A Playful Dive into the World of RL

Haozhe Tian,

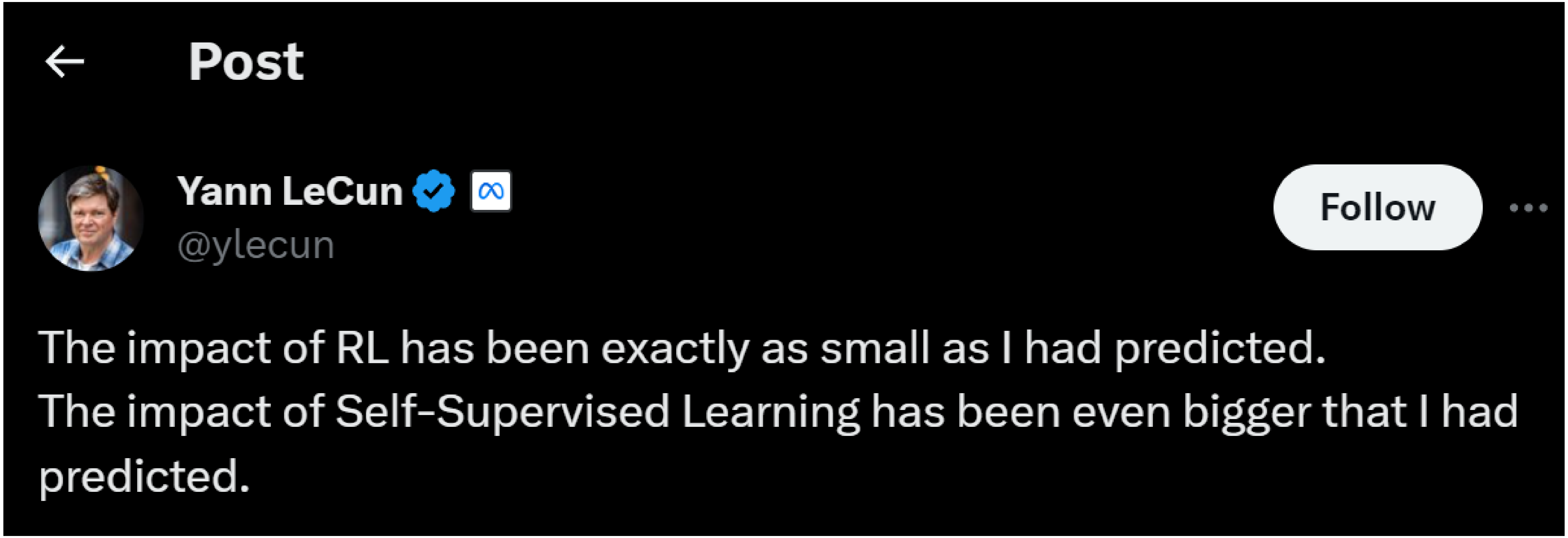
Research Postgraduate,

Dyson School of Design Engineering,

Imperial College London.

1. Overview

The Gloomy Comment



The RL Paradigm

s: state

R: reward

a: action

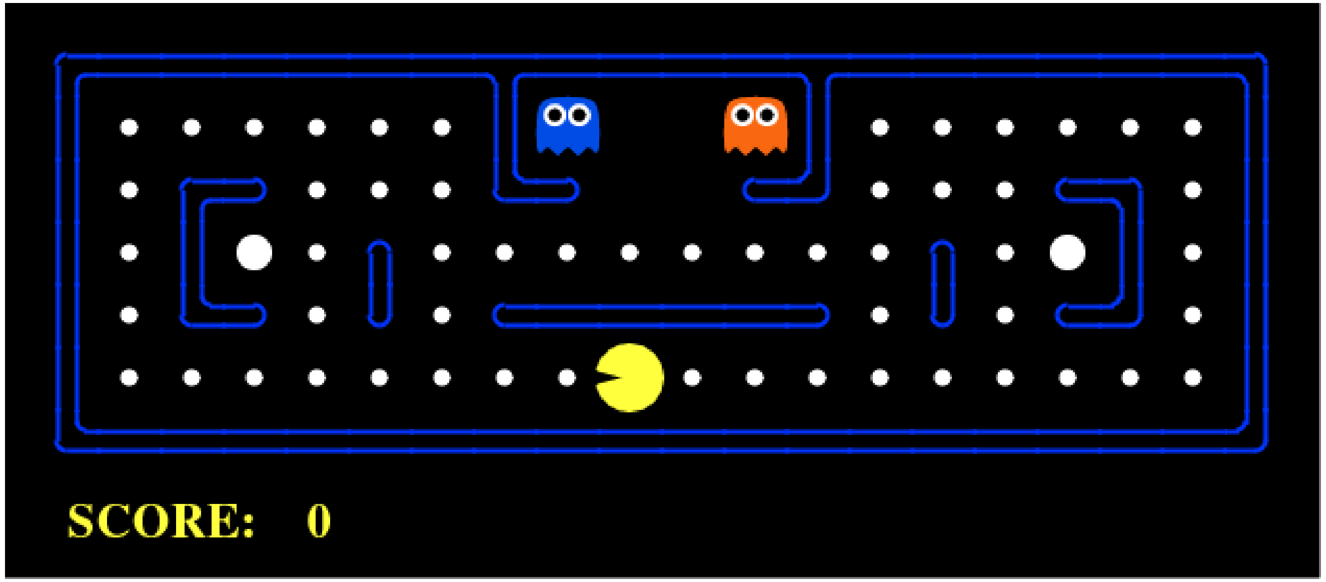
Most likely the agent does not know the inner working of the environment, i.e. model-free RL

An Example： Pac-Man

a: $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$

s_1: $\{I_0, I_{-1}, I_{-2}, I_{-3}\}$

s_2: $\{x_p, y_p, x_{g1}, y_{g2} \dots\}$



R_1: $\{+1, 0, -100\}$

R_2: $\{0, -100\}$

The choice of state and reward are fexible

2. Methodology

How RL Works

Reinforcement Learning

At state s_k, choose action a_k, that maximizes the **expected cumulative reward**. Formally:

$$a_k \leftarrow argmax_a \mathbf{Q}(\boldsymbol{s_k}, \boldsymbol{a})$$

RL Classification

- Planning — Model Predictive Control, Dynamic Programming, etc
- Model-free RL
  - Tabular Solution Methods
    - Monte-Carlo
    - Temporal Diff (TD)
      - Q-Learning, etc.
  - Approximated Solution Methods — DQN, etc.
    - Policy Gradient Methods

Tabular Solution Method Example: Q-Learning

Q-Learning

Reinforcement Learning

Find a_k that maximizes the **expected cumulative reward**.

$$a_k \leftarrow argmax_a \mathbf{Q(s_k, a)}$$

$$Q(s_k, a_k) = \mathbb{E}[R_k + \gamma R_{k+1} + \gamma^2 R_{k+2} + \cdots]$$

$$= R_k + \gamma \mathbb{E}[R_k + \gamma R_{k+2} + \cdots]$$

Assume a_k at s_k leads to determined s_{k+1}

$$= R_k + \sum_a P(a|s_{k+1}) Q(s_{k+1}, a)$$

Q-Learning

1. *Act*=>ε-greedy:

$$\begin{cases} a = \underset{a_i=-1/+1}{argmax}\, Q(s, a_i) & \text{=> Exploitation with } 1-\varepsilon \text{ possibility} \\ \text{act randomly} & \text{=> Exploration with } \varepsilon \text{ possibility} \end{cases}$$

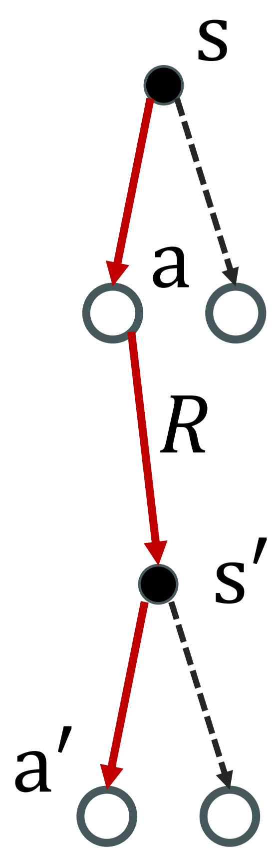
2. Update:

At state s, take a, update $Q(s, a)$:

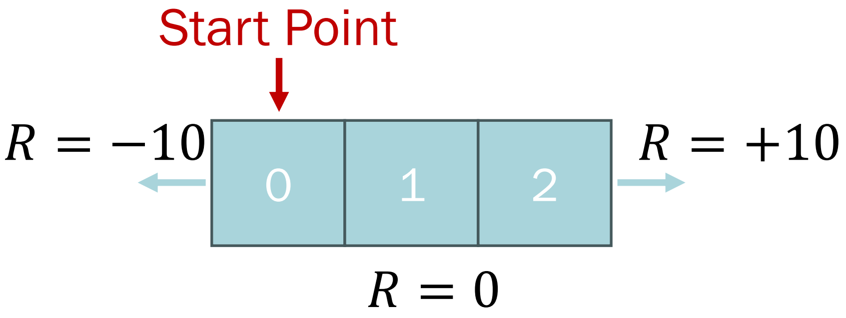
$$Q(s, a) \leftarrow Q(s, a) + lr[\underbrace{R + \gamma \max_{a'} Q(s', a')}_{\text{Bootstrapped estimation of } Q(s,a)} - Q(s, a)]$$

Bootstrapped estimation of $Q(s, a)$

Based on greedy assumption for future states



$\max_{a'} Q(s', a')$



1. Act:

$$a = \underset{a_i = -1/+1}{argmax}\, Q(s, a_i),$$

If $Q(s,-1) = Q(s,+1)$, act randomly

2. Update:

At state s, take a, update $Q(s,a)$:

$$Q(s,a) \leftarrow Q(s,a) + lr[R + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

1. Act:

Q(s, a)		Actions	
		-1	+1
States	0	-10	0
	1	0	0
	2	0	+10

Policy:

$S = 0, a = +1$

$S' = 1, R = 0$

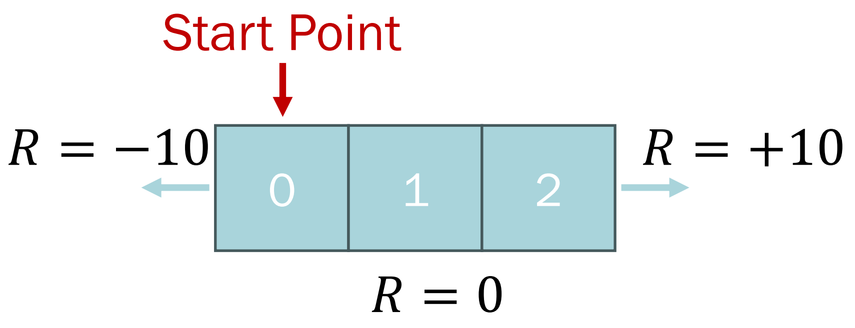
Initialized to be 0

2. Update:

Q(s, a)		Actions	
		-1	+1
States	0	-10	0
	1	0	0
	2	0	+10

Update:

$Q(0,+1) \leftarrow 0$



1. Act:

$$a = \underset{a_i=-1/+1}{argmax}\, Q(s, a_i),$$

If $Q(s, -1) = Q(s, +1)$, act randomly

2. Update:

At state s, take a, update $Q(s, a)$:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \max_{a'} Q(s', a') - Q(s,a)]$$

1. Act:

Q(s, a)		Actions	
		-1	+1
States	0	-10	0
	1	0	0
	2	0	+10

Policy:

$S = 1, a = -1$

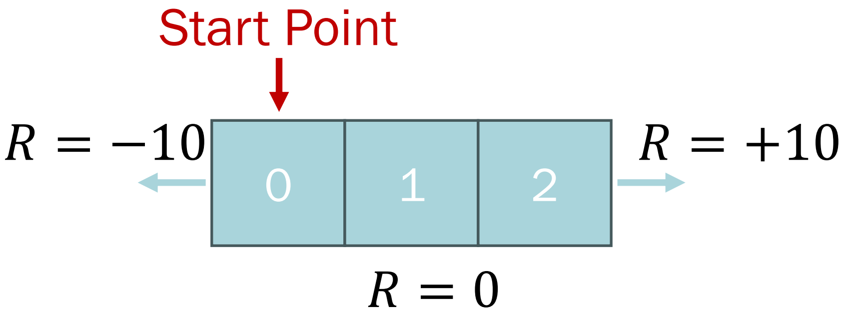
$S' = 0, R = 0$

2. Update:

Q(s, a)		Actions	
		-1	+1
States	0	-10	0
	1	0	+10
	2	0	+10

Update:

$Q(1, -1) \leftarrow 0$



1. Act:

$$a = \underset{a_i = -1/+1}{\operatorname{argmax}} Q(s, a_i),$$

If $Q(s, -1) = Q(s, +1)$, act randomly

2. Update:

At state s, take a, update $Q(s, a)$:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \max_{a'} Q(s', a') - Q(s,a)]$$

1. Act:

Q(s, a)		Actions: -1	Actions: +1
States	0	-10	0
	1	0	0
	2	0	+10

Policy:

$S = 1, a = +1$

$S' = 2, R = 0$

2. Update:

Q(s, a)		Actions: -1	Actions: +1
States	0	-10	0
	1	0	+10
	2	0	+10

Update:

$Q(1, +1) \leftarrow +10$

Q-Learning

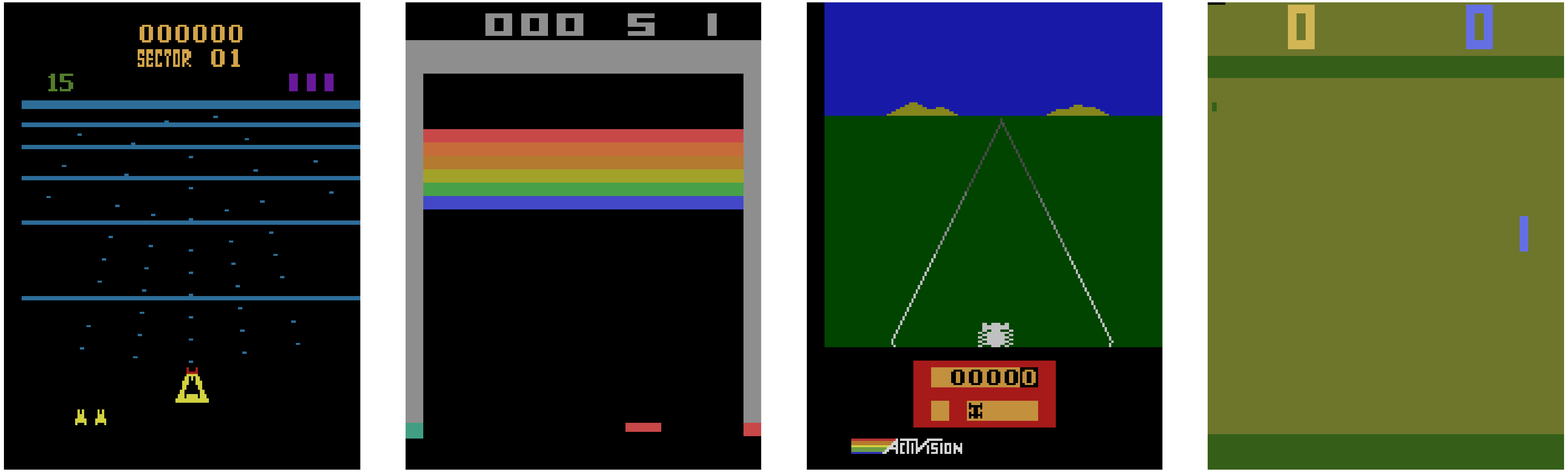
Q-learning: An off-policy TD control algorithm

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S' — **1. Act**
        $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$ — **2. Update**
        $S \leftarrow S'$
    until S is terminal

Approximated Solution Method Example: DQN

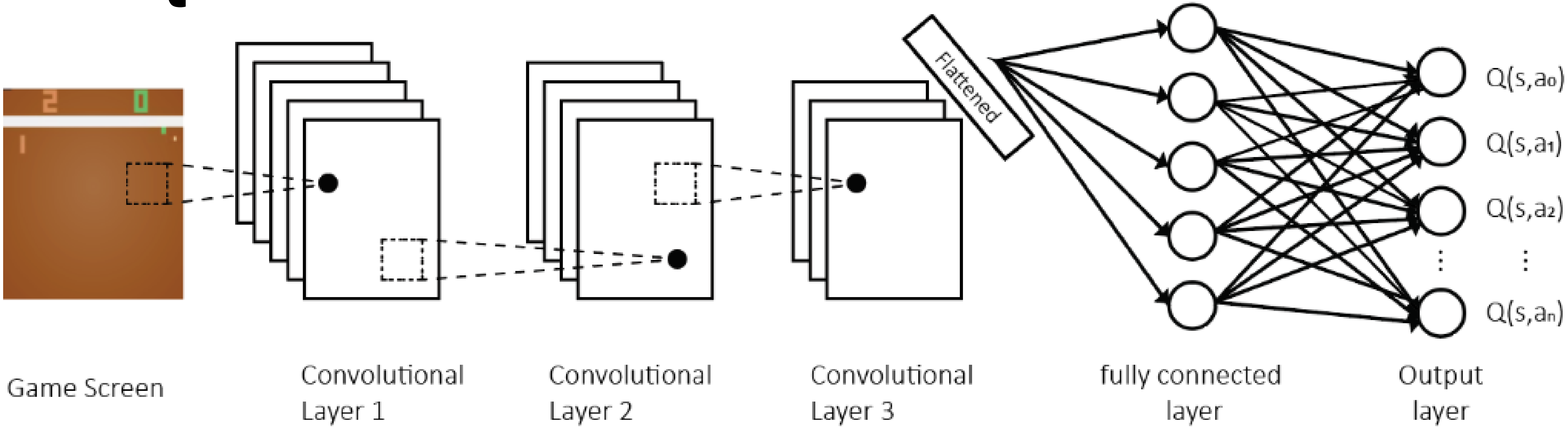
DQN

- Short for Deep Q-Network
- Proposed by Minh et al. in “Playing Atari with Deep Reinforcement Learning”



Figures from https://gymnasium.farama.org/environments/atari/complete_list/

DQN



- Use NN to approximate $Q_{\theta}(s, a)$.
- Suitable for large state and action space.
- Ability to generalize.

DQN

- To update θ in the NN
- Use NN to approximate $Q_\theta(s,a)$:

$$L = \frac{1}{2}(\underbrace{R + \max_a Q_\theta(s',a)}_{\text{Predicted Q value}} - Q_\theta(s,a))$$

Predicted Q value

.detach()

DQN

- Seems "straight-forward":

  Deeper -> More Powerful?
- In fact, the paper was not the first to propose deep networks for approximating $Q(s,a)$.
- The main contribution is the **Replay Memory**.

DQN: Replay Memory

- Save experience (s, a, R, s') in the Replay Buffer.
- In each iteration, sample a batch from the Replay Buffer.
- Benefits for doing this:
  - Breaks Correlation in Successive Samples
  - Promotes Sample Efficiency
  - Facilitates Learning from Rare Events
  - Improves Gradient Descent Stability (by having a batch).

3. Summary

Summary

Reinforcement Learning

RL learns from **trial and error** through interaction with an environment

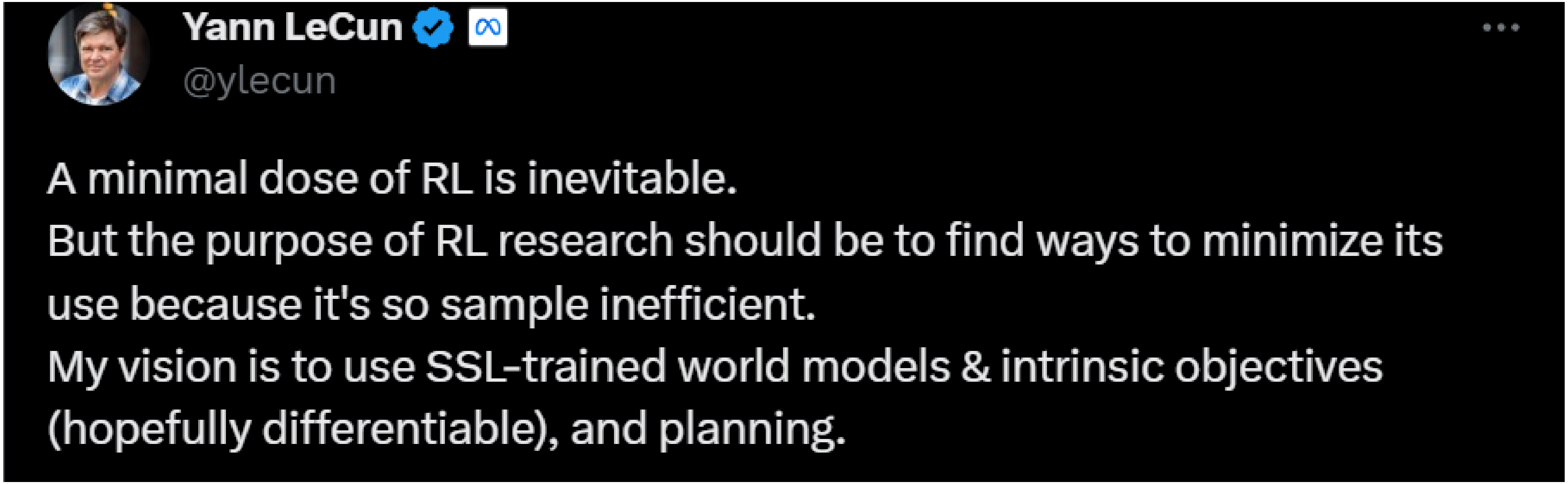
Compared with Other ML Paradigm

RL generates a sequence of decision each depending on previous actions; Data distribution changes according to the agent’s action.

Compared with Planning (DP, MPC)

No system model!!

Back to the Gloomy Comment


Yann LeCun
@ylecun

A minimal dose of RL is inevitable.
But the purpose of RL research should be to find ways to minimize its use because it's so sample inefficient.
My vision is to use SSL-trained world models & intrinsic objectives (hopefully differentiable), and planning.

If you are still interested

Sutton&Barto Book

Available free online: https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

David Silver UCL Lectures

Recording free on YouTube: https://www.youtube.com/watch?v=2pWv7GOvuf0

Thank you

Haozhe Tian,

Research Postgraduate,

Dyson School of Design Engineering,

Imperial College London.