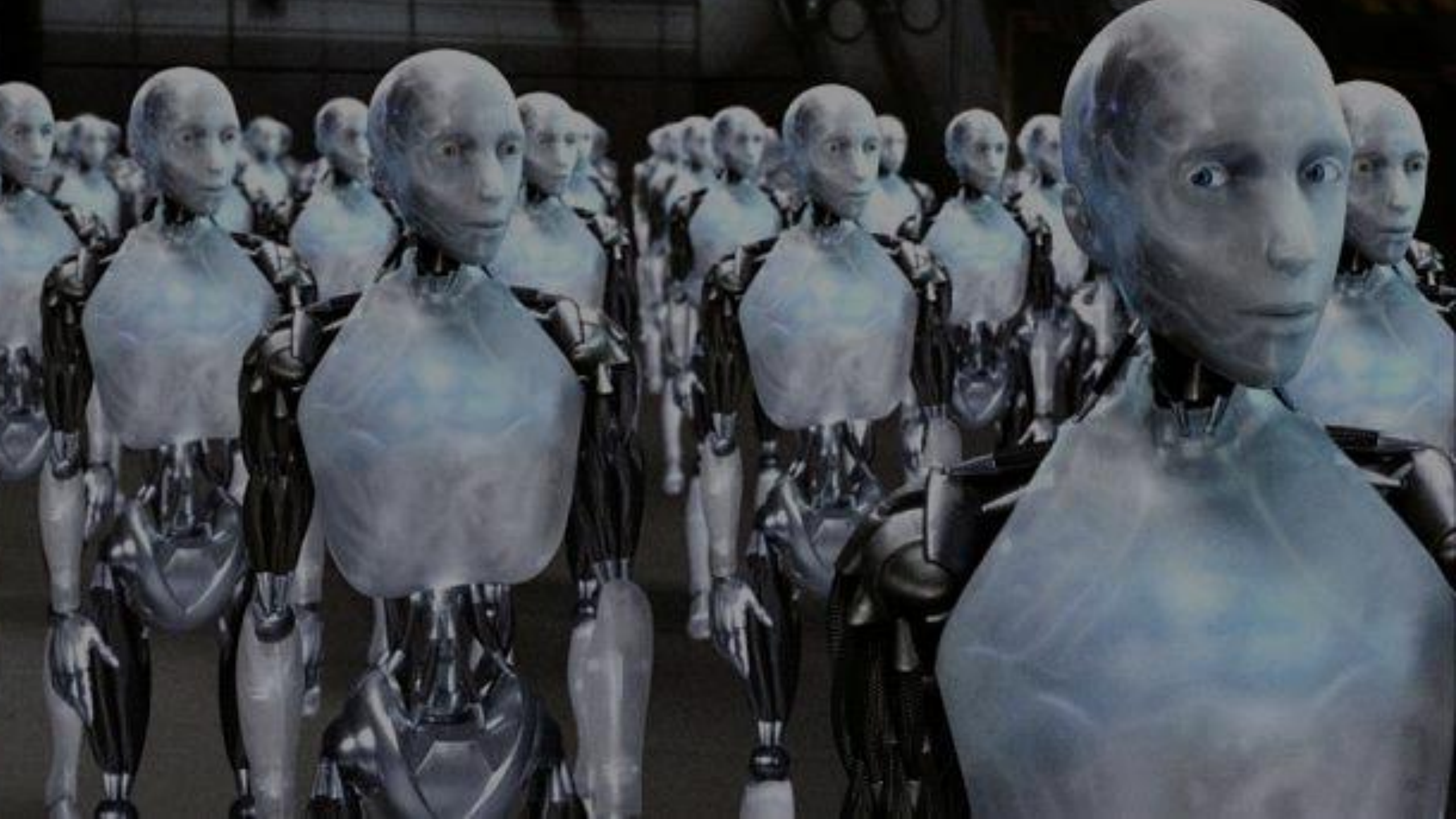
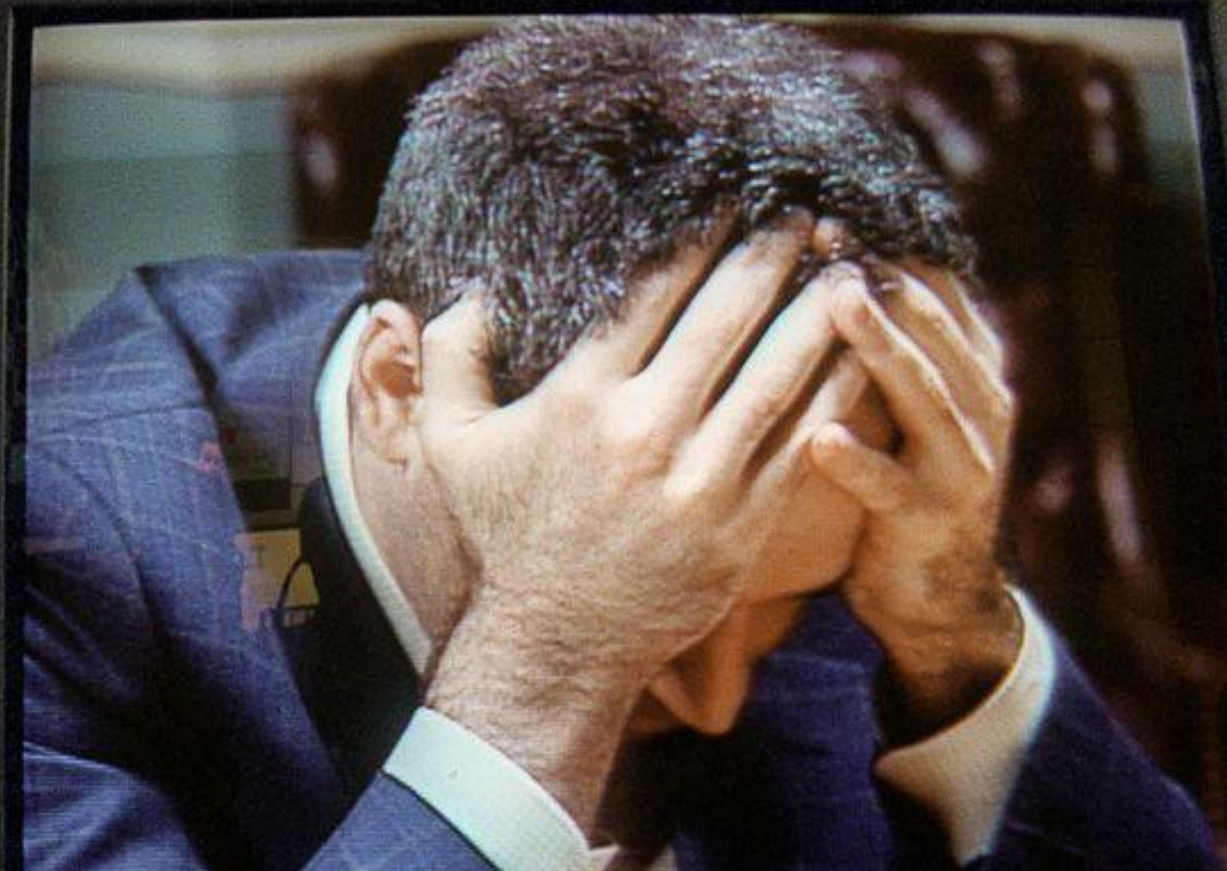


人工智能

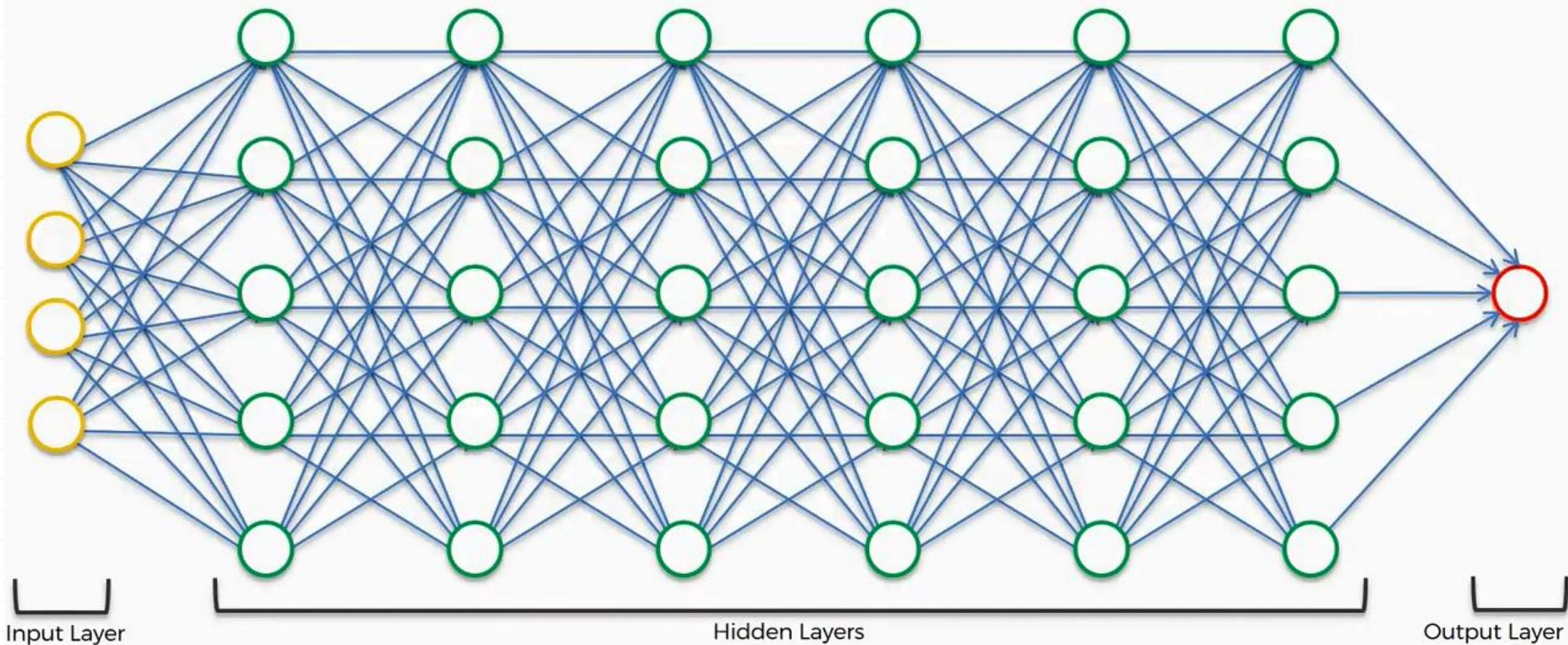
丘祐瑋
David Chiu

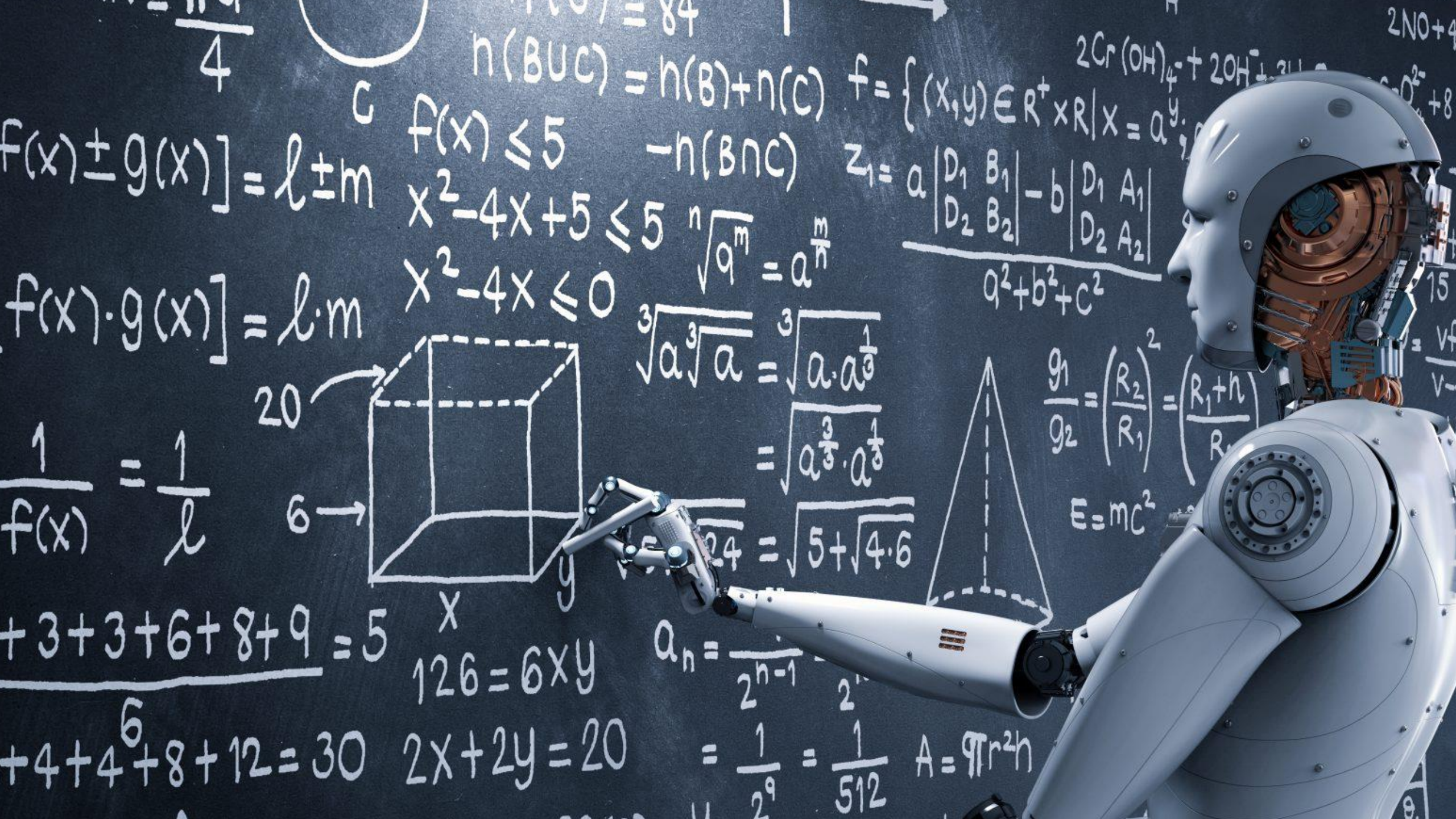




深度學習

- 機器學習的一部份，可以實作監督式學習與非監督式學習





$$\frac{1}{f(x)} = \frac{1}{l}$$
$$f(x) \pm g(x) = l \pm m$$
$$f(x) \cdot g(x) = l \cdot m$$
$$+3+3+6+8+9=5$$
$$+4+4+8+12=30$$
$$h(BUC) = h(B) + h(C)$$
$$f(x) \leq 5$$
$$x^2 - 4x + 5 \leq 5$$
$$x^2 - 4x \leq 0$$
$$126 = 6 \times y$$
$$2x + 2y = 20$$
$$-h(B \cap C)$$
$$\sqrt[n]{a^m} = a^{\frac{m}{n}}$$
$$\sqrt[3]{a^3 a} = \sqrt[3]{a \cdot a^3}$$
$$= \sqrt[3]{a^3 \cdot a^3}$$
$$= \sqrt[3]{5 + \sqrt{4 \cdot 6}}$$
$$a_n = \frac{1}{2^{n-1}}$$
$$= \frac{1}{2^9} = \frac{1}{512}$$
$$f = \{(x, y) \in \mathbb{R}^+ \times \mathbb{R} \mid x = a^y\}$$
$$z_1 = a \frac{\begin{vmatrix} D_1 & B_1 \\ D_2 & B_2 \end{vmatrix} - b \begin{vmatrix} D_1 & A_1 \\ D_2 & A_2 \end{vmatrix}}{a^2 + b^2 + c^2}$$
$$\frac{g_1}{g_2} = \left(\frac{R_2}{R_1}\right)^2 = \left(\frac{R_1 + h}{R_1}\right)^2$$
$$E = mc^2$$
$$A = \pi r^2 h$$
$$2Cr(OH)_4^- + 2OH^- + 2H^+ \rightarrow 2Cr(OH)_3 + 2H_2O$$
$$2NO + 4H^+ \rightarrow 2NO_2^- + 2H_2O$$
$$c^2 + d^2 + e^2 + 8$$
$$15$$
$$= \frac{v_f}{v_r}$$
$$20$$
$$6 \rightarrow$$
$$x$$
$$y$$
$$24$$
$$5$$

深度Q網路

■ DeepMind 於2014 年展示深度Q網路 (DQN, Deep-Q-Network) , 讓電腦自動學習雅達利 (Atari) 49 種遊戲

▣ Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529.





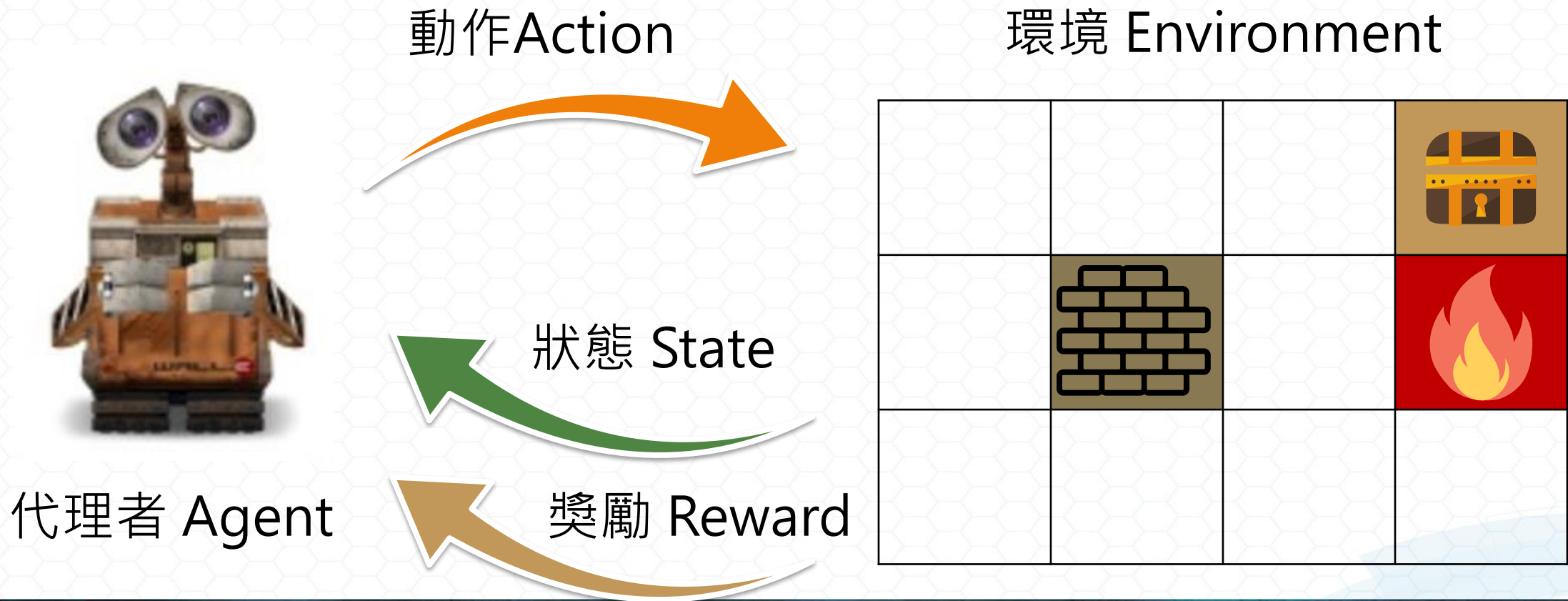
LEE SEDOL
00:01:00

d

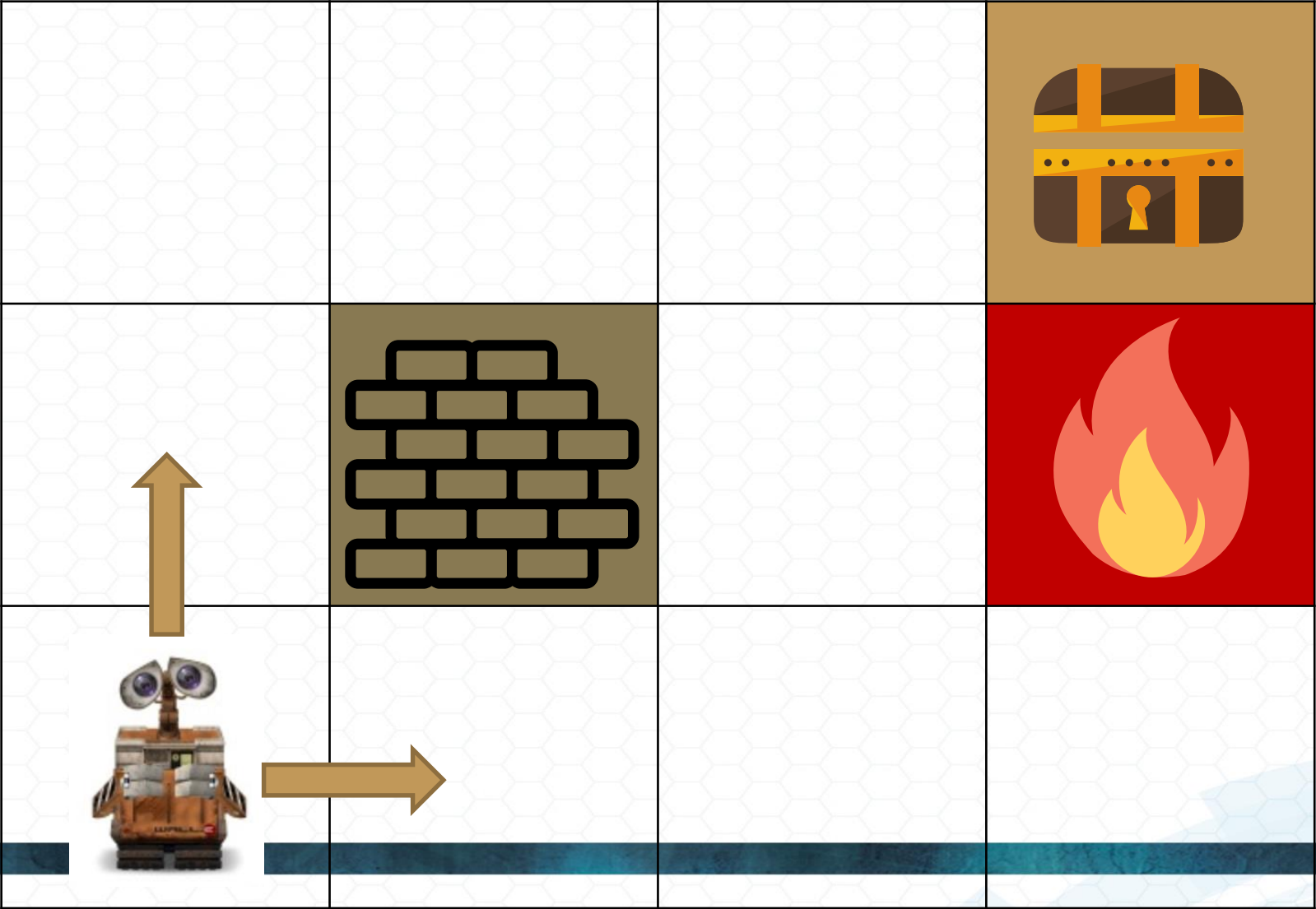


強化學習架構

- 代理者在環境中選擇做出某個動作以得到對應的獎勵
- 代理者選擇動作的方式稱為策略(Policy)



與環境互動過程



 $R = + 1$

 $R = - 1$

貝爾曼方程 (Bellman Equation)

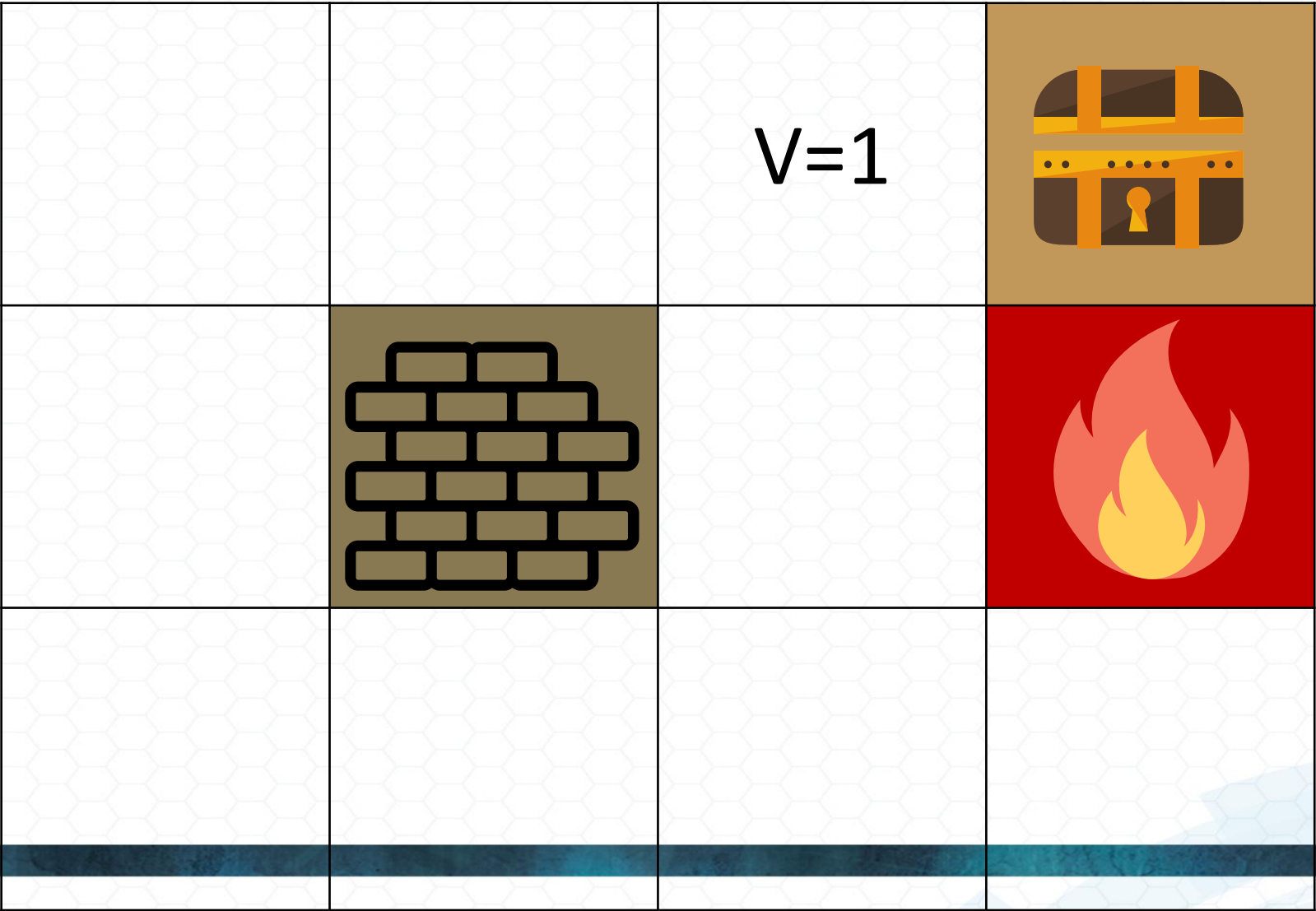
- 貝爾曼方程 (Bellman Equation) 也被稱作 動態規劃方程 (Dynamic Programming Equation)



An optimal trajectory on the time interval $[T_1, T]$ must be optimal also on each of the subintervals $[T_1, T_1 + \epsilon]$ and $[T_1 + \epsilon, T]$.




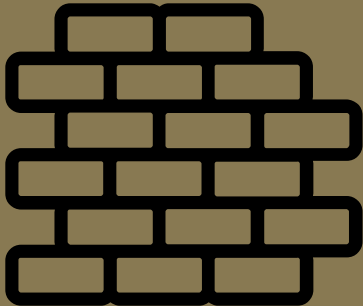

尋找最佳路徑



↖ $R = +1$

↙ $R = -1$

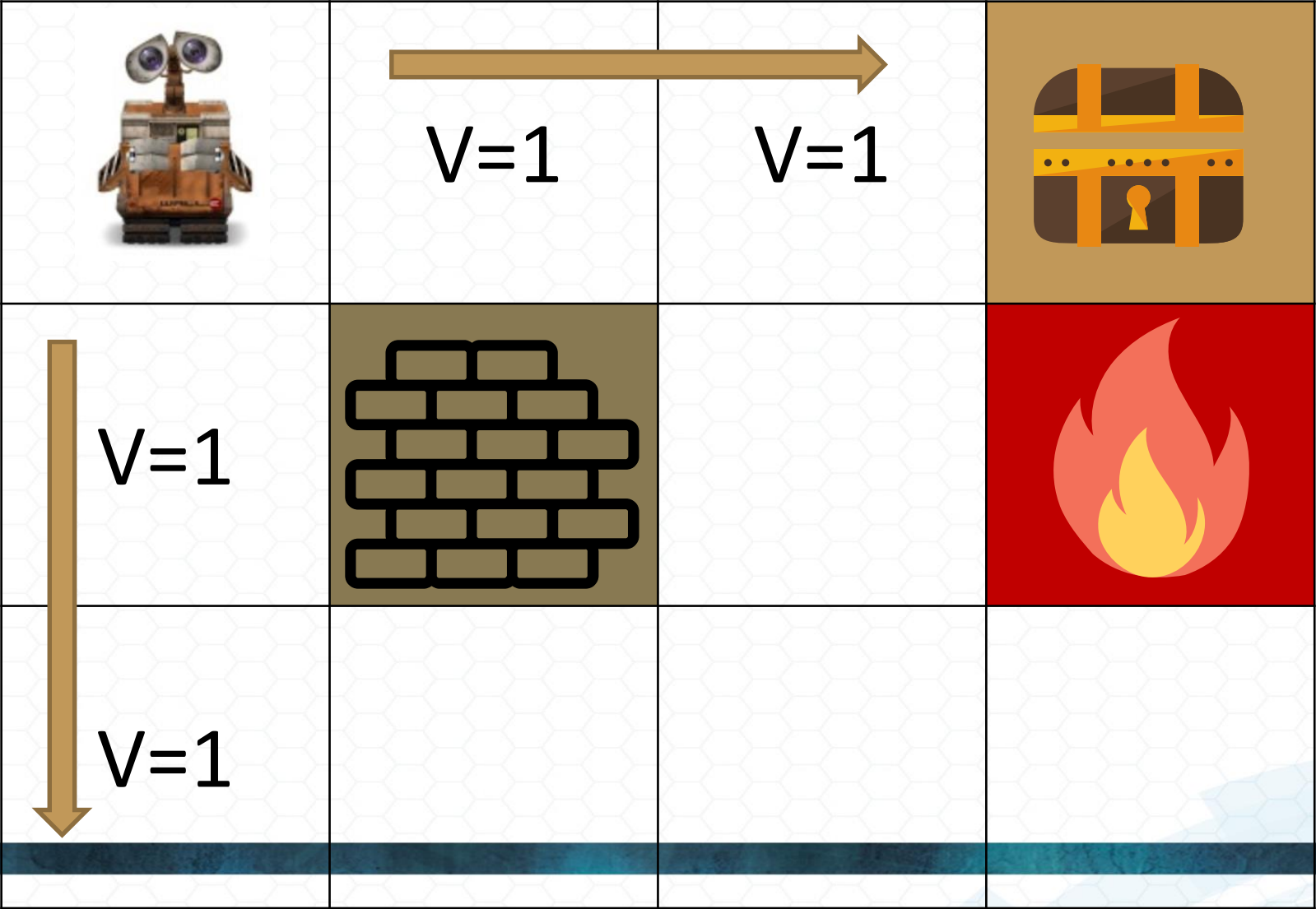
尋找最佳路徑

V=1	V=1	V=1	
V=1			
V=1			

 $R = +1$

 $R = -1$

尋找最佳路徑



貝爾曼方程

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

- S : 狀態 State
- a : 動作 Action
- R : 獎勵 Reward
- γ : 折扣 Discount

尋找最佳路徑

假設 $\gamma = 0.9$

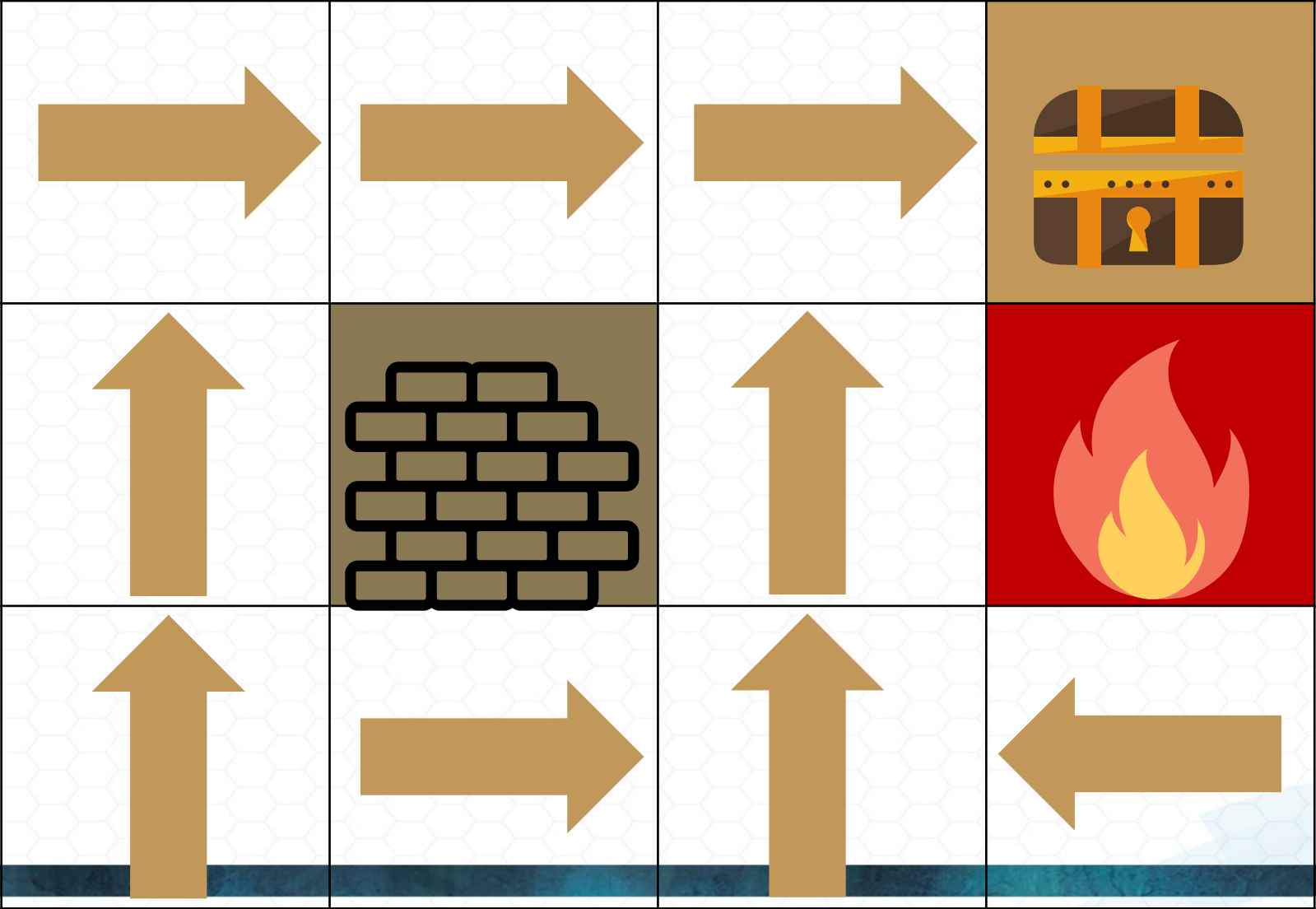
$V=0.81$	$V=0.9$	$V=1$	
$V=0.73$		$V=0.9$	
$V=0.66$	$V=0.73$	$V=0.81$	$V=0.73$

尋找最佳路徑

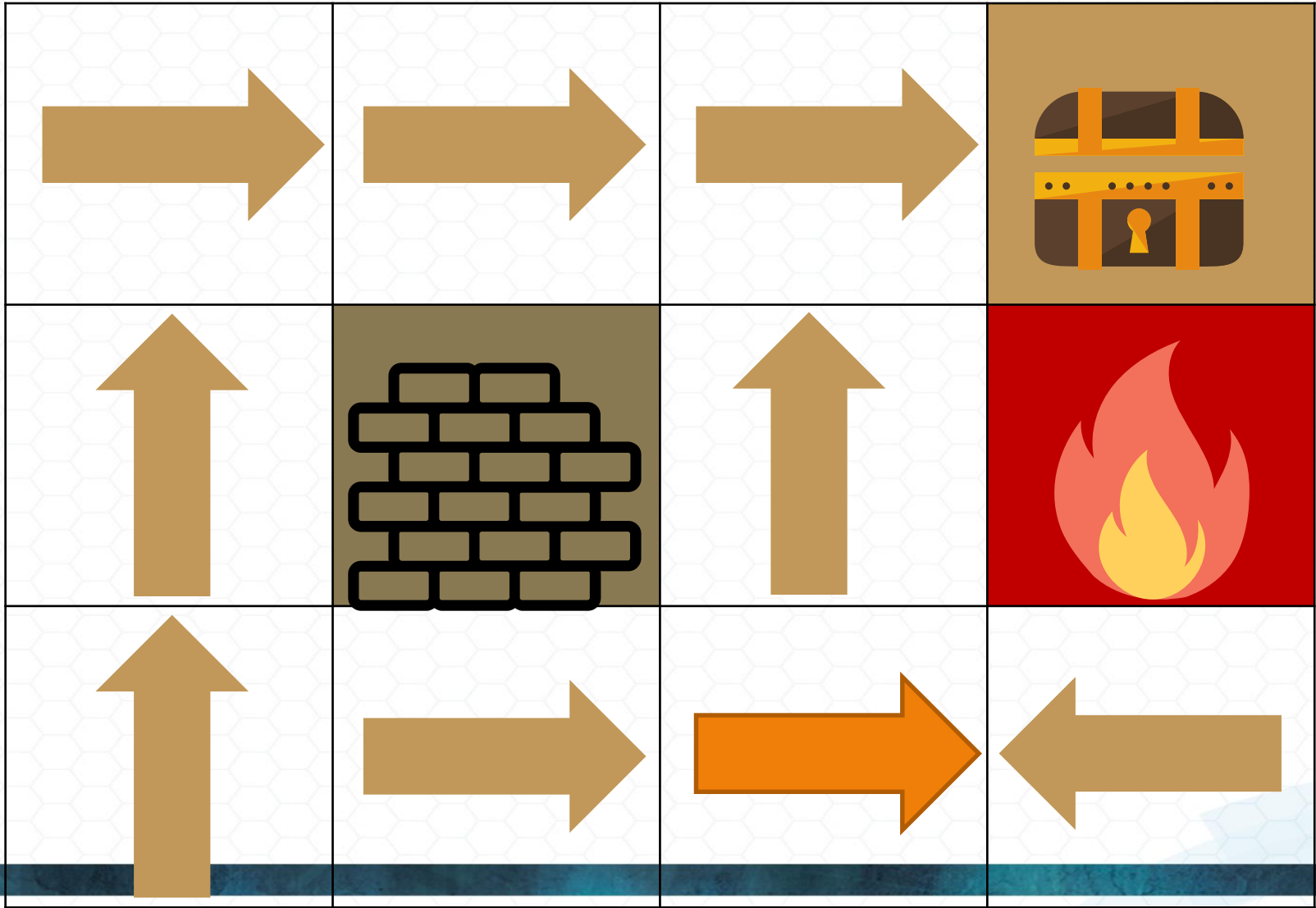
假設 $\gamma = 0.9$

$V=0.81$	$V=0.9$	$V=1$	
$V=0.73$		$V=0.9$	
$V=0.66$	$V=0.73$	$V=0.81$	$V=0.73$

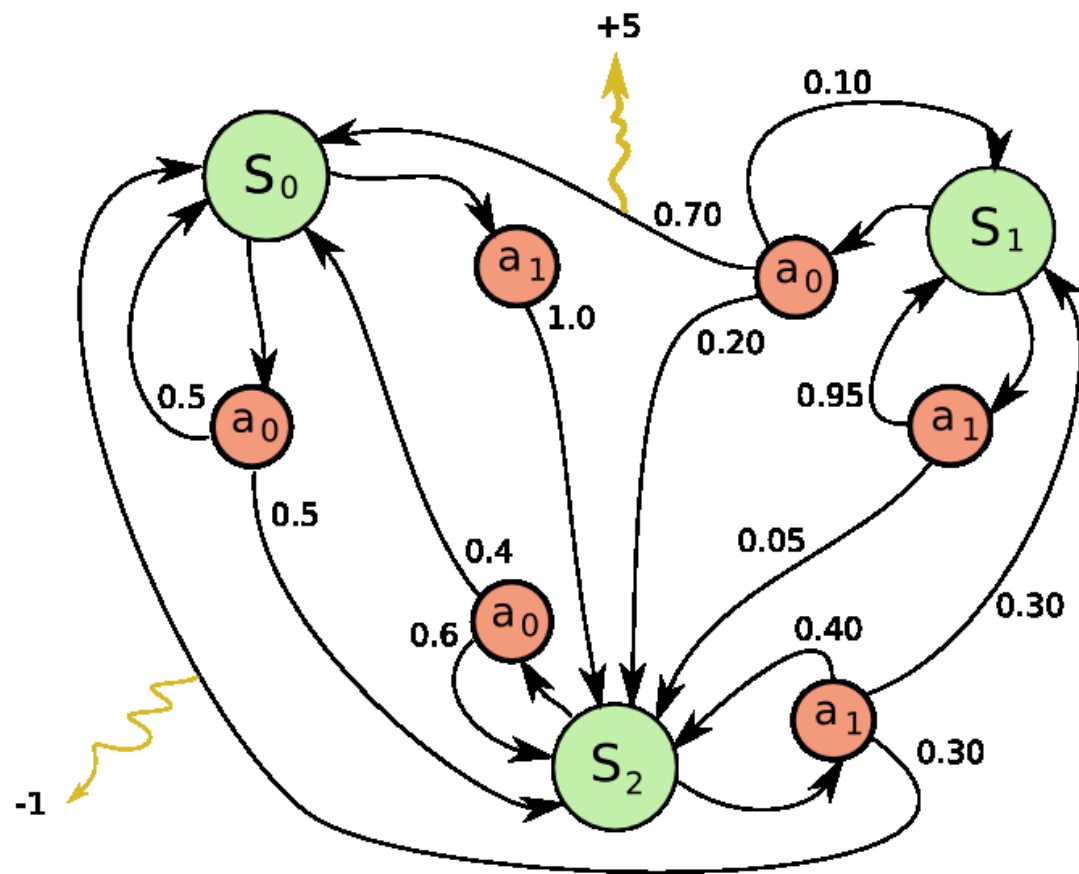
確定性策略



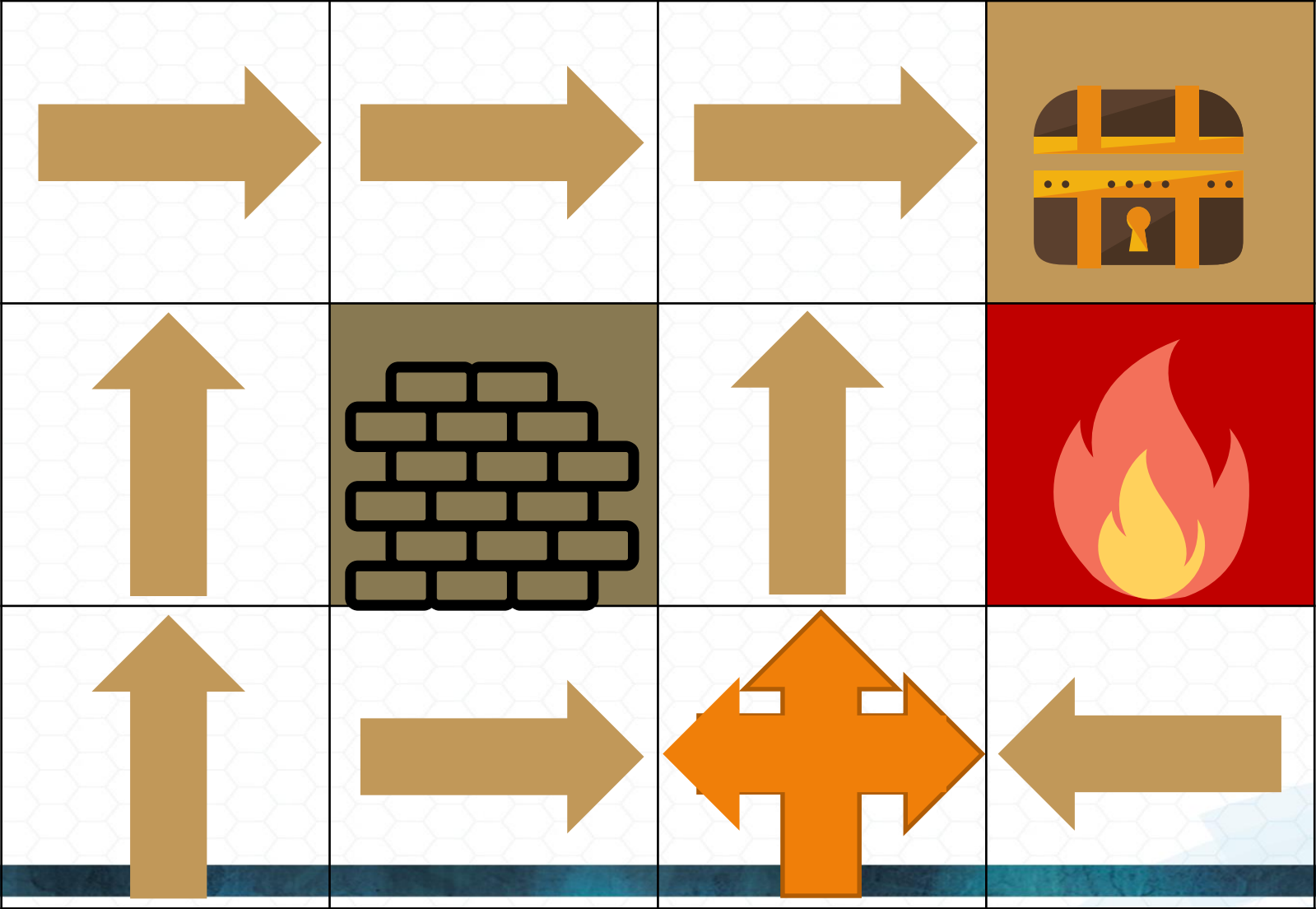
確定性策略的缺點



瑪律可夫決策過程



隨機策略



貝爾曼方程

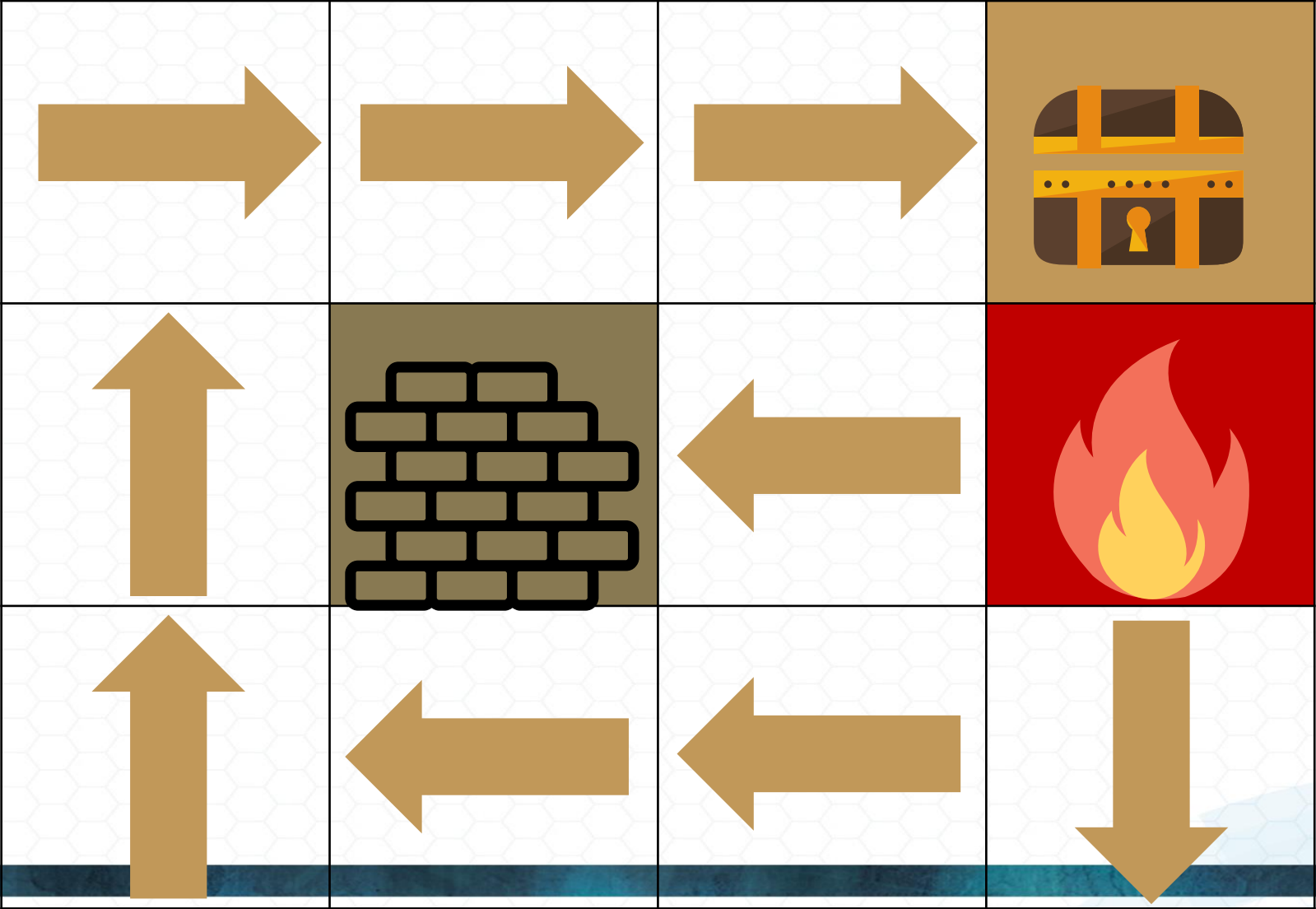
$$V(s) = \max_a (R(s, a) + \gamma \sum_{s'} P(s, a, s') V(s'))$$

- P : 機率 Probability
- S : 狀態 State
- a : 動作 Action
- R : 獎勵 Reward
- γ : 折扣 Discount

隨機策略

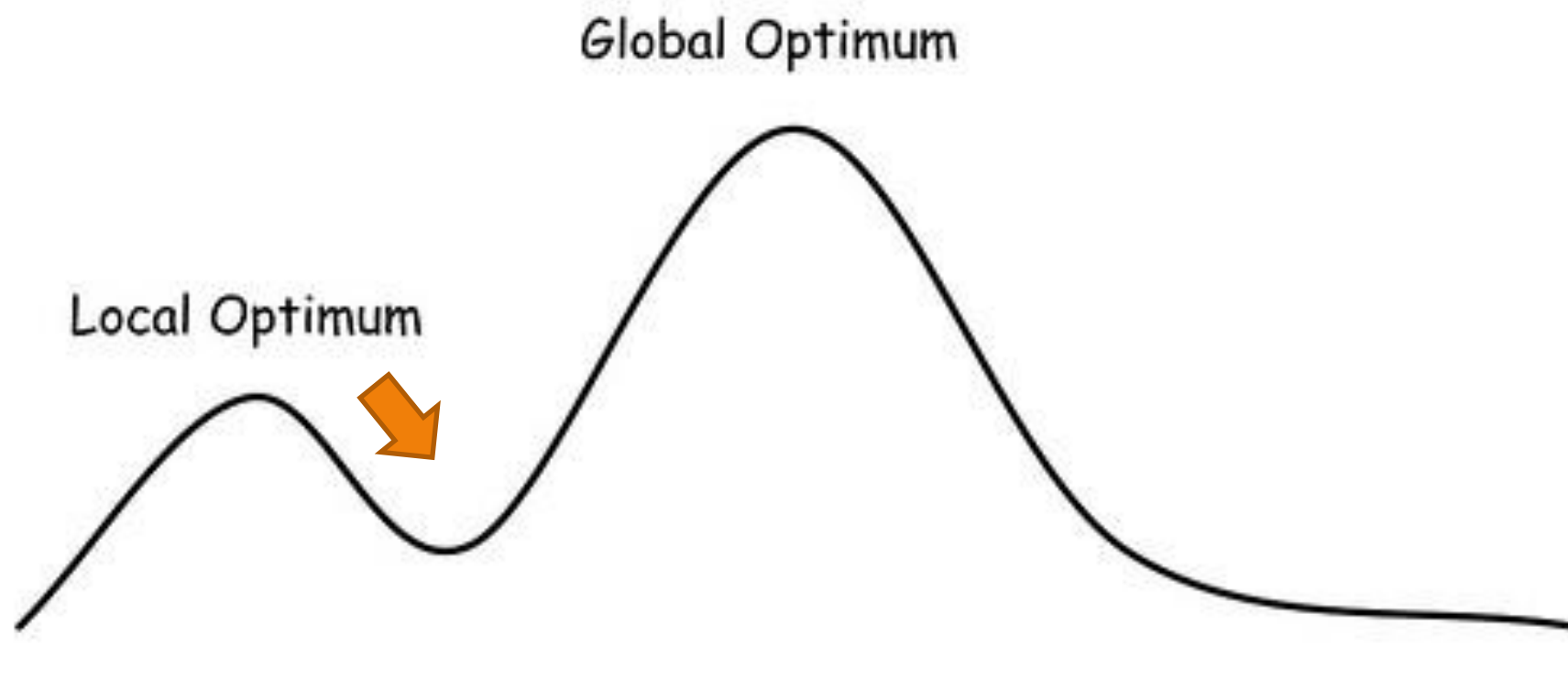
$V=0.7$	$V=0.75$	$V=0.85$	
$V=0.63$		$V=0.36$	
$V=0.55$	$V=0.45$	$V=0.3$	$V=0.2$

隨機策略路徑

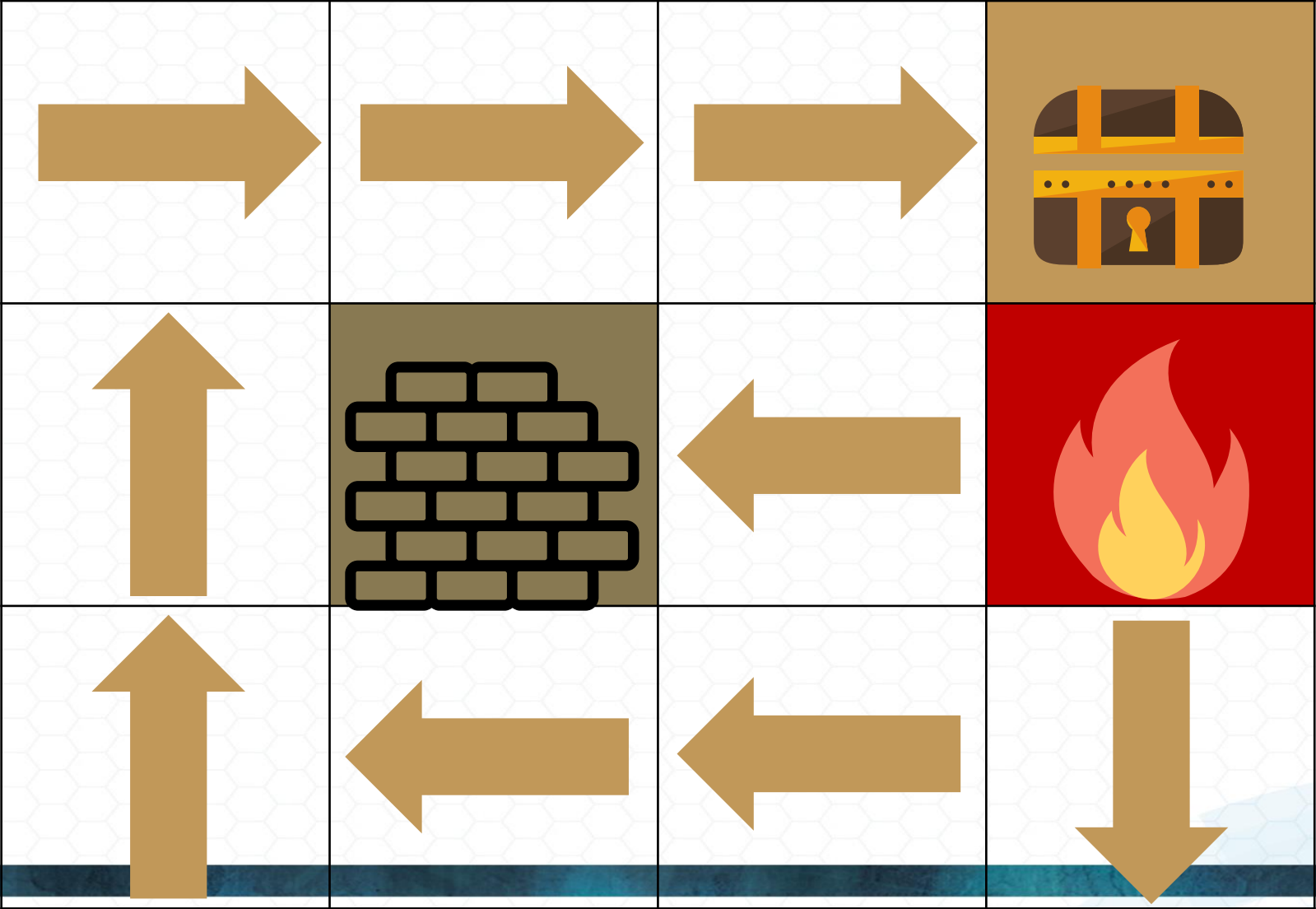


未來報酬與即時報酬


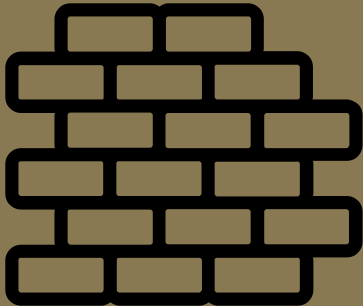

- 有時當下看是負面的行為，長遠來看有正面效果



考慮即時報酬

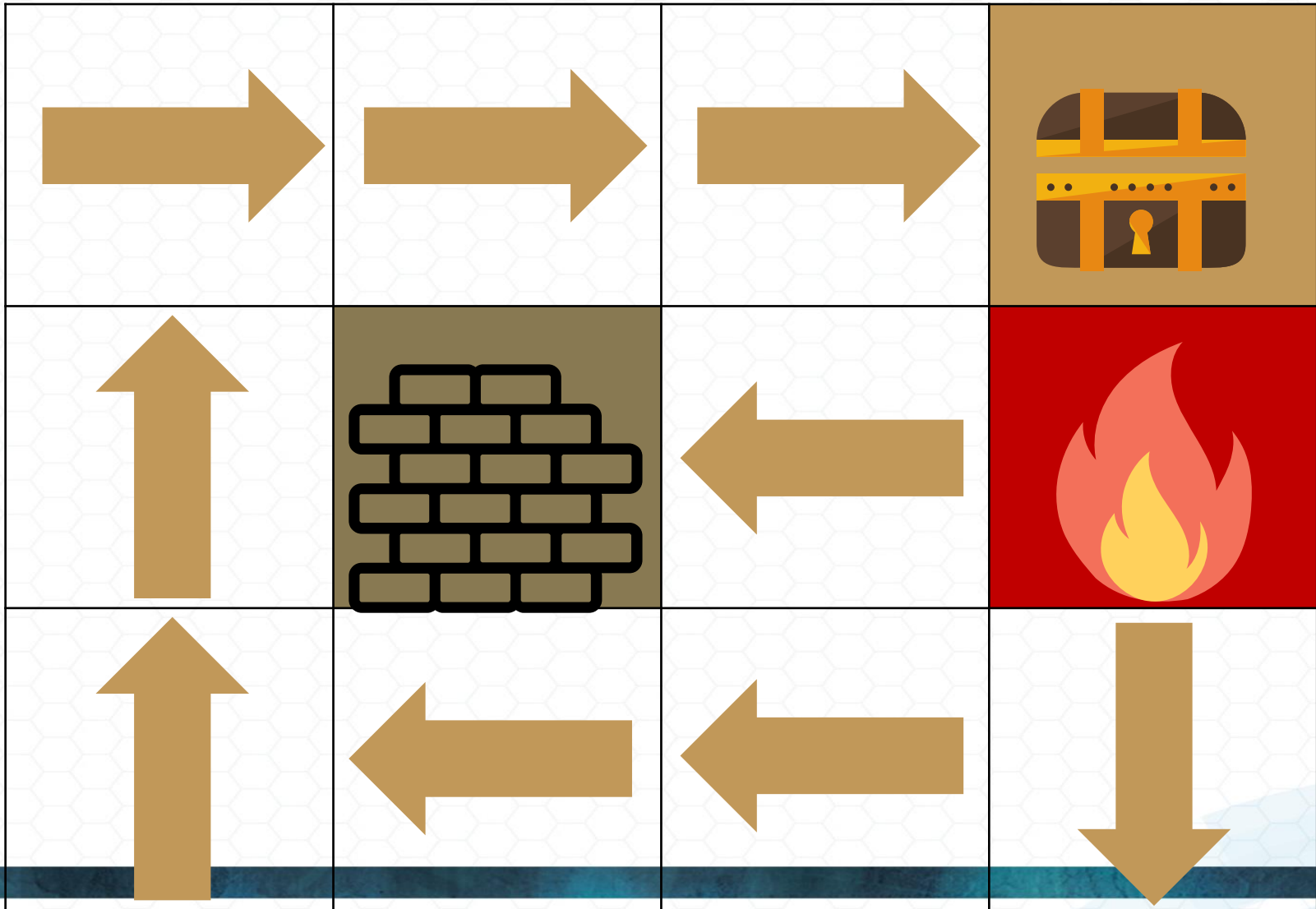


增加獎勵折扣

R=-0.04	R=-0.04	R=-0.04	
R=-0.04		R=-0.04	
R=-0.04	R=-0.04	R=-0.04	R=-0.04

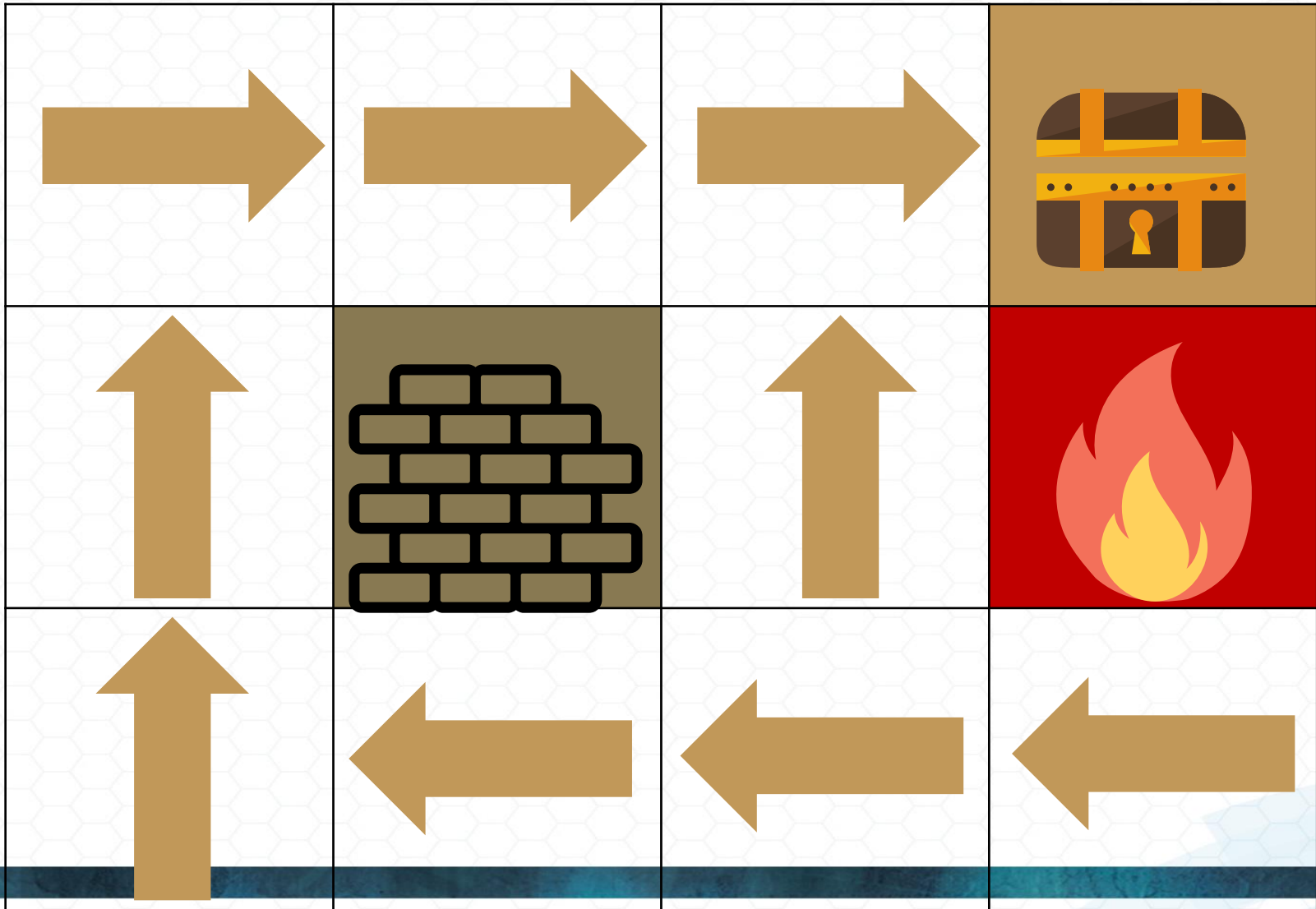
折扣未來報酬 $R(S)=0$

假設 $R(S)=0$



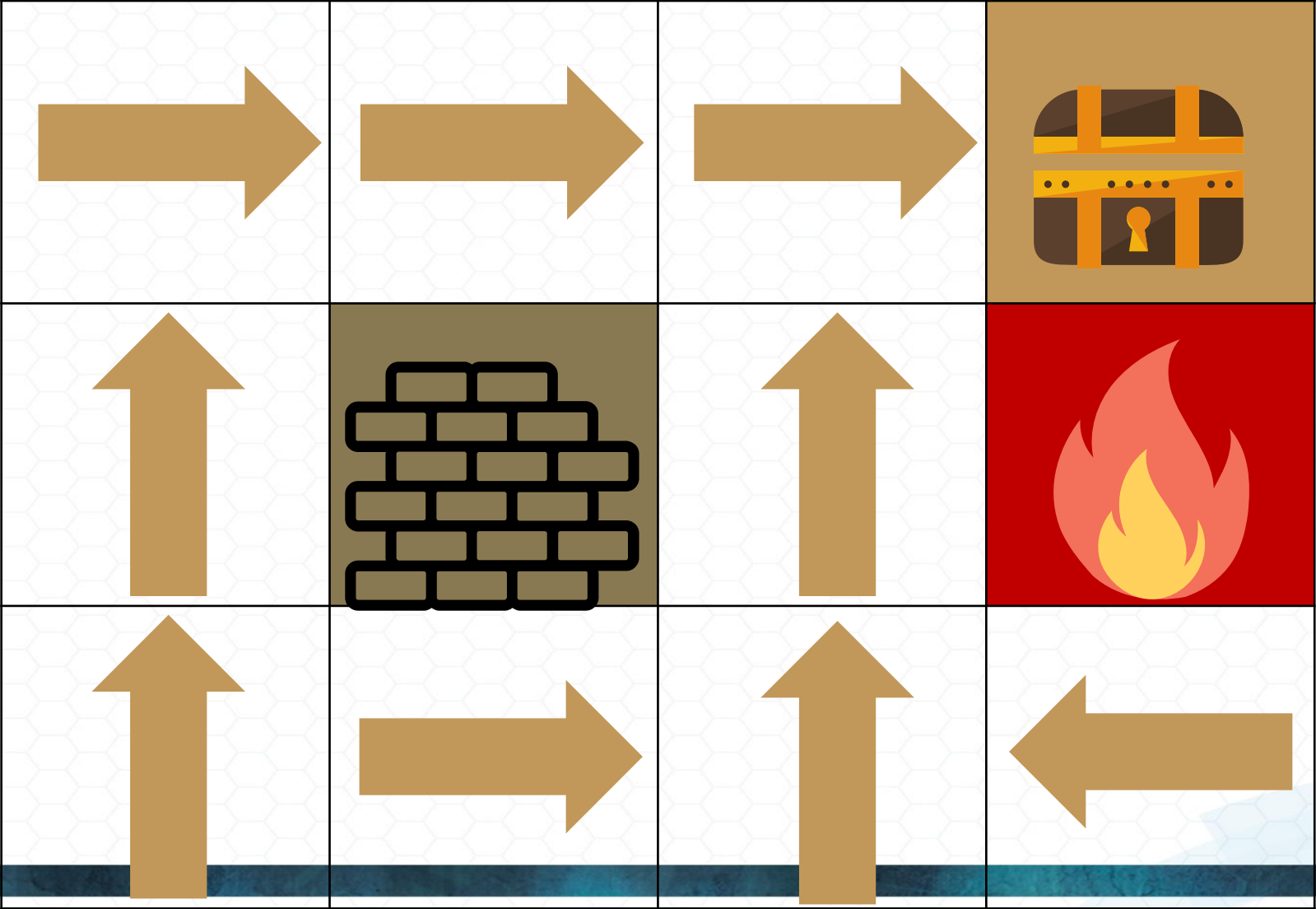
折扣未來報酬 $R(S) = -0.04$

假設 $R(S) = -0.04$

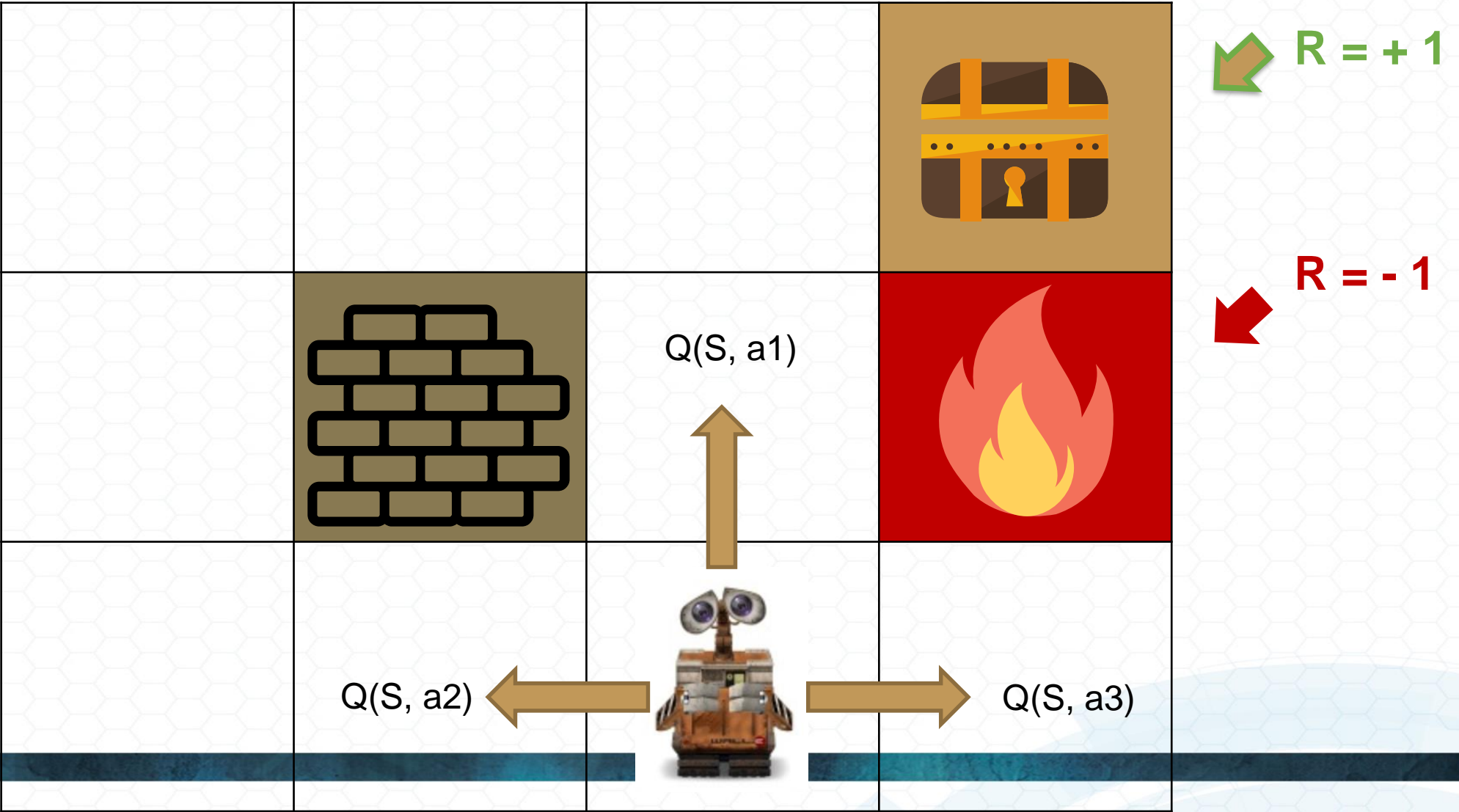


折扣未來報酬 $R(S) = -0.5$

假設 $R(S) = -0.5$



與環境互動過程



Q 函數

- 計算狀態(State) + 動作(Action) 所對應的品質(Quality)
- 計算在狀態 S 下做出動作 a 時，接下來每個動作都是完美的情況(未來報酬最大化)下所對應的結果

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') V(s')$$

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



$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') \max_{a'} (Q(s', a'))$$

隨機策略

$V=0.7$	$V=0.75$	$V=0.85$	
$V=0.63$		$V=0.36$	
$V=0.55$	$V=0.45$	$V=0.3$	$V=0.2$

價值反覆運算

- 根據未來的Q更新過去的Q
- 只要知道正確的Q值，該Q就是回合結束之前最後一個動作的Q。在該結果下我們可以確切知道下一個動作的獎勵

$$\widehat{Q}_j \rightarrow \widehat{Q}_{j+1} \rightarrow \widehat{Q}_{j+2} \rightarrow \cdots \rightarrow Q^*$$

$$TD(a, s) = R(s, a) + \gamma \max_{a'} Q(s', a') - Q_{t-1}(s, a)$$

$$Q_t(s, a) = Q_{t-1}(s, a) + \alpha TD_t(a, s)$$

$$Q_t(s, a) = Q_{t-1}(s, a) + \alpha (R(s, a) + \gamma \max_{a'} Q(s', a') - Q_{t-1}(s, a))$$

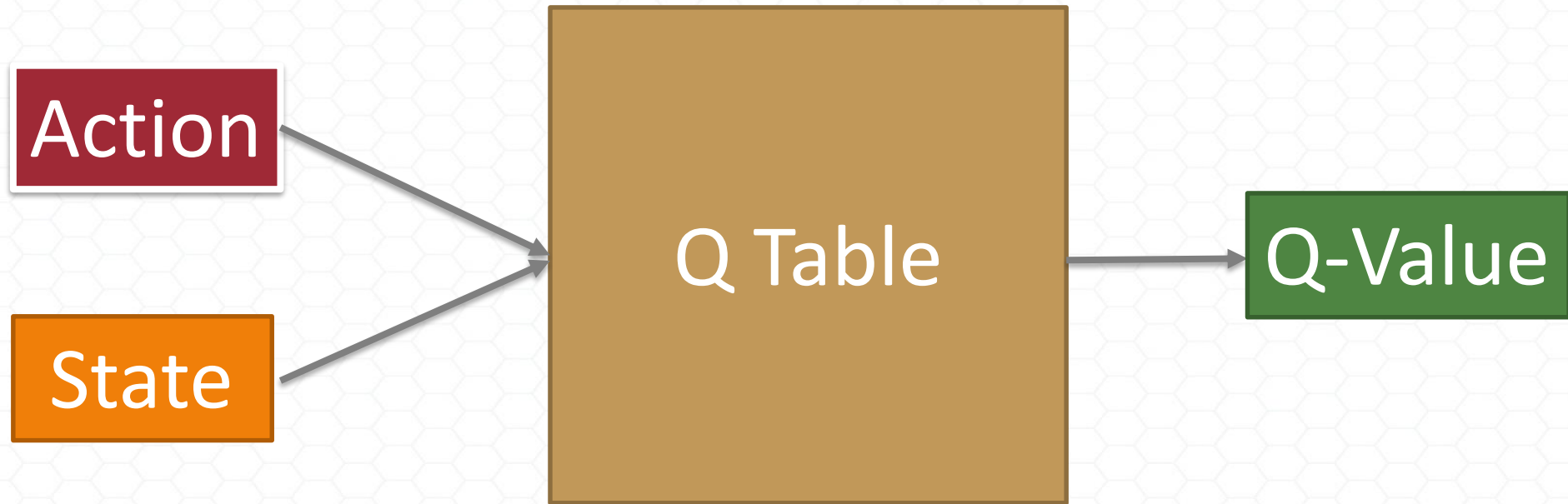
α 是學習率

0 代表 永遠不更新Q

1 代表遺忘過去更新到的Q

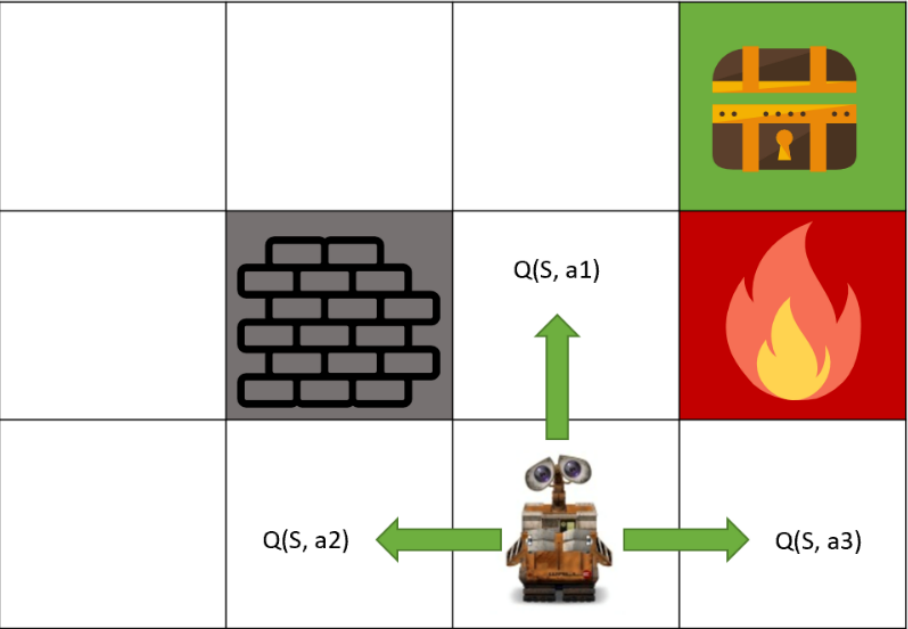
- 透過未來的Q更新現在的Q值，透過價值反覆運算，最後會收斂出較佳的Q

Q Table



更複雜的遊戲

狀態: 12種 動作: 4 種

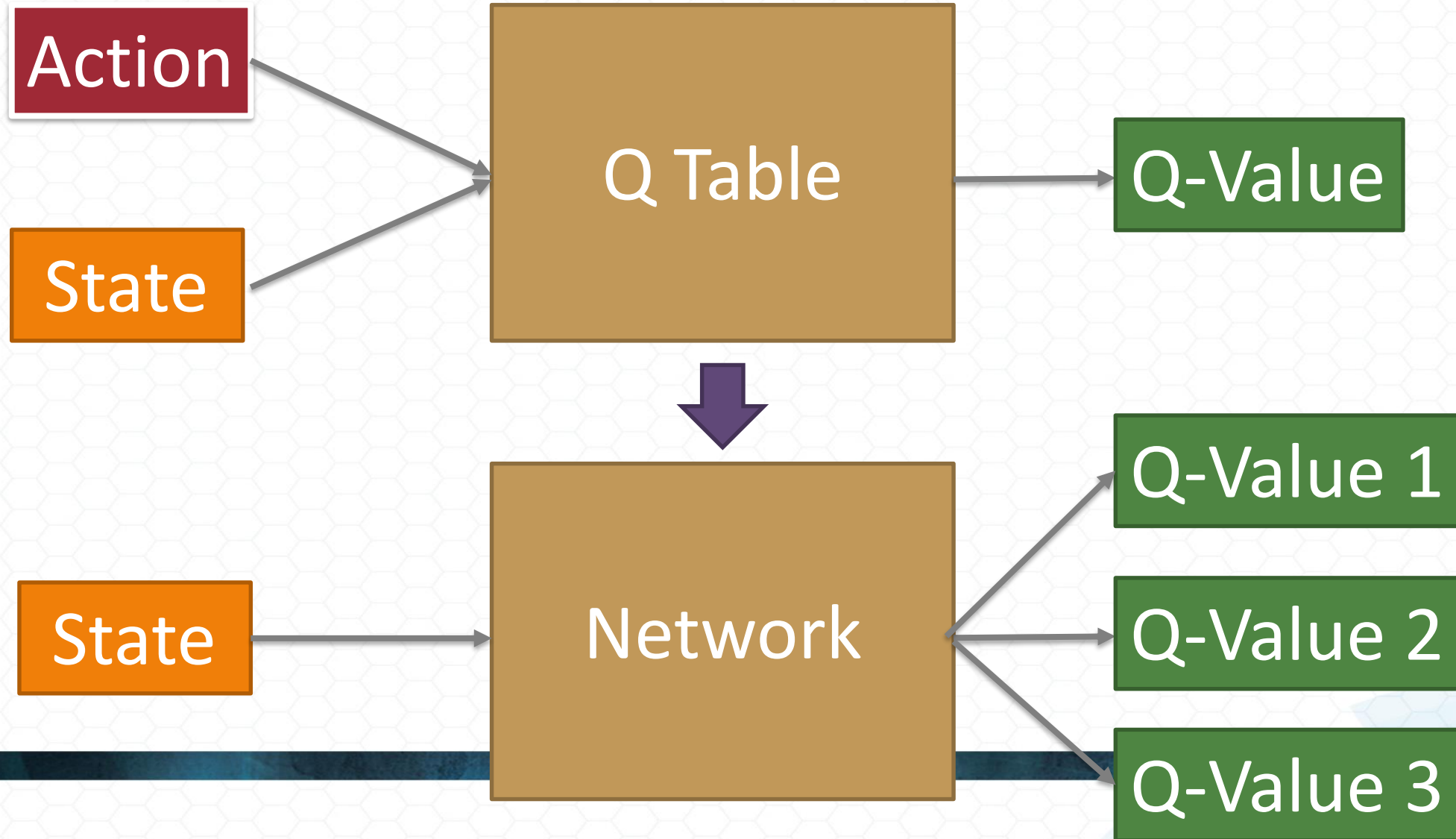


V.S.

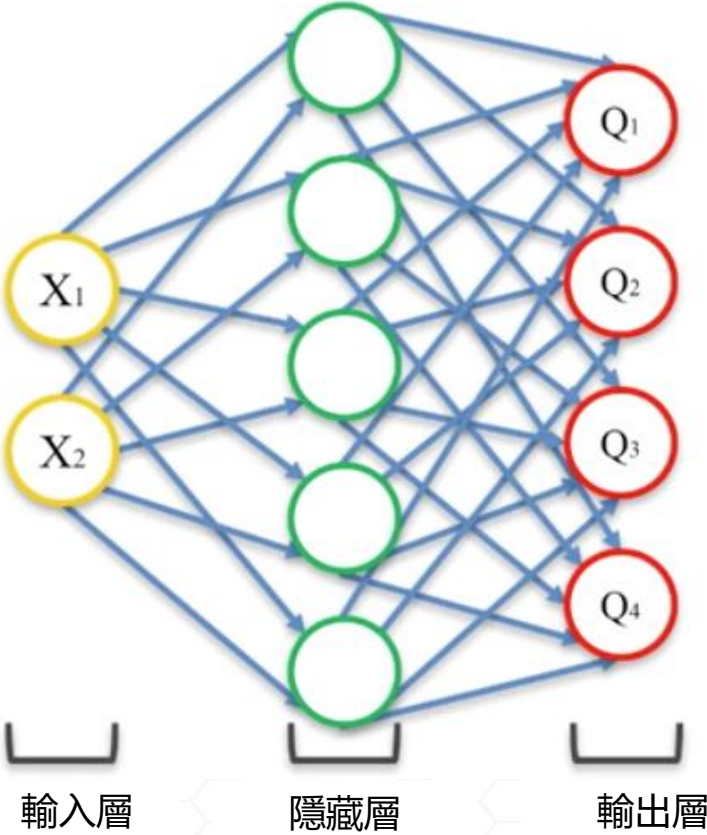
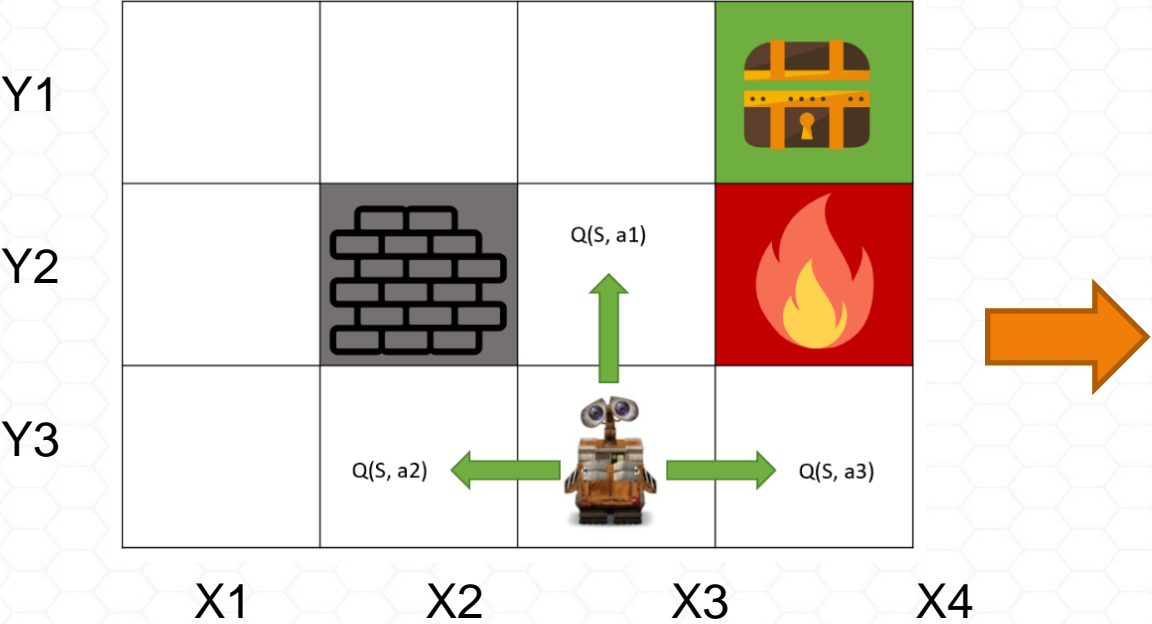
狀態: ?種 動作: ? 種



Deep Q Network



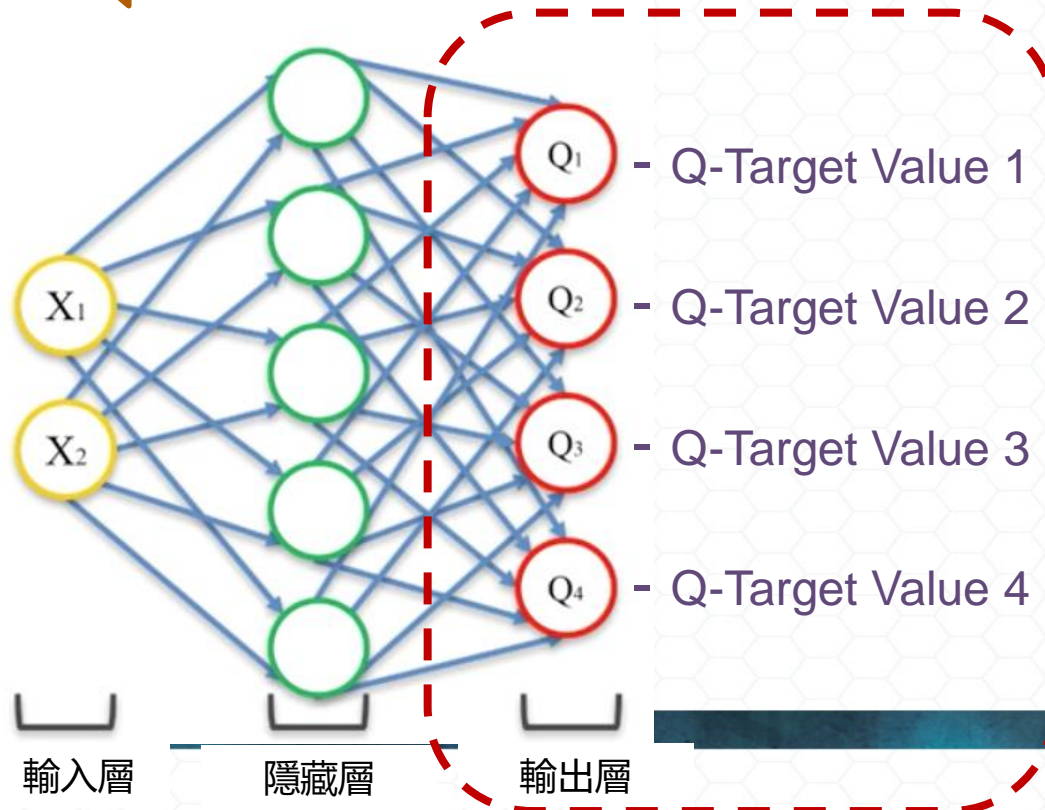
Deep Q Learning



Deep Q Learning

Minimize TD

$$TD(a, s) = R(s, a) + \gamma \max_{a'} Q(s', a') - Q_{t-1}(s, a)$$



$$L = \sum Q_{target} - Q$$

学习稳定性

- 由於損失函數是根據當前步驟與下個步驟的 Q 值差異計算得知，因此當參數更新時， Q 值就會產生固定的偏移，而我們又用偏移過後的 Q 值進行更新，將會引起雙重依賴的情況，導致損失無法收斂

- 解決穩定性問題

- 目標 Q 網路 (Target Q Network)
 - 經驗重播 (Experience Replay)



目標 Q 網路

■ 建立兩個 Q 網路

- ▣ 預測網路 (Prediction Network)
- ▣ 目標Q網路 (Target Q Network)

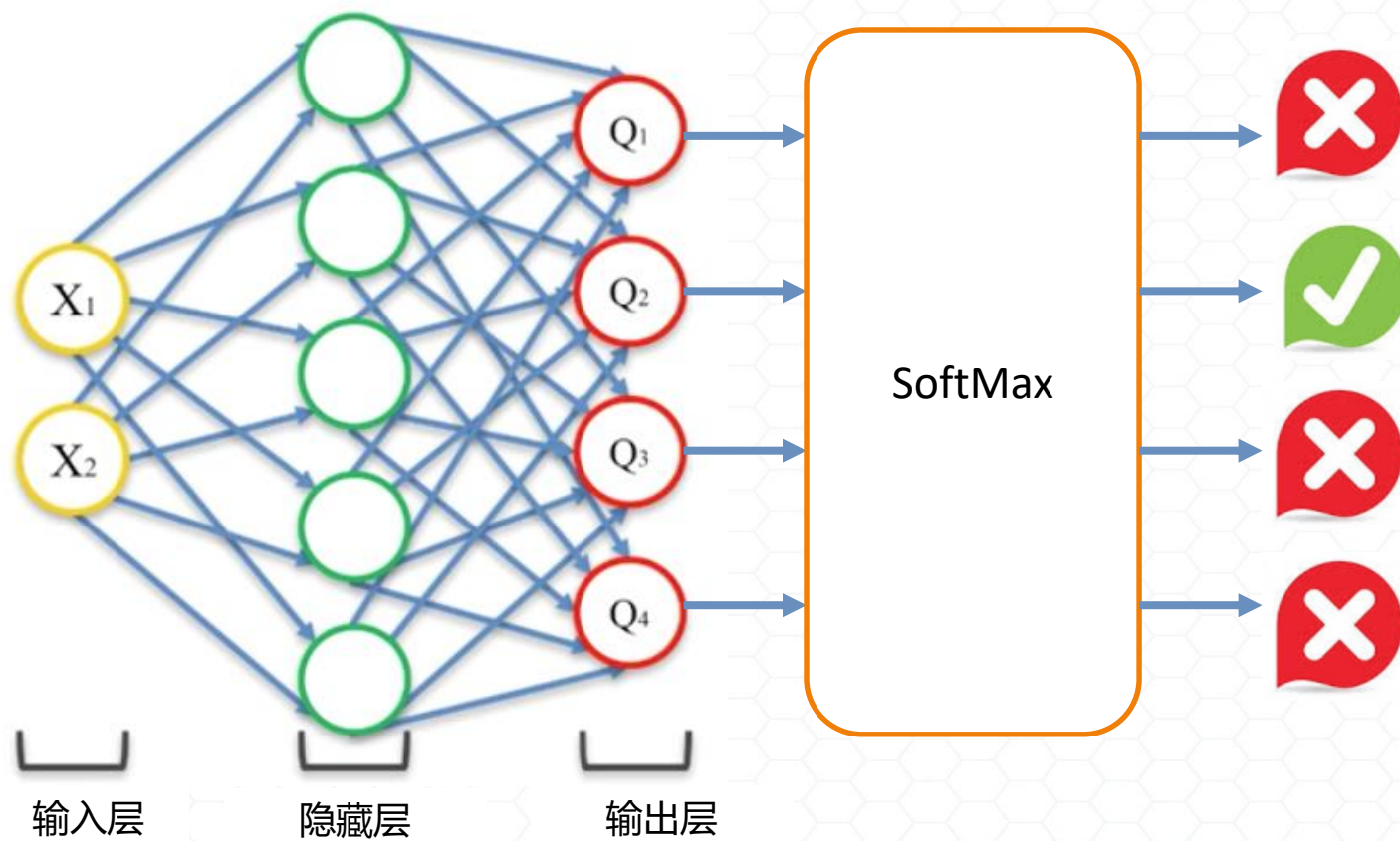
■ 目標Q網路

- ▣ 類似預測網路，但將參數更新延後好幾批資料才根據最新預測網路更新目標Q網路的參數

經驗重播

- 在訓練的時候存儲當前訓練的狀態到一表格中，更新參數的時候隨機從記憶表格中抽樣mini-batch進行更新
- 表格有最大長度的限制，以保證更新採用的資料都是最近的資料

使用 SoftMax 產生策略



Exploration v.s. Exploitation

Exploration :

在剛開始訓練的時候，為了能夠看到更多可能的情況，需要對action加入一定的隨機性。

Exploitation :

隨著訓練的加深，逐漸降低隨機性，也就是降低隨機action出現的概率。

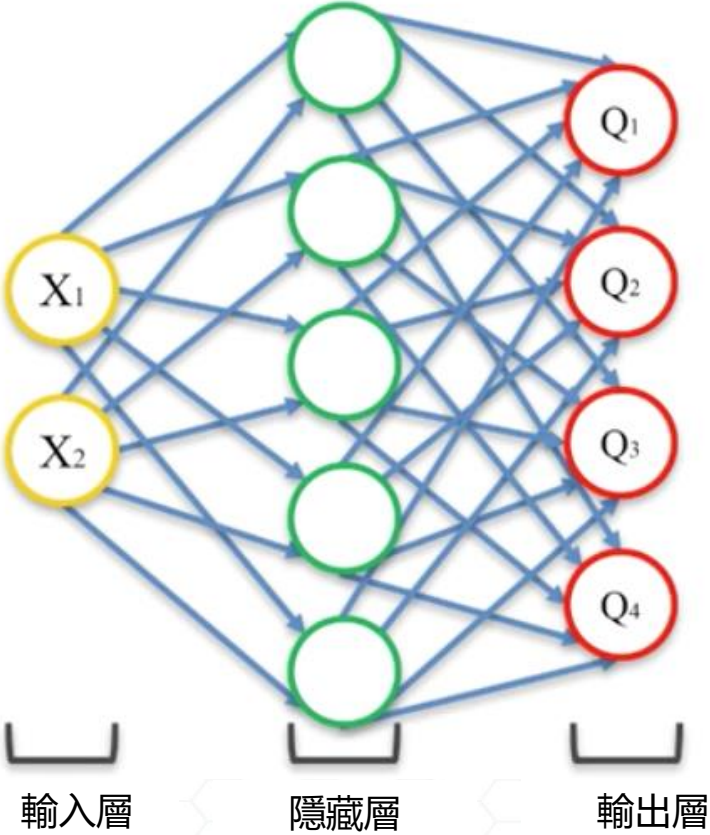
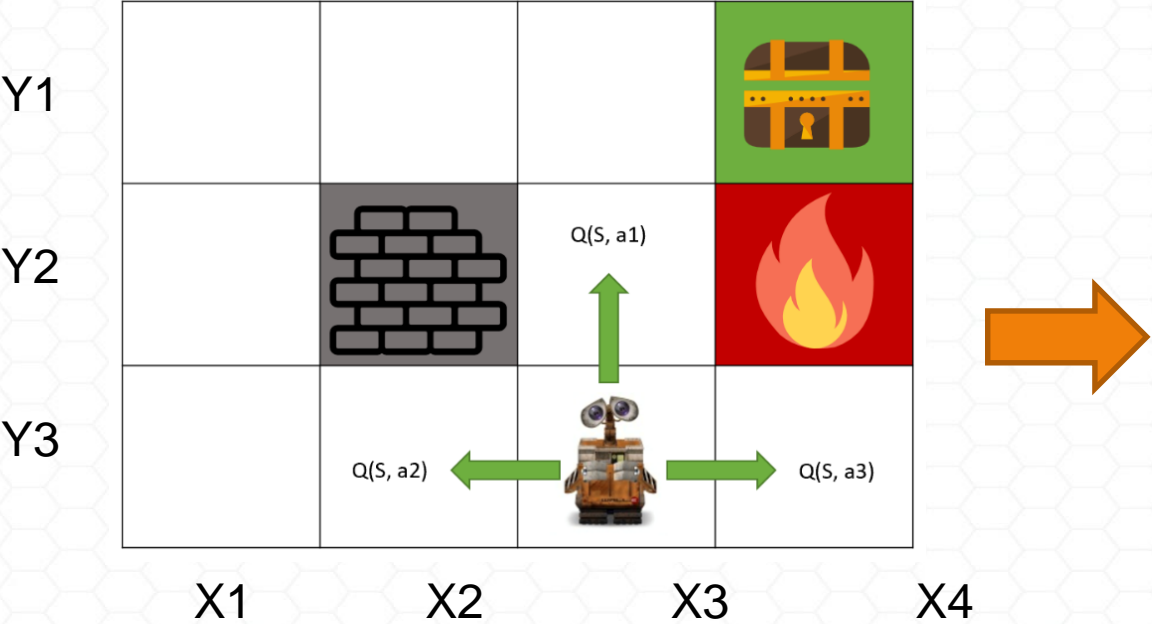
其他動作

■ epsilon-Greedy

{	$\operatorname{argmax}_a Q_a$	概率 $(1 - \epsilon)$
	隨機選擇一個行動	概率 ϵ

■ SoftMax

Deep Q Learning

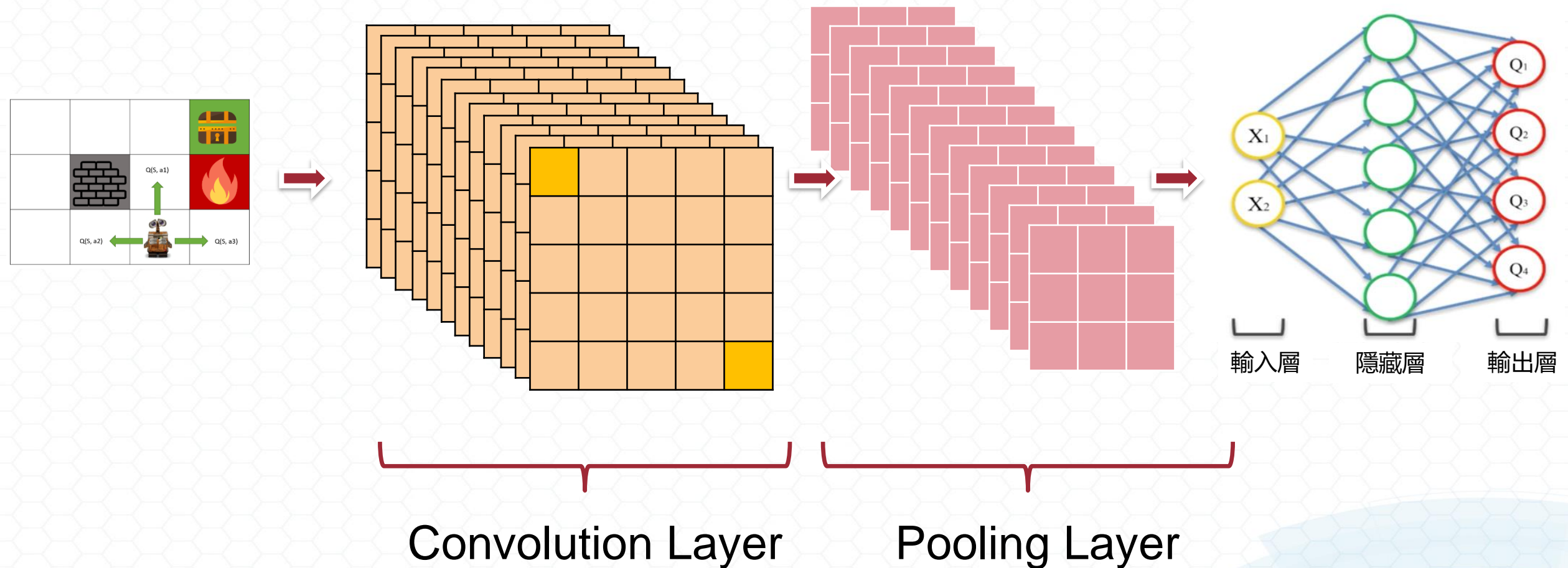


Deep Convolution Q-Learning

Convolution

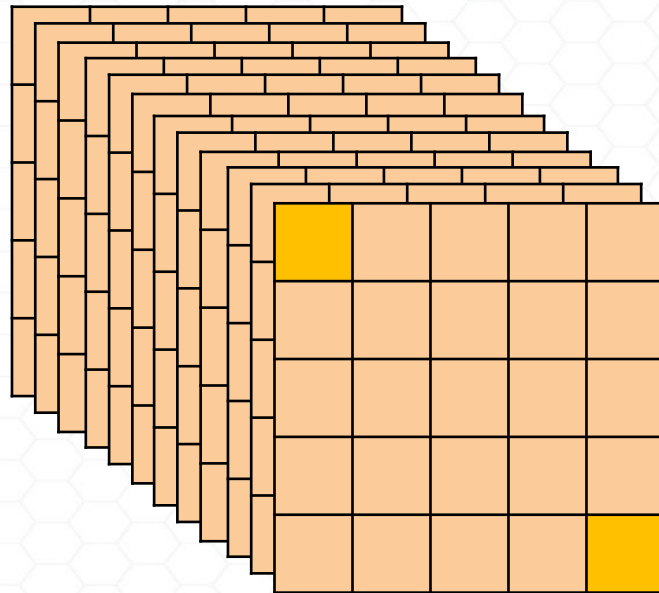
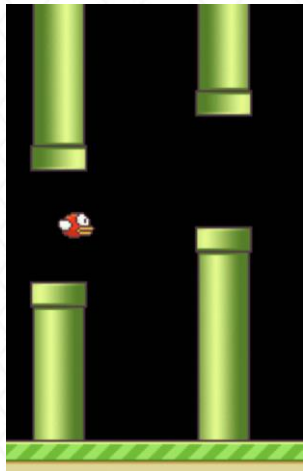
Pooling

Flattening



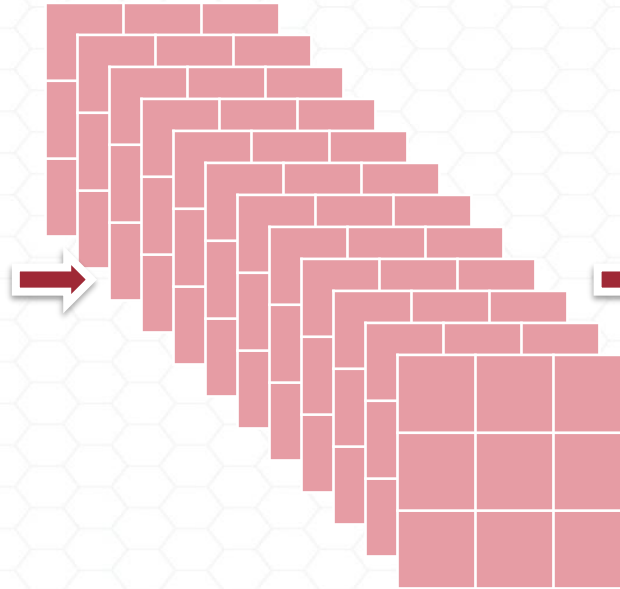
Deep Convolution Q-Learning

Convolution



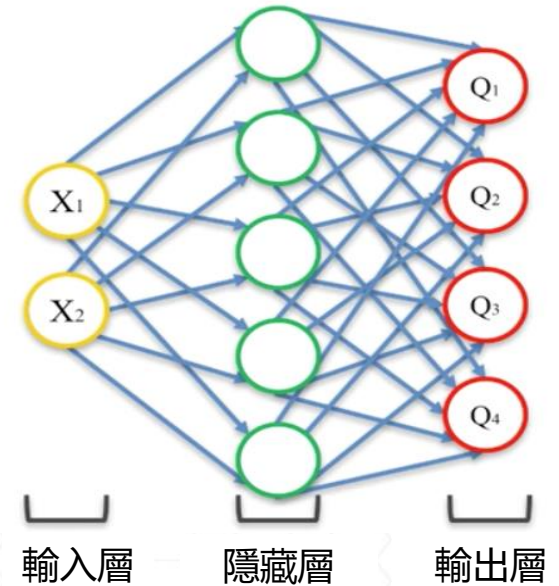
Convolution Layer

Pooling



Pooling Layer

Flattening



輸入層

隱藏層

輸出層

The background features a light gray hexagonal grid pattern. Overlaid on this is a series of concentric, semi-transparent circles in shades of light blue and white, creating a ripple effect. A solid dark blue horizontal line runs across the top of the image, and a darker, textured blue horizontal band is at the bottom.

THANK YOU