Deep Reinforcement Learning

A brief introduction

Pedro Atencio Ortiz
November 14, 2020

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Institución
Universitaria
Reacreditada en Alta Calidad

pedroatencio@itm.edu.co

Agenda

1. Context

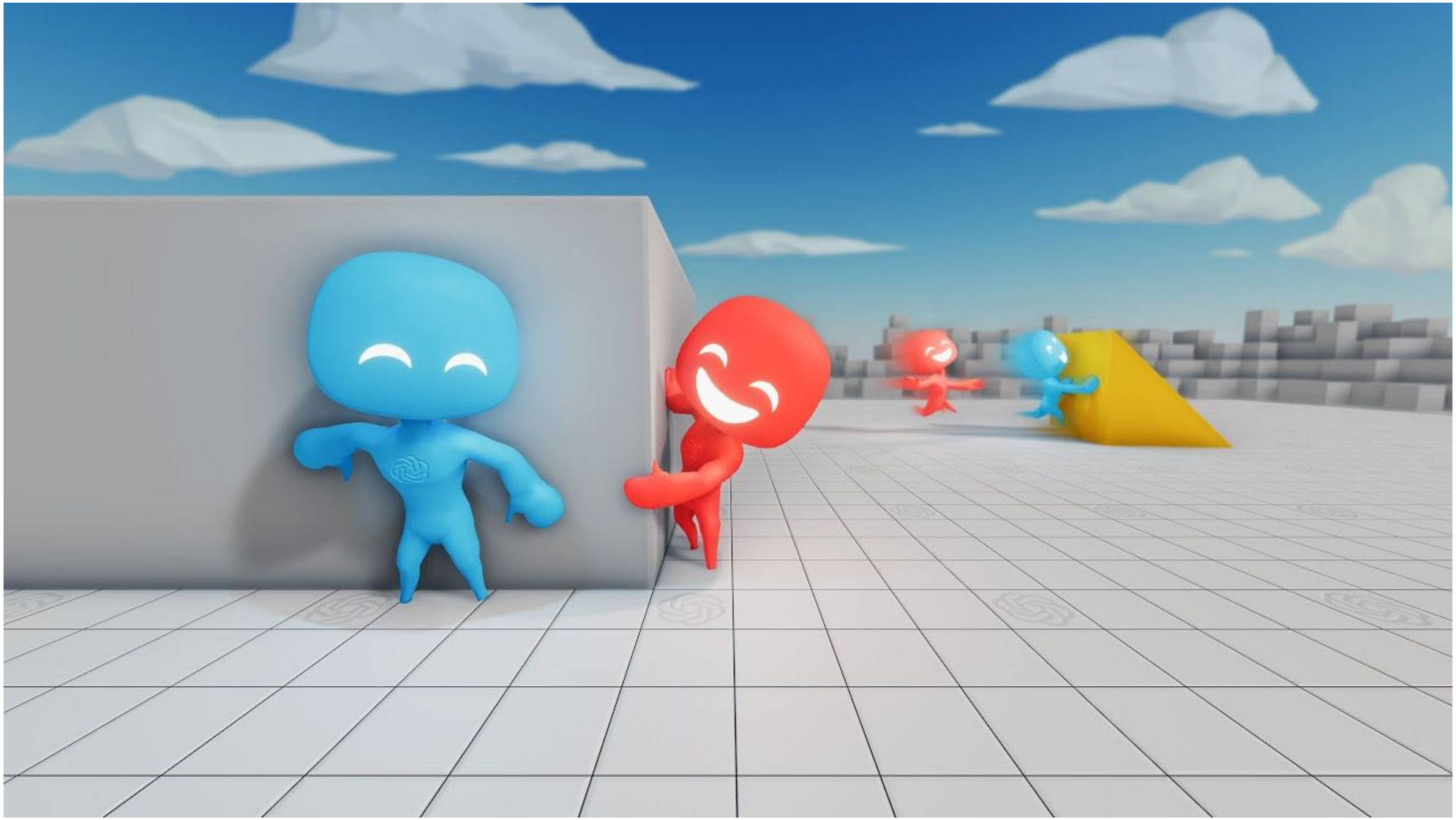
2. Deep Reinforcement Learning (Deep-RL)

3. Practice: Python and Tensorflow

4. Conclusions

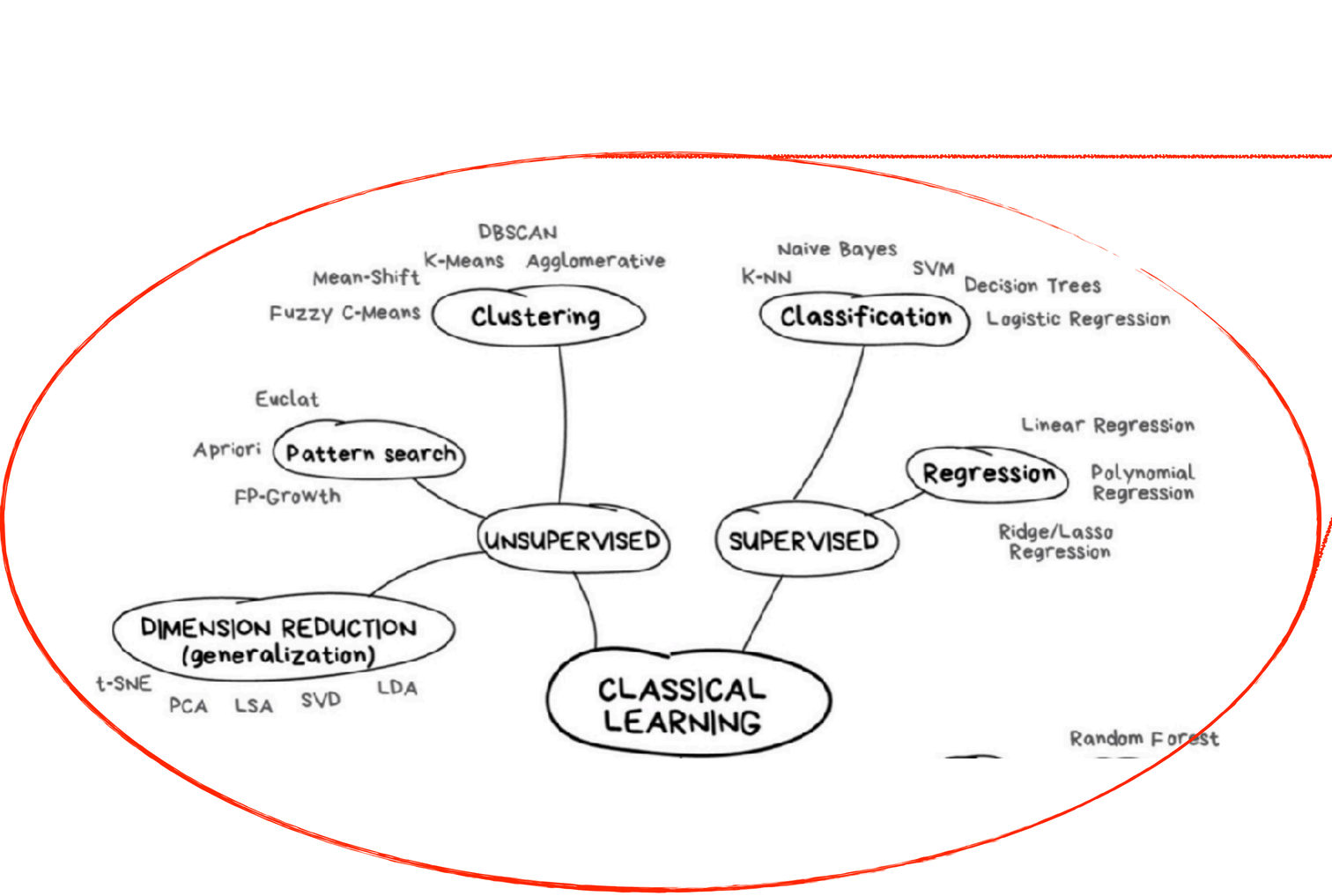
1. Context

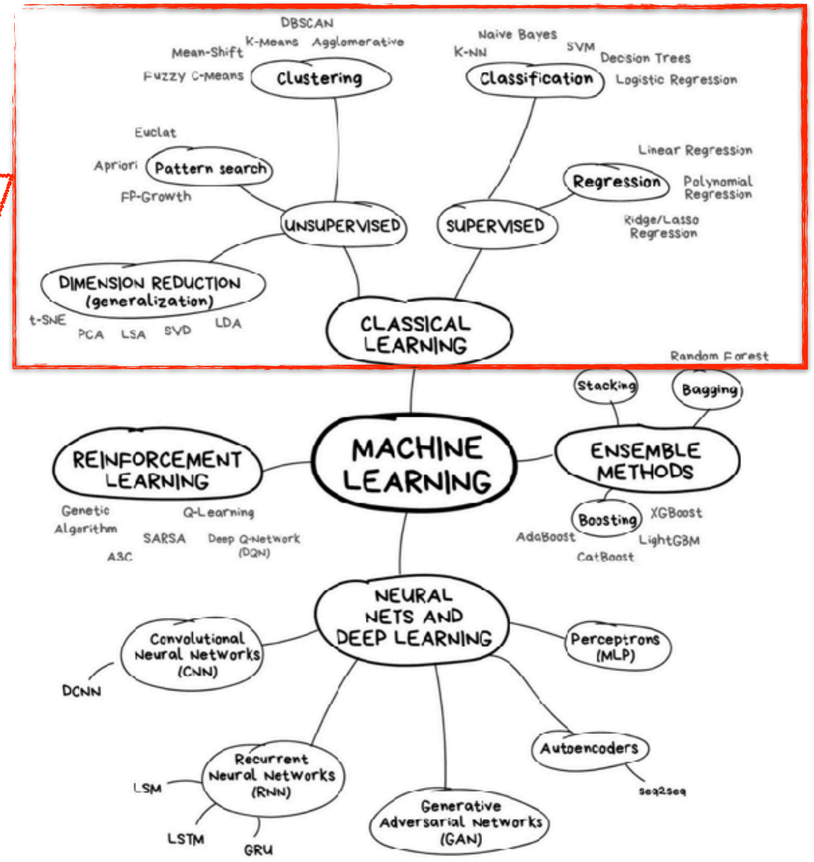
OpenAI: Emergent Tool Use from Multi-Agent Interaction



Source: https://www.youtube.com/watch?v=kopoLzvh5jY

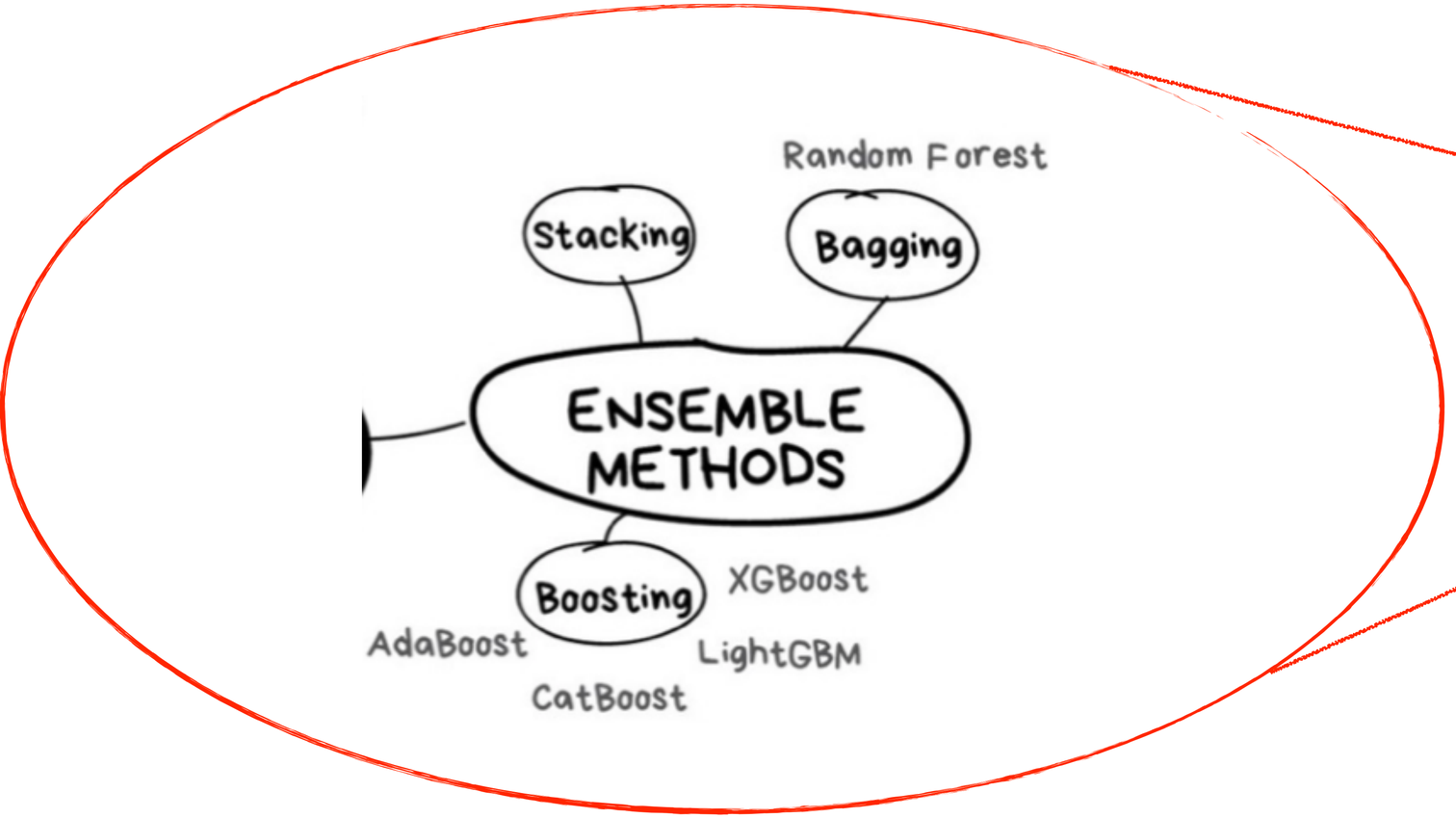
ML Landscape

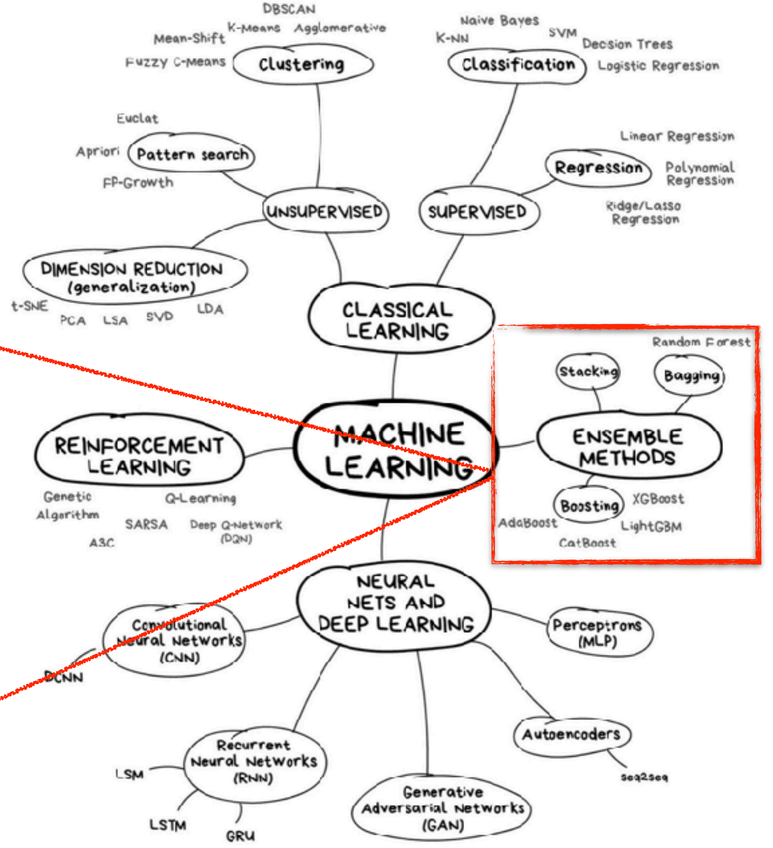




Source: https://medium.com/better-programming/from-machine-learning-to-reinforcement-learning-mastery-47f33d9f6b41

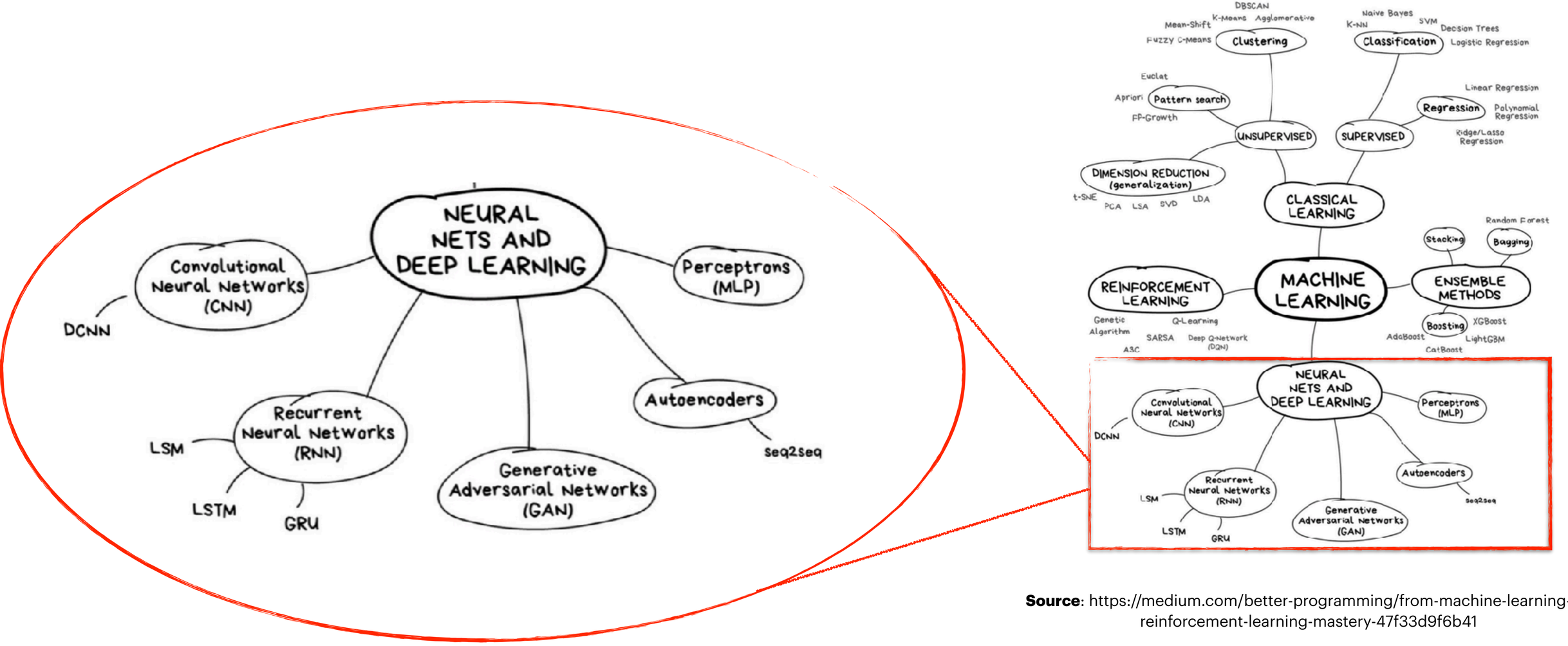
ML Landscape





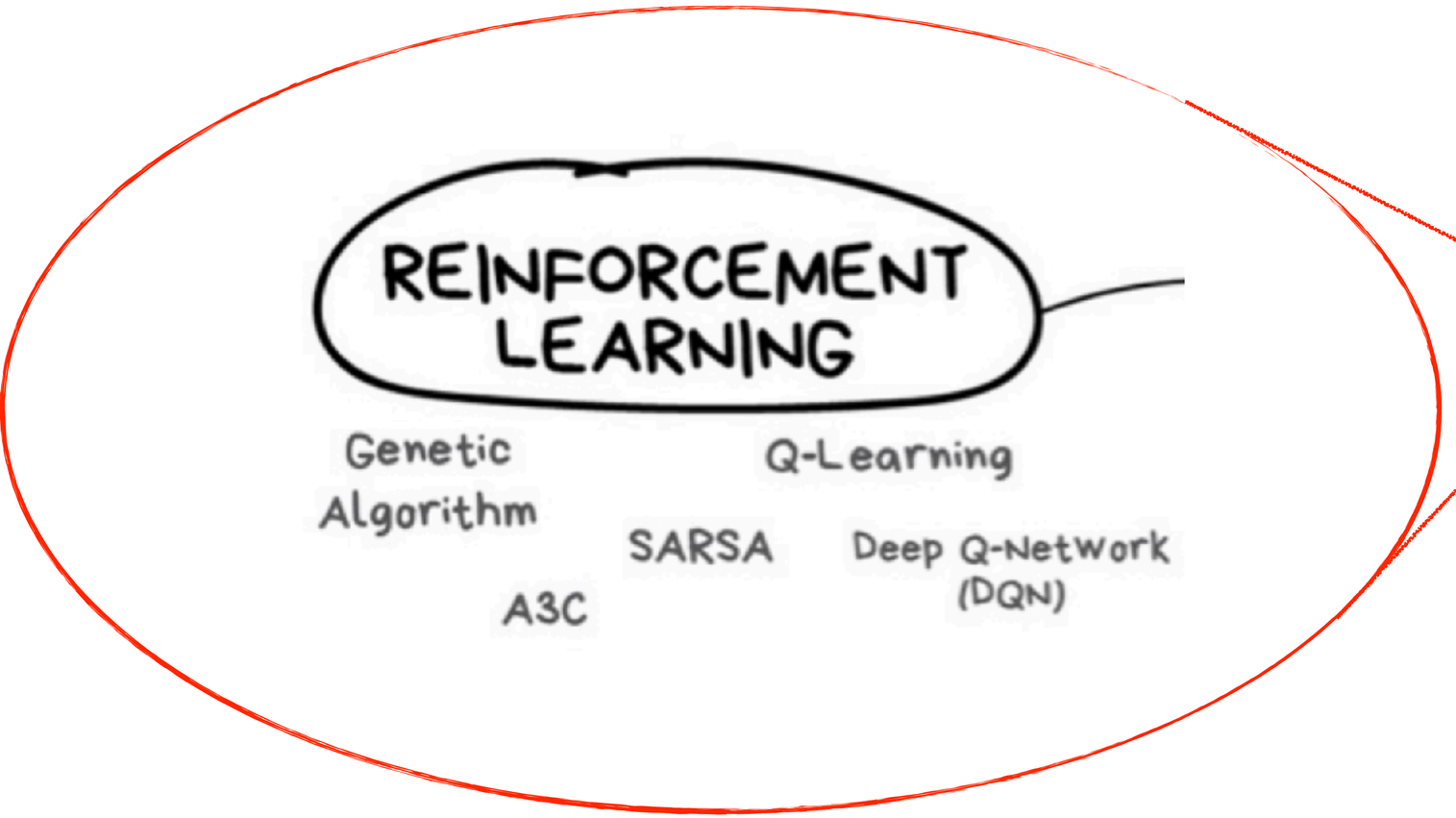
Source: https://medium.com/better-programming/from-machine-learning-to-reinforcement-learning-mastery-47f33d9f6b41

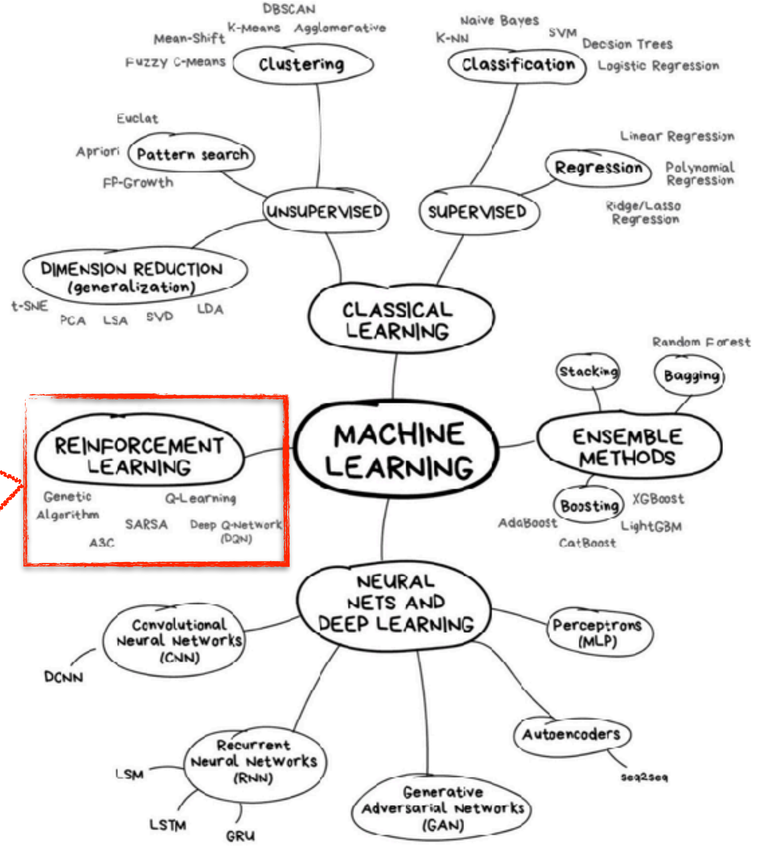
ML Landscape



Source: https://medium.com/better-programming/from-machine-learning-to-reinforcement-learning-mastery-47f33d9f6b41

ML Landscape





Source: https://medium.com/better-programming/from-machine-learning-to-reinforcement-learning-mastery-47f33d9f6b41

Classes of ML Problems

Supervised Learning

Goal: Maps inputs to outputs

This is a **mango**.
Mangoes **can be eaten**.
Birds eat mangoes.

Unsupervised Learning

Goal: Learn the underlying data patterns.

Looks like a mango.
Birds also eat it.
Maybe **I can eat it**.

Reinforcement Learning

Goal: Maximize future rewards over many time-steps.

When I eat peach, **i feel strong**.
I will **continue eating** peach.

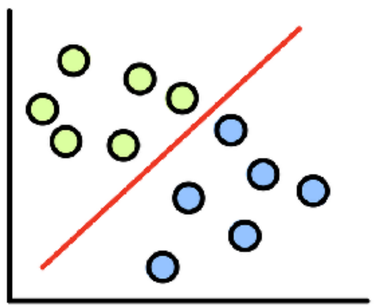
Classes of ML Problems

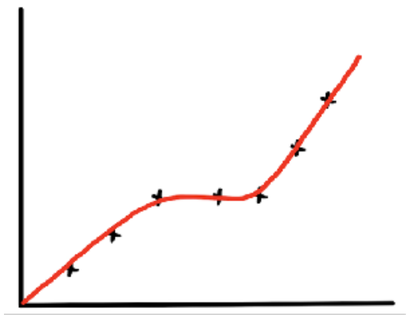
Supervised Learning

Data: (X, Y)

X is a set of input features, Y is a set of output values, labels or scalars.

Goal: $X \rightarrow Y$



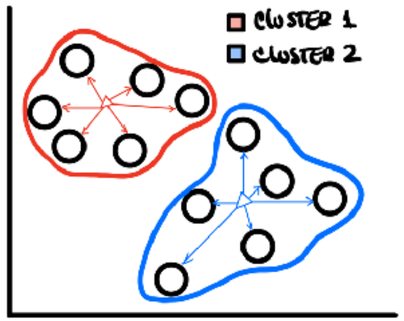


Unsupervised Learning

Data: X

X is a set of input features.

Goal: Learn the underlying data patterns.

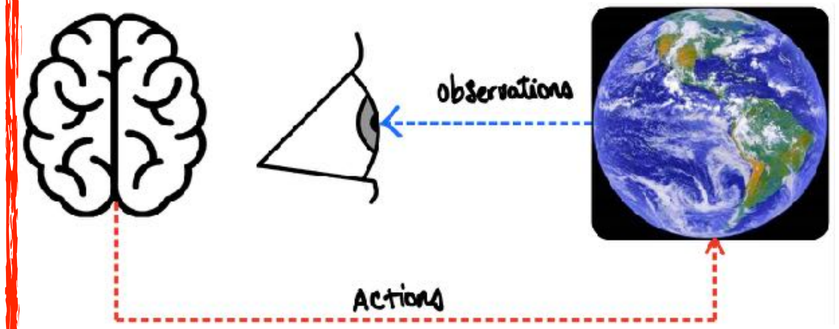


Reinforcement Learning

Data: (S, A)

S is a set of states, A is a set of actions.

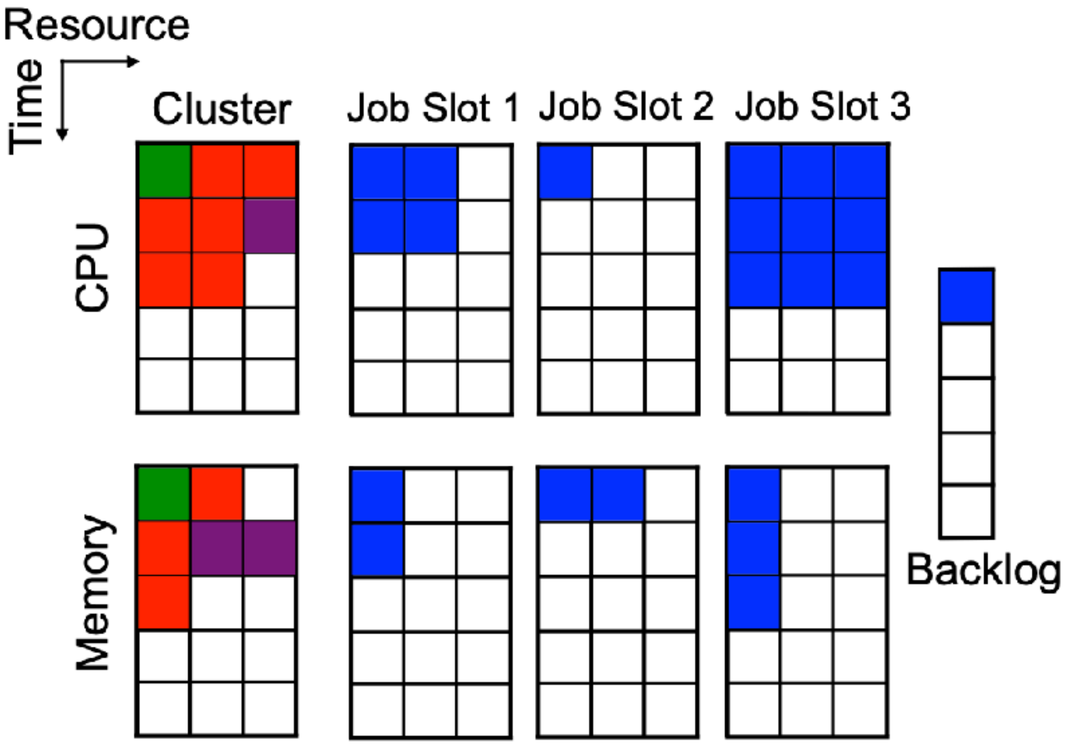
Goal: Maximize future rewards over many time-steps.



Applications

Resource management

- Job Scheduling in computer clusters, bit rate adaption in video streaming, virtual machine placement in cloud computing, congestion control.
- State space was formulated as the current resources allocation and the resources profile of jobs.

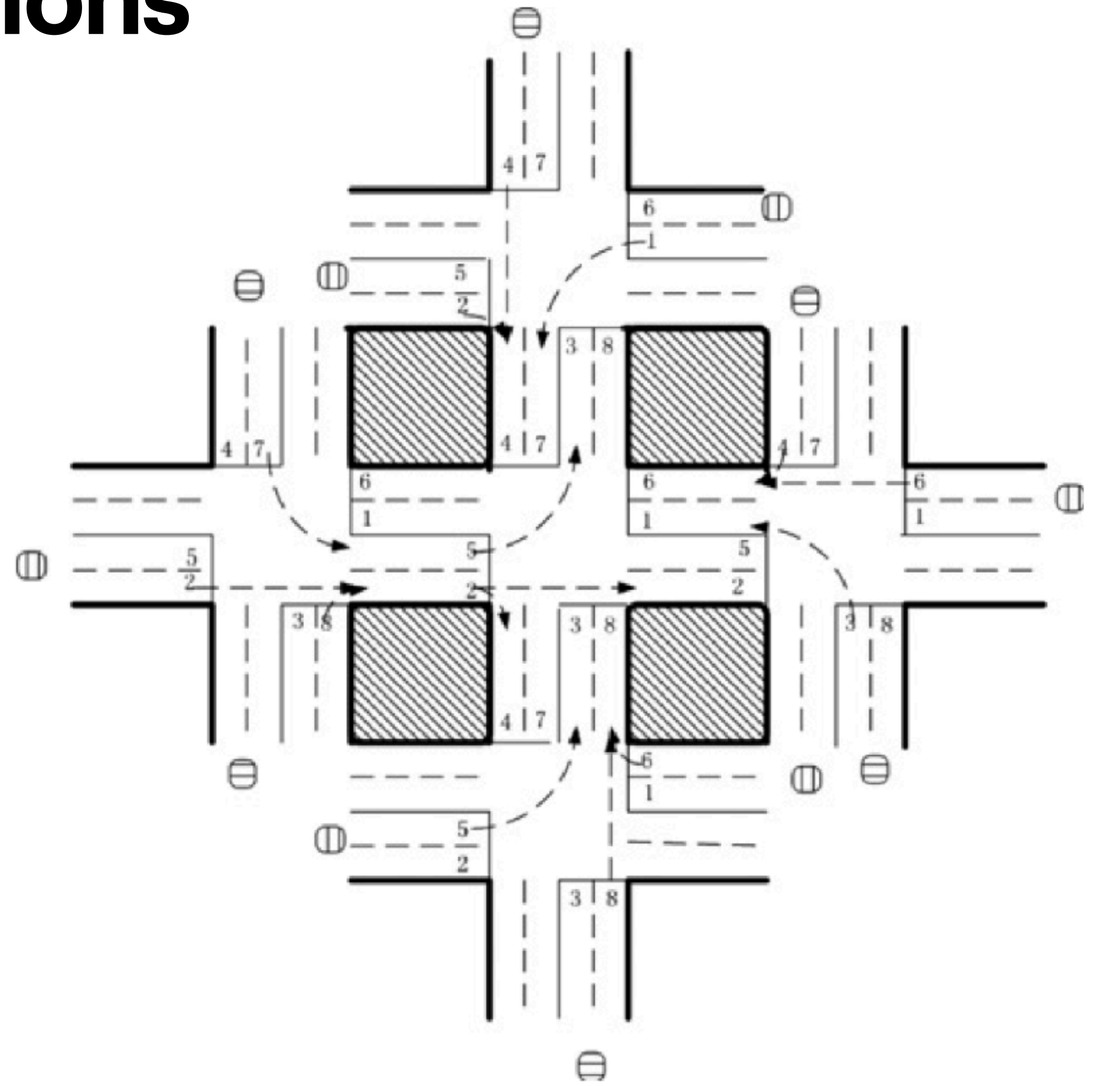


(Mao et. al., 2016). Resource Management with Deep Reinforcement Learning.

Applications

Traffic Light Control

- State is the relative traffic flow.
- Actions phase-based light control. 8 possible phases.
- Rewards related with low delay from the previous time step.

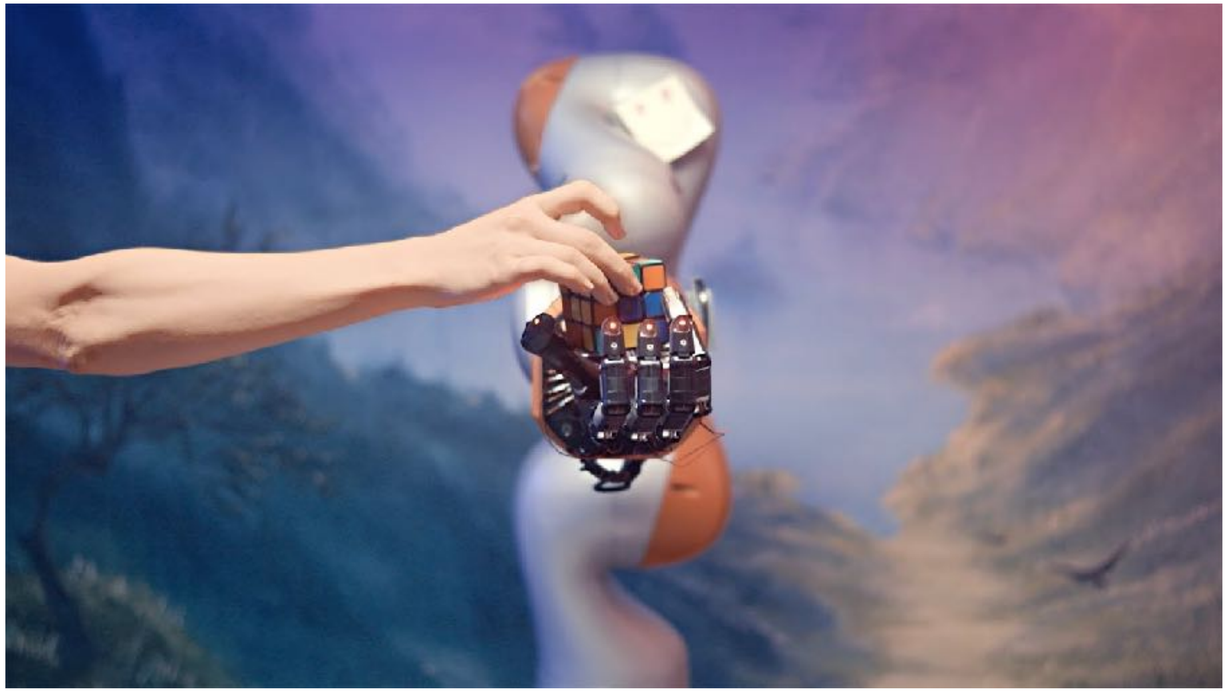


(Arel et. al., 2010). Reinforcement learning-based multi-agent system for network traffic signal control .

Applications

Robotics

- Walking robots.
- Grasping tasks.
- Vertical landing rockets: SpaceX.
- Autonomous driving: Kiwi Labs, Amazon prime drone system.

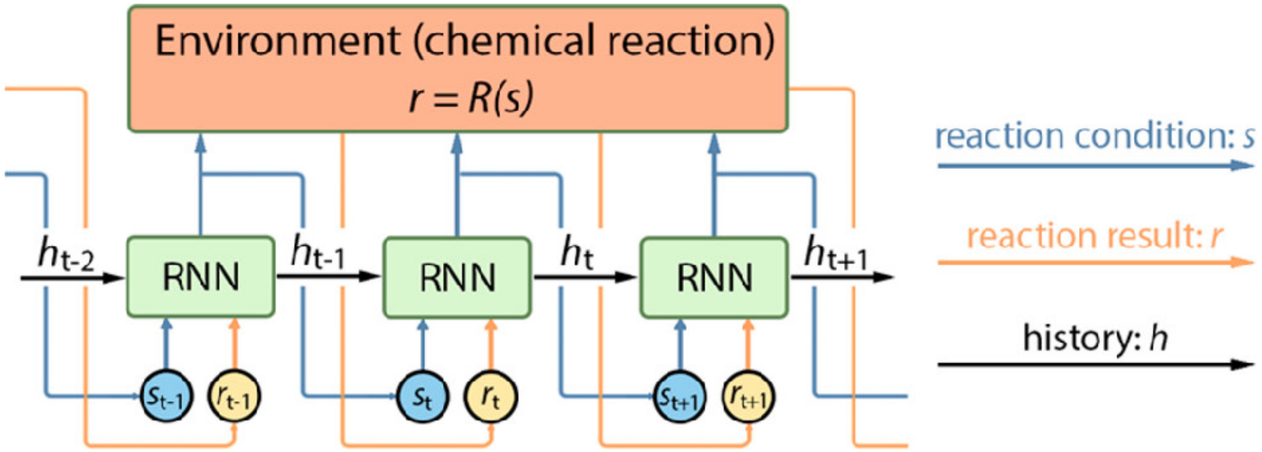


OpenAI. Solving Rubik’s cube with a robotic hand.

Applications

Chemistry and pharmaceutics

- Chemical reactions optimization.
- Drug discovering.
- States are the set of all possible combinations of experimental conditions.
- Actions are the set of all changes that can be made to experimental condition, for example, increase the temperature.
- Reward is a certain experimental condition that needs to be achieved.

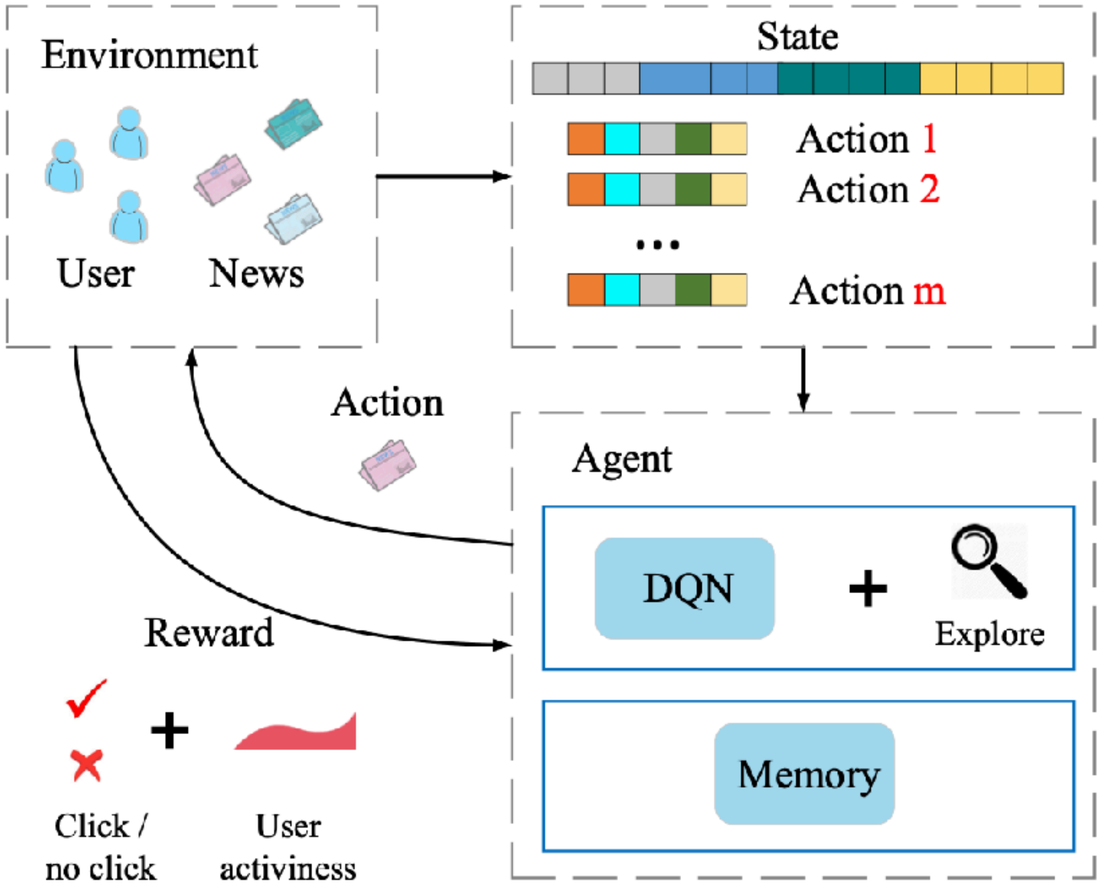


(Zhou et. al., 2017). Optimizing Chemical Reactions with Deep Reinforcement Learning.

Applications

Recommendation systems

- Movies recommendation.
- Products recommendations.
- Personalized advertising.
- States are a set of user features and context features.
- Actions are a set news features.
- Reward a click by the user to the recommended news.

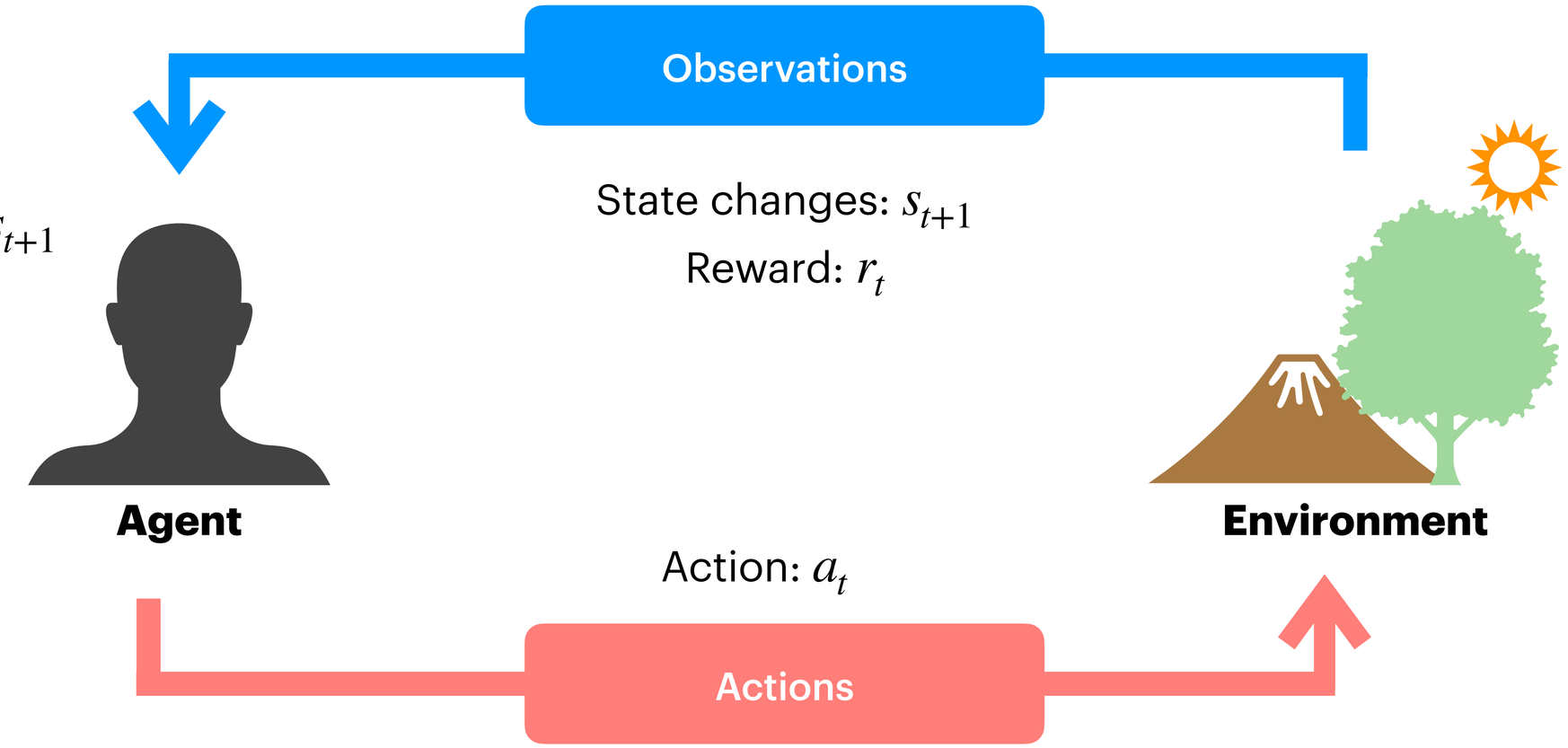


(Zheng et. al., 2018). DRN: A Deep Reinforcement Learning Framework for News Recommendation.

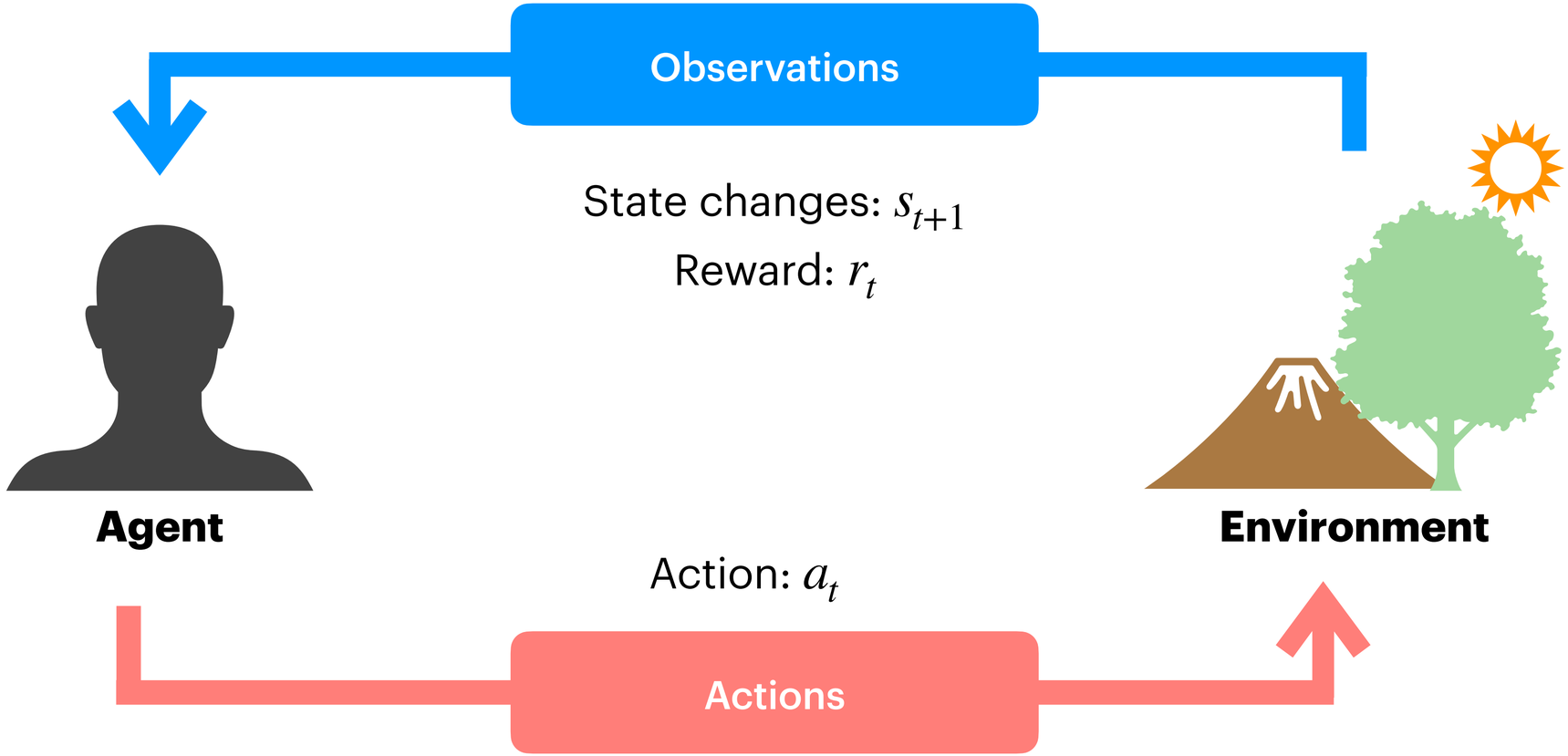
Reinforcement Learning (RL): Key Concepts

RL Timeline

- At each time-step the agent
  - Executes an action a_t
  - Receives a future observation s_{t+1}
  - Receives a scalar reward r_t
- The environment
  - Receives an action a_t
  - Emits a future observation s_{t+1}
  - Emits a scalar reward r_t
- t increments at each timestep



Reinforcement Learning (RL): Key Concepts



Total Reward

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} + \ldots + r_{t+n} + \ldots$$

Reinforcement Learning (RL): Key Concepts

Discounted Total Reward

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} + \ldots + \gamma^{t+n} r_{t+n} + \ldots$$

γ: discount factor

Reinforcement Learning (RL): Key Concepts

(-) reward

(+) reward

The Q Function

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} + \ldots + \gamma^{t+n} r_{t+n} + \ldots$$

The total reward (R_t)is the sum of all rewards from time t

This is what we want to learn → $Q(s, a) = \mathbb{E}[R_t]$

The Q function estimates / returns the expected ($\mathbb{E}$) total reward R_t an agent in a state s can perceive by executing an action a

The policy function π

$$Q(s, a) = \mathbb{E}[R_t]$$

$(state, action)$

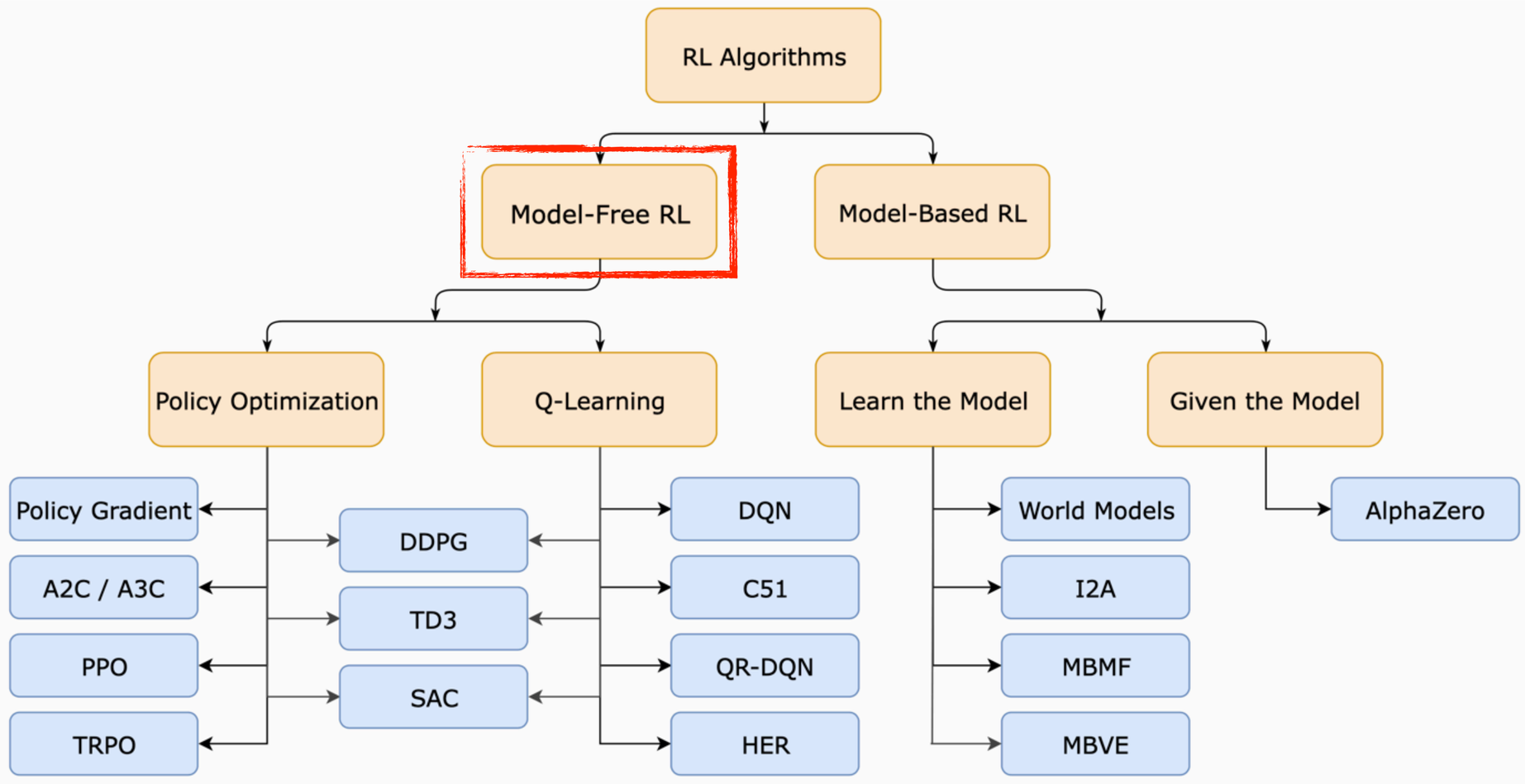
A **policy function** π is needed by the agent in order **to take the best action** from a given state s.

Strategy: select the action a which maximizes the expected total reward.

$$\pi^*(s) = \underset{a}{argmax}(Q(s, a))$$

2. Deep Reinforcement Learning (Deep-RL)

Deep-RL: A Taxonomy



Source: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#citations-below

Deep-RL: Main algorithms

Value Learning

Find $Q(s, a)$

$$a = \underset{a}{argmax}(Q(s, a))$$

Policy Learning

Find $\pi(s)$

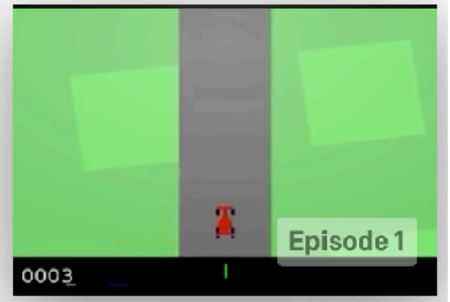
Sample $a \backsim \pi(s)$

Deep-RL: Environments

OpenAI-Gym

BipedalWalker-v2
Train a bipedal robot to walk.

BipedalWalkerHardcore-v2
Train a bipedal robot to walk over rough terrain.



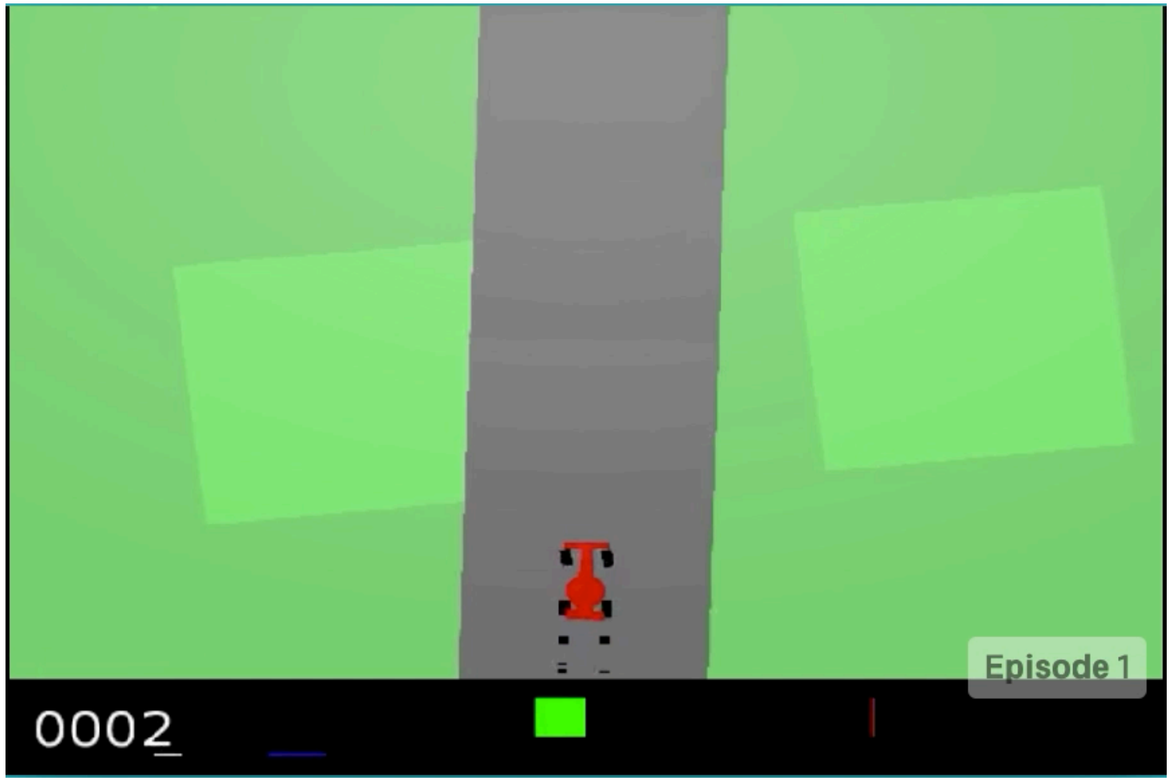
CarRacing-v0
Race a car around a track.

LunarLander-v2
Navigate a lander to its landing pad.

LunarLanderContinuous-v2
Navigate a lander to its landing pad.

Deep-RL: Environments

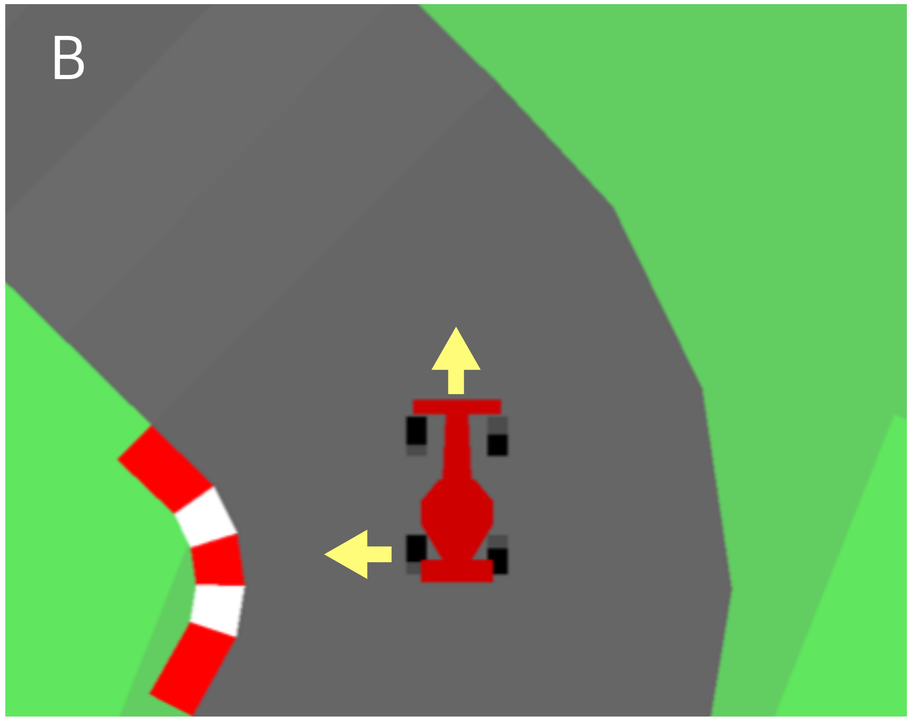
CarRacing-V0



- **State**(s_t): 96x96 pixels (a frame).
- **Actions** (a): [left, right,accel.,break]
- **Reward** (r_t): proportional to the completion of track; inversely to how many time to complete the track.

Deep-RL: Inside the Q-Function

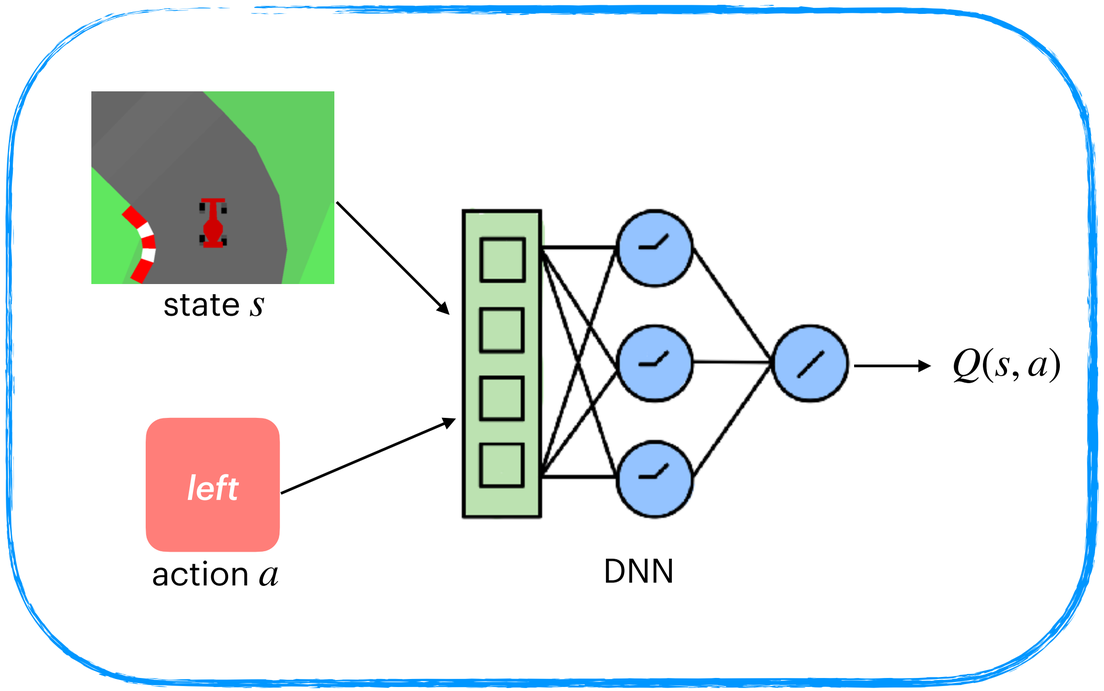
It could not be intuitively to estimate the Q-values



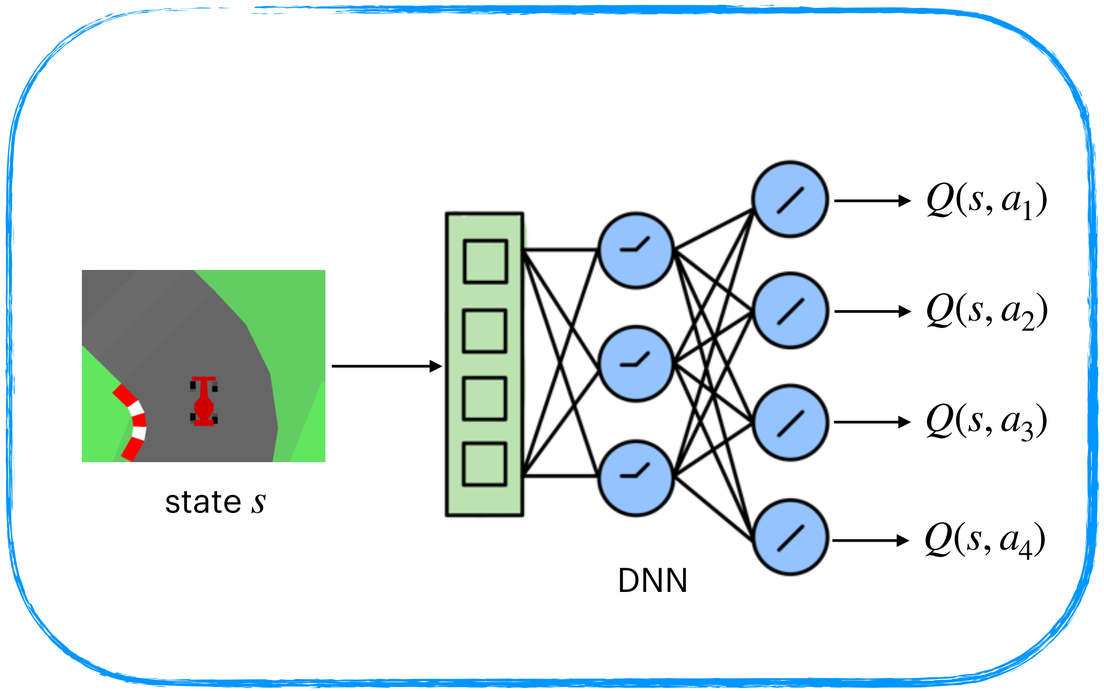
Which (s, a) pair has a higher Q-value?

Deep-RL: Deep Q Networks (DQN)

The use of Neural Networks and Deep Neural Networks to model the Q-functions



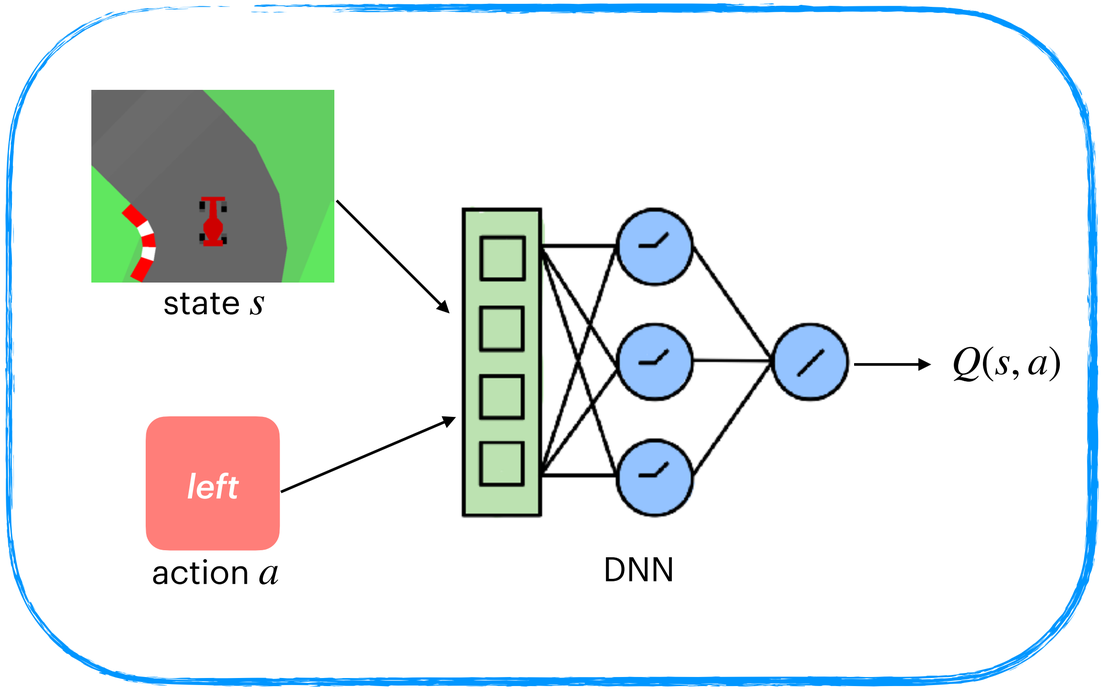
$n_{actions}$ time-steps

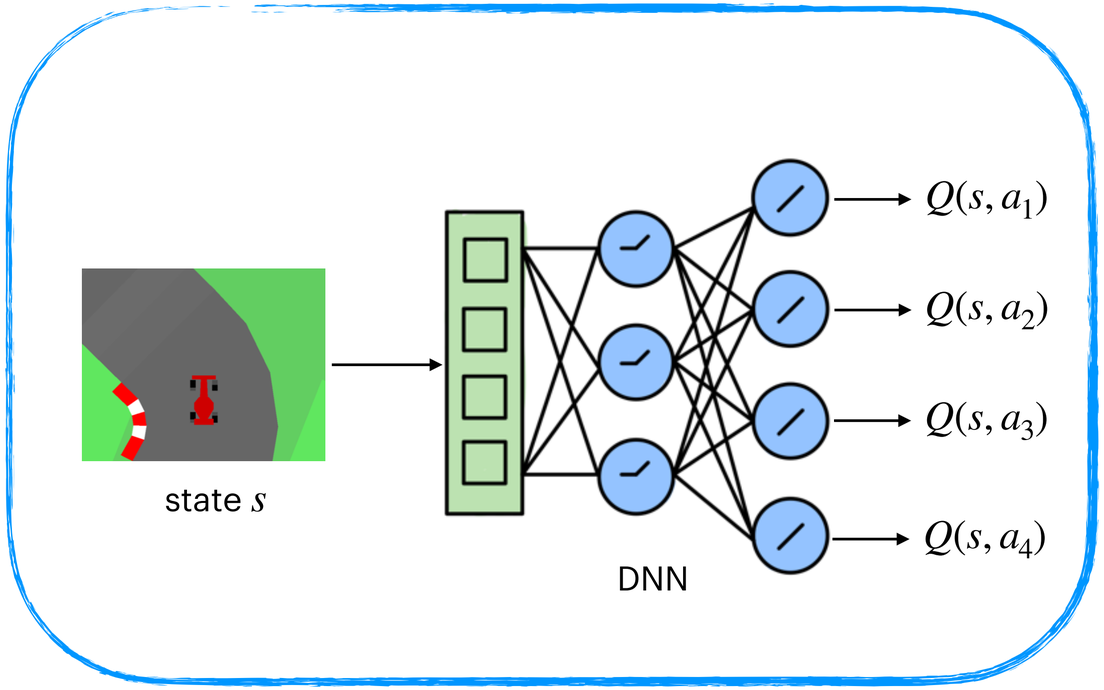


single time-step

Deep-RL: Deep Q Networks (DQN)

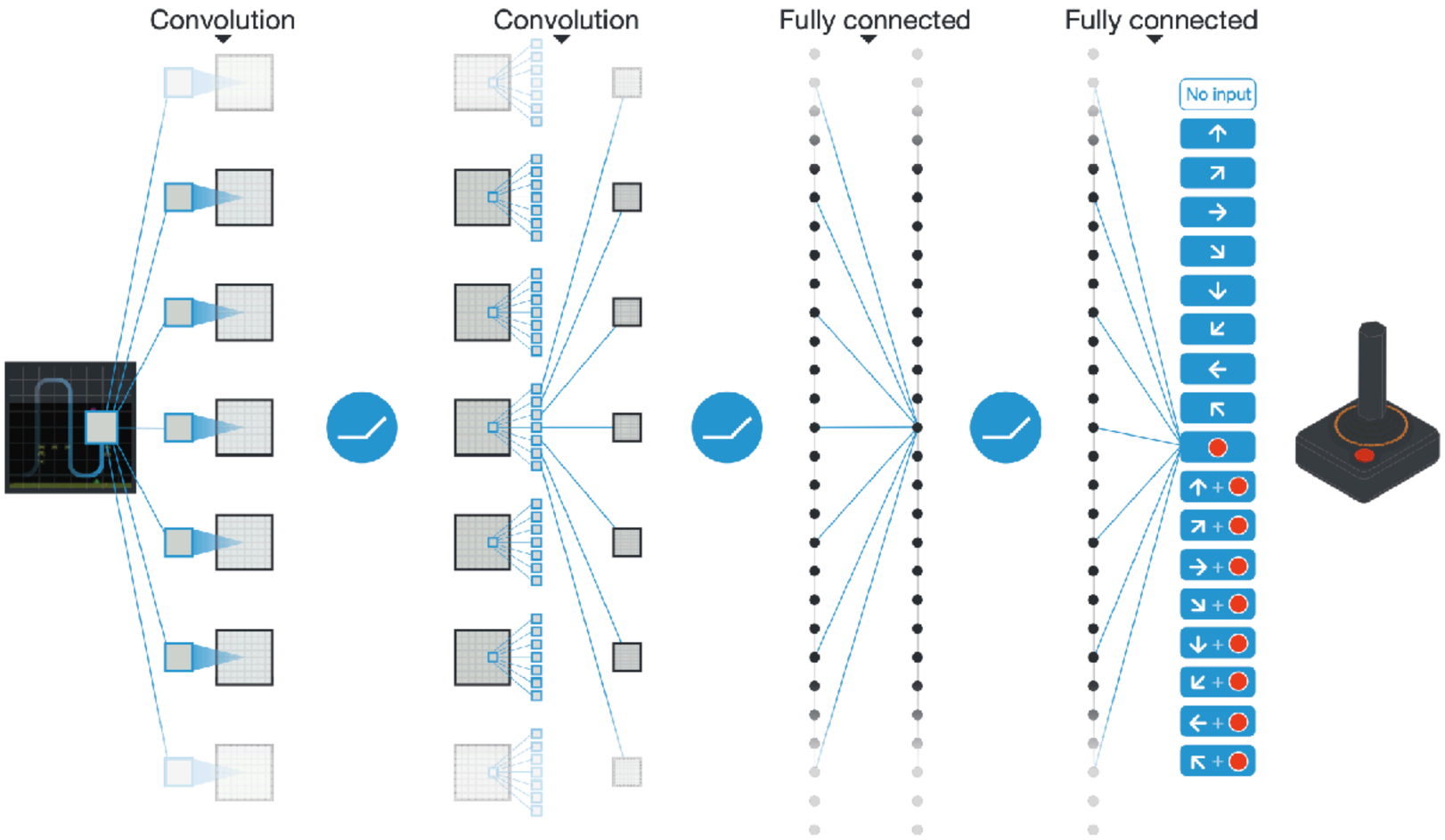
A regression problem!





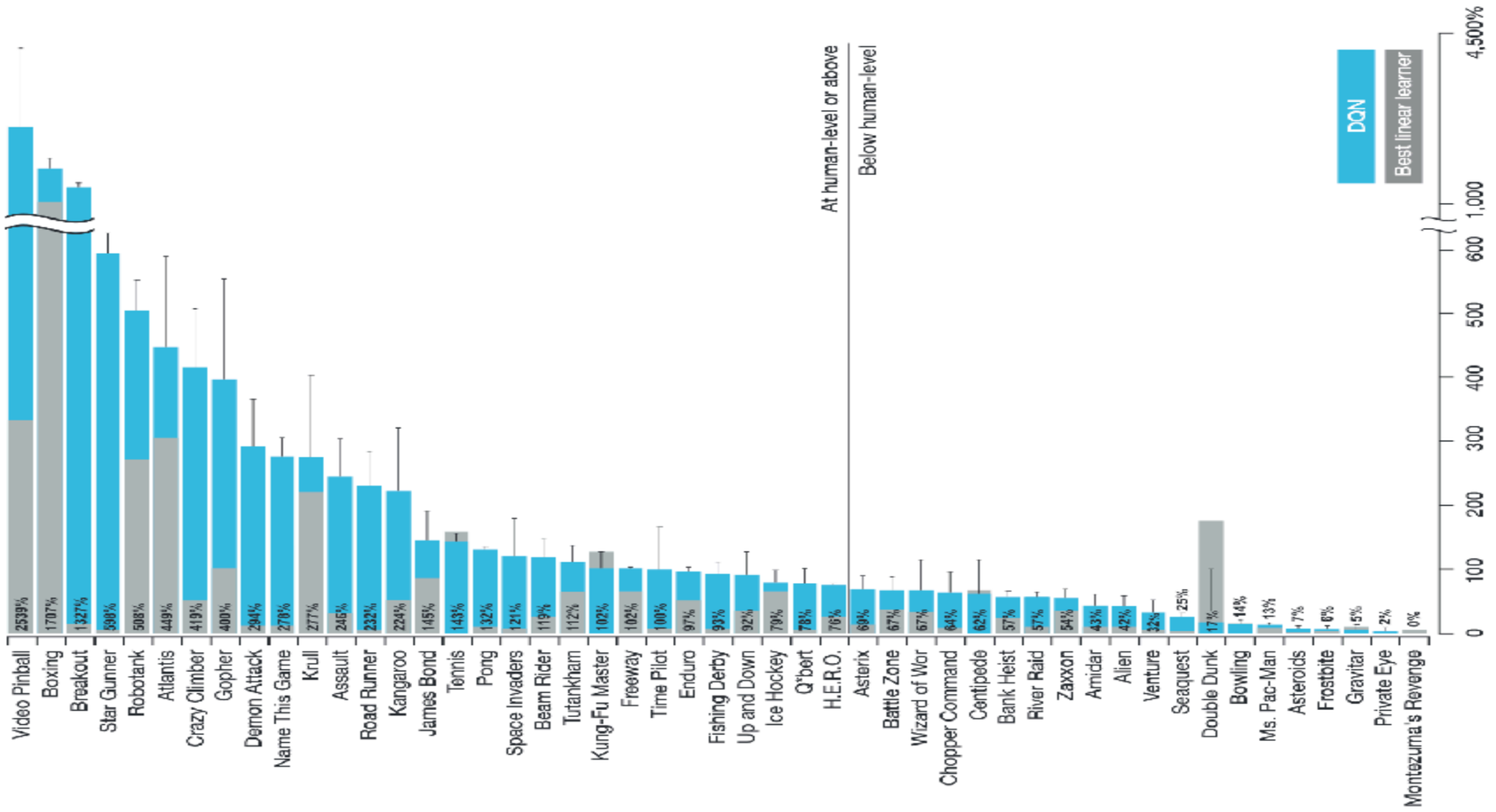
$$loss(s', a', a, s) = \mathbb{E}\left[\left\| \overbrace{\left(r + \gamma \,.\, \max_{a'}(s', a')\right)}^{\textbf{target}} - \overbrace{Q(s, a)}^{\textbf{prediction}} \right\|^2 \right]$$

Deep-RL: Deep Q Networks (DQN)



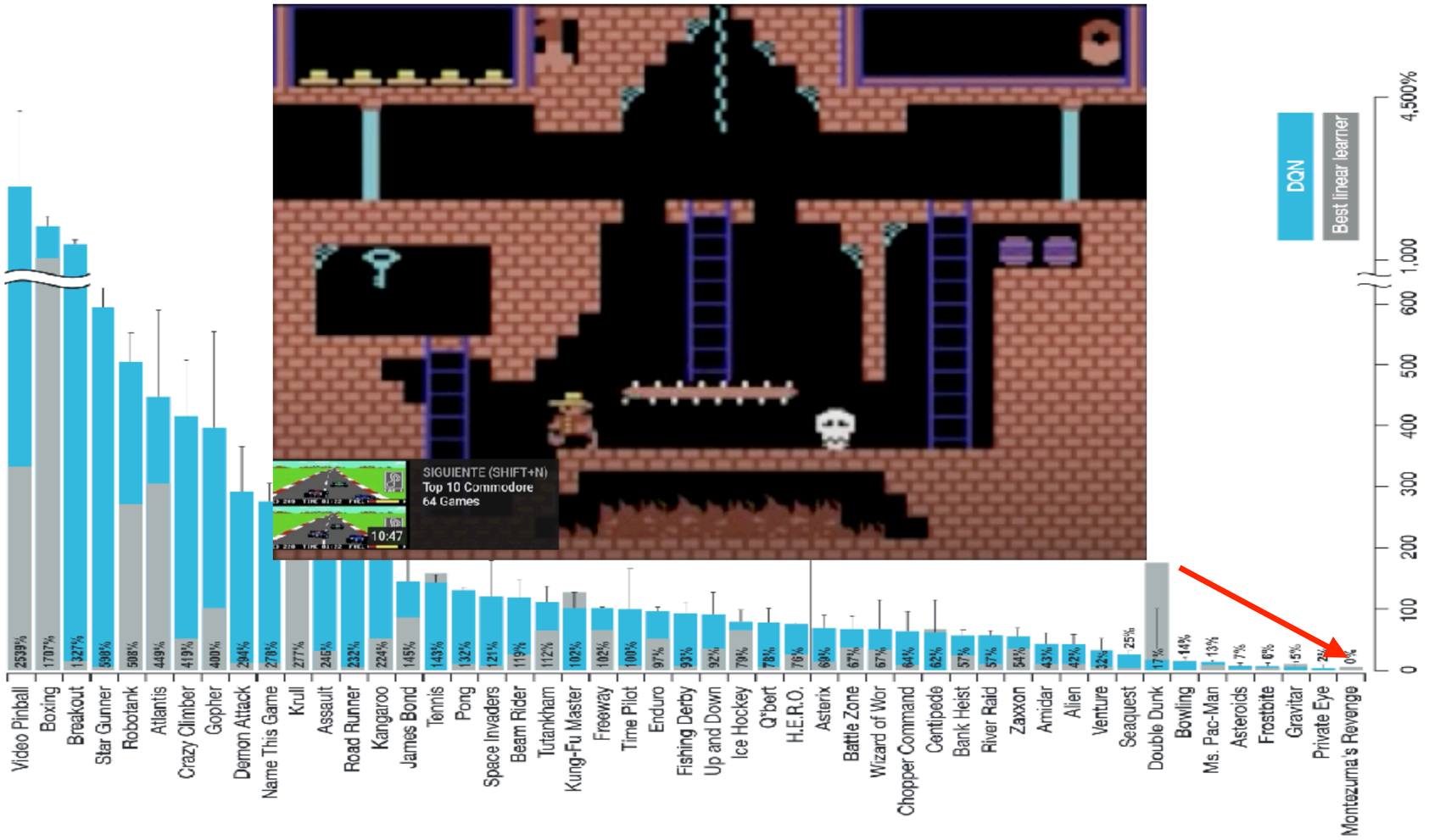
Google DeepMind - Human-level control through deep reinforcement learning, 2015.

Deep-RL: Deep Q Networks (DQN)



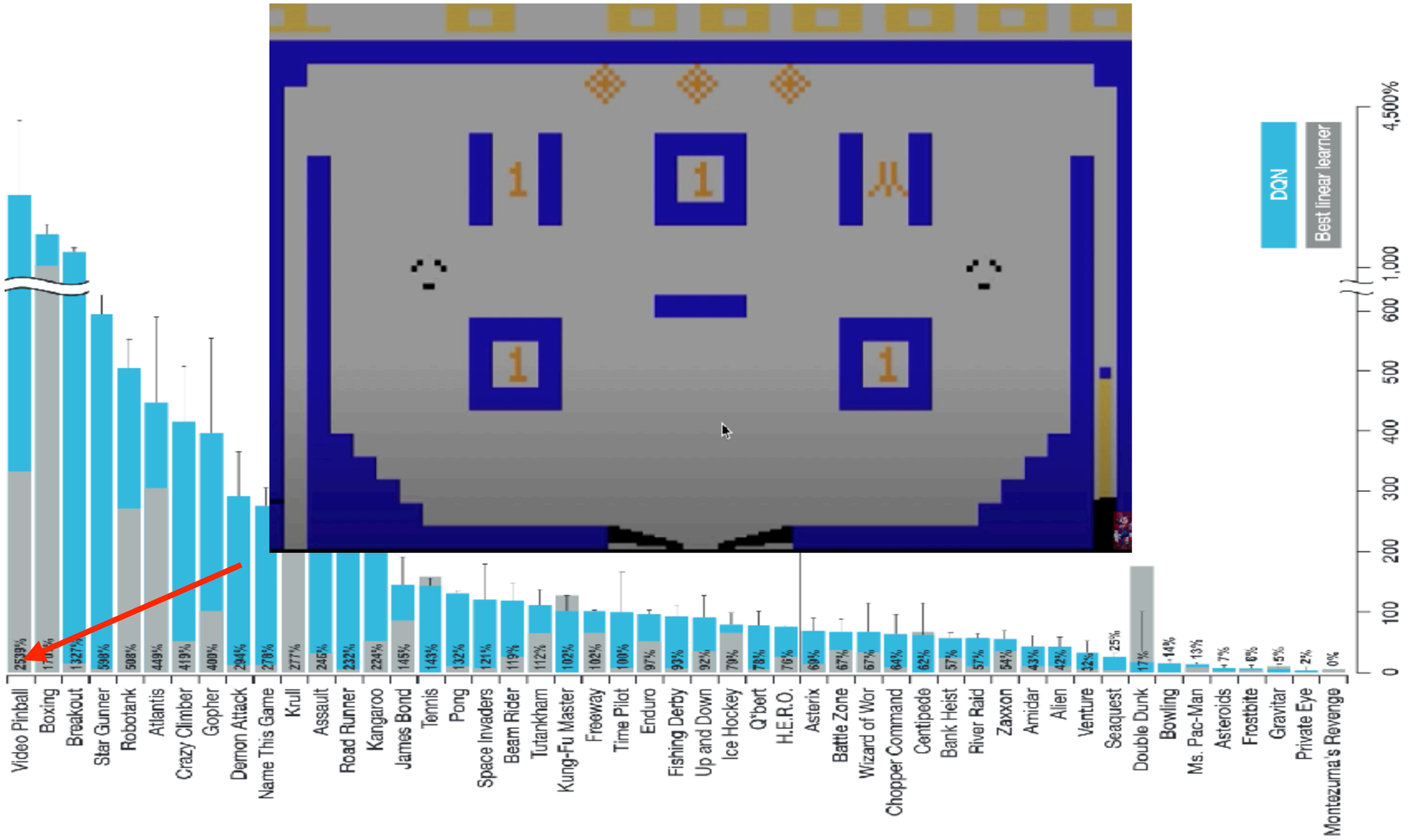
Google DeepMind - Human-level control through deep reinforcement learning, 2015.

Deep-RL: Deep Q Networks (DQN)



Google DeepMind - Human-level control through deep reinforcement learning, 2015.

Deep-RL: Deep Q Networks (DQN)



Google DeepMind - Human-level control through deep reinforcement learning, 2015.

What can we expect from DQN?

- Can model scenarios where the number of actions (action space) is discrete and small.
- Cannot model continuous actions spaces, for example, steering angle in autonomous car driving.
- As the actions taken by the agent comes from argmax from predicted Q, cannot learn stochastic policies.

Deep-RL: Main algorithms

Value Learning

Find $Q(s, a)$

$$a = \underset{a}{argmax}(Q(s, a))$$

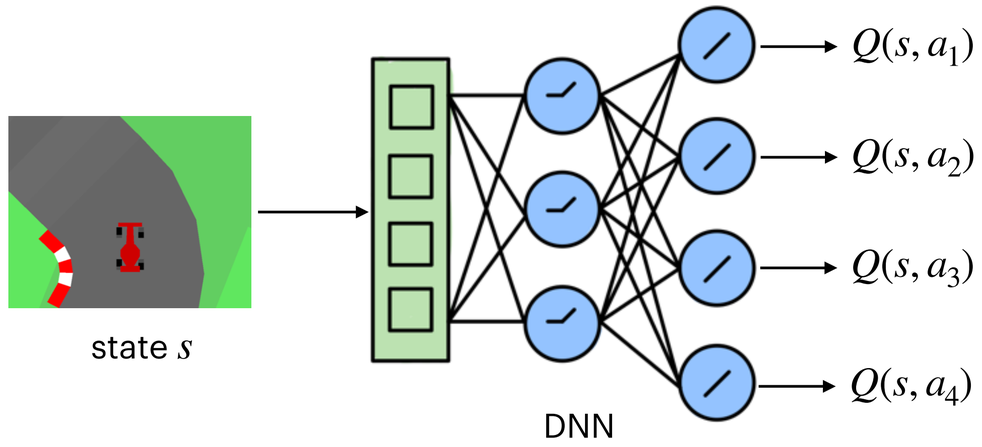
Policy Learning

Find $\pi(s)$

Sample $a \backsim \pi(s)$

Deep-RL: Policy Gradient

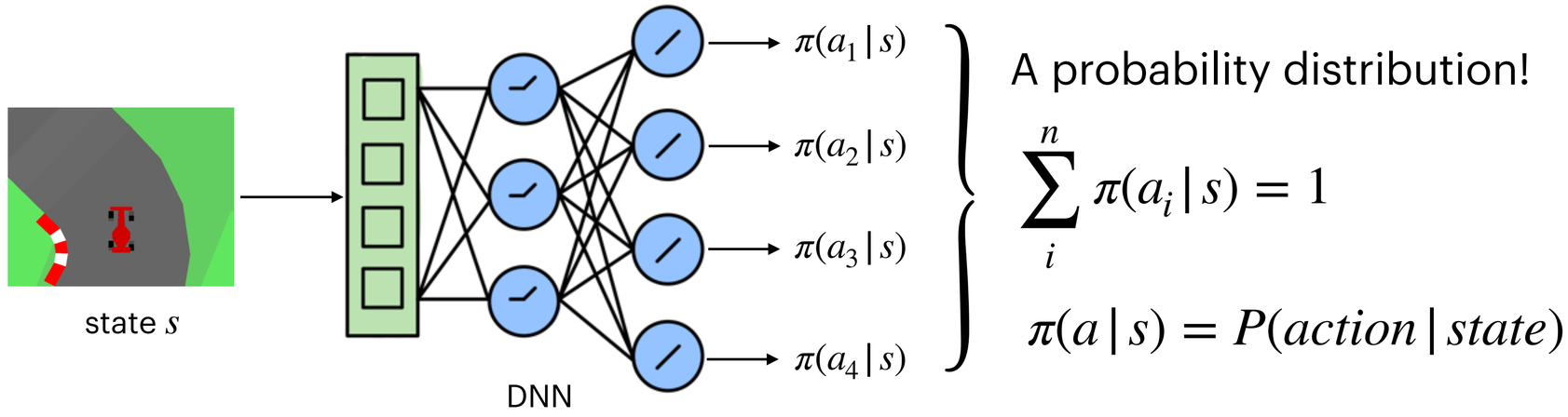
DQN: Approximate Q from a DNN, and infer the optimal policy (action) from it.



Deep-RL: Policy Gradient

DQN: Approximate Q from a DNN, and infer the optimal policy (action) from it.

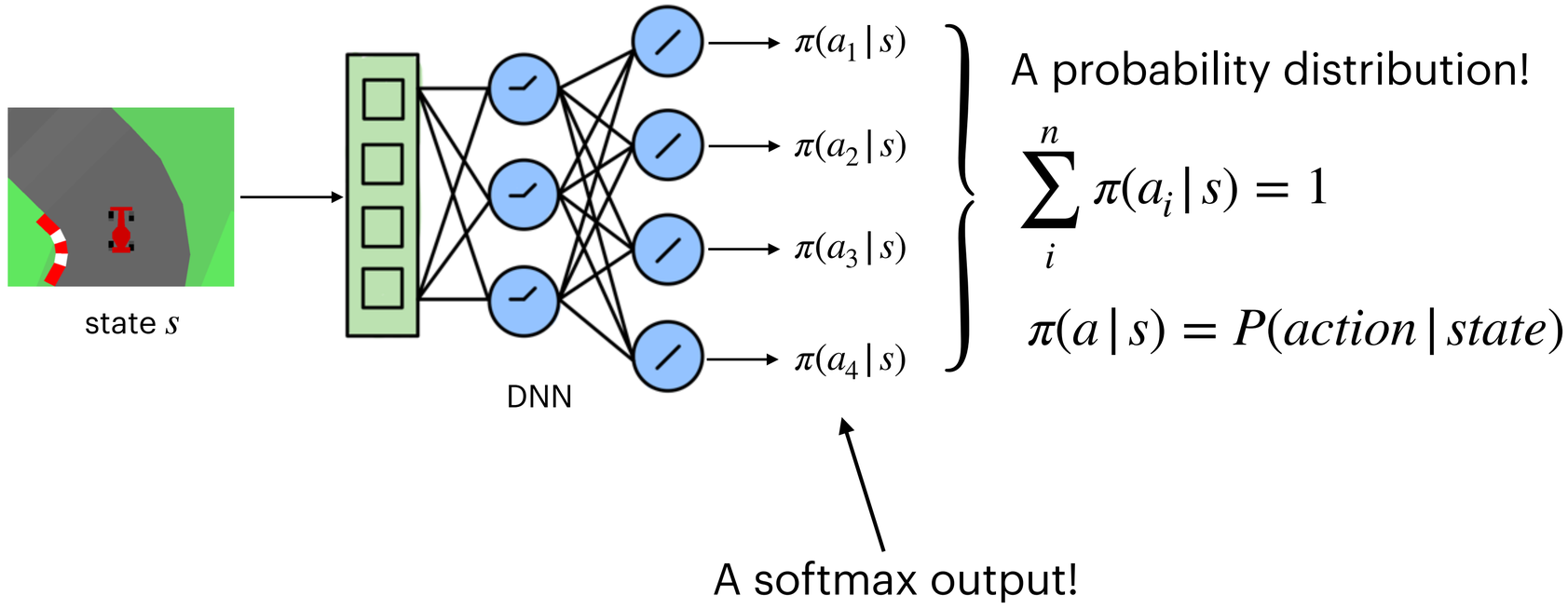
Policy Gradient: Estimate the policy π directly from the DNN.



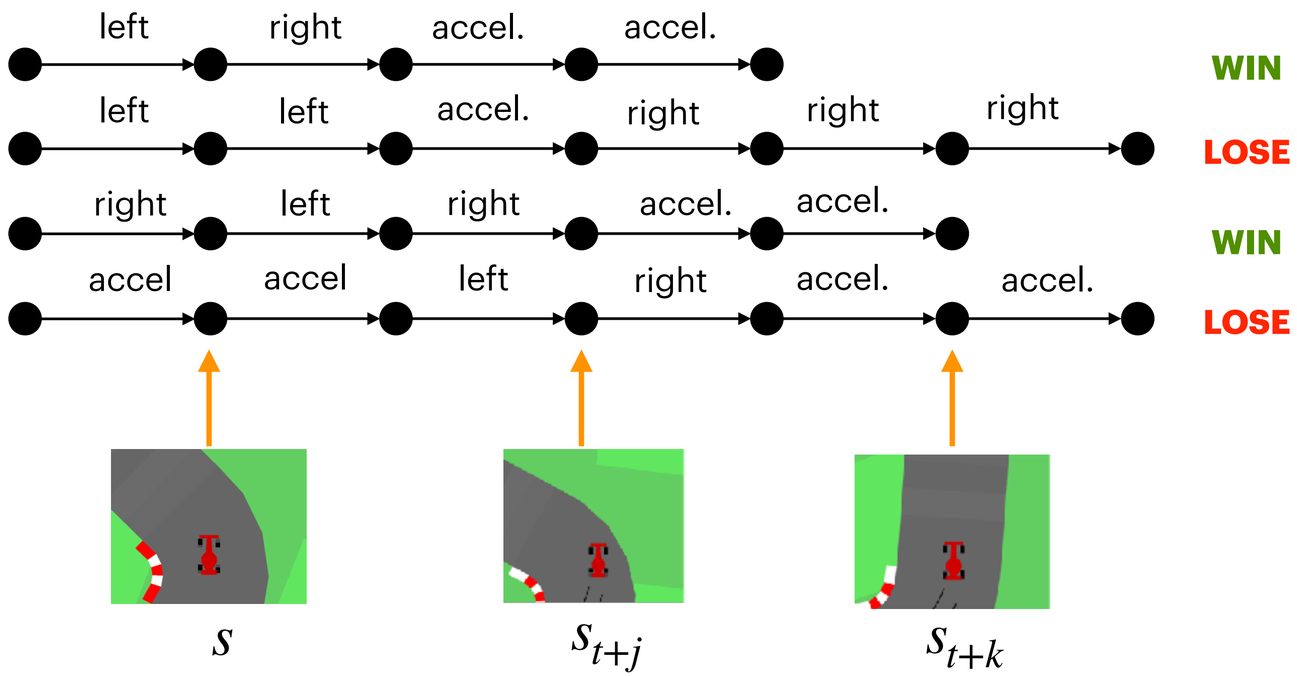
Deep-RL: Policy Gradient

DQN: Approximate Q from a DNN, and infer the optimal policy (action) from it.

Policy Gradient: Estimate the policy π directly from the DNN.



Deep-RL: Policy Gradient - Training



General scheme

1.Runs a policy for a while.

2.Increase probability of actions that lead to high rewards from a given state s.

3.Decrease probability of action that lead to low/no rewards from a given state s.

Deep-RL: Policy Gradient - Training

Training algorithm

Initialize θ ← DNN parameters

for $episode \backsim \pi_\theta$

$\{s_i, a_i, r_i\}_{i+1}^{T-1} \leftarrow episode$

for $t = 1 \rightarrow T - 1$

$\nabla \leftarrow \nabla_\theta log \pi_\theta(a_t | s_t) R_t$ ← DNN params gradients; Discounted total reward

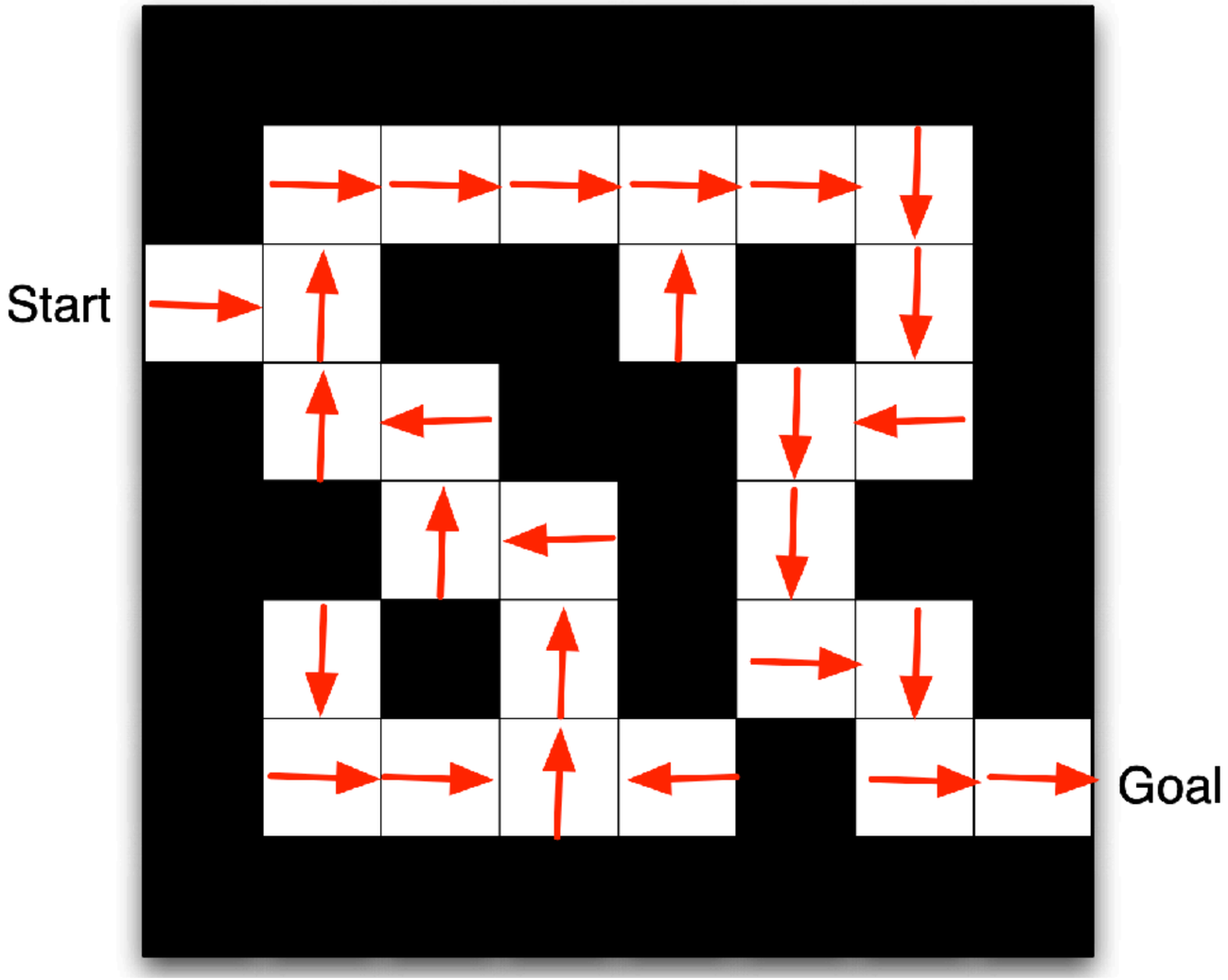
$\theta \leftarrow \theta - \alpha \nabla$ ← Gradient descent

return θ

If R_t is high, increase the probability of seeing the actions given by π in the future.

If R_t is negative or zero, decrease that probability.

Deep-RL: Policy Gradient - Training

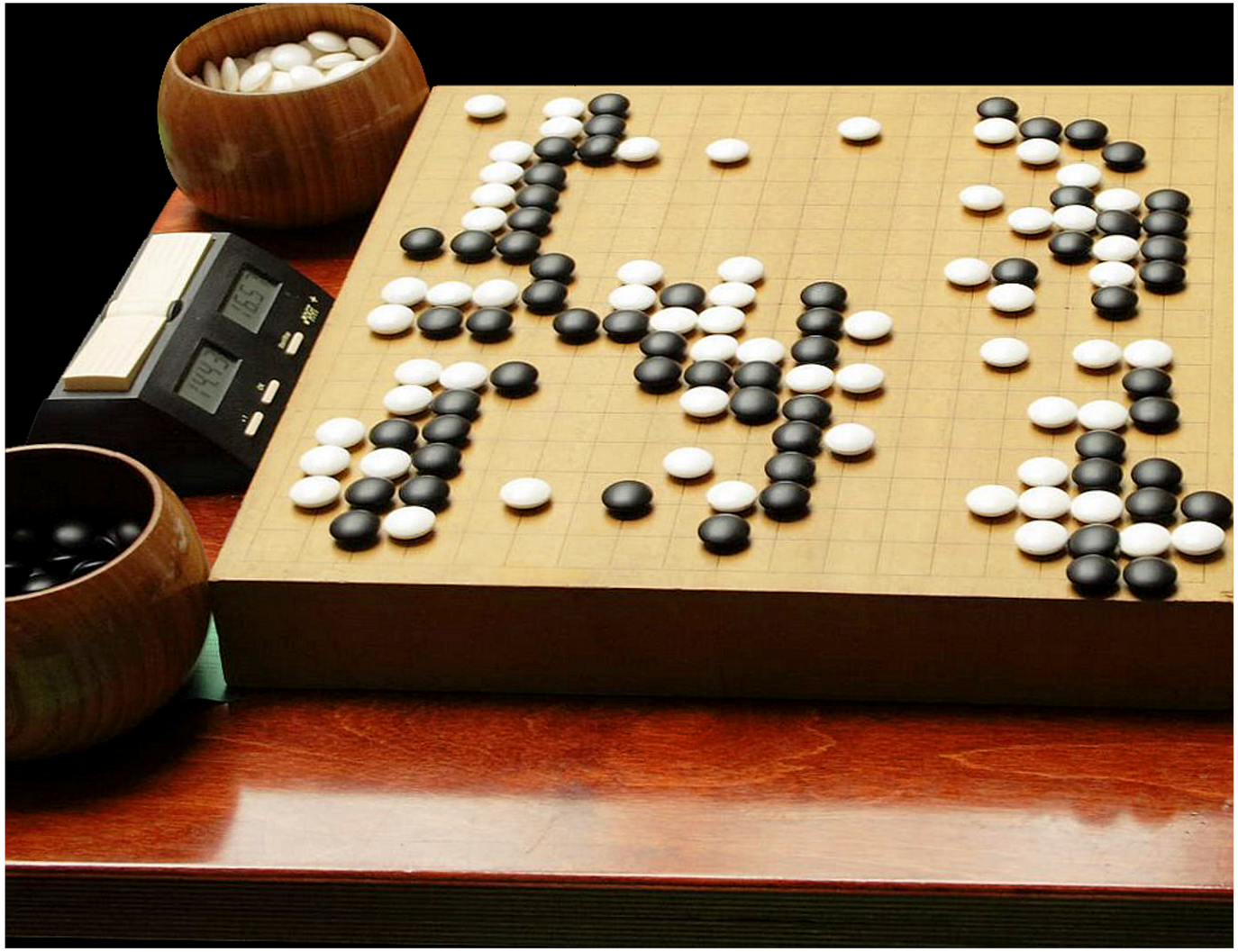


Source: https://www.davidsilver.uk/wp-content/uploads/2020/03/intro_RL.pdf

An example of a policy π for a maze game

- The image of the maze with the agent in a specific cell is an state s.
- The red arrow illustrates an action estimated by π from current state.

Deep-RL: Policy Gradient - AlphaGo (2016)



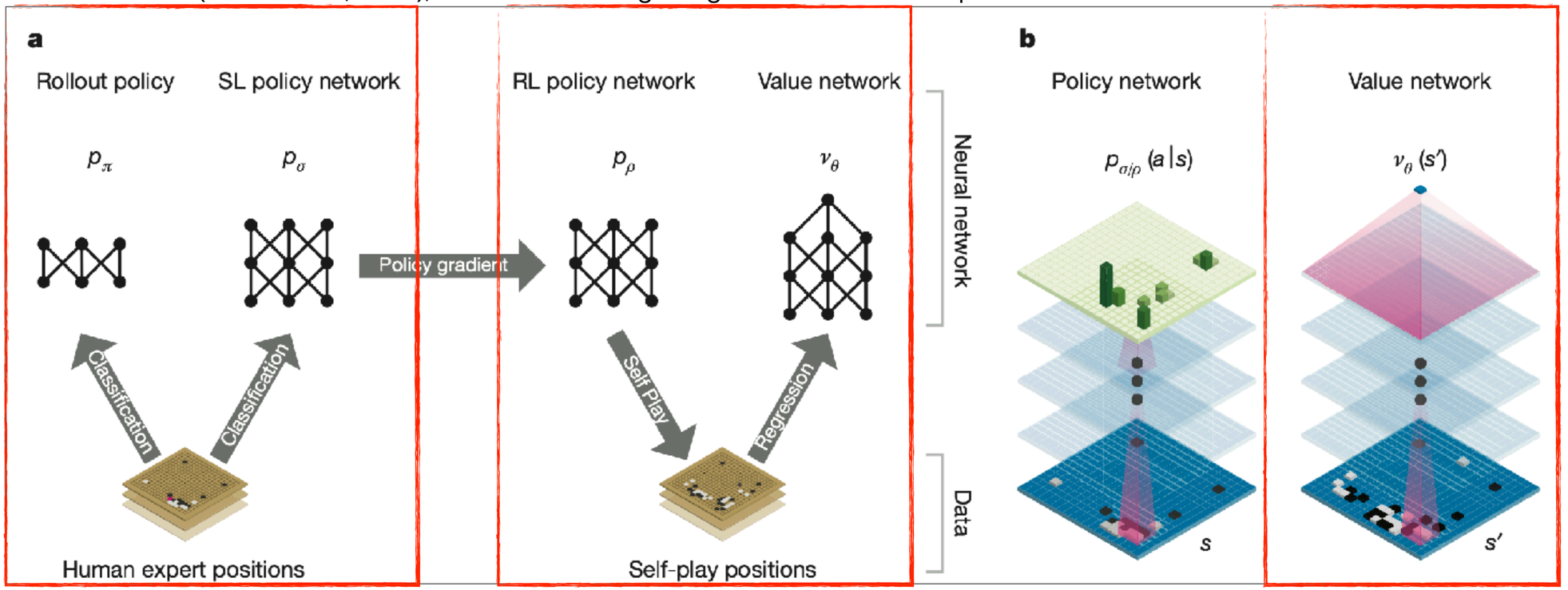
Source: https://es.wikipedia.org/wiki/Go#/media/Archivo:Go-Equipment-Narrow-Black.png

Go

- High complexity game > Chess.
- Complete board 19x19.
- A cell can be black, white or empty.
- A policy network requires 19x19 and 19x19 output.

Deep-RL: Policy Gradient - AlphaGo (2016)

(Silver et. al., 2016), Nature. Mastering the game of Go with deep neural networks and tree search

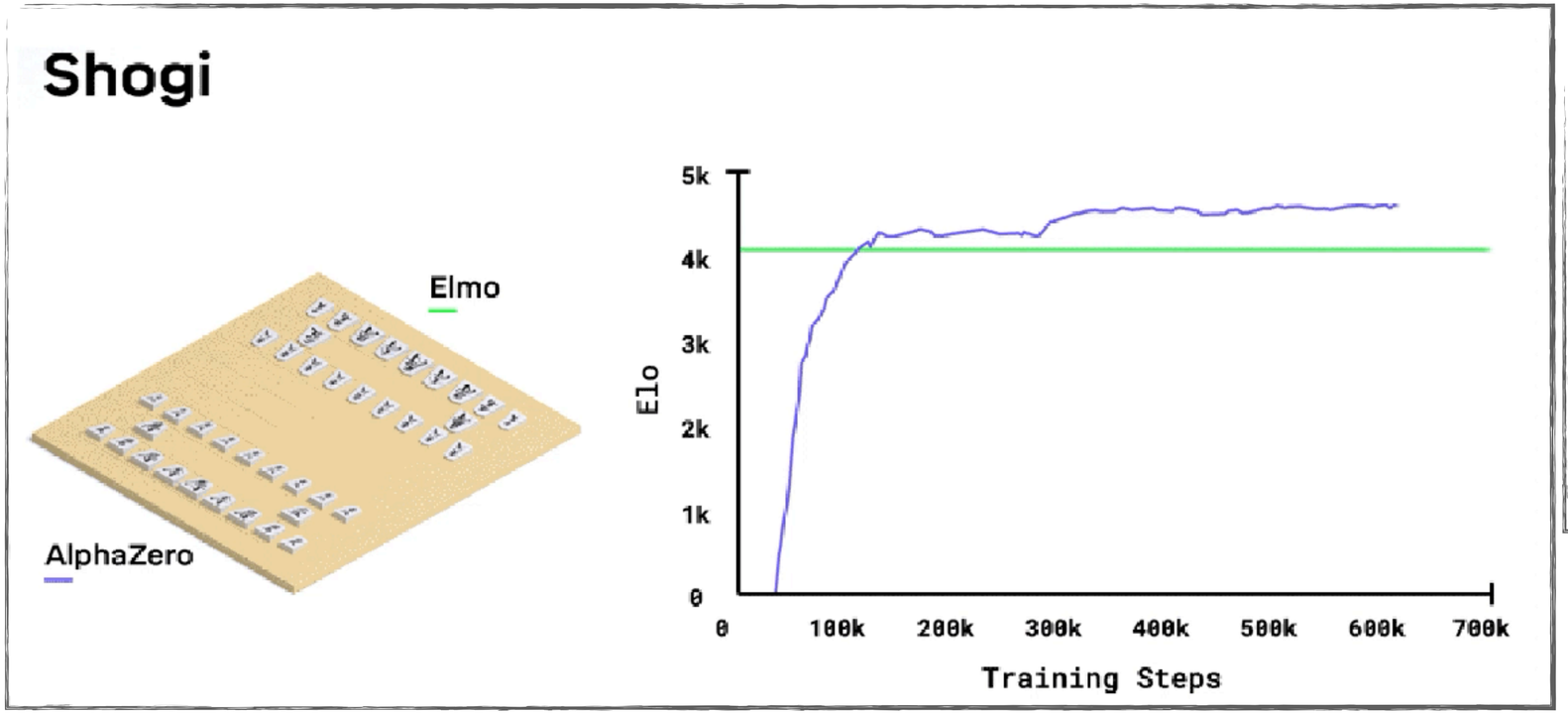


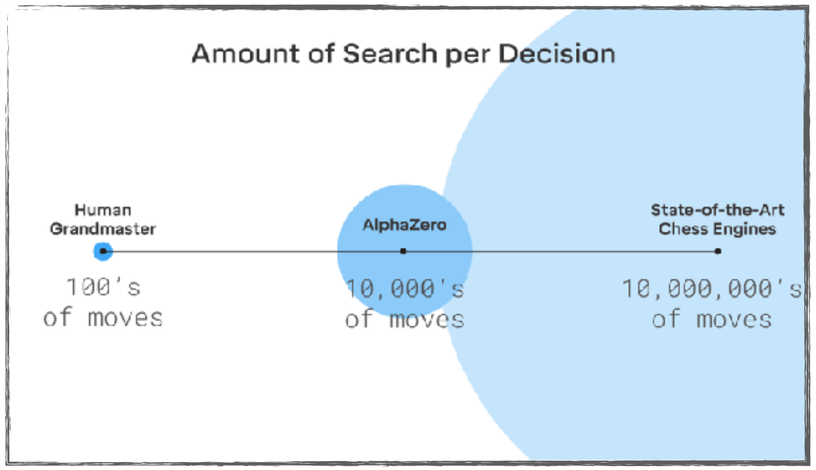
1. **(Supervised)** A fast rollout policy and supervised learning (SL) policy network are trained to predict human expert moves in a data set of positions.

2. **(Reinforcement)** A RL policy network is initialized to the SL policy network, and is then improved by policy gradient learning to maximize the outcome.

3. **(Value / heuristic network (Q-Function))** value network similarly uses many convolutional layers with parameters, but outputs a scalar that predicts the expected outcome in

Deep-RL: Policy Gradient - AlphaZero (2017)





Monte Carlo Tree Search (MCTS) at its core.

Source: https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go

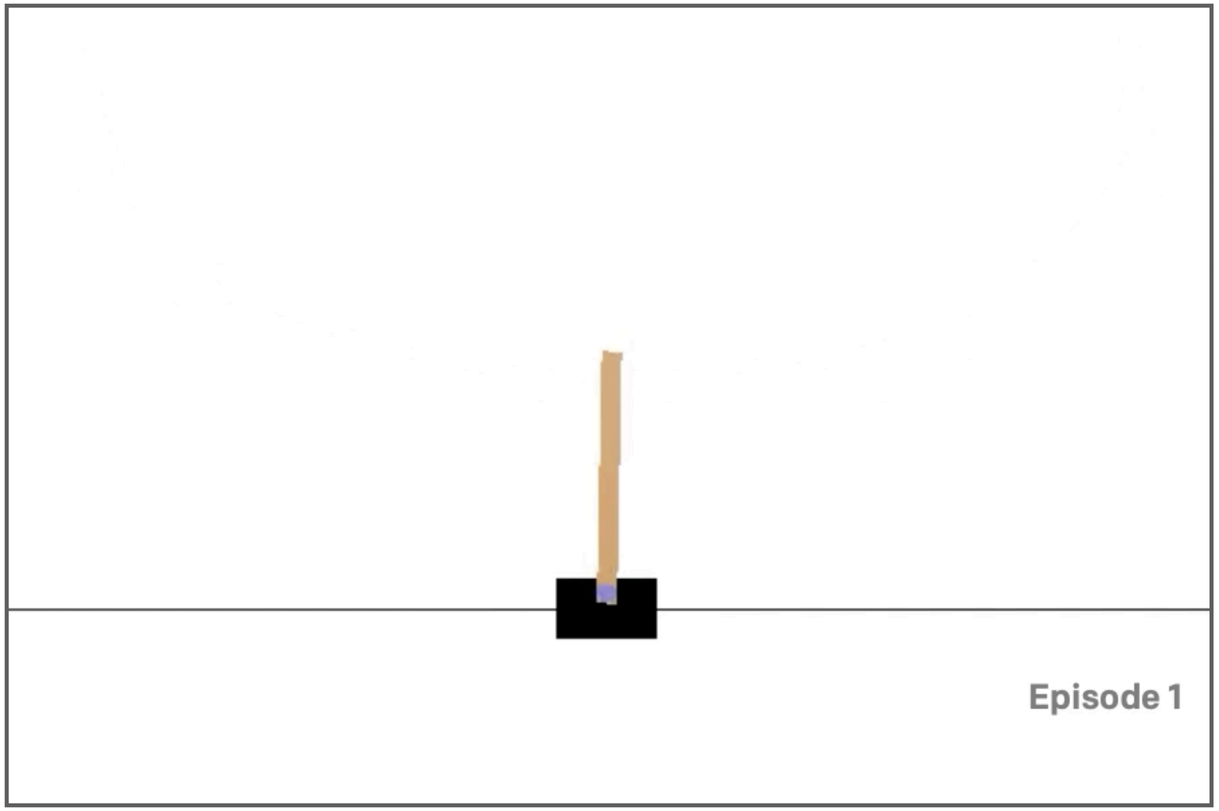
See the video

3. Python practice

OpenAI Gym: CartPole-v1

CartPole-v1 environment

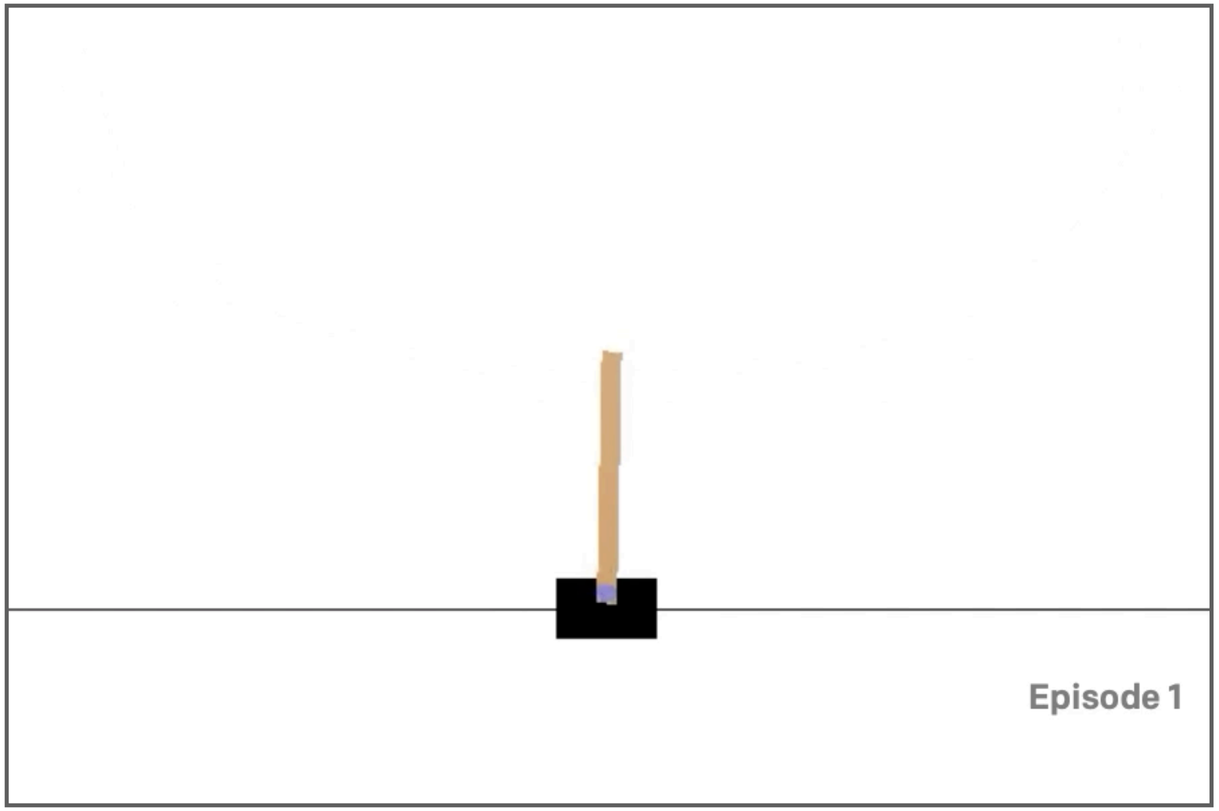
- **Objective**: pole starts vertical. the objective is to prevent the pole from falling over.
- **State**: Cart position, cart velocity, pole angle, pole rotation rate.
- **Actions**: Left force (-1) and right force (+1).
- Episode **ends** when the pole is more than 15 degrees from vertical.



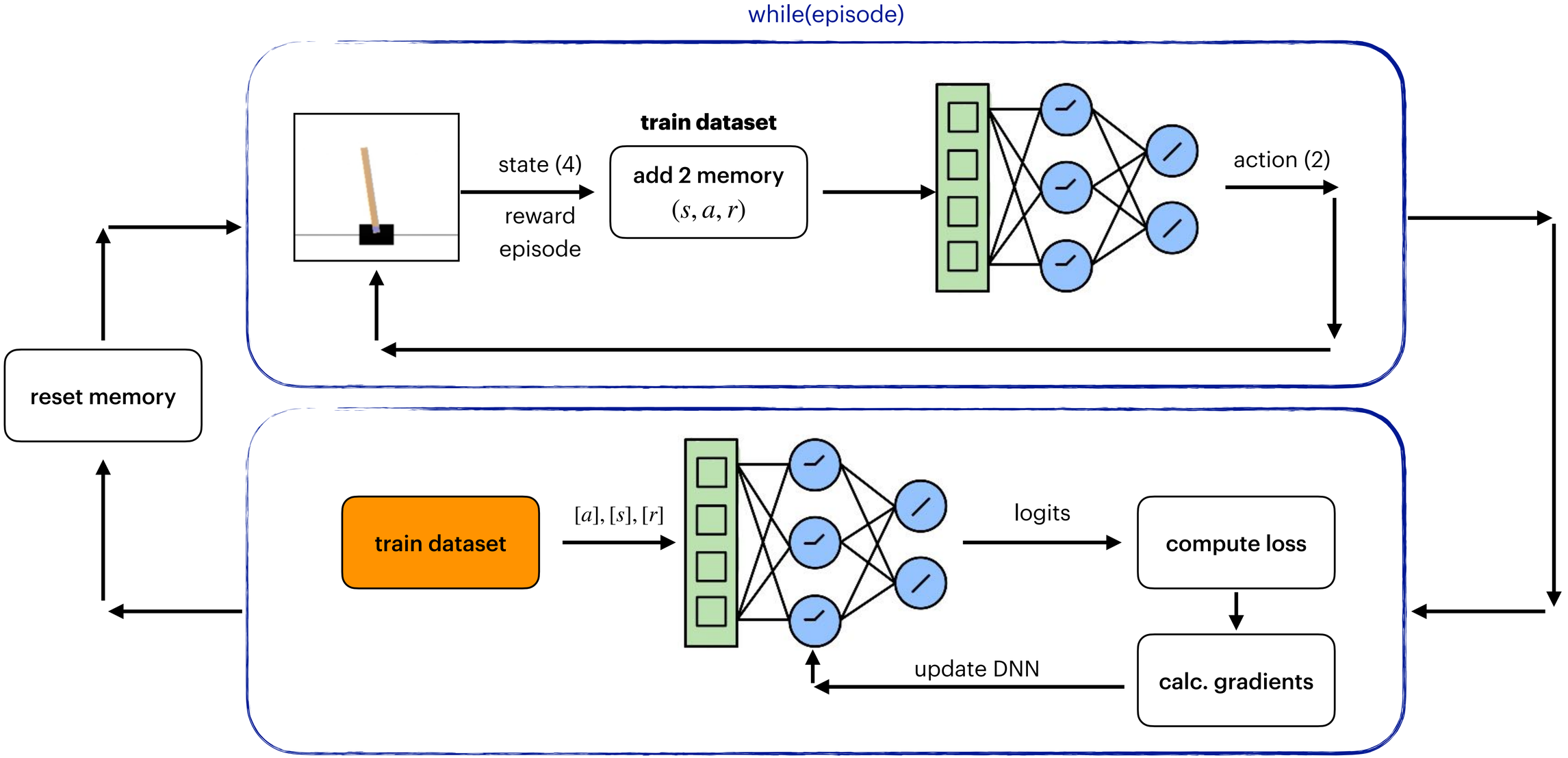
OpenAI Gym: CartPole-v1

Deep-RL approach

- We are going to implement a **Policy Gradient** approach in order to maximize the likelihood of actions that result in large rewards.
- This is the **same to minimize the negative likelihood** of the same actions.



CartPole-v1: Training framework (1 epoch)



OpenAI Gym: CartPole-v1

Open the Colab Notebook

Open the Colab Notebook (Solution)

These notebooks were taken from: https://github.com/aamini/introtodeeplearning/tree/master/lab3

4. Conclusions

Conclusions

- Environment could be hard to simulate: actions, states, rewards.
- Partial observability impose an important constraint to how fast an agent can get a reward (positive or negative).
- Deep-RL it is a relatively new paradigm when compared to other deep learning paradigms: could be hard to get base implementations for industrial projects.
- SOTA still have limitations in problems with non-fully observable environments and a high degree of complexity in actions and rewards.