A Playful Dive into the World of RL

Haozhe Tian,

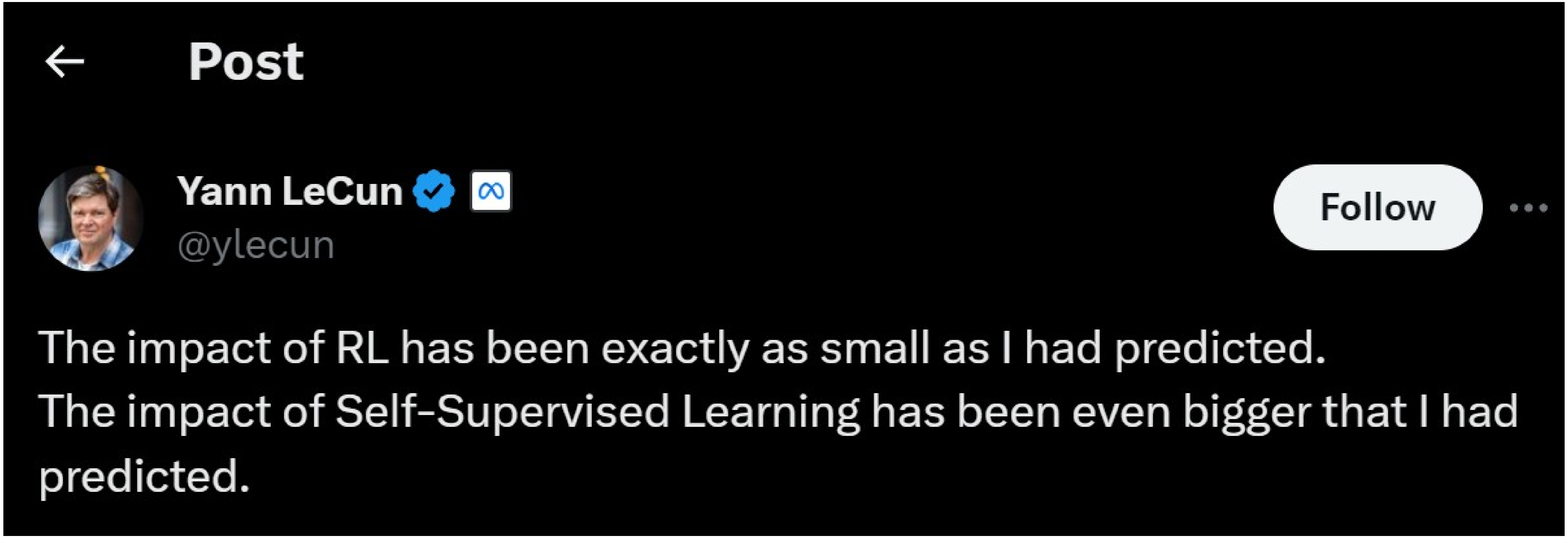
Research Postgraduate,

Dyson School of Design Engineering,

Imperial College London.

1. Overview

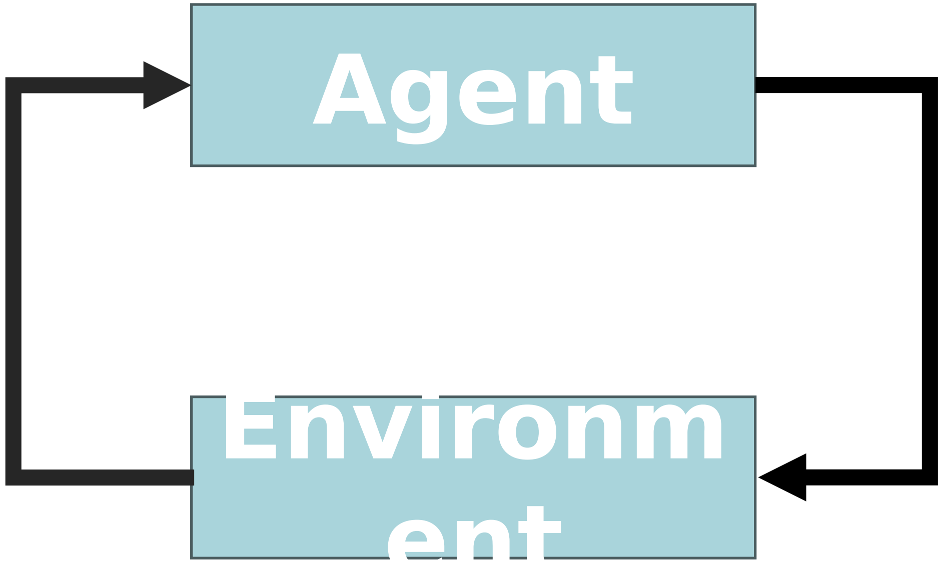
The Gloomy Comment



Yann LeCun
@ylecun

The impact of RL has been exactly as small as I had predicted.
The impact of Self-Supervised Learning has been even bigger that I had predicted.

The RL Paradigm



Agent

Environment

: state

: reward

: action

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

An Example： Pac-Man

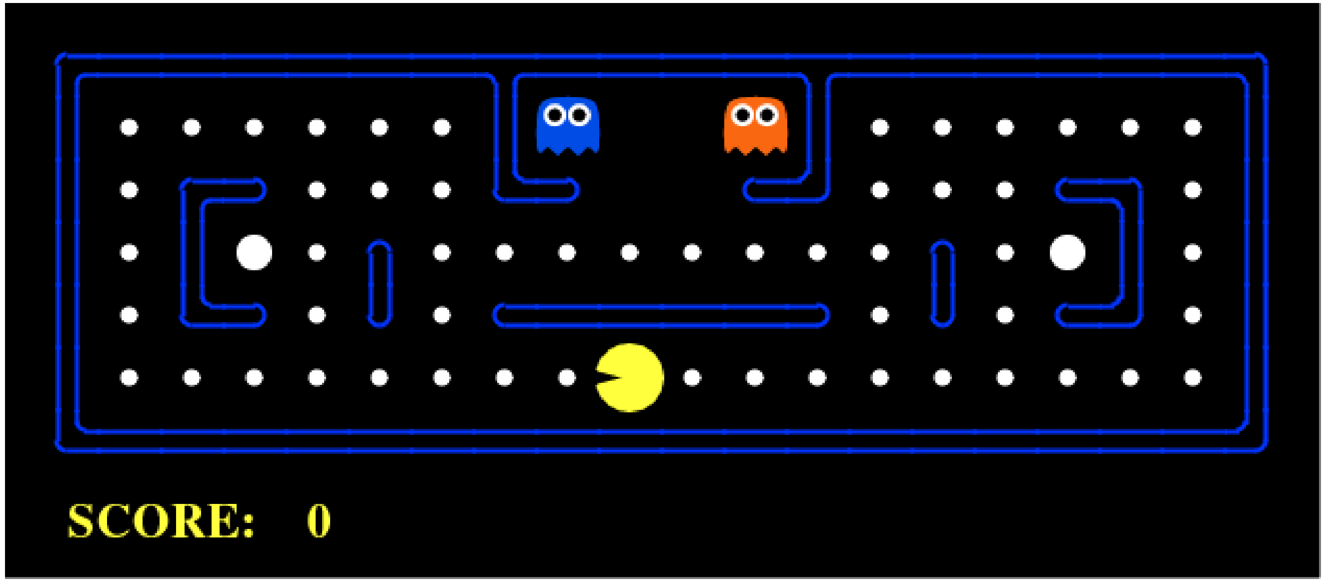
$: \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$

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

$: \{\ldots\}$

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

$: \{0, -100\}$



The choice of state and reward are fexible

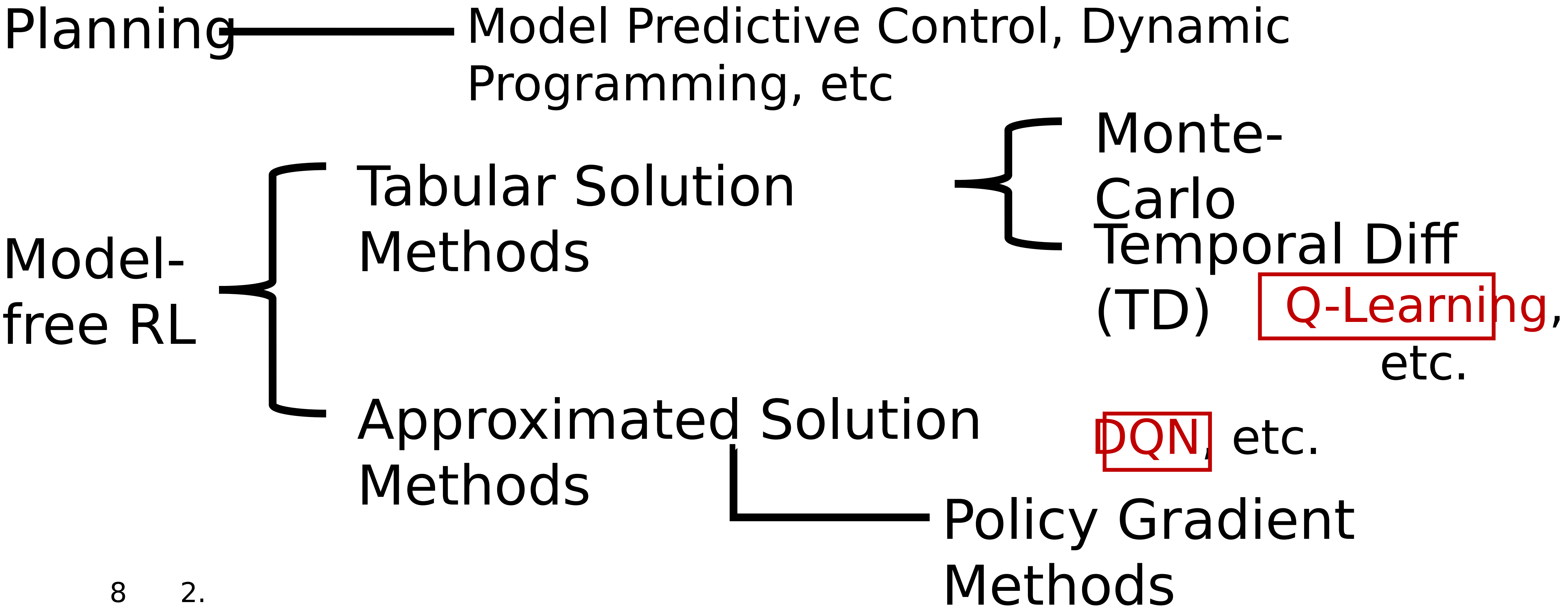
2. Methodology

How RL Works

Reinforcement Learning

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

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 that maximizes the **expected cumulative reward**.

$+$

Assume at leads to
determined

Q-Learning

1. Act=>-greedy:

=> Exploitation with possibility

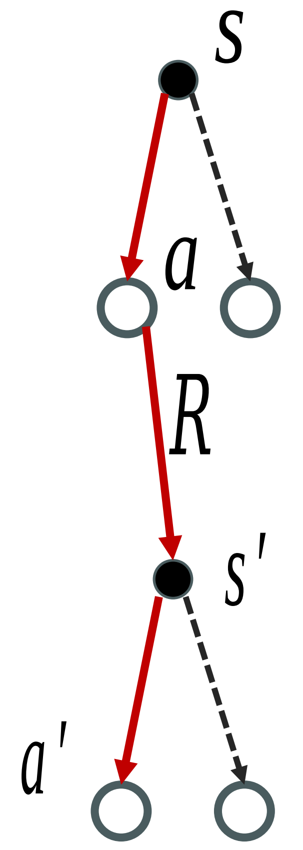
act randomly   => Exploration with possibility

2. Update:

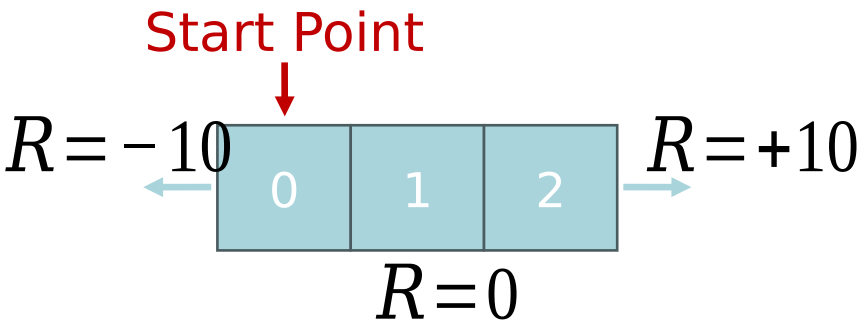
At state , take , update :

Bootstrapped estimation of

Based on greedy assumption for future states



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



1. Act:

,

If , act randomly

2. Update:

At state , take , update :

1. Act:

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

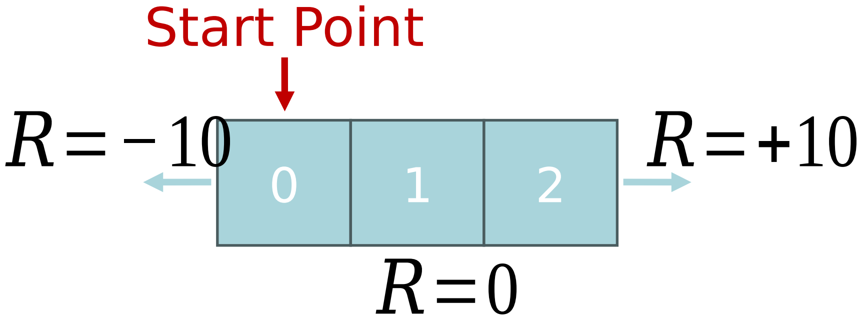
Initialized to be 0

Policy:

2. Update:

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

Update:



1. Act:

,

If , act randomly

2. Update:

At state , take , update :

1. Act:

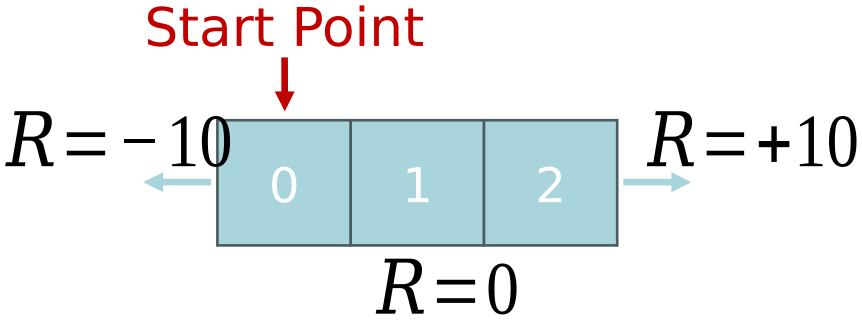
Q(s, a)		Actions	
		-1	+1
States	0	-10	0
	1	0	0
	2	0	+10

Policy:

2. Update:

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

Update:



1. Act:

,

If , act randomly

2. Update:

At state , take , update :

1. Act:

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

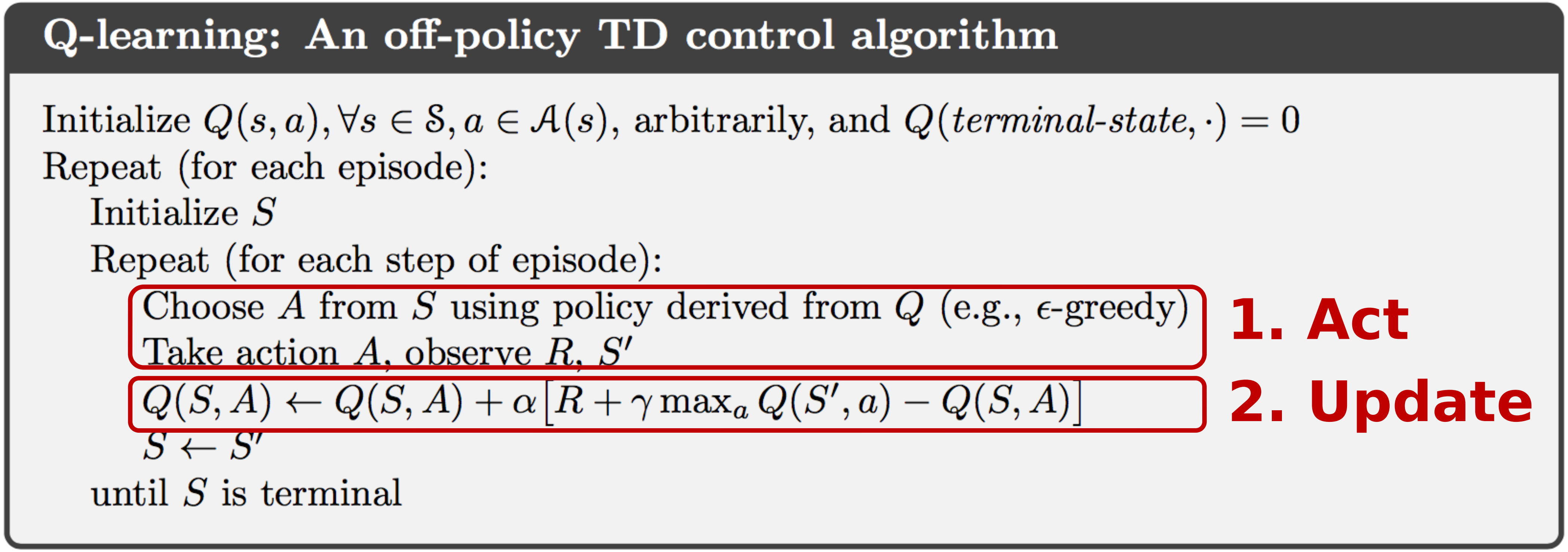
Policy:

2. Update:

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

Update:

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., $\epsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S,A) \leftarrow Q(S,A) + \alpha \left[ R + \gamma \max_a Q(S',a) - Q(S,A) \right]$
        $S \leftarrow S'$
    until $S$ is terminal
```

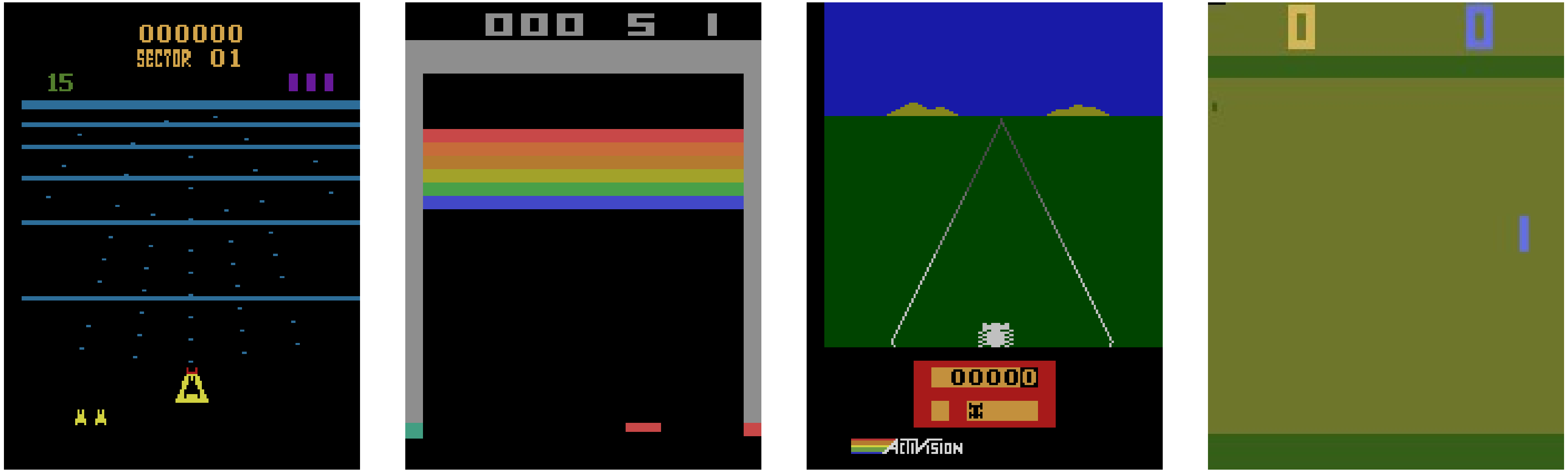
1. Act

2. Update

Approximated Solution Method
Example: DQN

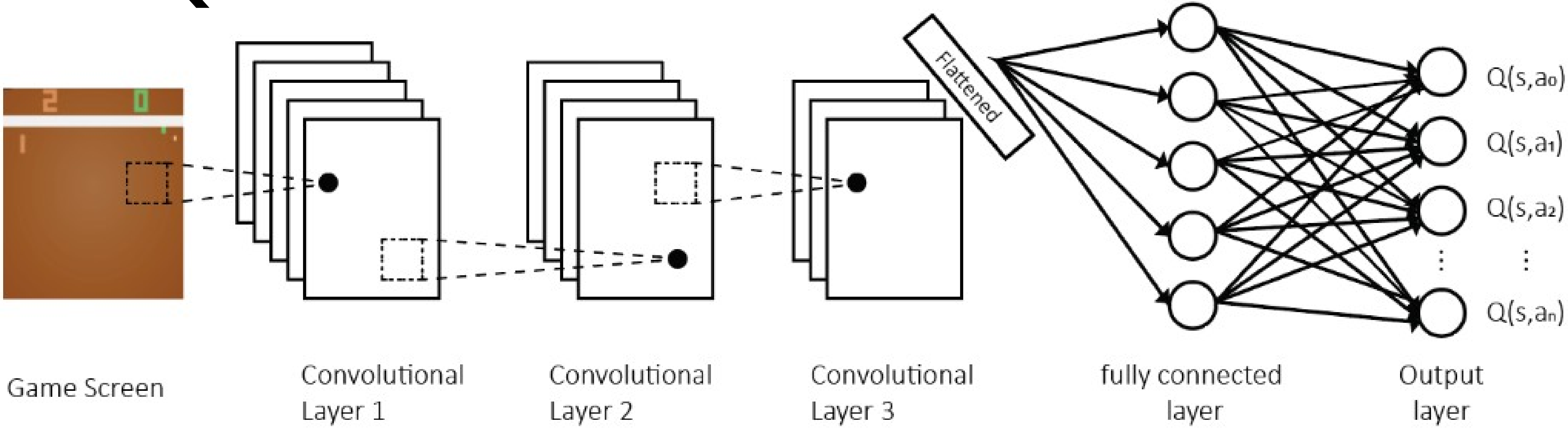
DQN

- 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 .
- Suitable for large state and action space.
- Ability to generalize.

DQN

- To update in the NN
- Use NN to approximate

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 .
- The main contribution is the **Replay Memory**.

DQN: Replay Memory

- Save experience 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

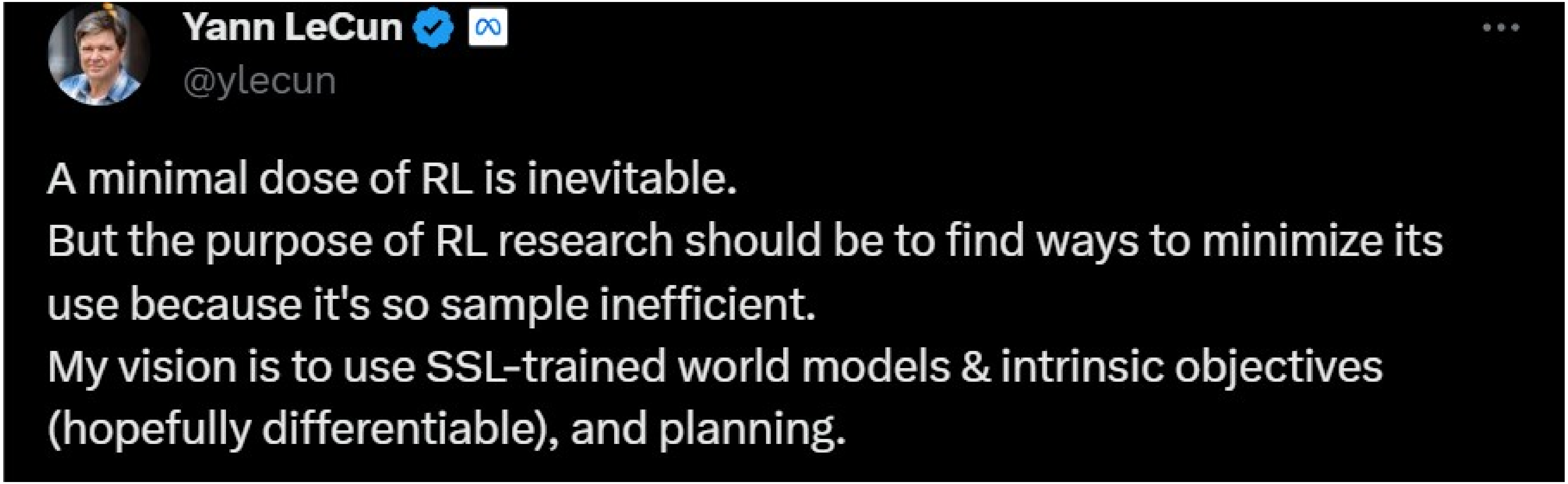
Compared with Other ML Paradigm

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.