Học viện
Công nghệ Bưu chính Viễn thông

PTIT x JAIST AI Summer School 2024

Mastering Reinforcement Learning: Insights from Winning the AI Challenge 2023 with Tetris Battle

Presenter: TS. Trần Tiến Công

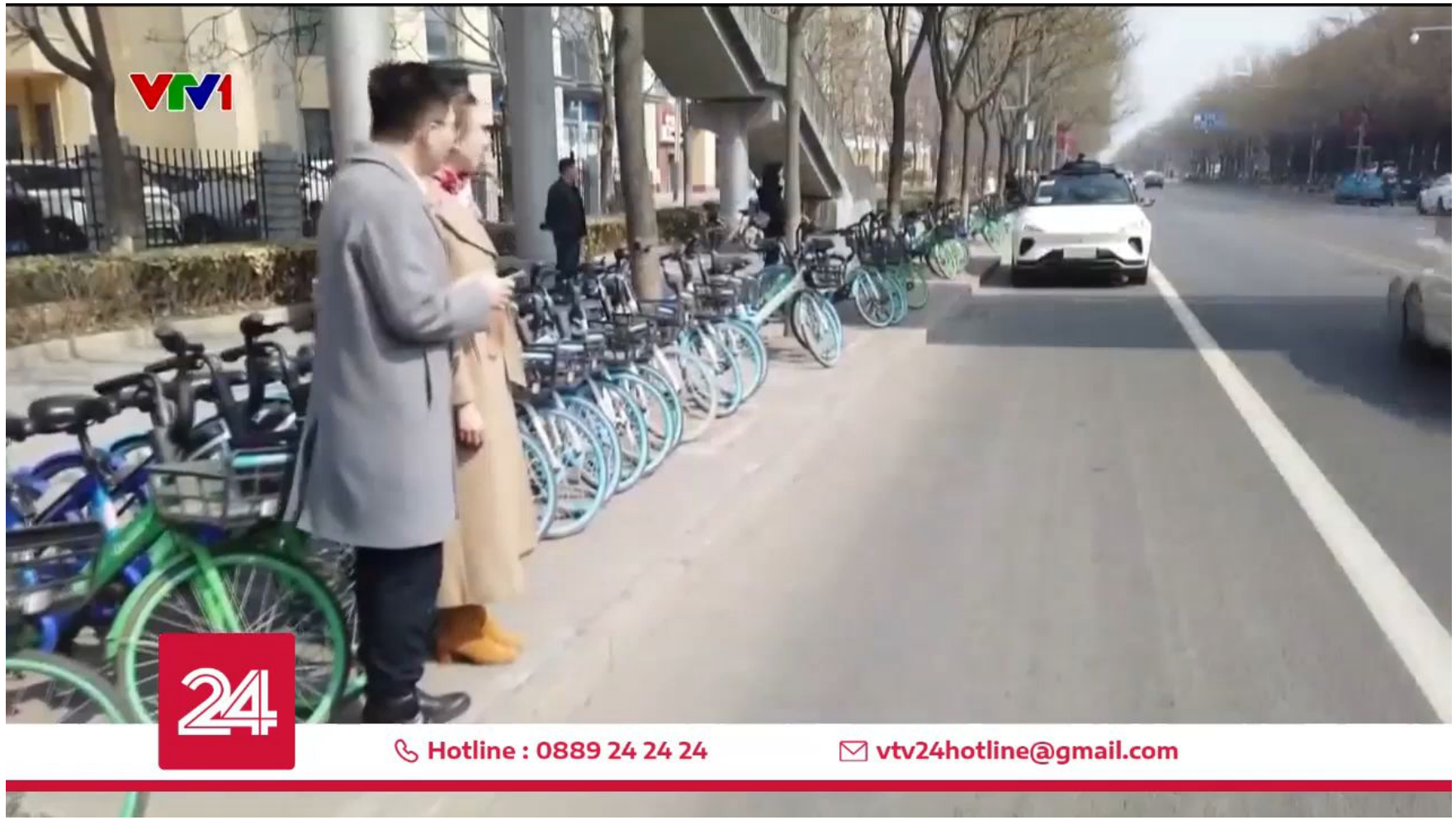
Email: congtt@ptit.edu.vn

Contents

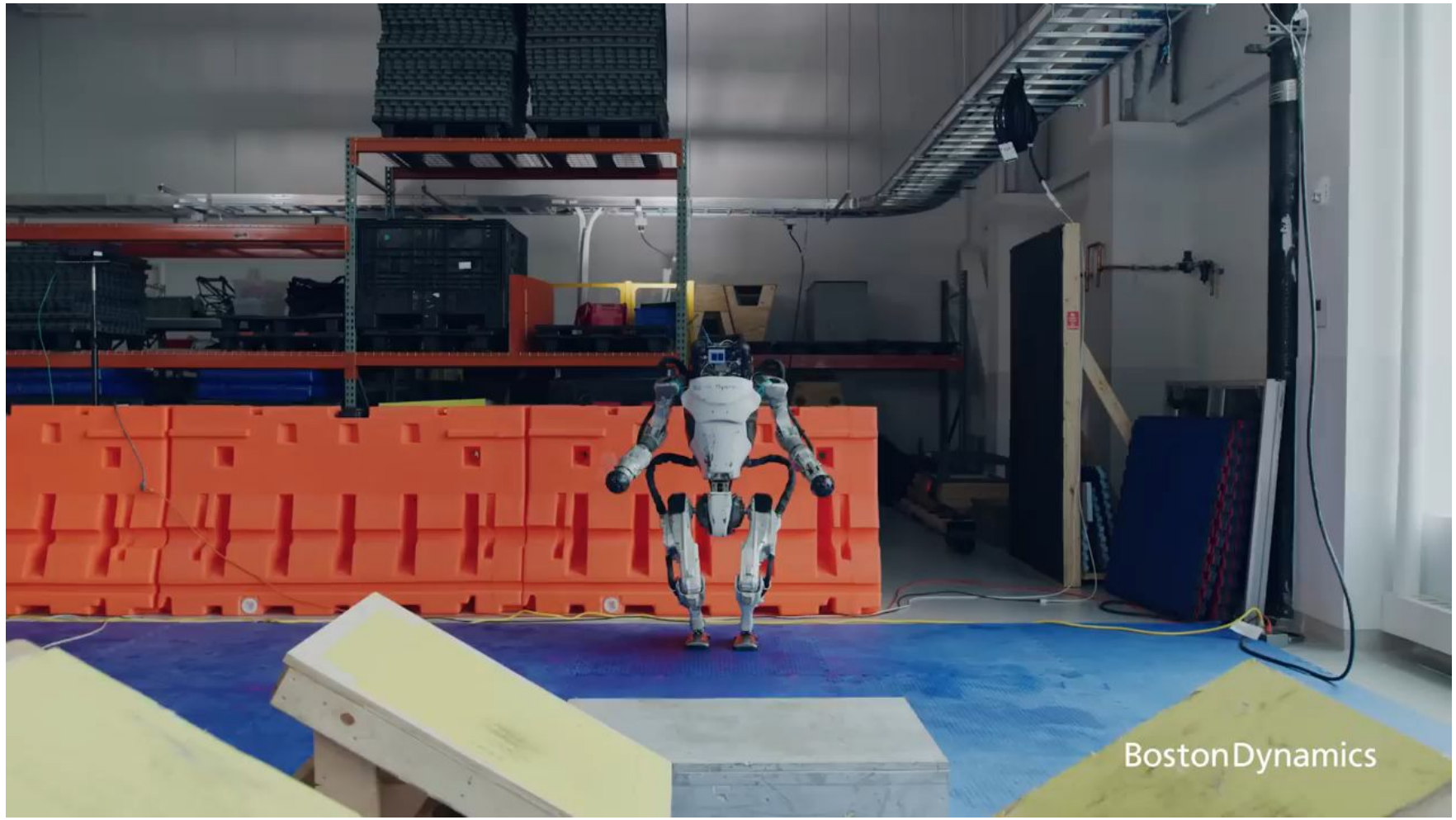
Reinforcement learning

What do you know about the applications of RL beside games?

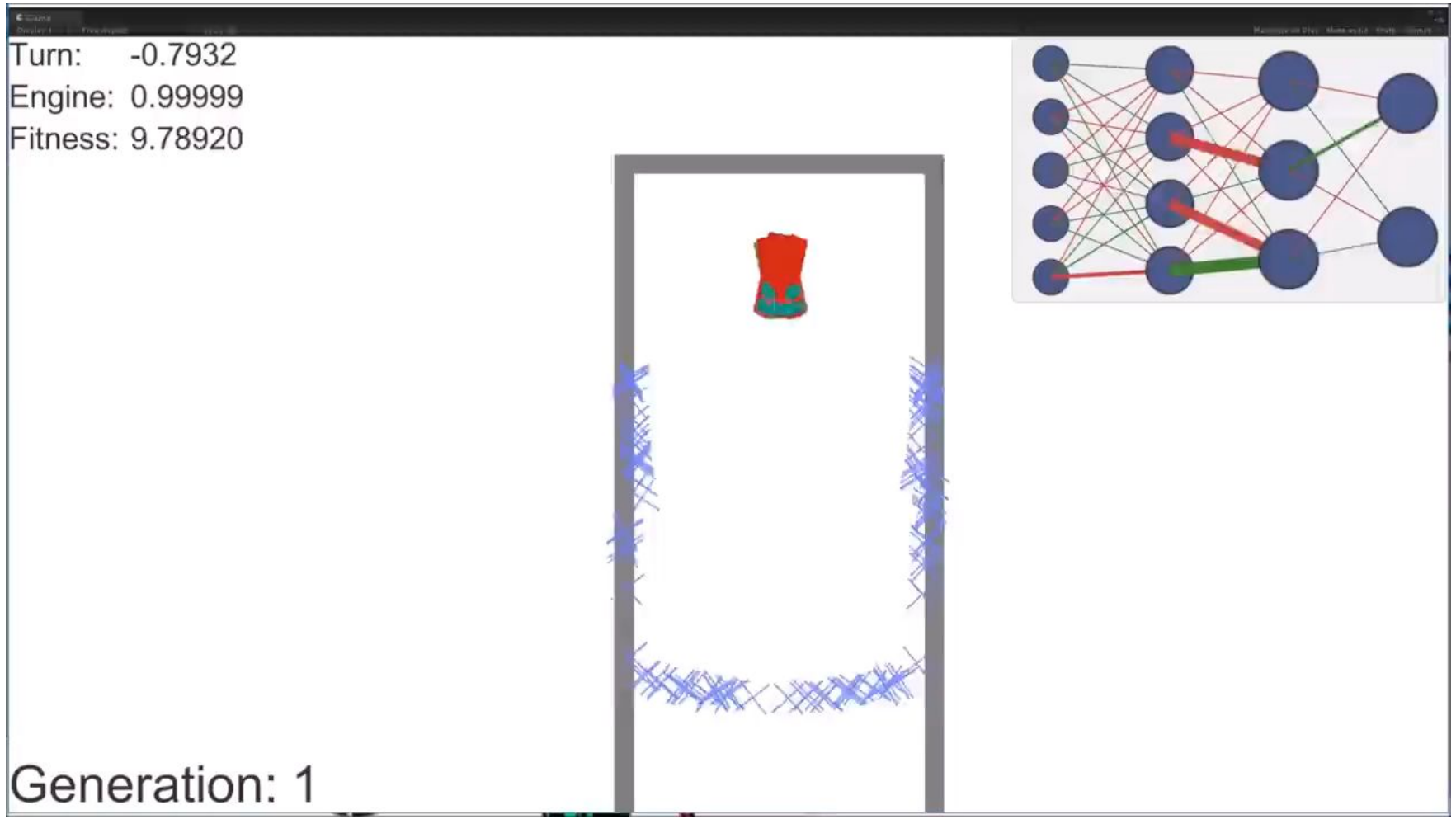
Autonomous car



Autonomous Robots

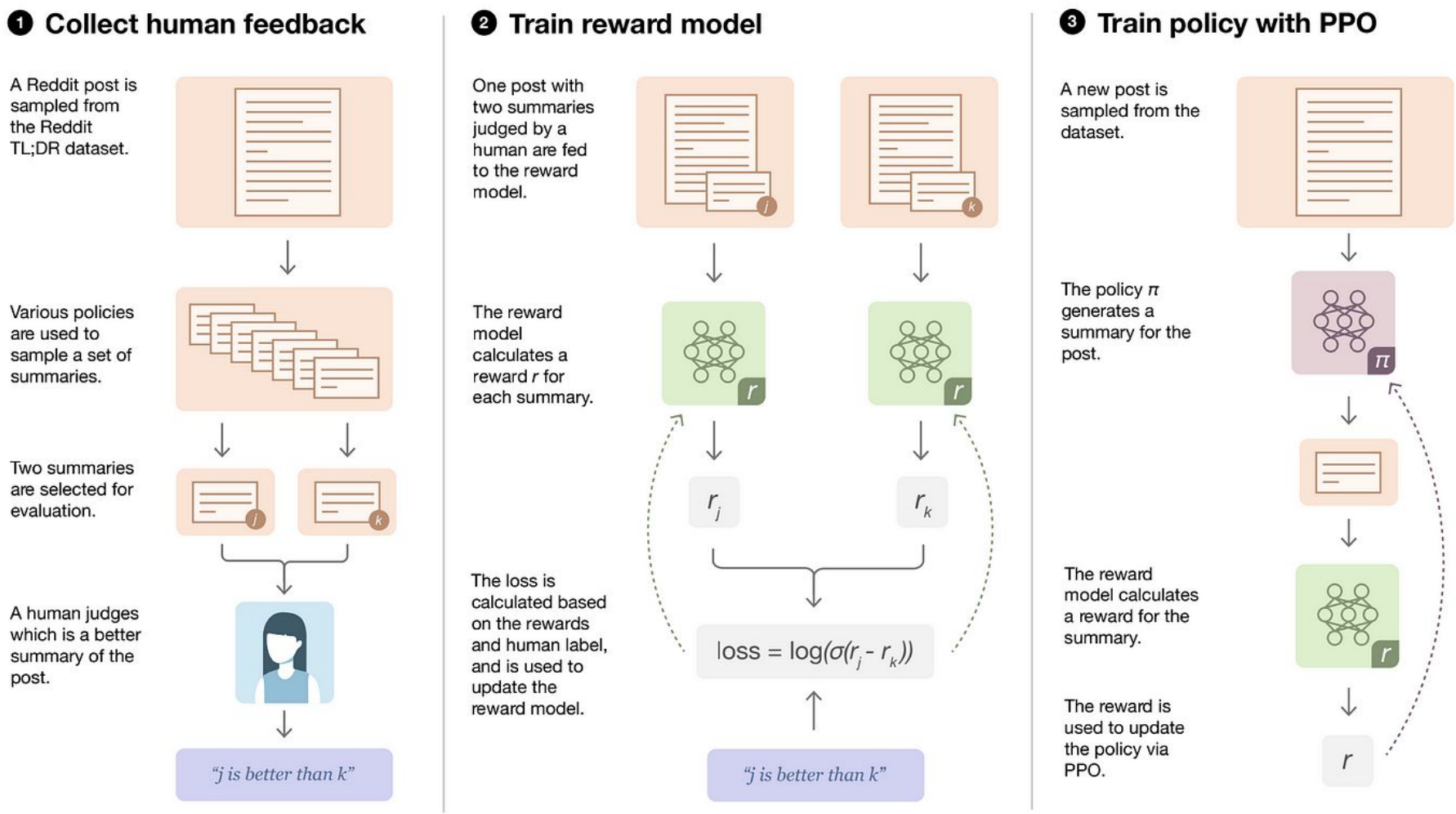


Secret behind the autonomous cars/robots



Secret behind the autonomous cars/robots

ChatGPT


1 Collect human feedback
A Reddit post is sampled from the Reddit TL;DR dataset.
Various policies are used to sample a set of summaries.
Two summaries are selected for evaluation.
A human judges which is a better summary of the post.
"j is better than k"
2 Train reward model
One post with two summaries judged by a human are fed to the reward model.
The reward model calculates a reward r for each summary.
The loss is calculated based on the rewards and human label, and is used to update the reward model.
loss = log(σ(r_j - r_k))
"j is better than k"
3 Train policy with PPO
A new post is sampled from the dataset.
The policy π generates a summary for the post.
The reward model calculates a reward for the summary.
The reward is used to update the policy via PPO.

Reinforcement learning

Task

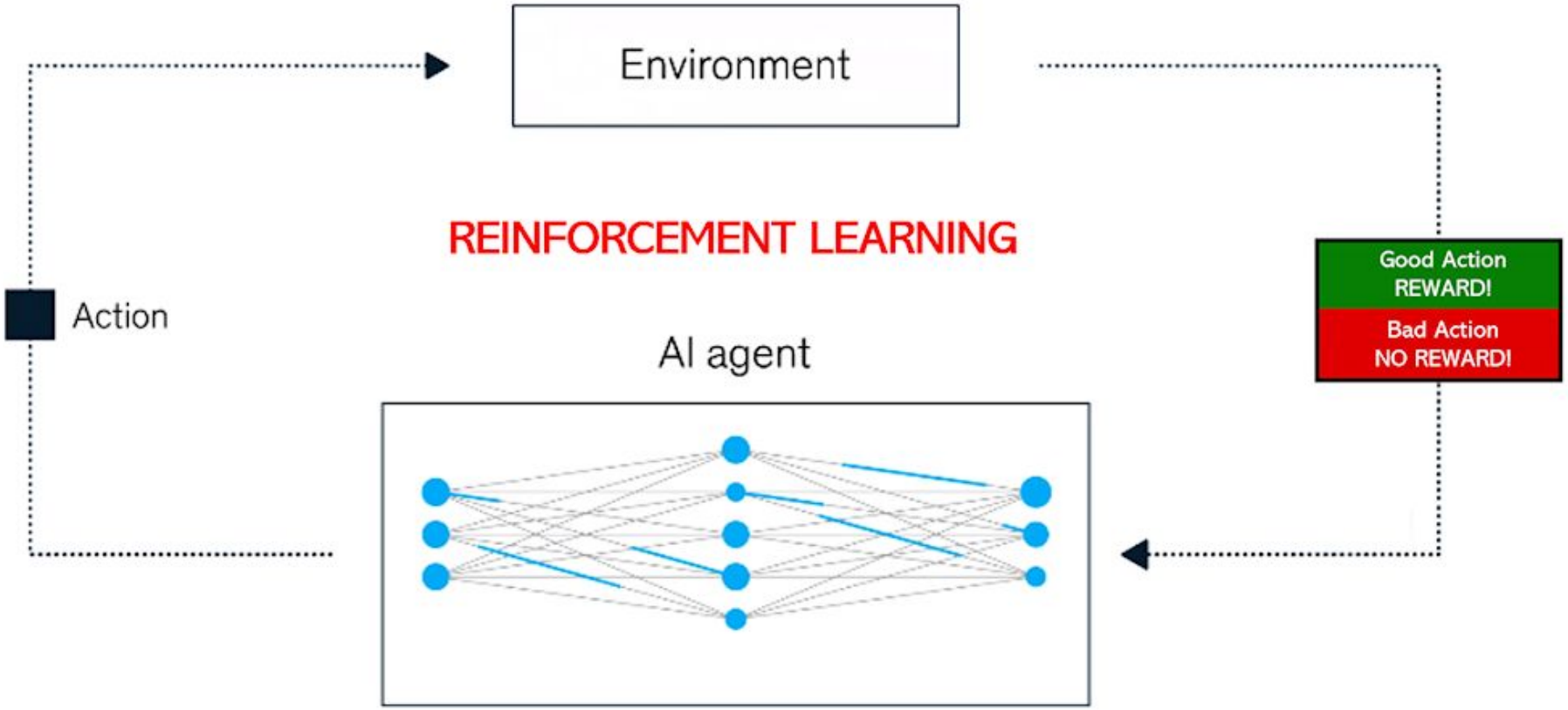
Learn how to behave successfully to achieve a goal while interacting with an external environment

- *Learn via experiences!*

Examples

- Game playing: player knows whether it win or lose, but not know how to move at each step
- Control: a traffic system can measure the delay of cars, but not know how to decrease it.

RL is learning from interaction



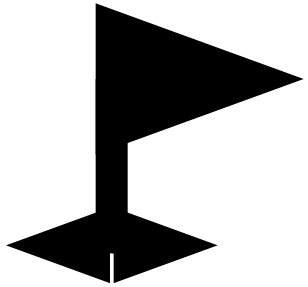
Robot in a room

			+1
			-1
START			

actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT



reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

states
actions
rewards

what is the solution?

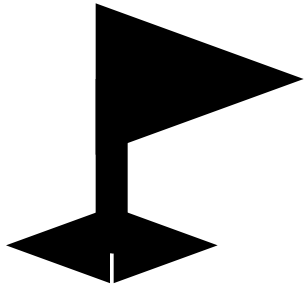
Robot in a room

			+1
			-1
START			

actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT

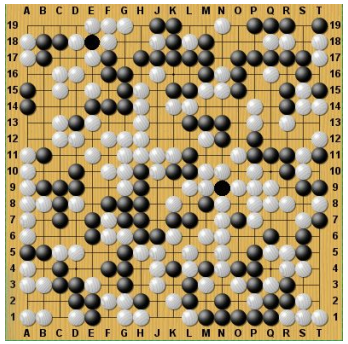


reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

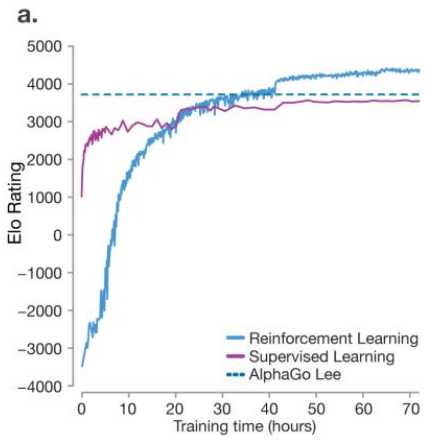
what's the strategy to achieve max reward?
what if the actions were deterministic?

Alpha Go

- Learning how to beat humans at ‘hard’ games (search space too big)
- Far surpasses (Human) Supervised learning
- Algorithm learned to outplay humans at chess in 24 hours



State: Board State
Actions: Valid Moves
Reward: Win or Lose



https://deepmind.com/documents/119/agz_unformatted_nature.pdf

Markov Decision Process (MDP)

set of states S, set of actions A, initial state S_0

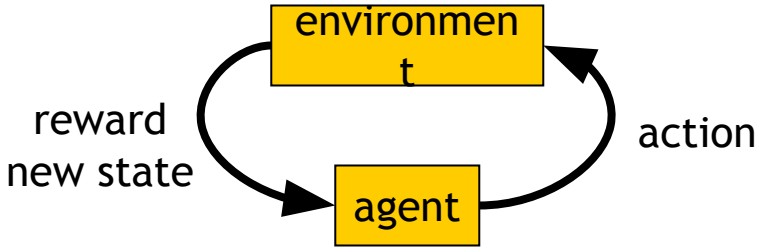
transition model P(s,a,s')

- P([1,1], up, [1,2]) = 0.8

reward function r(s)

- r([4,3]) = +1

goal: maximize cumulative reward in the long run

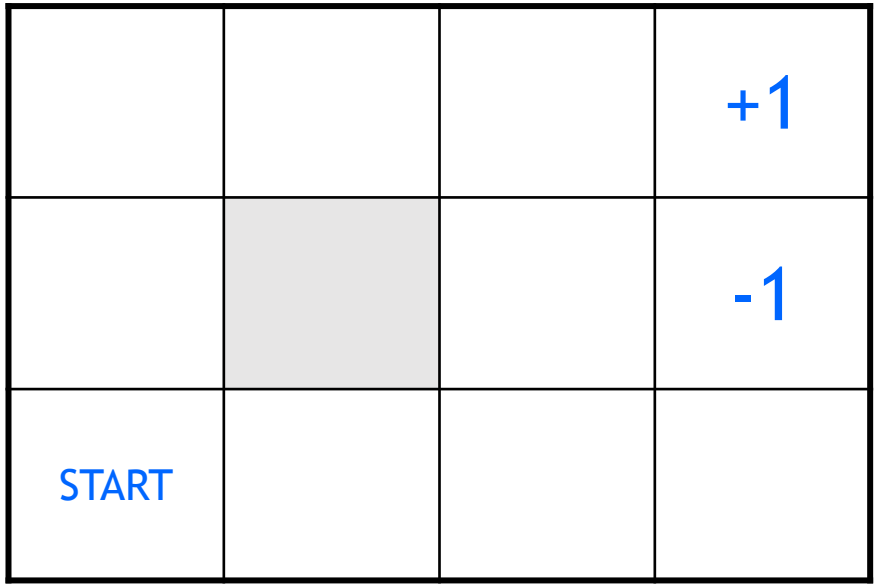


policy: mapping from S to A

- $\pi(s)$ or $\pi(s,a)$ (deterministic vs. stochastic)

reinforcement learning

- transitions and rewards usually not available
- how to change the policy based on experience
- how to explore the environment



Computing return from rewards

episodic (vs. continuing) tasks

- “game over” after N steps
- optimal policy depends on N; harder to analyze

additive rewards

- $V(s_0, s_1, \ldots) = r(s_0) + r(s_1) + r(s_2) + \ldots$
- infinite value for continuing tasks

discounted rewards

- $V(s_0, s_1, \ldots) = r(s_0) + \gamma * r(s_1) + \gamma^2 * r(s_2) + \ldots$
- rewards received in the future are worth less than rewards received immediately
- value bounded if rewards bounded

Bellman Equation

For a given state s, the Bellman equation expresses the value function $V(s)$ in terms of the value of successor states:

$$V(s) = \max_a \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma V(s')]$$

Where:

- s is the current state.
- a is the action taken from state s.
- s' is the successor state resulting from action a.
- $P(s' \mid s, a)$ is the probability of transitioning to state s' from state s by taking action a.
- $R(s, a, s')$ is the reward received after transitioning from state s to state s' via action a.
- γ is the discount factor (where $0 \leq \gamma \leq 1$), which accounts for the present value of future rewards.

Robot in a room

Assume $P(s' \mid s, a) = 1$, reward 0 for each step, and $\gamma = 0.9$

Bellman Equation:

$$V(s) = \max_a R(s, a, s') + 0.9V(s')$$

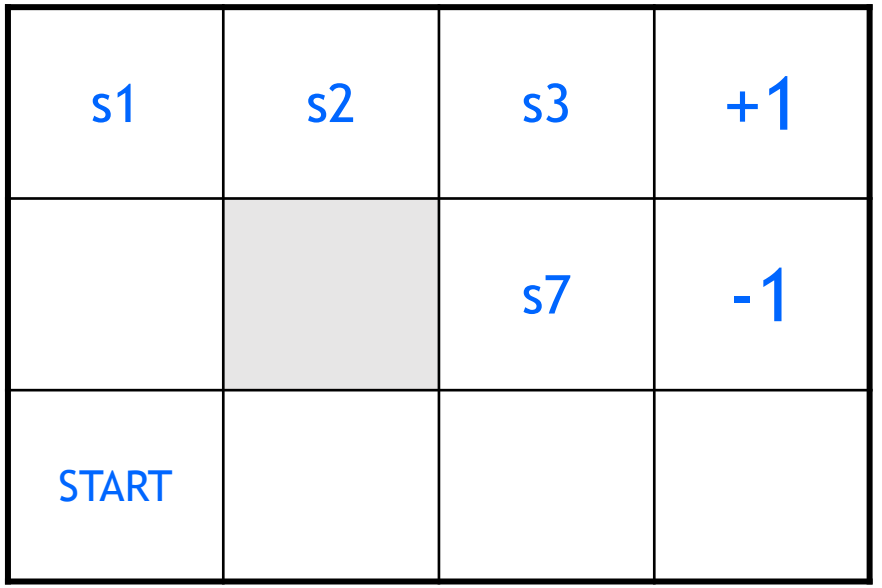
Solve using dynamic programming

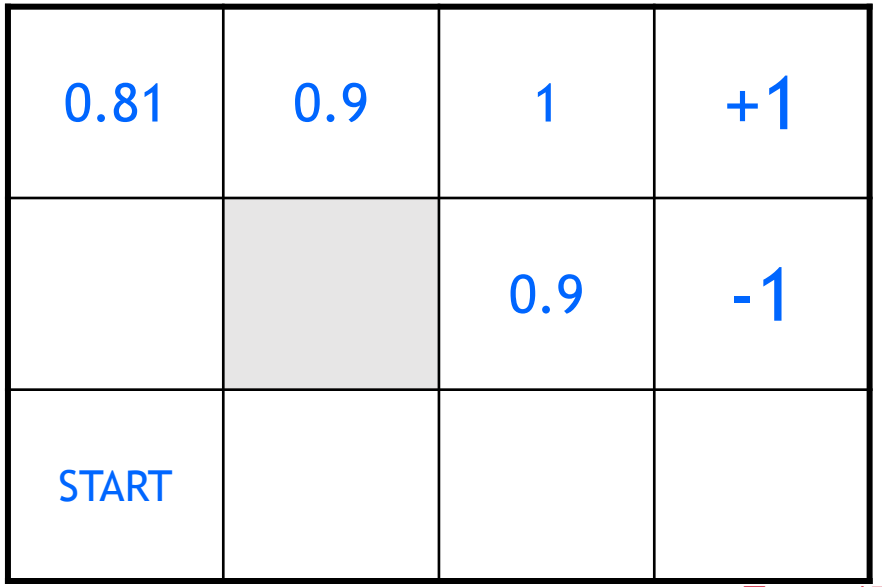
V(s3) = 1

For 2nd block:

V(s2) = max [R(s,a) + γV(s')], V(s')= 1, and R(s, a)= 0, because there is no reward at this state.

V(s2)= max[0.9(1)]=> V(s)= max[0.9]=> **V(s2) =0.9**





Robot in a room

For 3rd block:

V(s1) = max [R(s,a) + γV(s`)], V(s')= 0.9, and R(s, a)= 0, because there is no reward at this state also.

V(s1)= max[0.9(0.9)]⇒ V(s3)= max[0.81]⇒ **V(s1) =0.81**

For 4th block:

V(s5) = max [R(s,a) + γV(s`)], V(s')= 0.81, and R(s, a)= 0, because there is no reward at this state also.

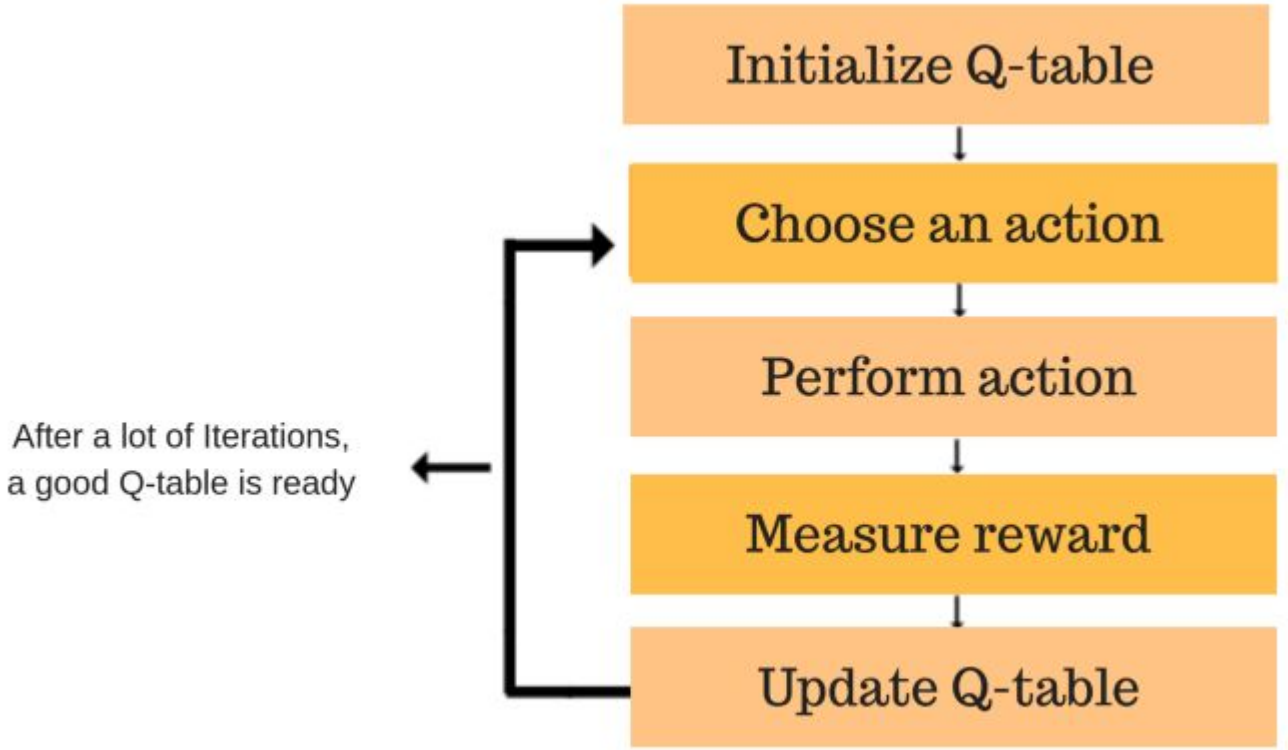
V(s5)= max[0.9(0.81)]⇒ V(s5)= max[0.81]⇒ **V(s5) =0.73**

s1	s2	s3	+1
s5		s7	-1
START	s10	s11	s12

0.81	0.9	1	+1
0.73		0.9	-1
START	0.73	0.81	0.73

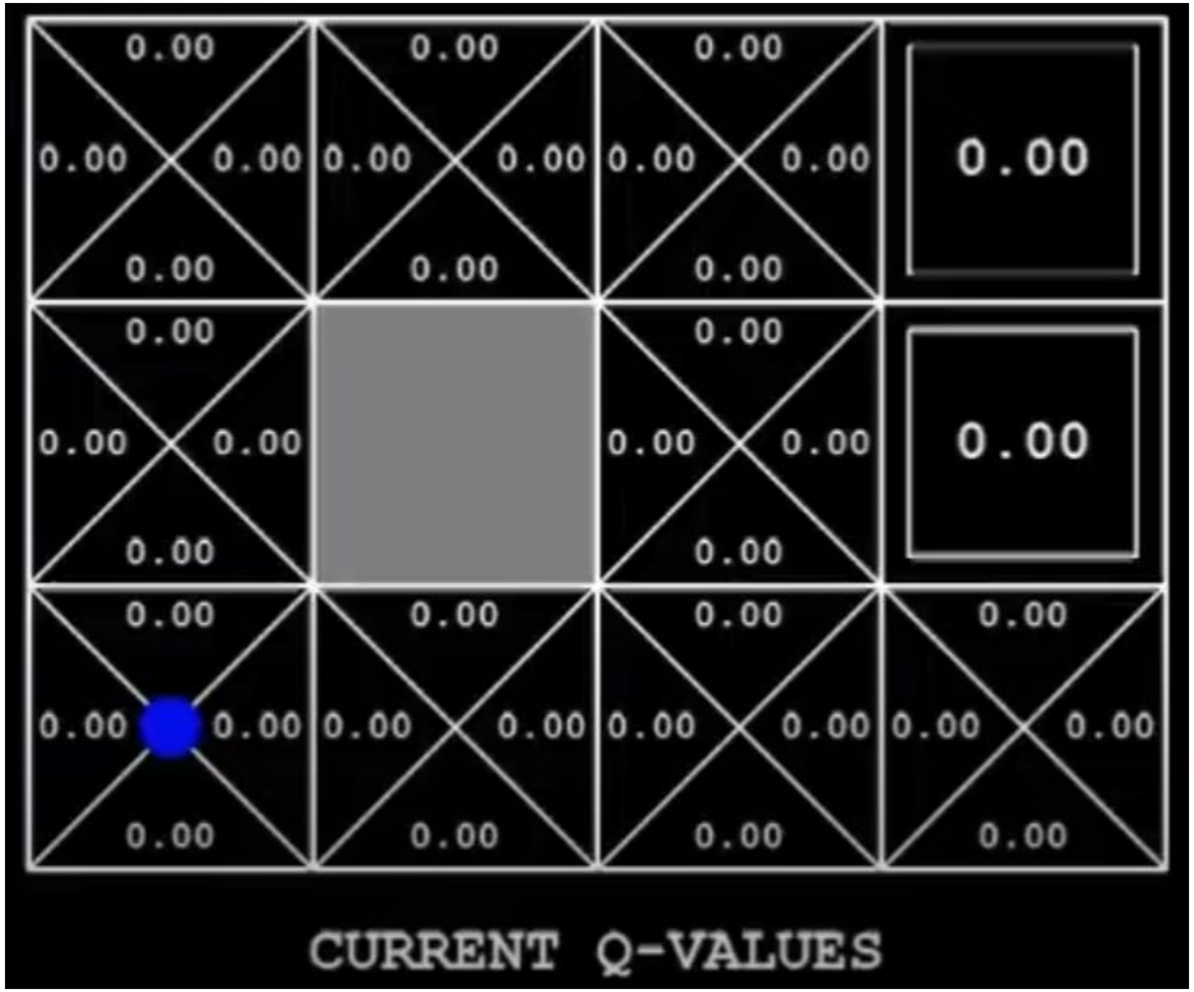
Q-learning

Q-learning is an algorithm that uses an approximation of the Bellman equation to learn the optimal policy for an agent



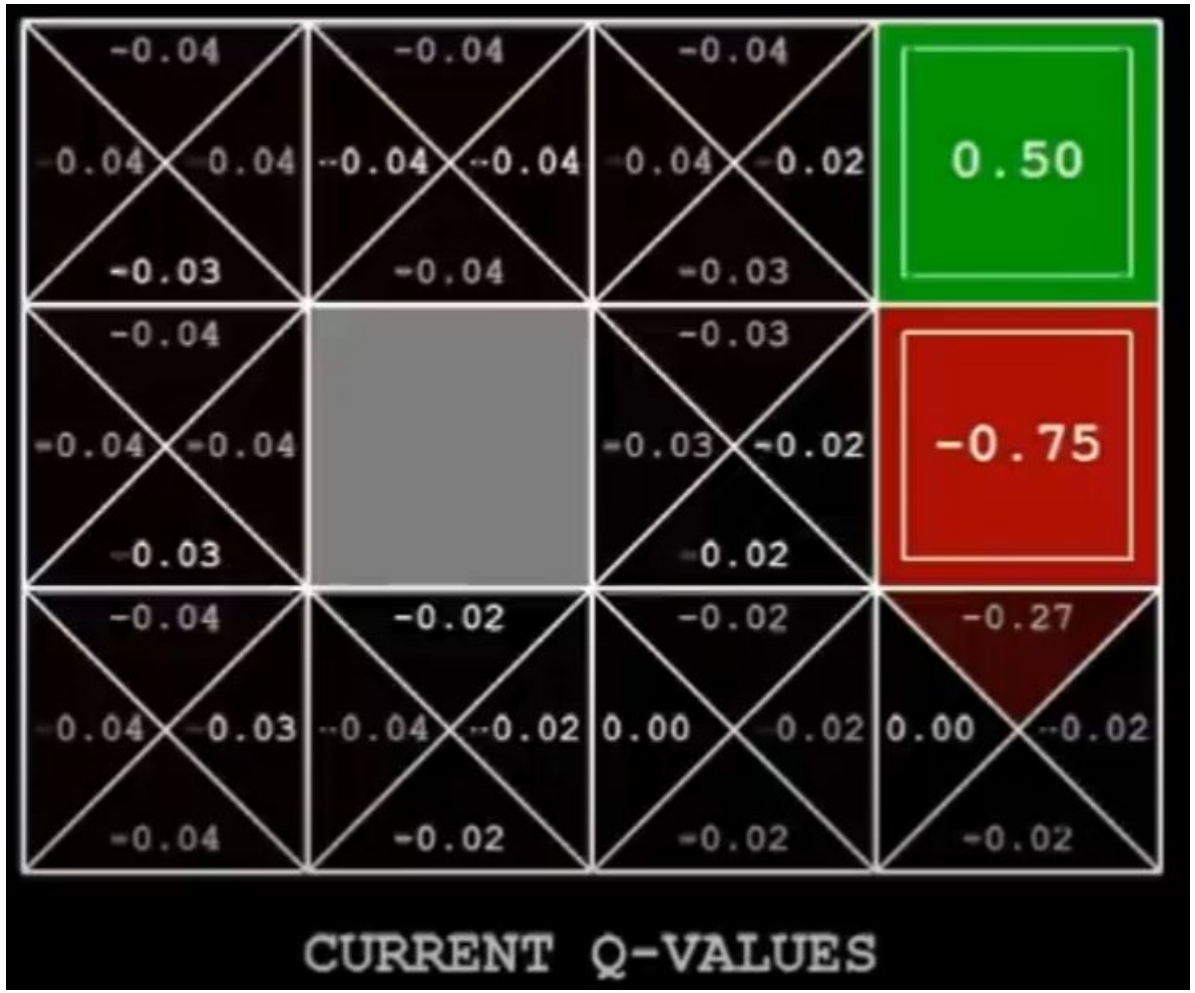
	↑	→	↓	←
s1	0	…	…	…
s2	…	0.1	…	…
s3	…	…	…	…
s4	…	…	0.2	…
s5	…	…	…	…
s6	…	…	…	…
s7	…	…	…	…
s8	…	…	…	…

Initialization

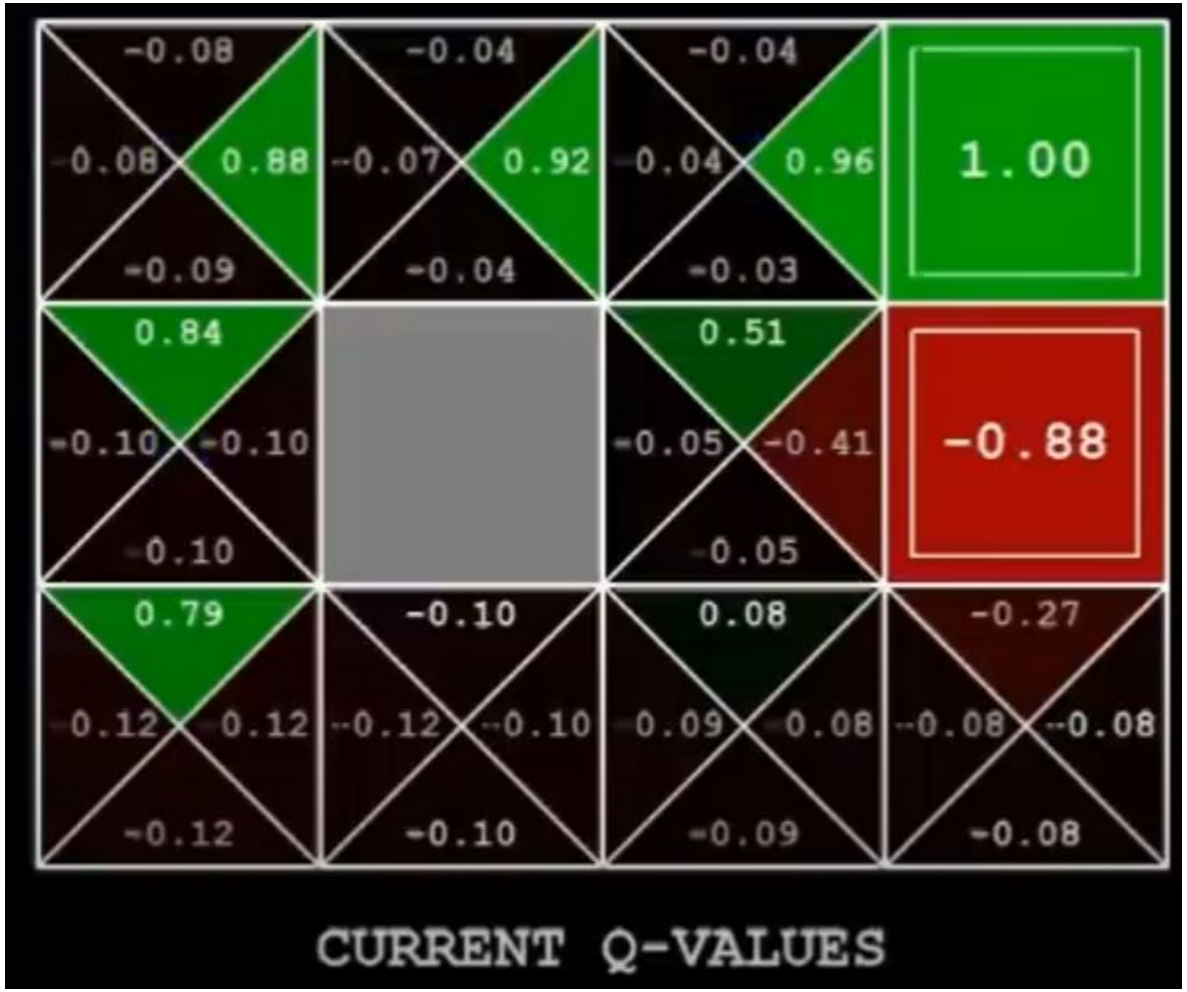


http://ai.berkeley.edu/reinforcement.html

After some iterations



Good enough

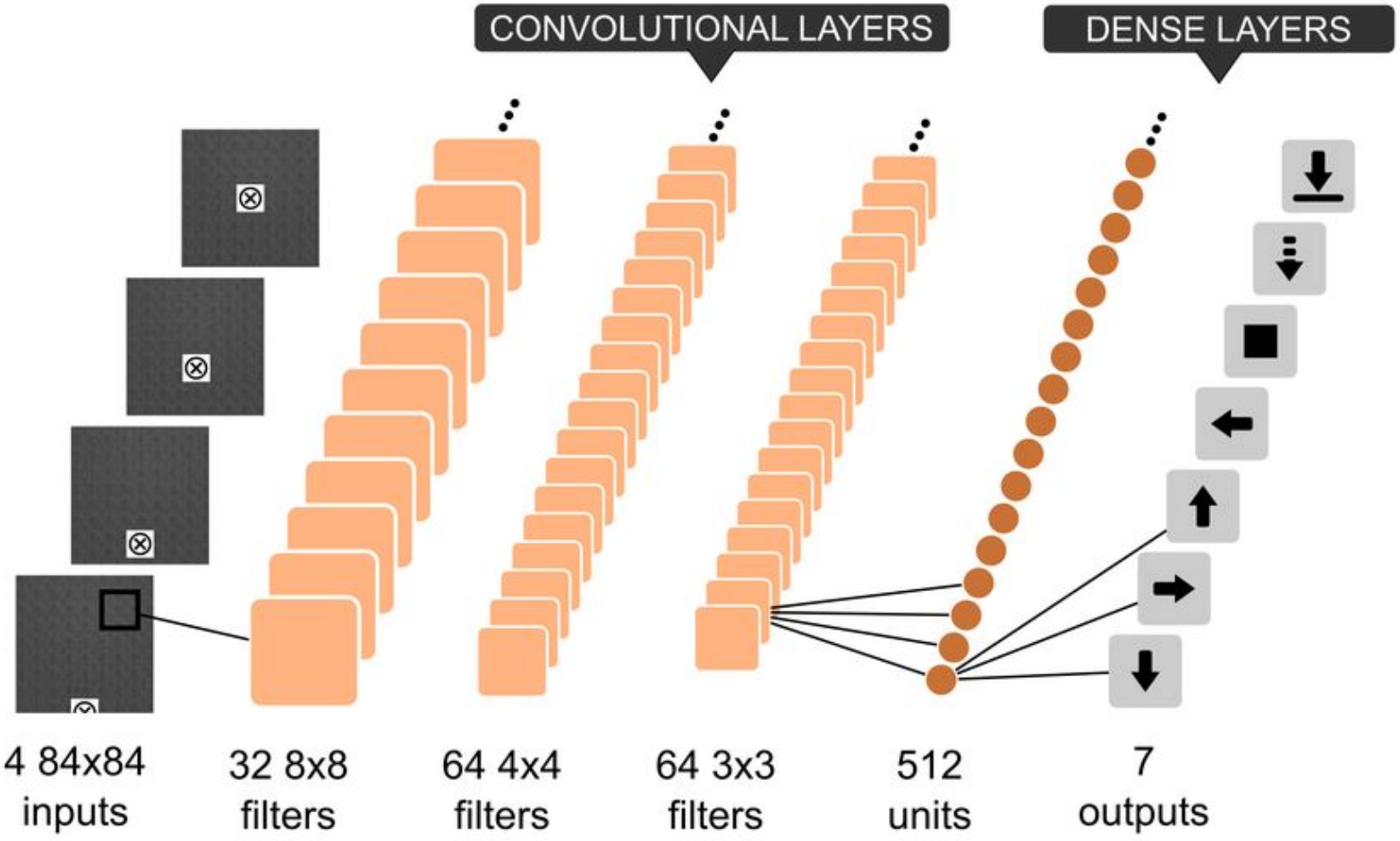

-0.08
-0.08
0.88
-0.09
-0.04
-0.07
0.92
-0.04
-0.04
-0.04
0.96
-0.03
1.00
0.84
-0.10
-0.10
0.10
0.51
-0.05
-0.41
0.05
-0.88
0.79
-0.12
0.12
-0.12
-0.10
-0.10
0.08
-0.09
0.08
-0.09
-0.27
-0.08
-0.08
-0.08
CURRENT Q-VALUES

Q-learning

updating Q values which denotes value of performing action *a* in state *s*

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Deep Q-learning



PTIT AI Challenge 2023 - Tetris Battle

PTIT AI Challenge 2023 - Tetris Battle

Tetris được phát triển bởi Alexey Pajitnov vào năm 1984. Luật chơi của Tetris cơ bản bao gồm:

– Người chơi sẽ bắt đầu với một màn hình trống và một khối hình, gọi là Tetrimino, xuất hiện từ trên cùng của màn hình.
– Người chơi phải xoay và di chuyển khối hình này để đặt nó vào vị trí một cách phù hợp trên màn hình.
– Mục tiêu của người chơi là xếp các khối hình lại với nhau để tạo thành một hàng ngang đầy đủ.
– Khi một hàng ngang được tạo ra, nó sẽ biến mất và các khối hình ở trên hàng này sẽ rơi xuống để điền vào khoảng trống.
– Điểm số của người chơi sẽ tăng lên mỗi khi tạo ra một hàng ngang hoàn chỉnh và hàng ngang được tạo ra càng nhiều thì điểm số càng cao.
– Trò chơi sẽ kết thúc khi các khối hình không thể được xếp vào màn hình nữa.

PTIT AI Challenge 2023 - Tetris Battle

Với TetrisBattle (chế độ đối kháng), các luật đối kháng sẽ được áp dụng như sau (lưu ý rằng các luật này có thể thay đổi tùy vào cài đặt của người tổ chức trò chơi):

– Thời gian giới hạn của Tetris Battle là 2 phút.
– Trong Tetris Battle, người chơi có thể thấy trước 5 Tetriminos tiếp theo.
– Ở chế độ Double, người chơi có thể sử dụng “dòng rác” để gây trở ngại cho đối thủ. Khi người chơi gửi đi n “dòng”, nó sẽ không thể bị xóa và được gửi đến bảng của đối thủ. Những “dòng rác” này sẽ được tạo ra sau khi Tetrimino của đối thủ rơi xuống đất. Nếu người chơi đạt được m “dòng” bằng cách rơi Tetrimino đó, số lượng “dòng rác” sẽ giảm xuống còn n – m.
– Ở chế độ Double, khi bảng của người chơi không còn chỗ trống để đặt Tetrimino tiếp theo, những “garbage lines” sẽ được xóa bỏ, và đối thủ của người chơi sẽ bị hạ gục (KO).

PTIT AI Challenge 2023 - Tetris Battle

Các nhóm sẽ phải lập trình một agent sử dụng AI sao cho ở mỗi bước (tương ứng với 0.1 giây) trong trò chơi có thể đưa ra được hành động (action) tương ứng của khung đỡ, bao gồm:

- 0: “NOOP”,
- 1: “hold”,
- 2: “drop”,
- 3: “rotate_right”,
- 4: “rotate_left”,
- 5: “right”,
- 6: “left”,
- 7: “down”

Agent này được đóng gói trong 1 hàm có input là file json chứa thông tin hiện tại của ván đấu trong từng bước và output là 1 trong 7 action kể trên. Hàm này sẽ được nạp và chạy ở từng bước trong máy chủ trò chơi. Yêu cầu về tốc độ của hàm này: trả lại action với thời gian dưới 0.1 giây.

PTIT AI Challenge 2023 - Tetris Battle

Đội Toi_Yeu_PTIT gồm 3 thành viên (B20DCCN295 Nguyễn Mạnh Hùng, B20DCCN279 Nguyễn Trọng Hoàng, B20DCCN274 Lê Phúc Hoàng thuộc Khoa Công nghệ thông tin 1 giành giải nhất cuộc thi với chiến thắng tuyệt đối 2-0 về hiệp đấu

Thank you!