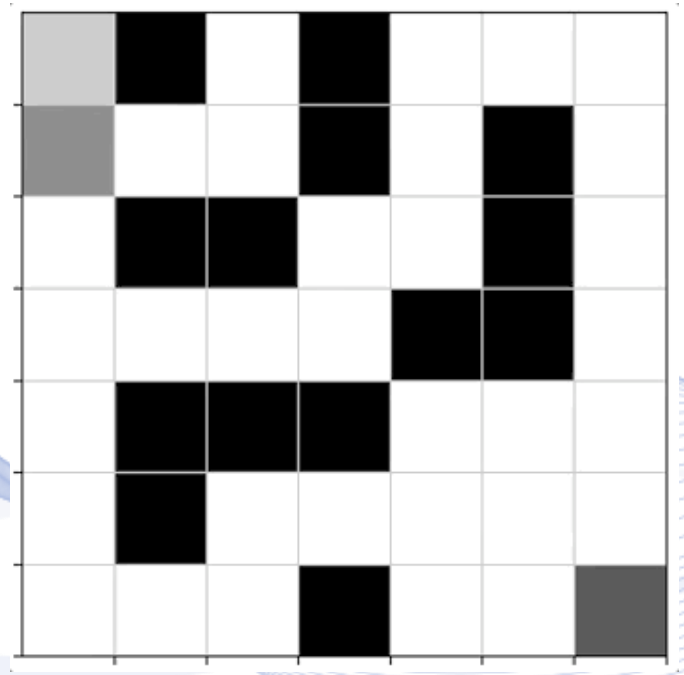
- Deep Neural Network implemented in Keras:
  - 5 fully connected (dense) NN layers;
  - PReLU activator function;
  - Adam optimization algorithm.
- DNN input: a maze layout in the form of a $N \times M$ matrix:
  - 0: filled cells.
  - 1: valid (empty) cell that the robot can occupy (one at each time instance).

```
maze =  np.array([
    [ 1.,  0.,  1.,  1.,  1.,  1.,  1.],
    [ 1.,  1.,  1.,  0.,  0.,  1.,  0.],
    [ 0.,  0.,  0.,  1.,  1.,  1.,  0.],
    [ 1.,  1.,  1.,  1.,  0.,  0.,  1.],
    [ 1.,  0.,  0.,  0.,  1.,  1.,  1.],
    [ 1.,  0.,  1.,  1.,  1.,  1.,  1.],
    [ 1.,  1.,  1.,  0.,  1.,  1.,  1.]
])
```

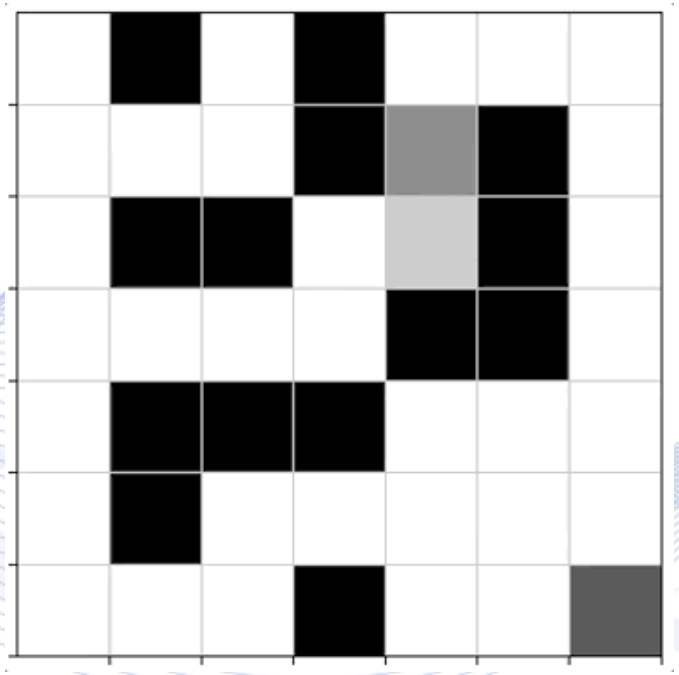
Maze matrix.

Maze: Result

We can see the optimized policy for all points:



Starting Point at (0, 0)



Starting Point at (2, 4)

Maze: Conclusions

- Maze example is a great introduction to RL.

- Due to its triviality and low complexity:
  - It does not demonstrate the unique RL ability, as it could be solved using other methods.
  - It fails to demonstrate the importance of experimentation with parameters, such as exploration rate or size of experience and their effect on result quality.

Reinforcement Learning (RL)

- Introduction to RL
- RL algorithms
- Deep RL. DQN
- Policy Gradient methods
- Actor Critic Methods
- Maze example
- **Autonomous Driving and Piloting**
- Current Research Challenges

Autonomous Drone Cinematography

- There are 26 different types of drone cinematography.
- Drone motion (position/orientation) and gimbal should be controlled for autonomous drone cinematography.

- RL can be used for such drone piloting and gimbal control.

Drone Motion Type Modeling

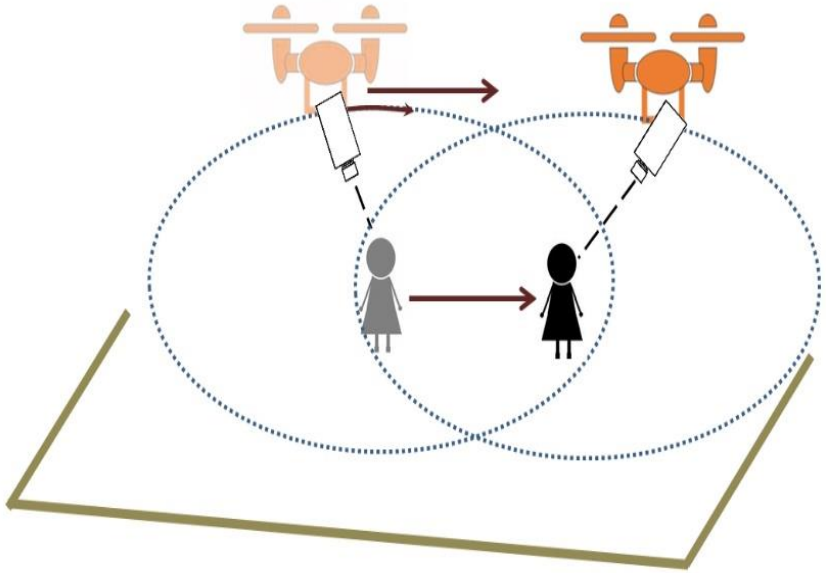
Example: ORBIT.

- The camera always looks at the target while the drone moves on a circle in the TCS:

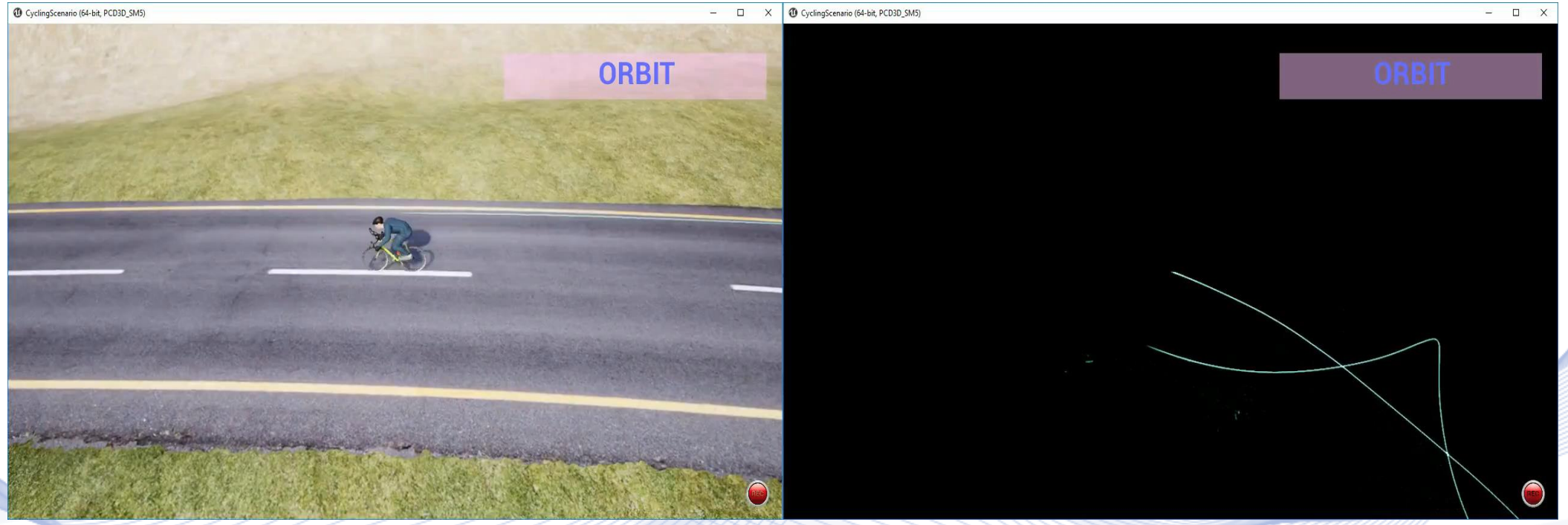
$$\mathbf{X}_t = \left[\lambda \cos\left(t\frac{\omega}{T} + \theta_0\right), \lambda \sin\left(t\frac{\omega}{T} + \theta_0\right), x_{t3}\right]^T \quad (1)$$

$$\mathbf{L}_t = \mathbf{P}_t.$$

- $\mathbf{P}_t = [p_{t1}, p_{t2}, p_{t3}]^T$: Target position in WCS.
- $\mathbf{X}_t = [x_{t1}, x_{t2}, x_{t3}]^T$: Drone position in TCS.
- $\mathbf{L}_t$: Camera Look-At Point in WCS.



Drone Motion Type Modeling



ORBIT.

Autonomous Driving

- Autonomous driving is an evolving research field in computer vision and control systems.

- Companies, such as Google, Tesla, NVIDIA try to develop advanced autonomous driving cars.

- Practical RL challenges in real RL applications are:
  - high-dimensionality of state space and
  - non-trivial large action range.

Autonomous Driving Examples

- AWS Deep Racer (self-driving robot).
- Voyage Deep Drive (simulation platform).

Reference: [MED-RLAD]

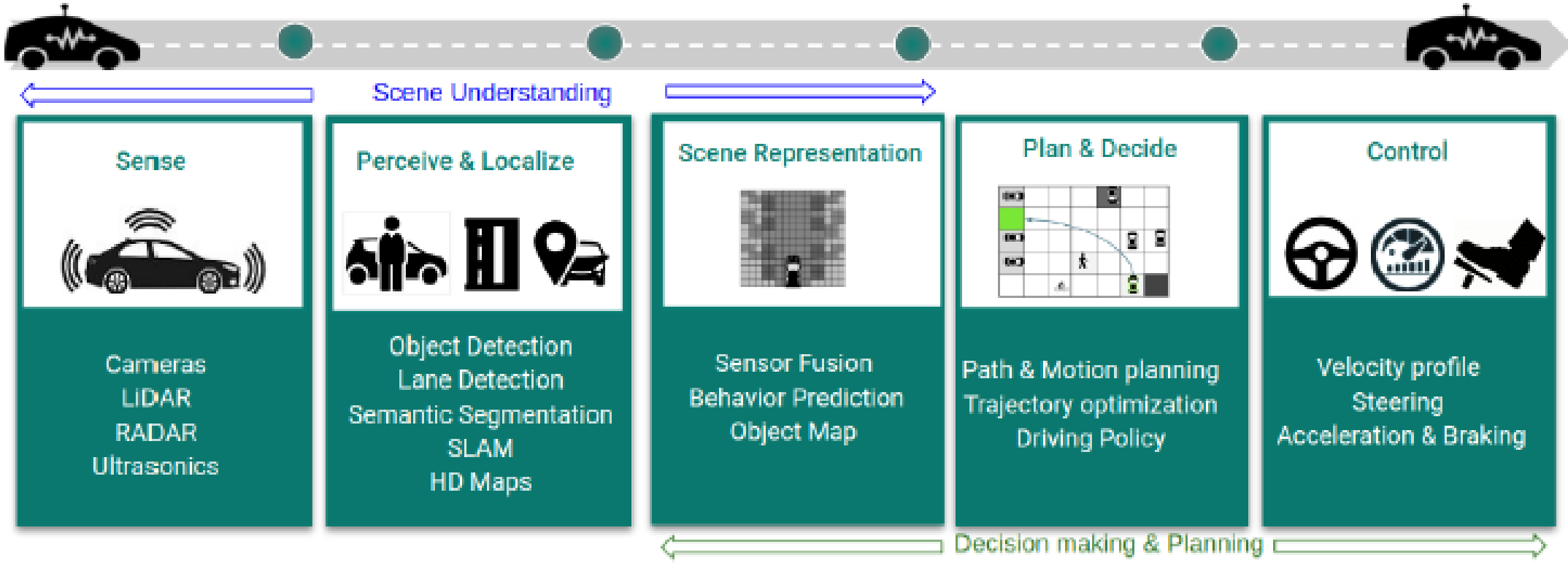
Reference: [MED-DRLAD]

Autonomous Driving

RL can be beneficial in the following autonomous driving tasks:

- Motion Planning
- Overtaking
- Intersections/Merging
- Lane Change
- Lane Keep
- Automated Parking.

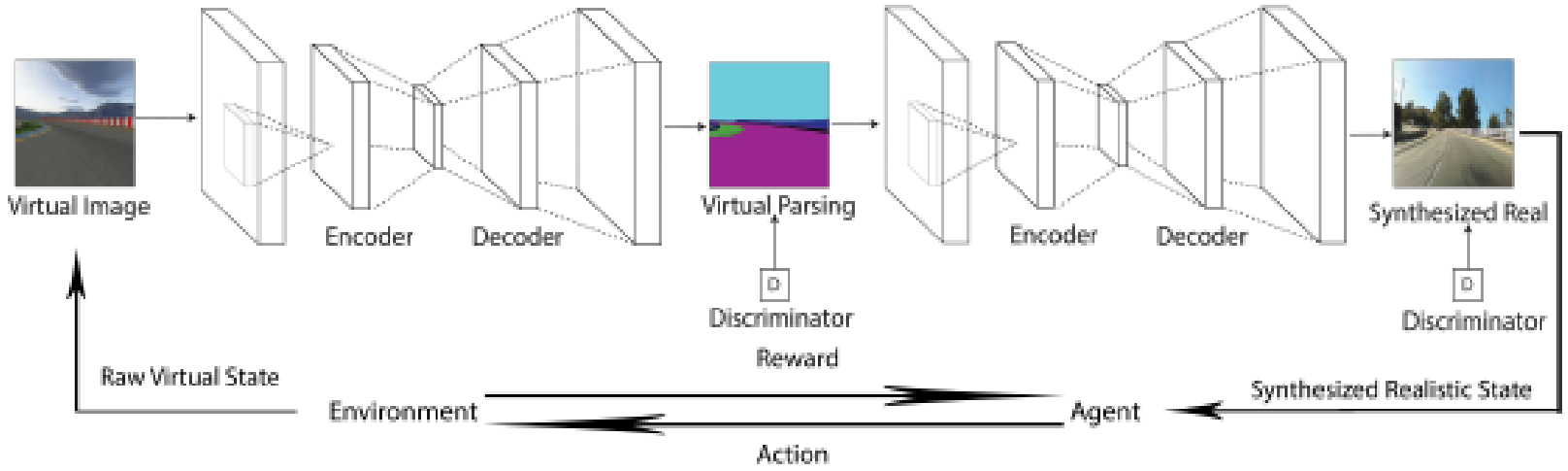
Autonomous Driving



Autonomous driving system architecture [TAL2019].

DRL Framework for Autonomous Driving

- RL: In order to train a driving policy the realistic images were fed into A3C.
- Evaluation: The trained policy was examined on real world driving data, in order to assess how accurately it can predict the steering angle.



Reference: [PAN2017]

Reinforcement Learning (RL)

- Introduction to RL
- RL algorithms
- Deep RL. DQN
- Policy Gradient methods
- Actor Critic Methods
- Maze example
- Autonomous Driving and Piloting
- **Current Research Challenges**

Current Research Challenges

- Exploration vs Exploitation (***Long horizon reasoning***):
  - ε -soft policies, noisy nets, upper confidence bound (maximize $E[R] + \kappa\sigma[R]$, where $\sigma[R]$ is the standard deviation of the return and $\kappa > 0$, stimulating exploration in areas with high uncertainty and moderate expected return), etc.
- ***Hierarchical RL***:
  - Policies that run other policies.
  - Higher-level policies pay attention to higher-level goals, while sub-policies are in charge of fine control.

Current Research Challenges

- ***Imitation Learning***:
  - Supervised Learning.
  - It cannot adapt to new situations that have not been demonstrated.
  - Thus, first supervised learning is performed and then RL is used for fine-tuning.
  - The agent tries to evaluate an unknown reward function ***using distributions provided from an expert*** that direct him towards a specific solution.

Current Research Challenges

- Memory and Attention:
  - Use of different architectures, for example an RNN-architecture instead of a convolutional network to address network memory.
- Transfer Learning:
  - We can not easily train an agent for new tasks, in the real world.
  - Knowledge transfer from one task to another.
- Benchmarking:
  - Mostly Video Games, since the success story of DQN.

Q & A

Thank you very much for your attention!

Contact: Prof. I. Pitas
pitas@csd.auth.gr