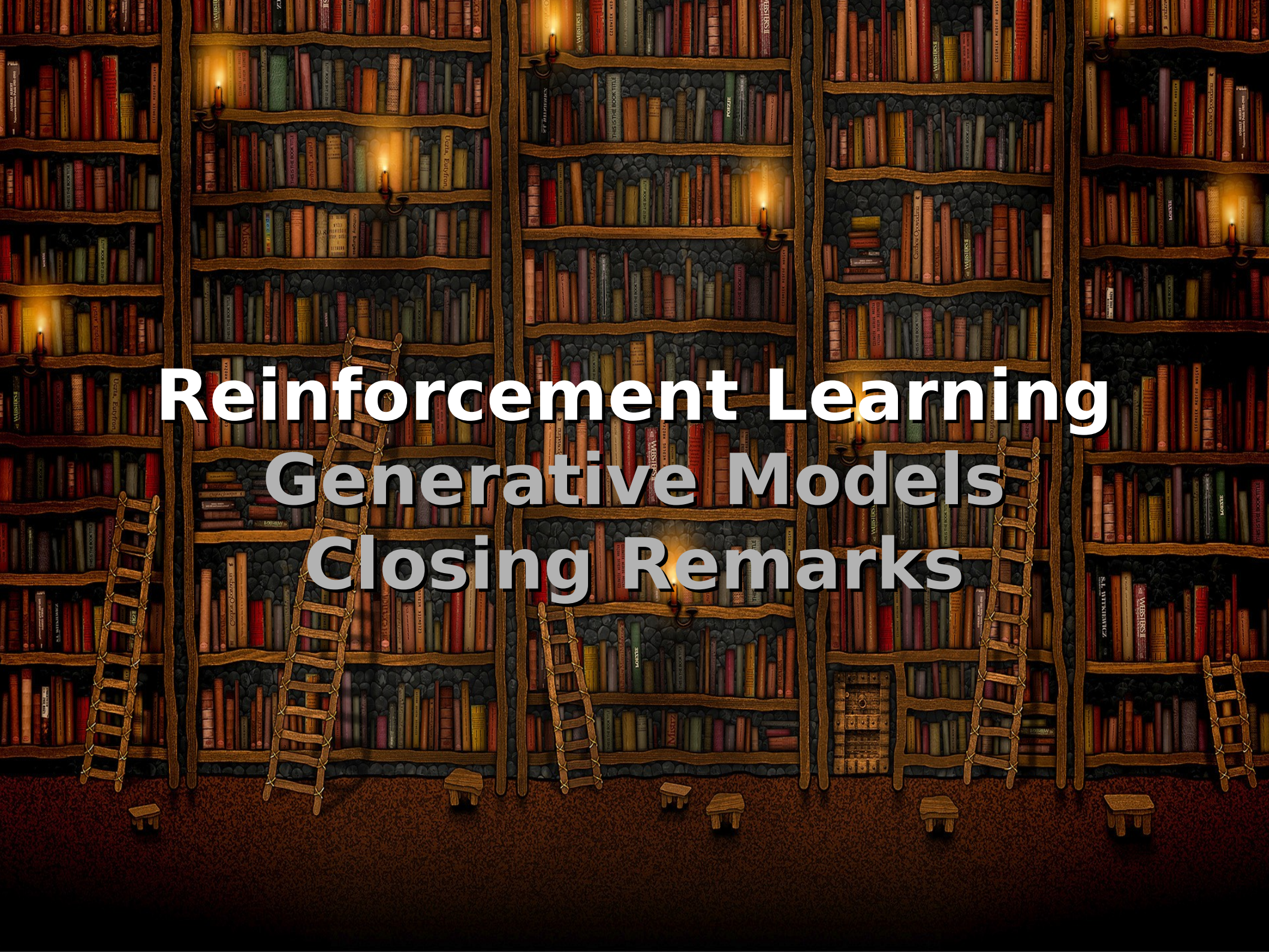
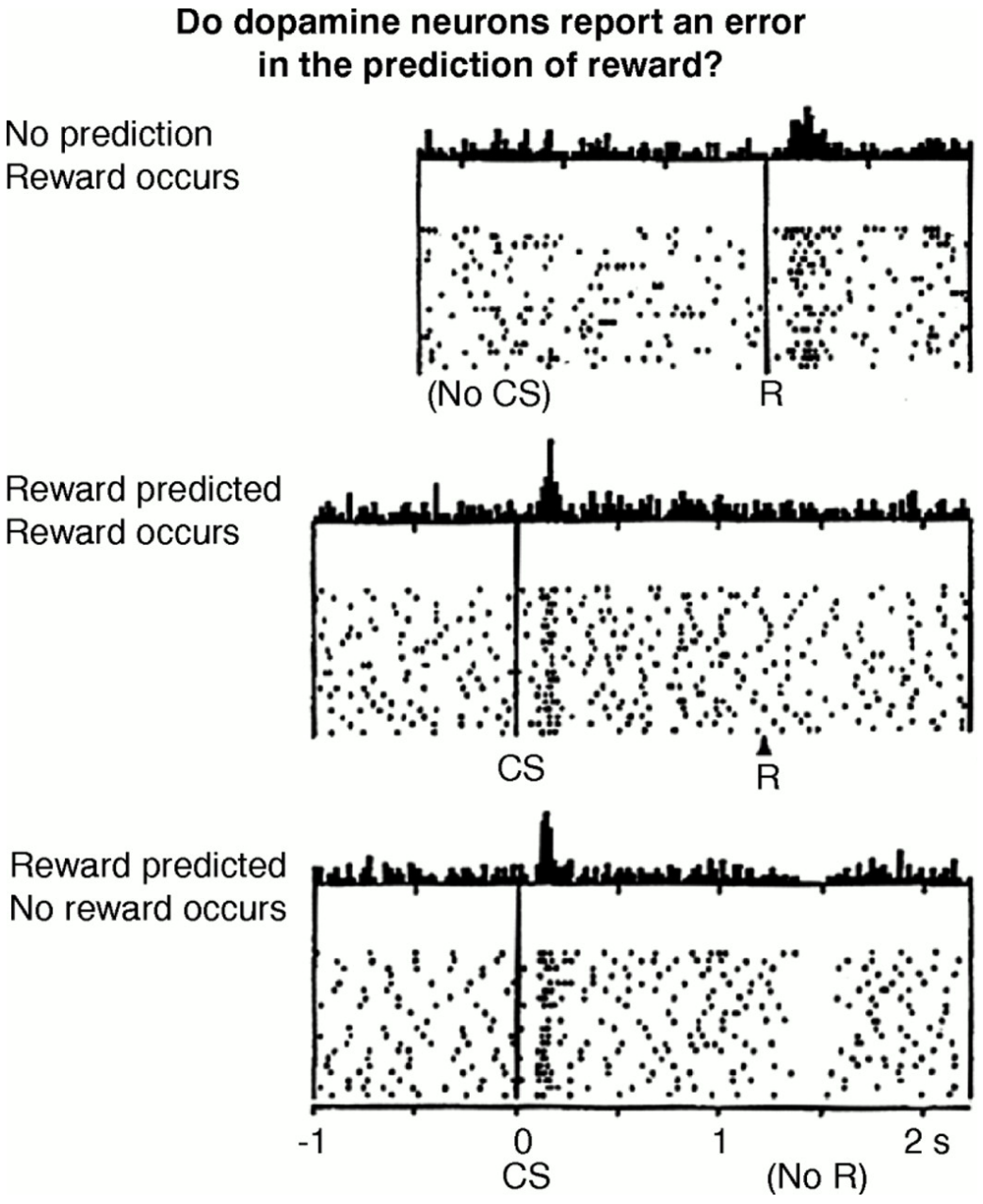
# 神經與行為模型建構
# (Neural & Behavioral Modeling)

課號：Psy7277　　教室： 北 206
識別碼：227M9280　　時間： 五 234


今天來把這個層次了結
！


Reinforcement Learning
Generative Models
Closing Remarks

# 獎賞與多巴胺 (Dopamine)




古典制約學習
可看成是學 S-S 連結
或是刺激替代

注意左圖神經元反應的
是 " 獎賞的預測錯誤 "
而非 " 獎賞 " 本身

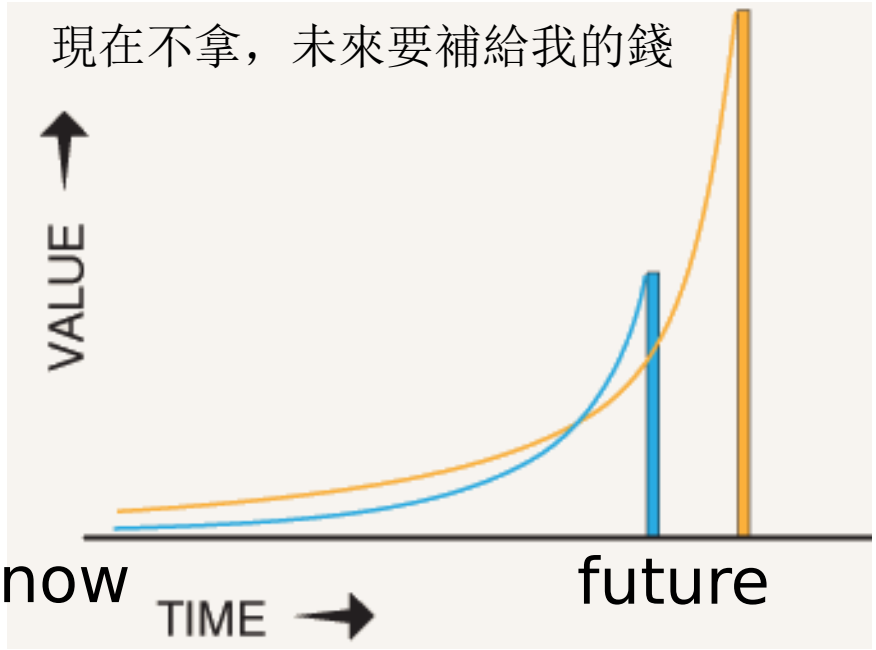
可用 temporal δ rule
來解釋 DA dip=r-r̂

$$\hat{r}^t = \sum_i w_i^{t-1} x_i^{t-1}$$

$$\Delta w_i = x_i^{t-1} \delta^t = x_i^{t-1} \left( r^t - \hat{r}^t \right)$$

# 更精確的獎賞估計應考慮長遠的未來

## 沒帶來立即獎賞的 CS 其實預測之後的各種獎賞




$$\hat{V}^t = \sum_i w_i^{t-1} x_i^{t-1}$$

預期 reward

$$\Delta w_i = x_i^{t-1}\left(V^t - \hat{V}^t\right) = x_i^t \delta V^t$$

因不知 V 無法計算 δV

$$V^t = r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} \ldots$$

真實reward(要考慮temporal discounting)

若 γ=0 回到上頁公式

上式可改寫成遞迴形式：

$$V^t(t-1) = r^t(t) + \gamma V^{t+1}(t)$$

預測將來第t期的酬賞

現在時間點t-1預測

獎賞的時間離當下愈遠價值愈低 (temporal discounting)

學習目標： 上面等式成立

學習過程：讓 $\delta V \equiv r(t) + \gamma \hat{V}(t) - \hat{V}(t-1) \to 0$

# 古典制約的 TD Learning

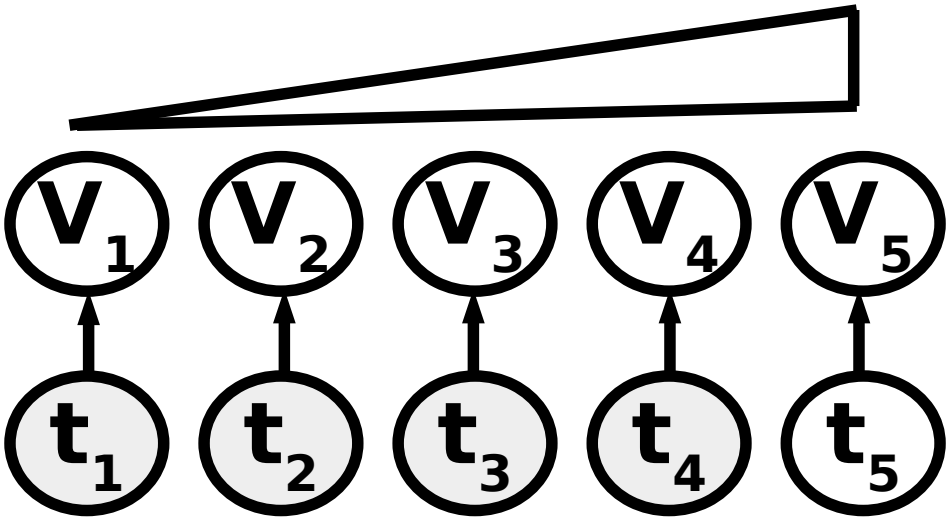
TD Learning $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

Previous estimate

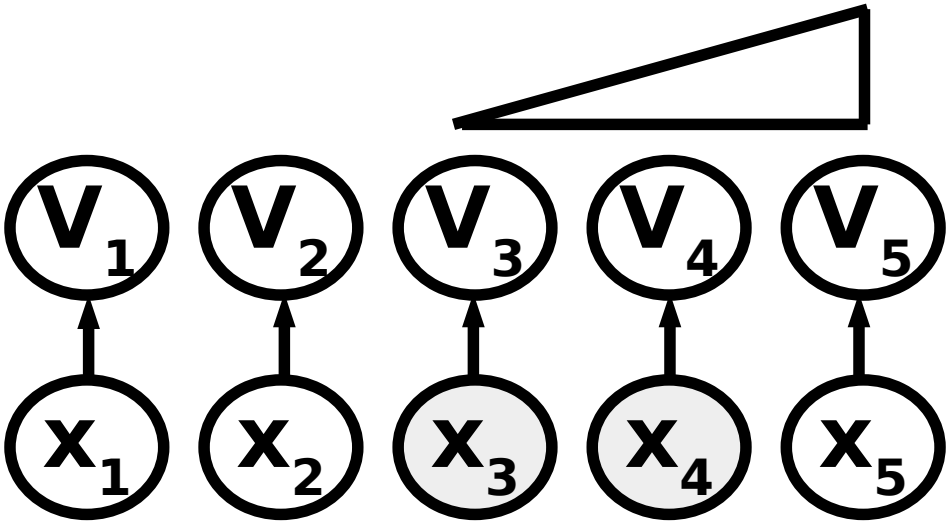
Reward t+1

Discounted value on the next step

TD Target




若 s=t 則 V(t) 變為
無邊界的時間折價函數
（如無刺激出現但
固定 5 秒後出現酬賞）




若 s=x(t) 則 V(x) 變為
有邊界的時間折價函數
（如有刺激出現，且之後
固定 3 秒後出現酬賞）

# 操作制約的 Q-Learning(1/2)

## Q(s,a) 是狀態 s 與行動 a 的函數

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

## 在 t 時要選什麼動作變成下一個狀態要根據既定 policy

## 計算上常離散化 s 與 a 來做成一個 Q(s,a) 的查找表

**Game Board:**

Current state (*s*): 000 010

**Q Table:** γ = 0.95

| | 000 100 | 000 010 | 000 001 | 100 000 | 010 000 | 001 000 |
|---|---|---|---|---|---|---|
| ⇧ | 0.2 | 0.3 | 1.0 | -0.22 | -0.3 | 0.0 |
| ⇩ | -0.5 | -0.4 | -0.2 | -0.04 | -0.02 | 0.0 |
| ⇨ | 0.21 | 0.4 | -0.3 | 0.5 | 1.0 | 0.0 |
| ⇦ | -0.6 | -0.1 | -0.1 | -0.31 | -0.01 | 0.0 |

一開始都是0，在學習的過程中 trial and error，跟新這些數值
完成 Q(s, a) table，最後就能找到最佳的 policy

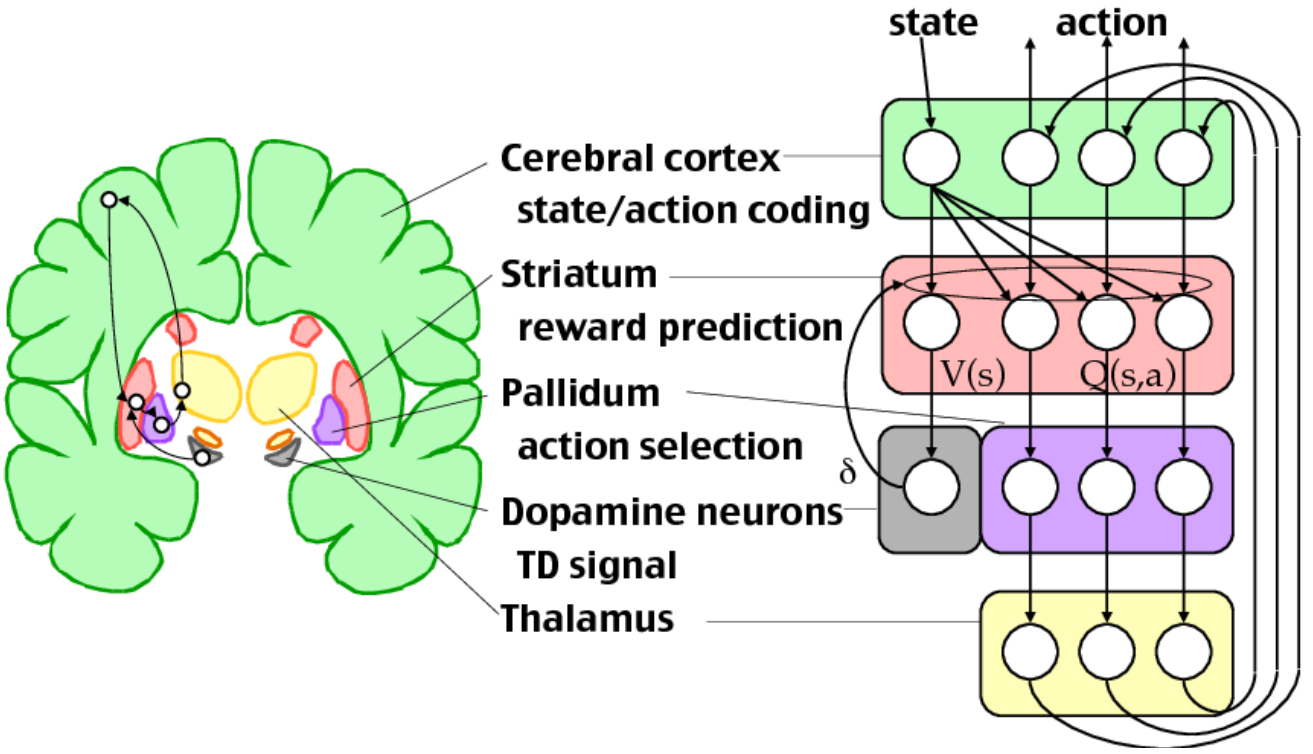
# 操作制約的 Q-Learning(2/2)

Q(s,a) 是狀態 s 與行動 a 的函數

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

在 t 時要選什麼動作變成下一個狀態要根據既定 policy

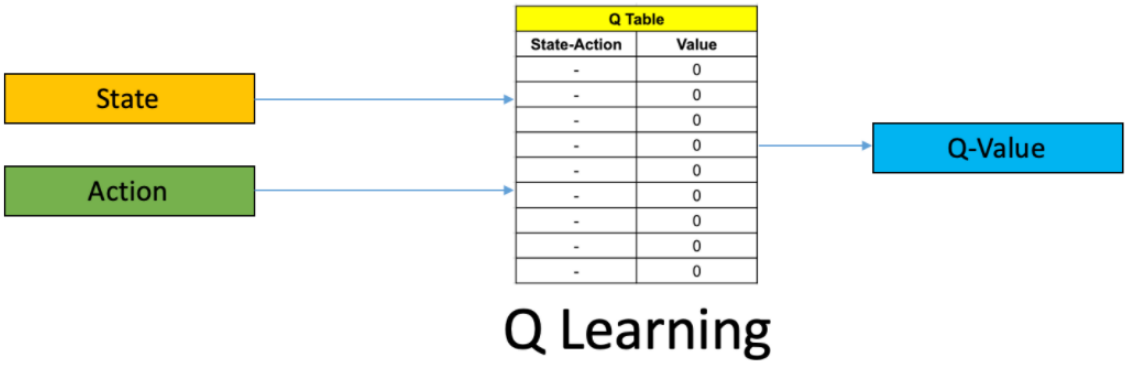
計算上常離散化 s 與 a 來做成一個 Q(s,a) 的查找表

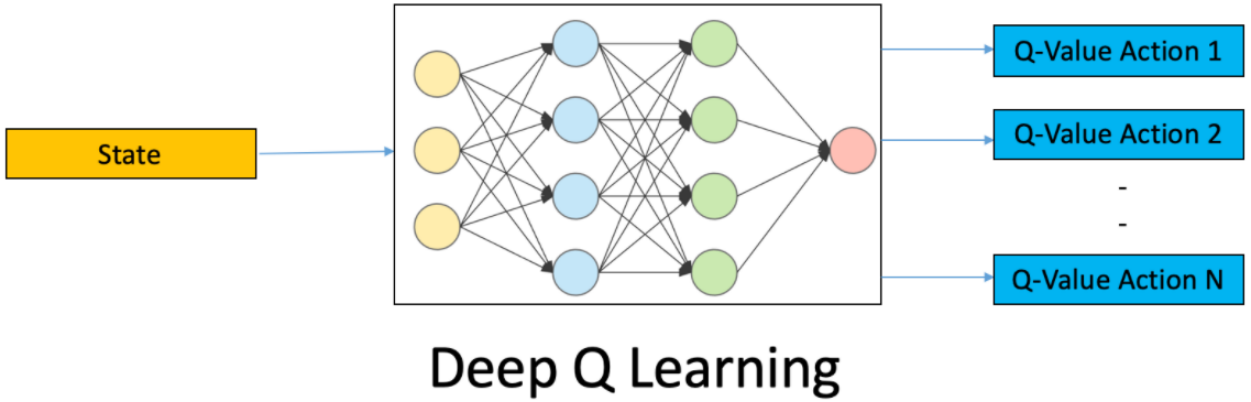


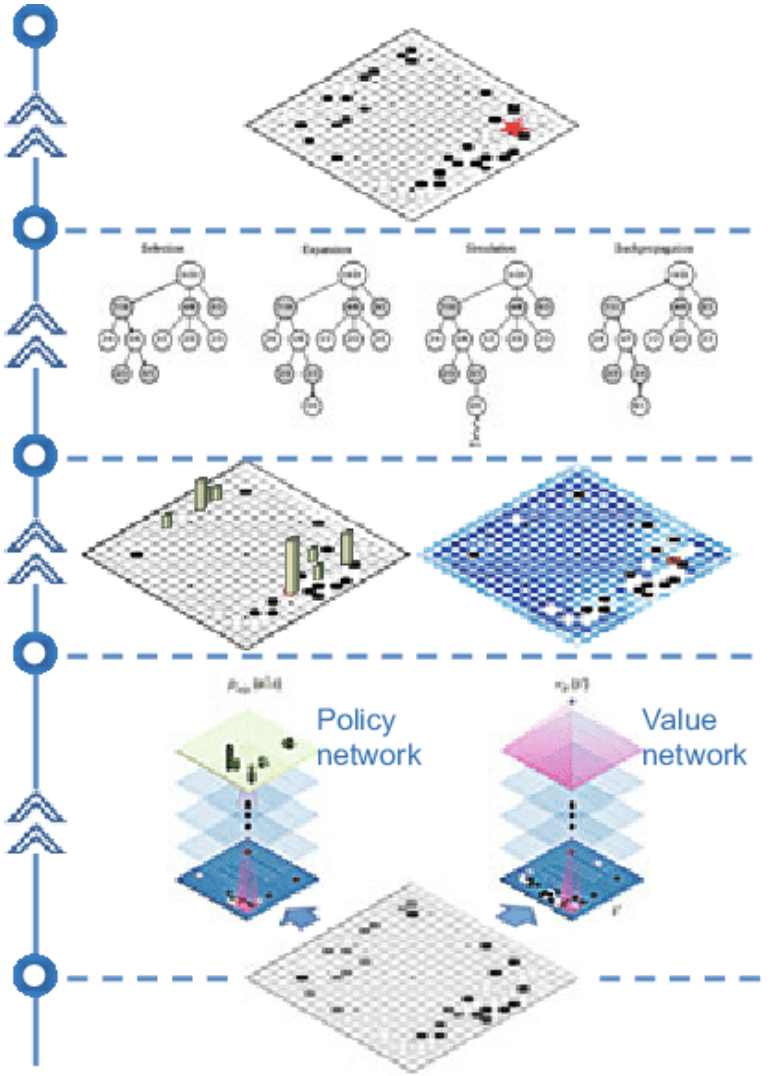
# Deep Q Network (1/5)

2015 年

2016 年



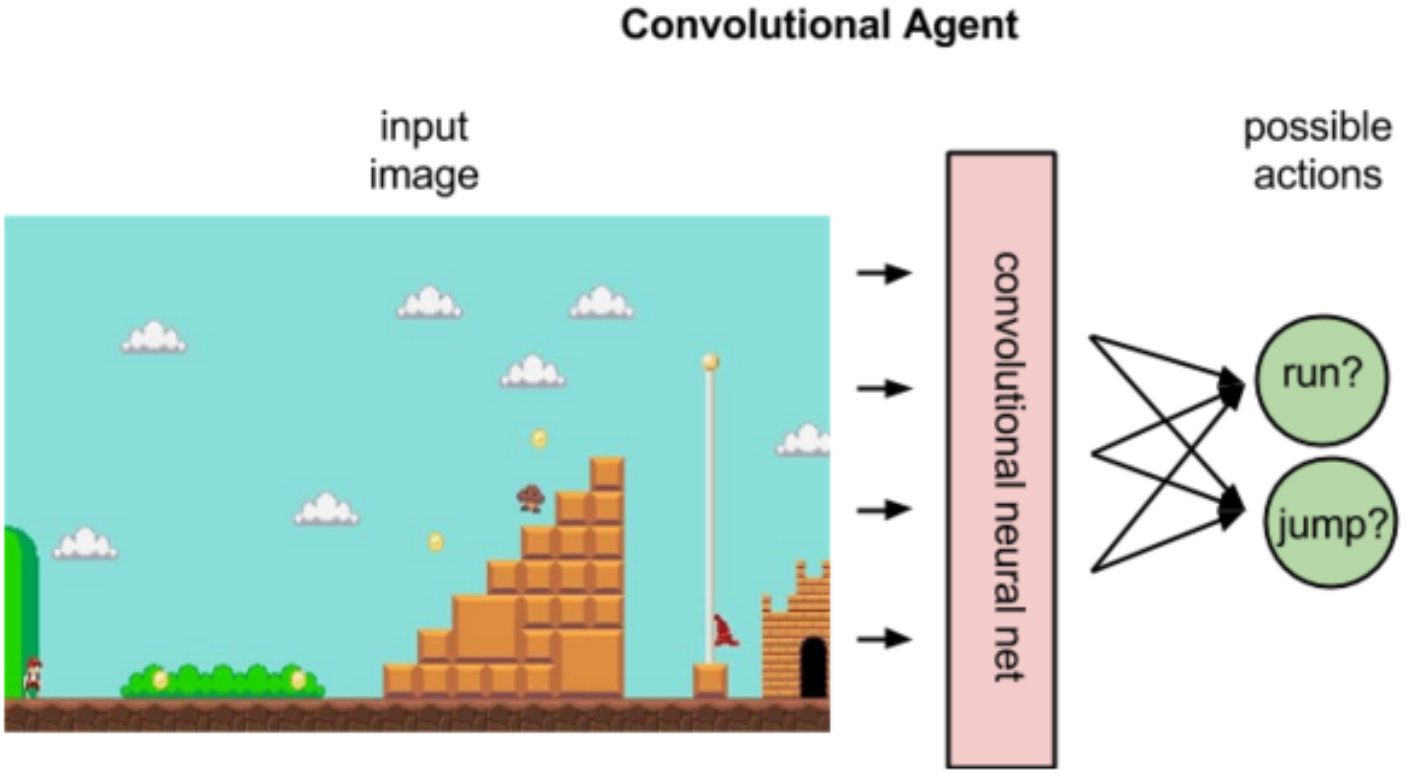




用一 NN 來逼近 Q Table

（因為 state 不離散）

# Deep Q Network (2/5)

學習上的困難 :X(t)~X(t+1) 但 Y(t)≠Y(t+1)

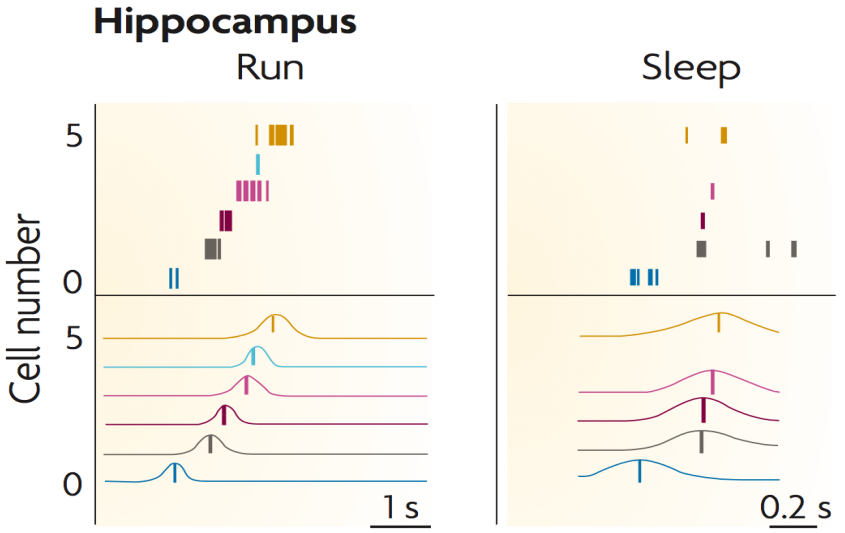


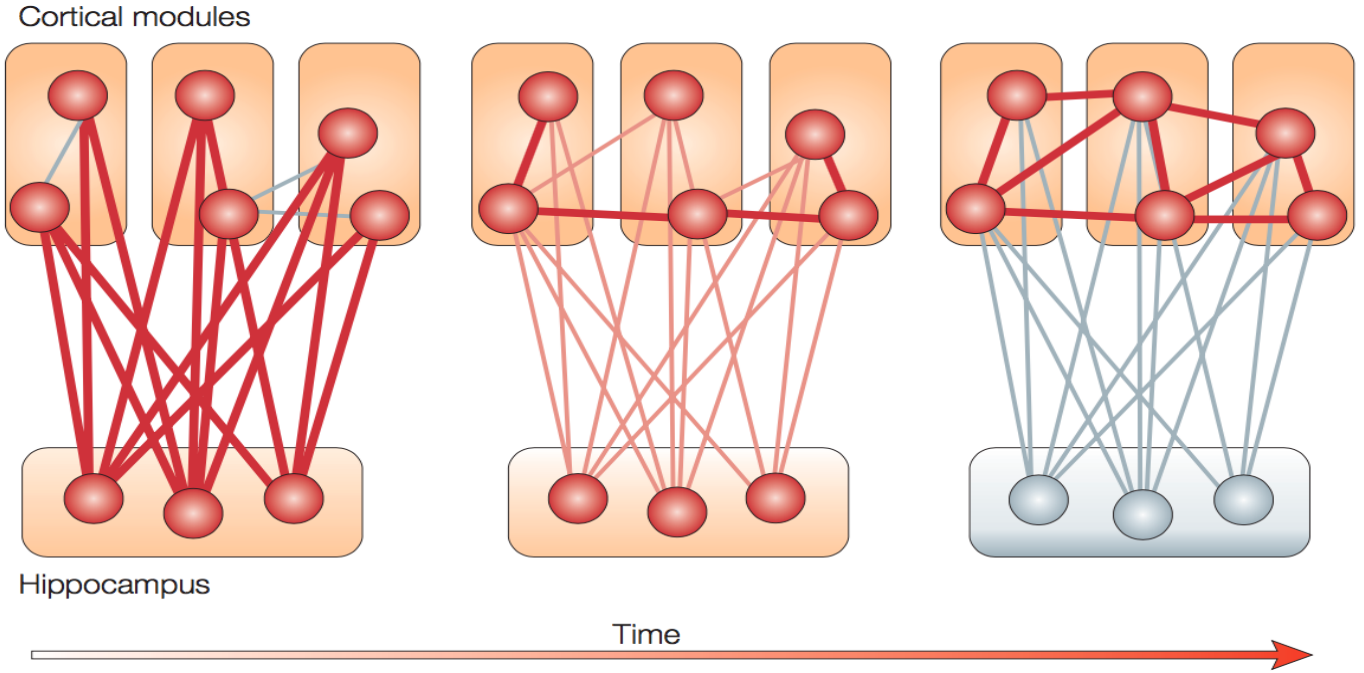

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

也讓快學的完美記憶教慢學的 Q-learning

# Deep Q Network (3/5)

## 引入海馬 Experience Replay 解決穩定性 - 彈性兩難



# Deep Q Network (4/5)

## 引入分開的 Target Network 隔一陣子才更新

**Algorithm 1: deep Q-learning with experience replay.**

Initialize replay memory $D$ to capacity $N$

Initialize action-value function $Q$ with random weights $\theta$

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

**For** episode $= 1, M$ **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

**For** $t = 1$,T **do**

With probability $\varepsilon$ select a random action $a_t$ action selection

otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$ Experience Replay

Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{target Network otherwise} \end{cases}$

Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters $\theta$ 更新Q-network权重

Every $C$ steps reset $\hat{Q} = Q$ 每隔C步更新target network

**End For**

**End For**

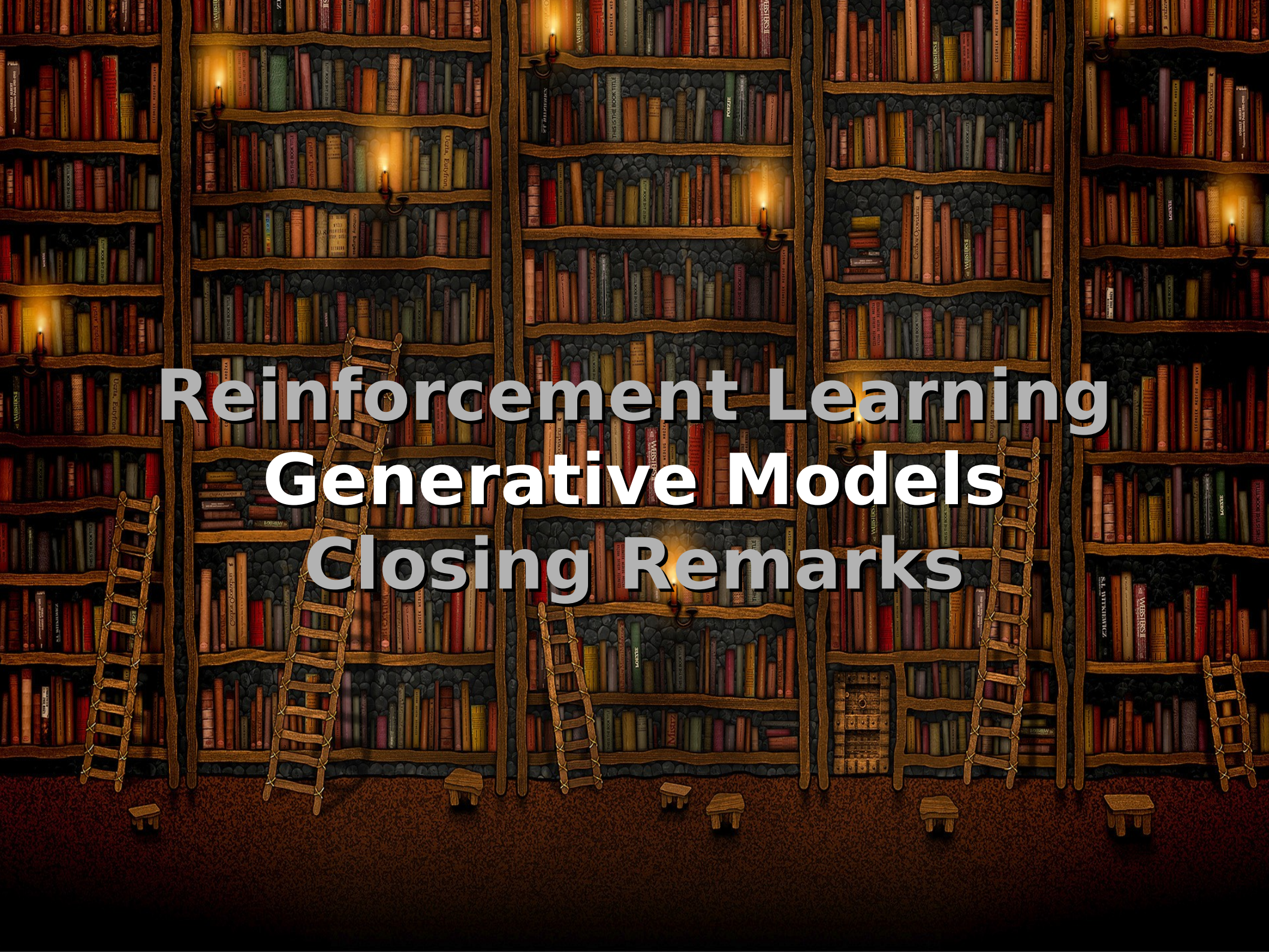
# Deep Q Network (5/5)

## 獨立的 Q-Net 與 Experience Replay 效果卓著

**Extended Data Table 3 | The effects of replay and separating the target Q-network**

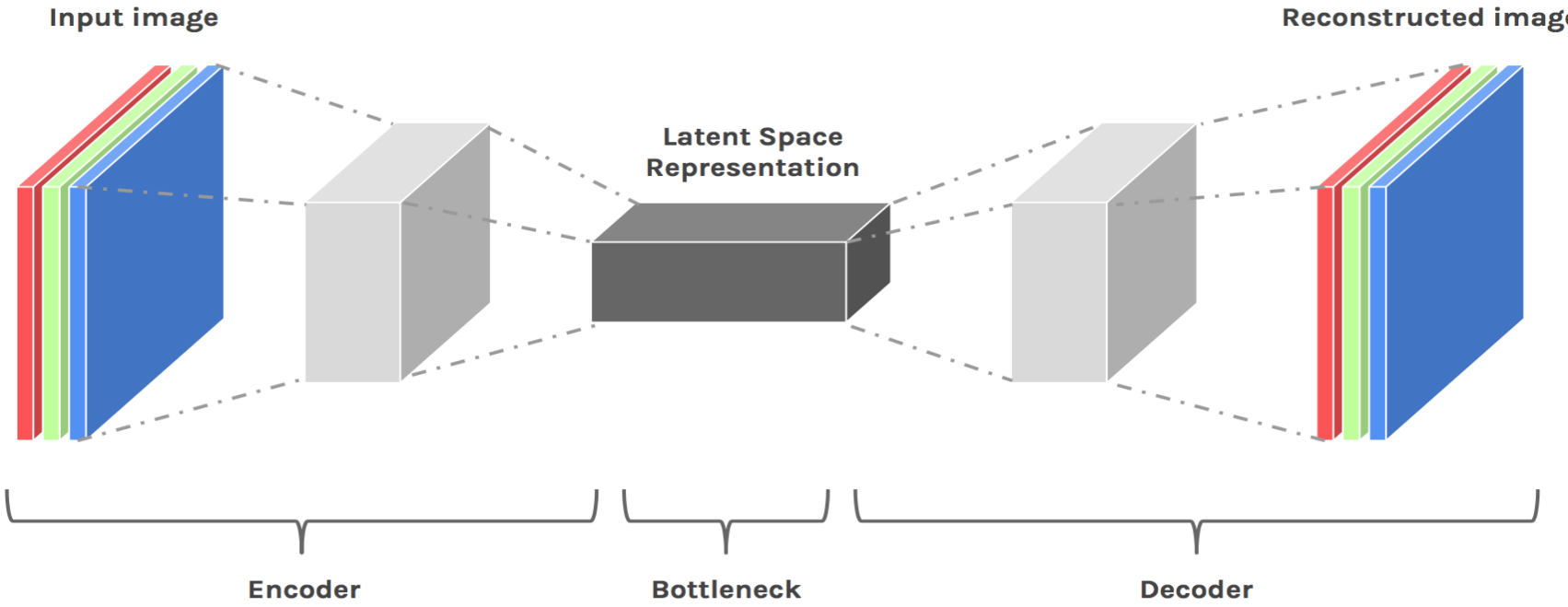
| Game | With replay, with target Q | With replay, without target Q | Without replay, with target Q | Without replay, without target Q |
|---|---|---|---|---|
| Breakout | 316.8 | 240.7 | 10.2 | 3.2 |
| Enduro | 1006.3 | 831.4 | 141.9 | 29.1 |
| River Raid | 7446.6 | 4102.8 | 2867.7 | 1453.0 |
| Seaquest | 2894.4 | 822.6 | 1003.0 | 275.8 |
| Space Invaders | 1088.9 | 826.3 | 373.2 | 302.0 |

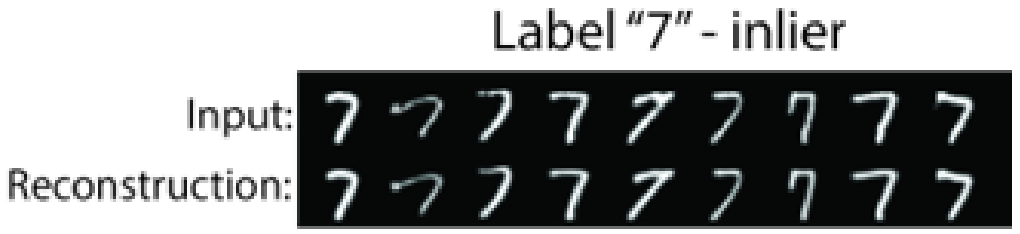
DQN agents were trained for 10 million frames using standard hyperparameters for all possible combinations of turning replay on or off, using or not using a separate target Q-network, and three different learning rates. Each agent was evaluated every 250,000 training frames for 135,000 validation frames and the highest average episode score is reported. Note that these evaluation episodes were not truncated at 5 min leading to higher scores on Enduro than the ones reported in Extended Data Table 2. Note also that the number of training frames was shorter (10 million frames) as compared to the main results presented in Extended Data Table 2 (50 million frames).

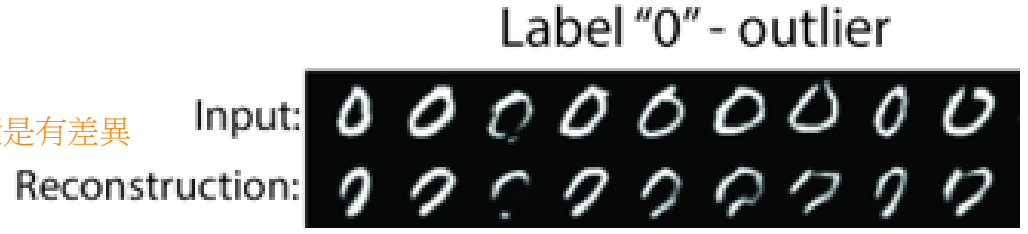

Reinforcement Learning
Generative Models
Closing Remarks

# AutoEncoder

## 可以產生一個 latent/embedding representation



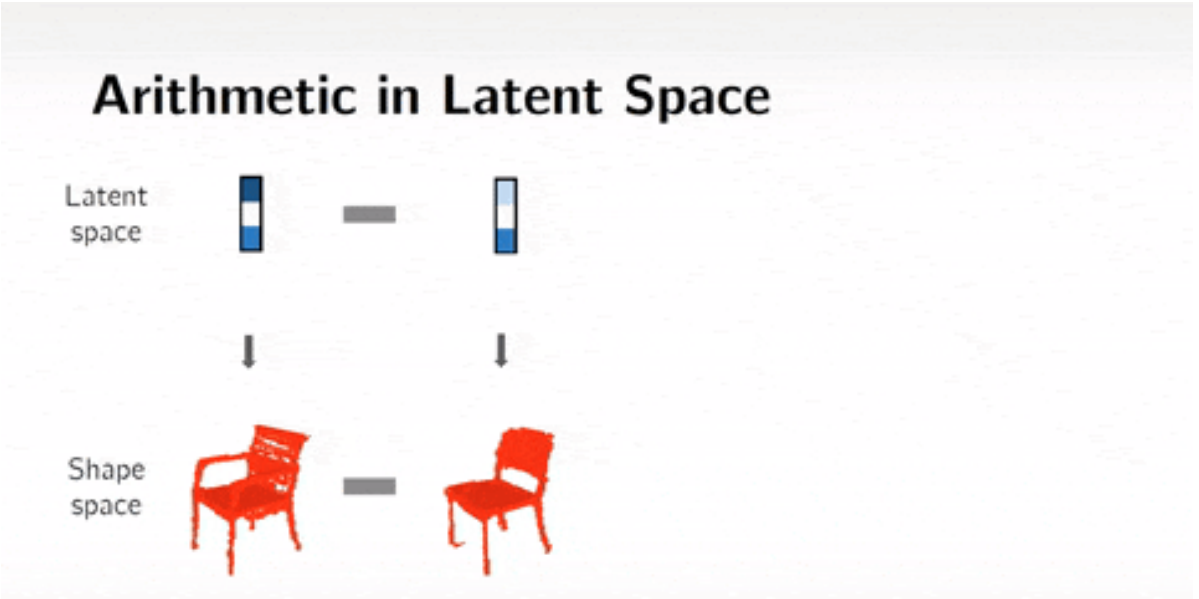




這邊是用針對7(常態)的decoder
他會想把0(異常)解壓縮為7，但還是有差異
運用異常值的偵測

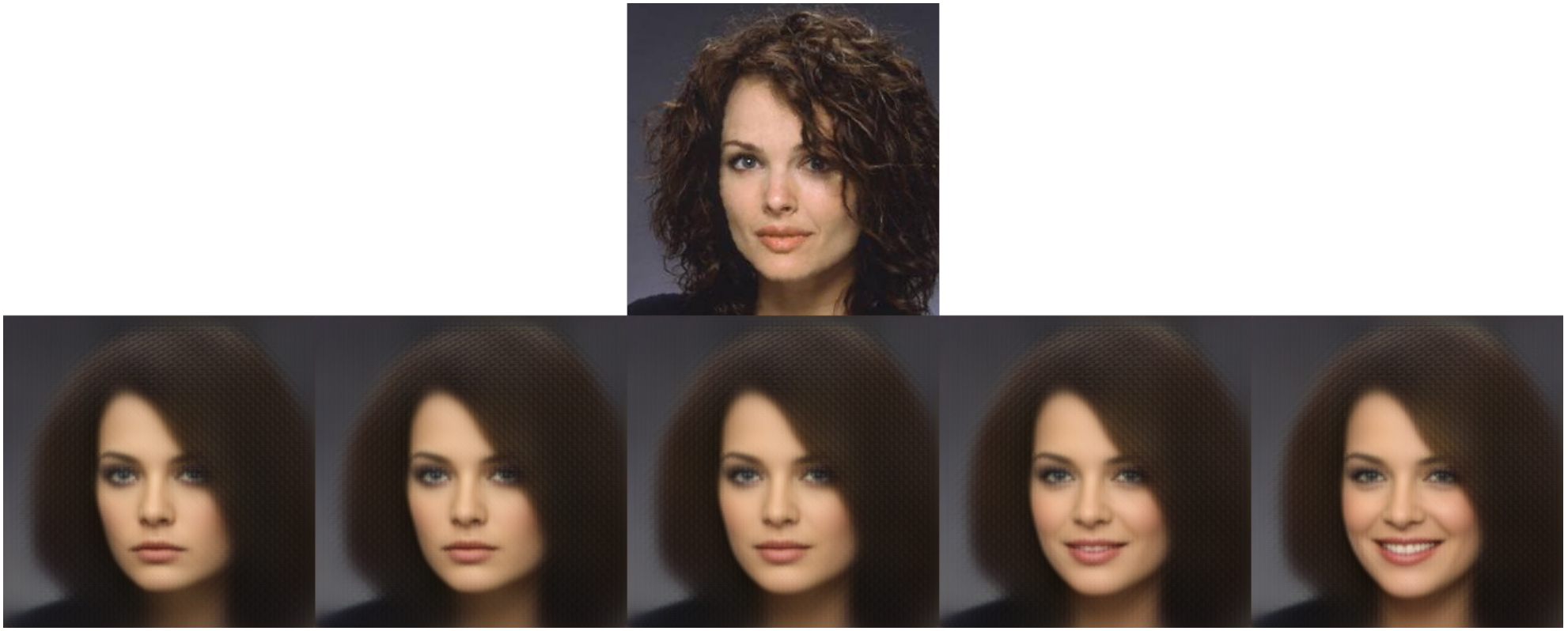
# Latent Space Sampling (1/2)

若 Latent Space 有好的結構則可以更好地 editing

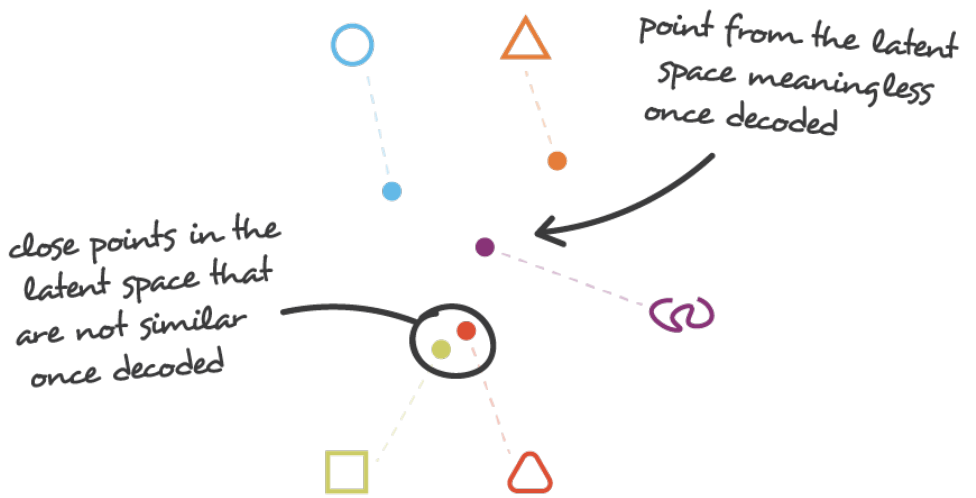
# Latent Space Sampling (2/2)

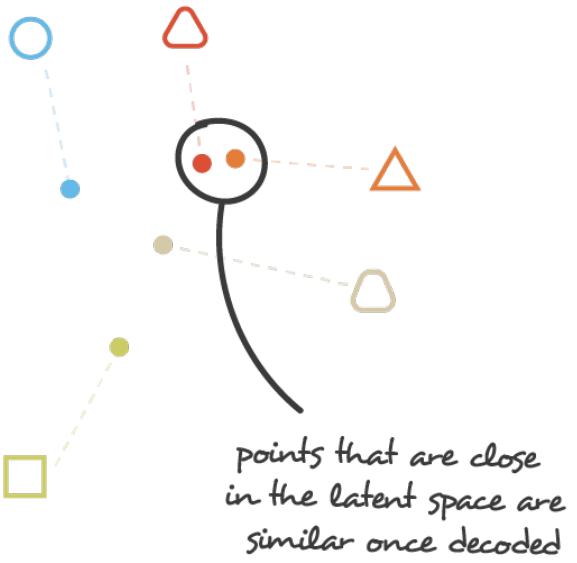
+/- smile vector



# Variational AutoEncoder (1/3)

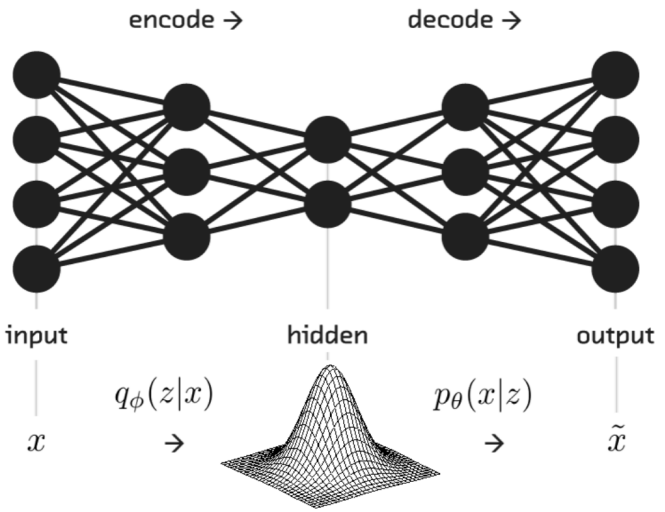
## 強迫 latent space 有連續變化 (cf. Kohonen SOM)



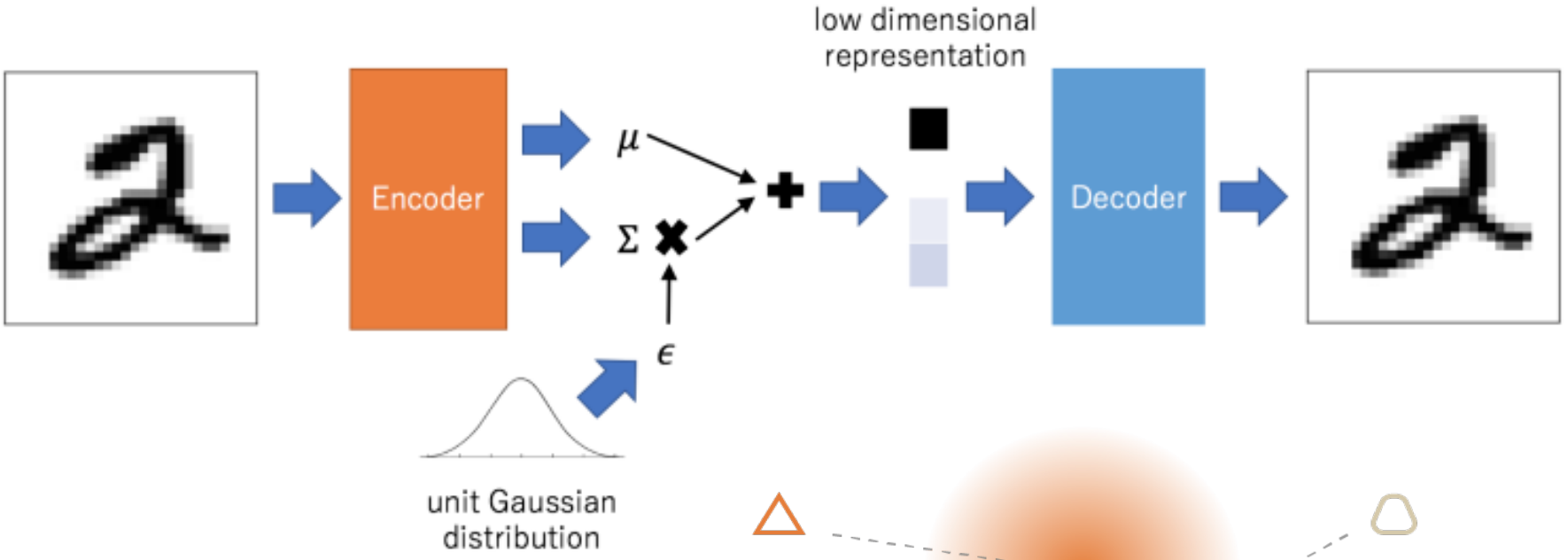

**irregular latent space**

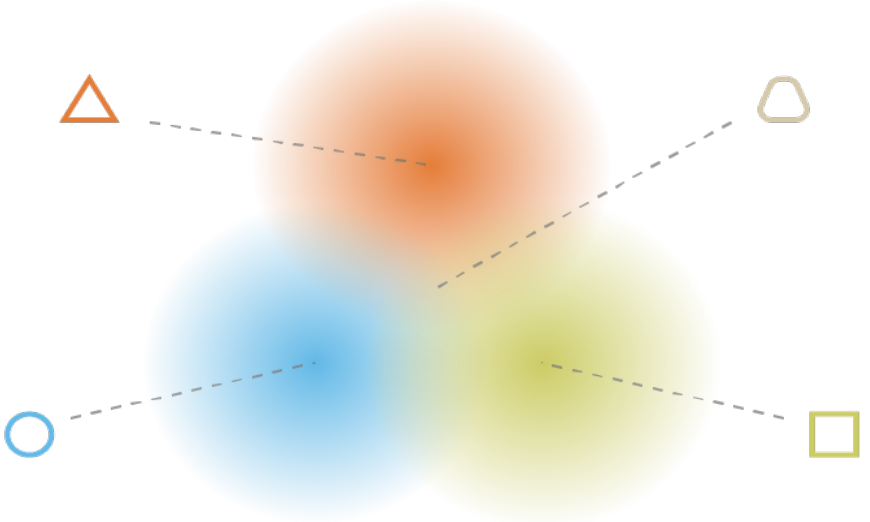
**regular latent space**



# Variational AutoEncoder (2/3)

具體而言讓整個 Gaussian 映射到同一個 output

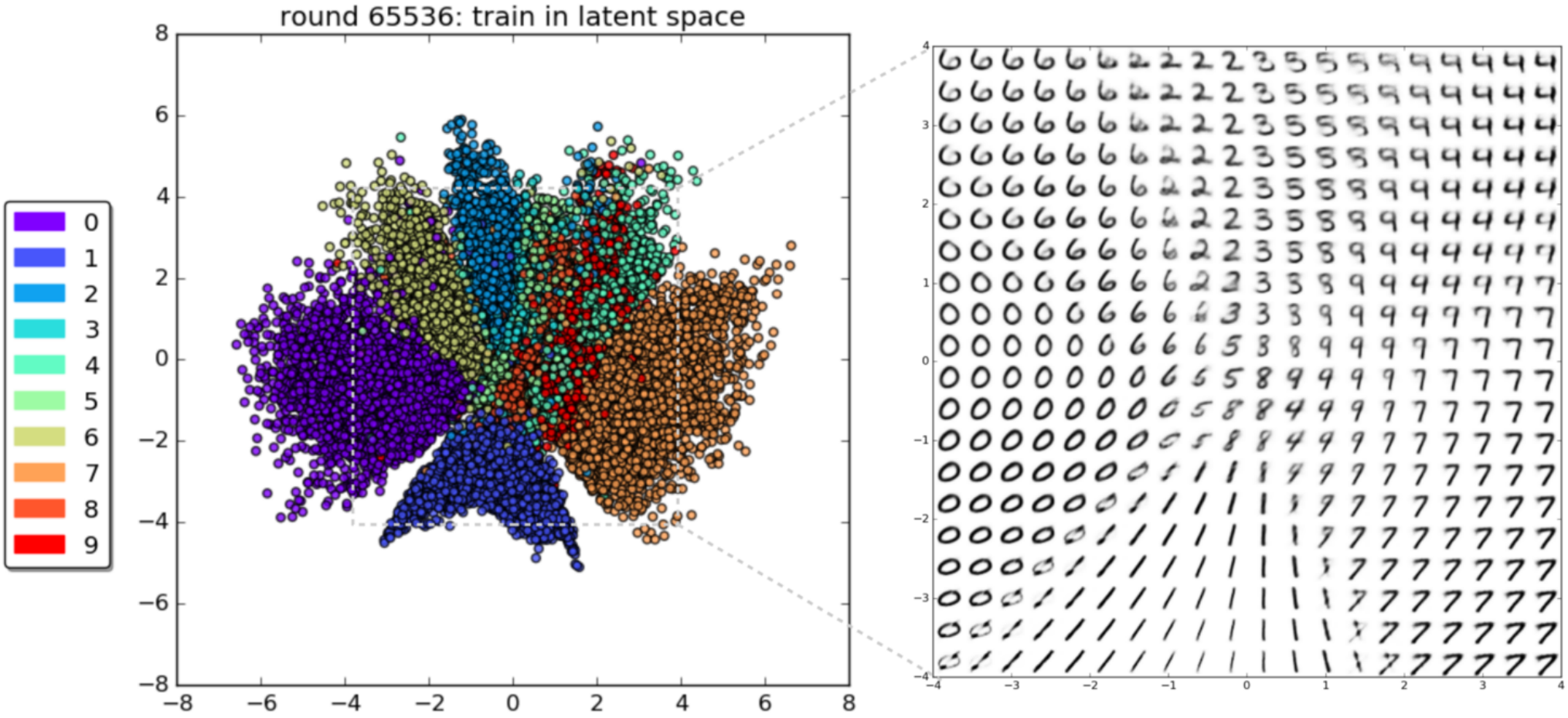


what we want to obtain with regularisation

# Variational AutoEncoder (3/3)

MNIST 經過 VAE 學習 2 維 embedding 後



# Generative Adversarial Network (1/2)

可看成更進階的 agent-based modeling

# Generative Adversarial Network (2/2)

## 關鍵在設計好 loss functionS 與 D/G 的非同步學習

maximize

$$\frac{1}{m}\sum_{1}^{m}[logD(X)+log(1-D(G(z))]$$

synthetic data

$X'$

pretrained

MLP D

discriminator

P(data is from X)

MLP G

generator

minimize

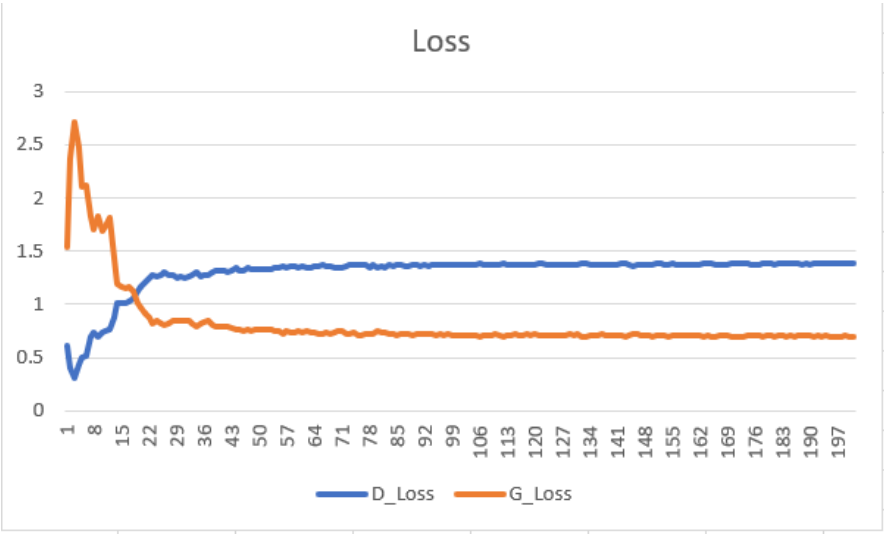
$$\frac{1}{m}\sum_{1}^{m}log(1-D(G(z))$$

z

random noise

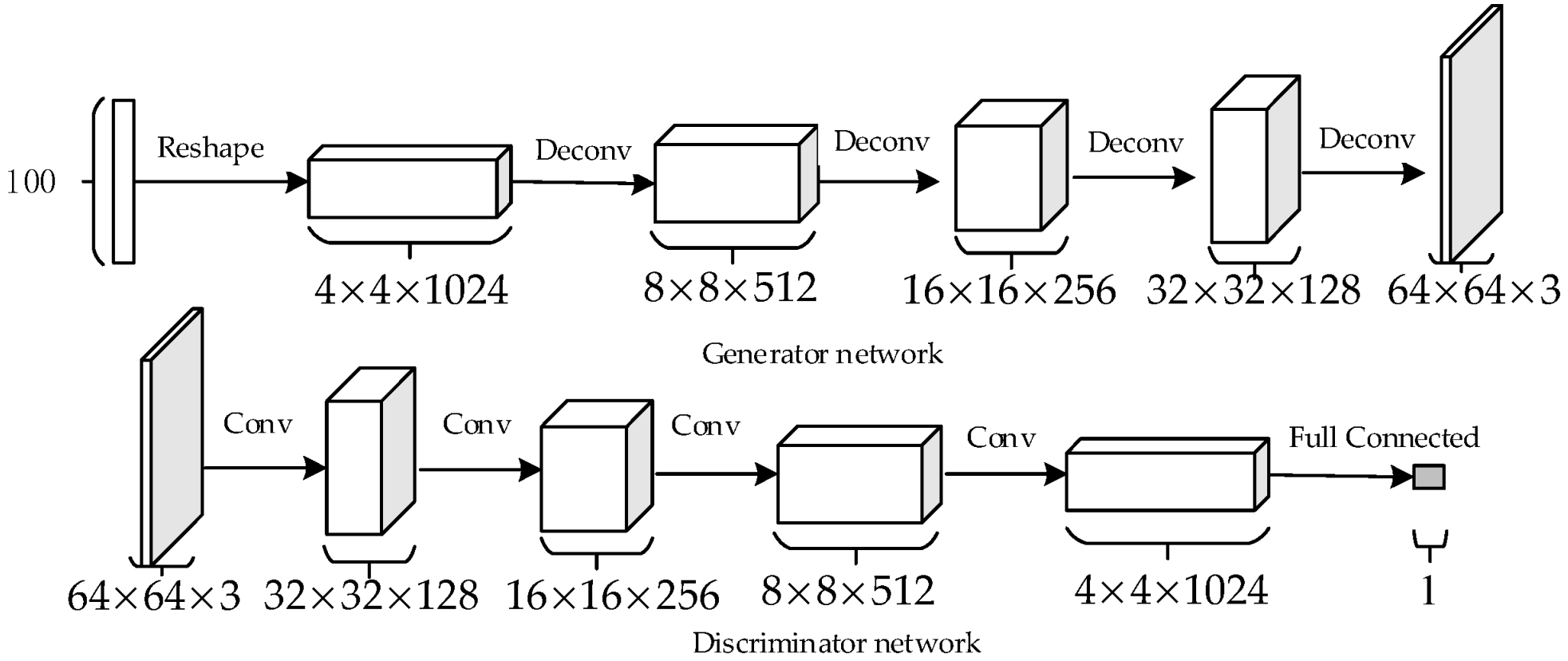
X

training data



# Deep Convolutional GAN (DCGAN)

## Convolution & Deconvolution

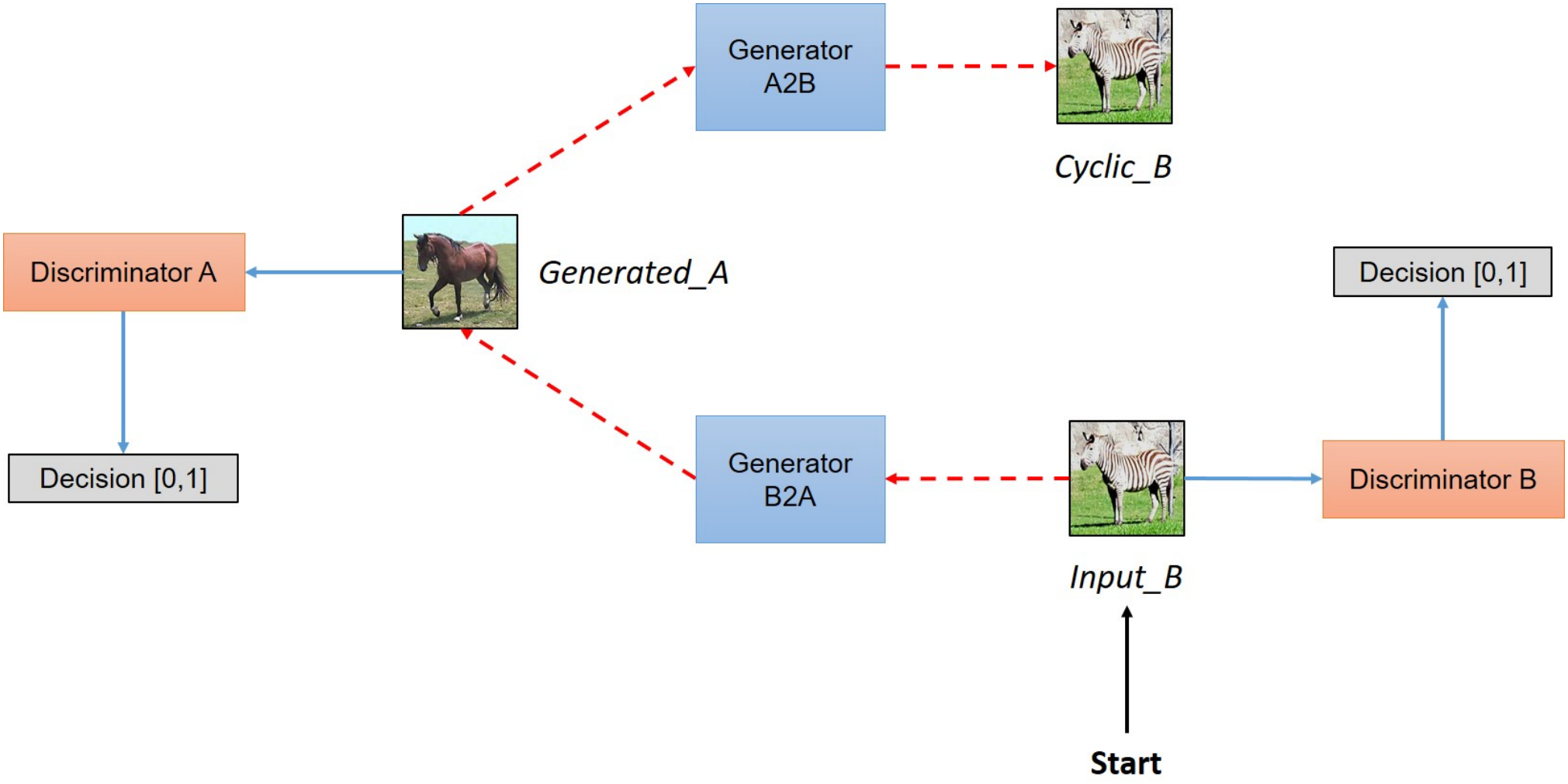


# DCGAN（續）

Generator 最後可以以假亂真

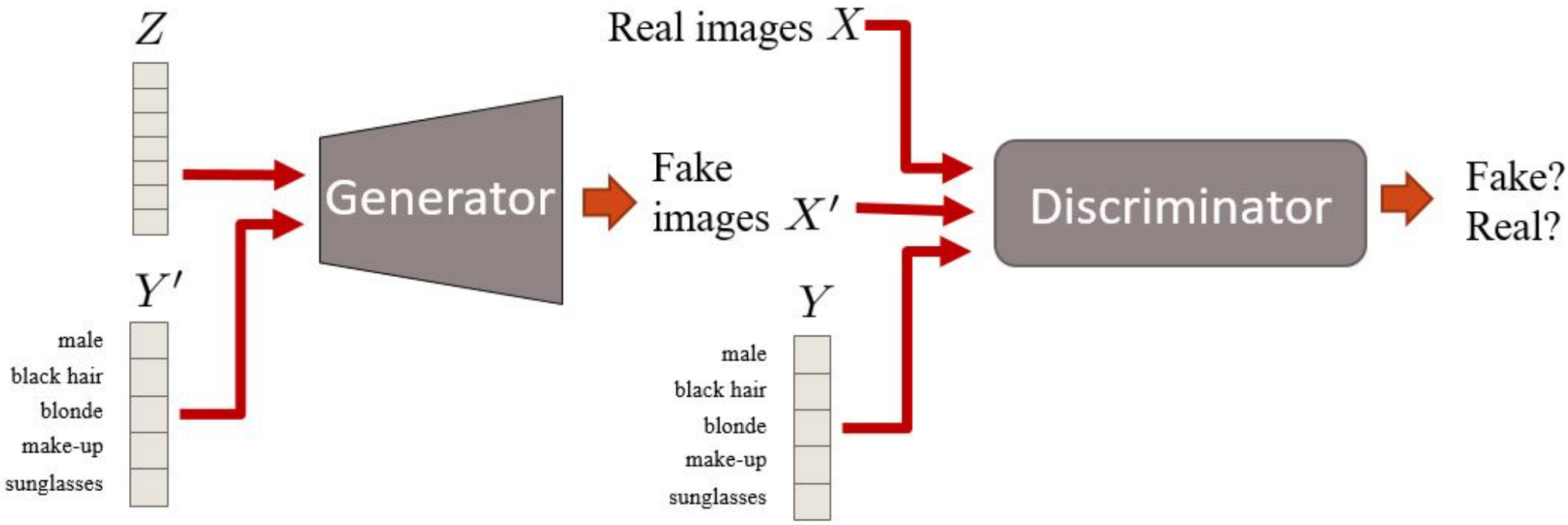
# CycleGAN

## 調包遊戲



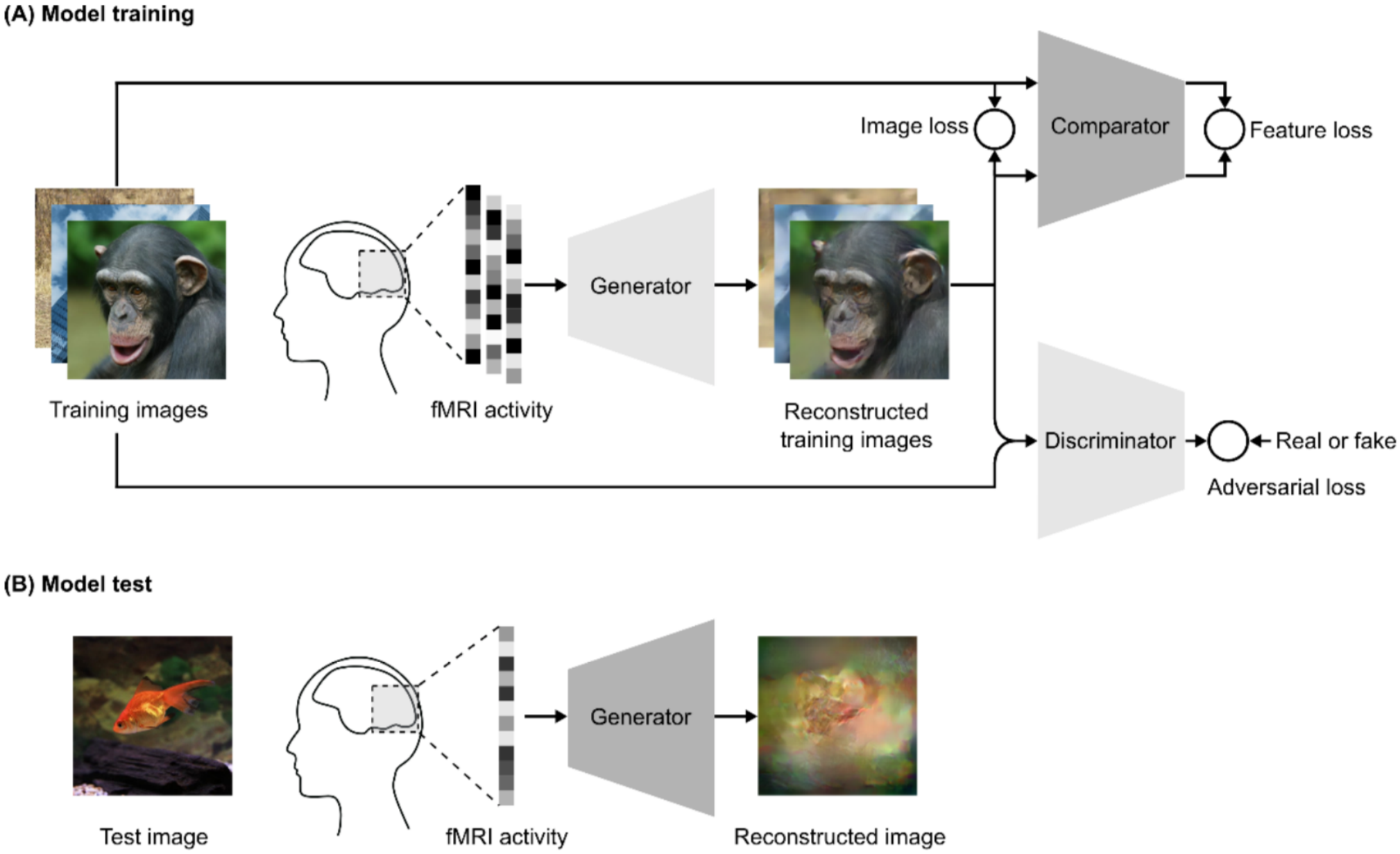
# Conditional GAN

更能根據需要產生樣本
(cf. latent space editing)



# Applications of GAN

例如用來從腦訊號重建外接刺激



# Neural Style Transfer (1/4)

如果風格與內容能夠分開

1 **Upload photo**

The first picture defines the scene you would like to have painted.

2 **Choose style**

Choose among predefined styles or upload your own style image.

3 **Submit**

Our servers paint the image for you. You get an email when it's done.

# Neural Style Transfer (2/4)

Styles 藏在細節裡 ( 即空間中的高頻資訊 )

新細明體 焉能辨我是宋明？

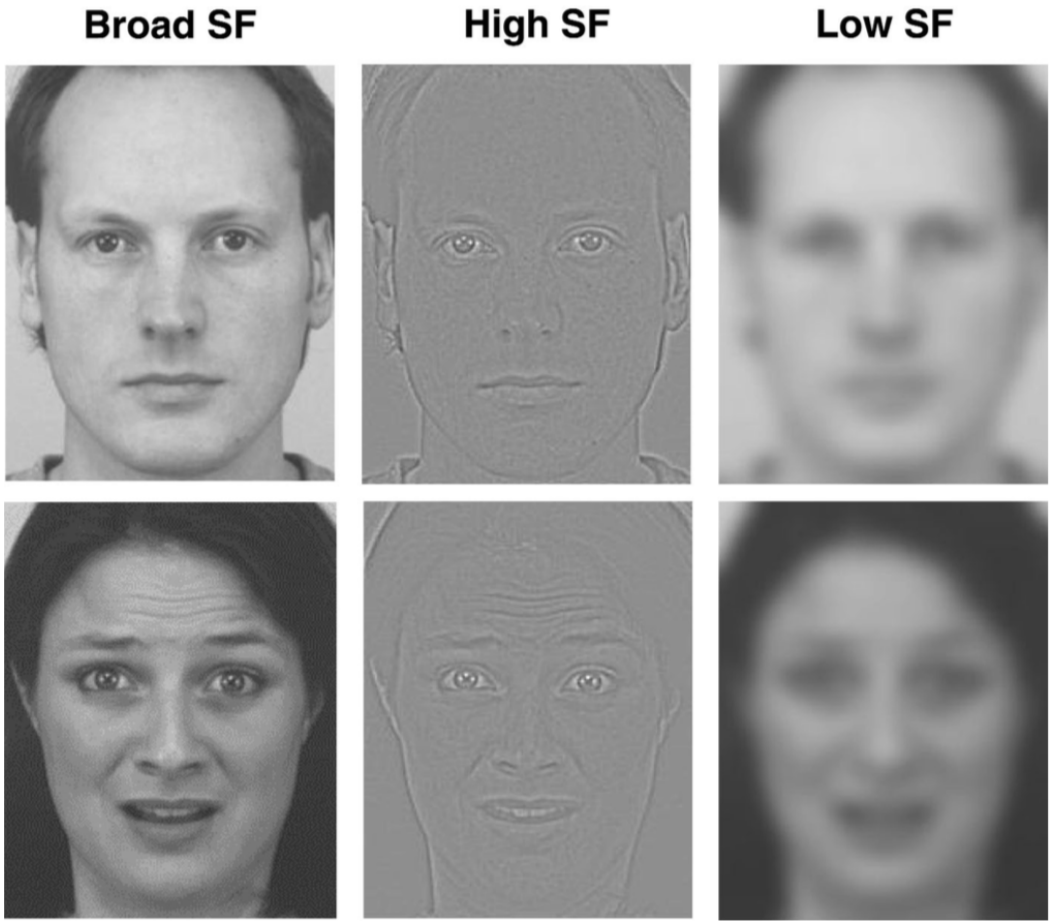
華康粗明體 焉能辨我是宋明？

華康儷中宋 焉能辨我是宋明？

文鼎中明 焉能辨我是宋明？

文鼎粗標準宋體 焉能辨我是宋明？

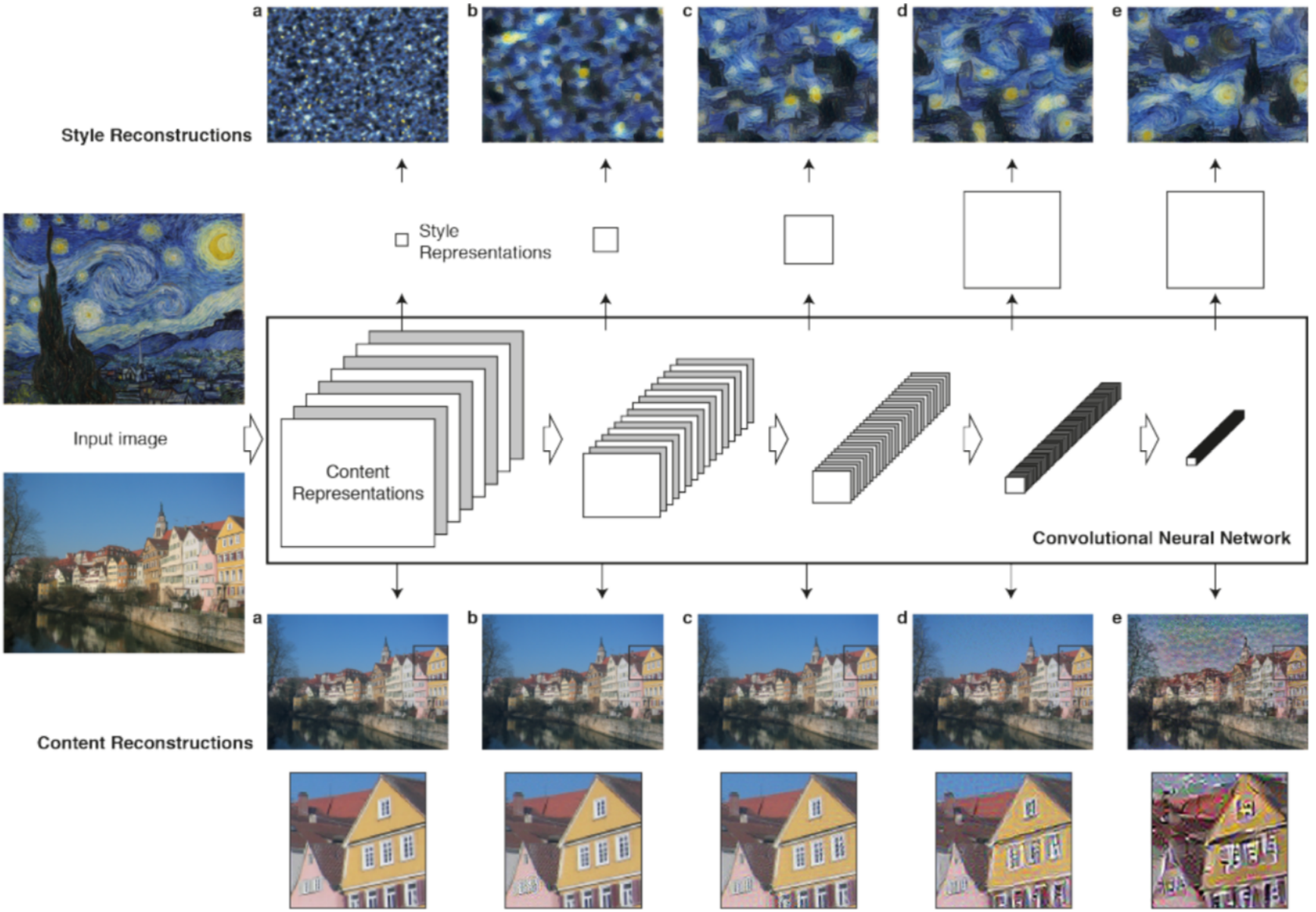
蒙納繁中宋 焉能辨我是宋明？




點開連結然後放大

# Neural Style Transfer (3/4)

## CNN 的下游層比較有內容 (gist)



# Neural Style Transfer (4/4)

L=D(content(G)-content(O))+D(style(G),style(T))

$$\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{content}(\vec{p},\vec{x}) + \beta\mathcal{L}_{style}(\vec{a},\vec{x})$$

$\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the weighting factors for content and style reconstruction

$$\mathcal{L}_{content}(\vec{p},\vec{x},l) = \frac{1}{2}\sum_{i,j}(F_{ij}^{l}\text{-}P_{ij}^{l})^2$$

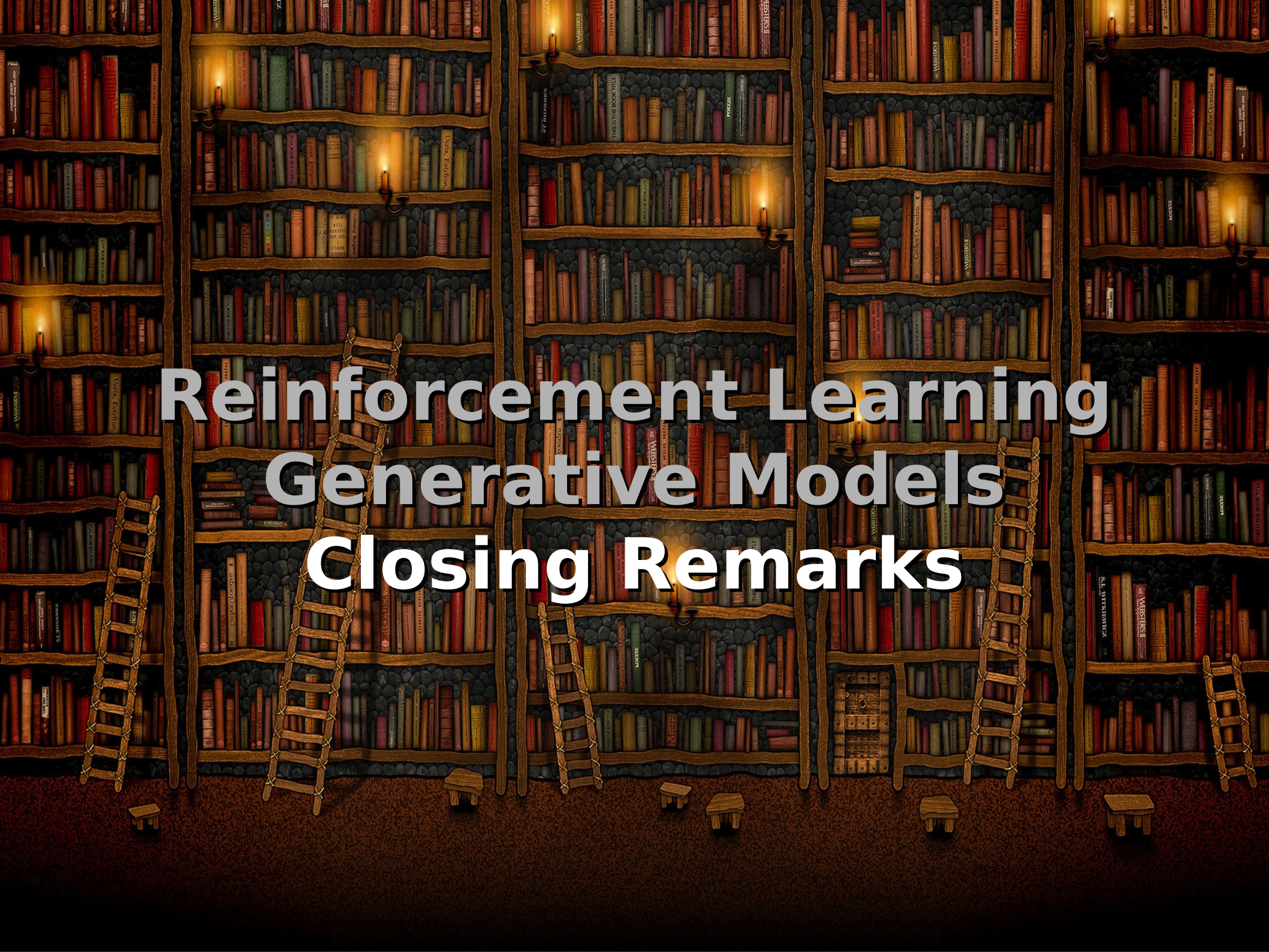
$\vec{\boldsymbol{p}}$ The original image
$\vec{\boldsymbol{x}}$ The generated image
$\boldsymbol{l}$ Layer
$\boldsymbol{F_{ij}^{l}}$ Activation of the $i^{th}$ filter at position j in the feature representation of $\vec{x}$ in $l$
$\boldsymbol{P_{ij}^{l}}$ Activation of the $i^{th}$ filter at position j in the feature representation of $\vec{p}$ in $l$

$$\mathcal{L}_{style}(\vec{a},\vec{x}) = \sum_{l=0}^{L} w_l E_l$$

$$E_l = \frac{1}{4N_l^2M_l^2}\sum_{i,j}(G_{ij}^{l}\text{-}A_{ij}^{l})^2$$

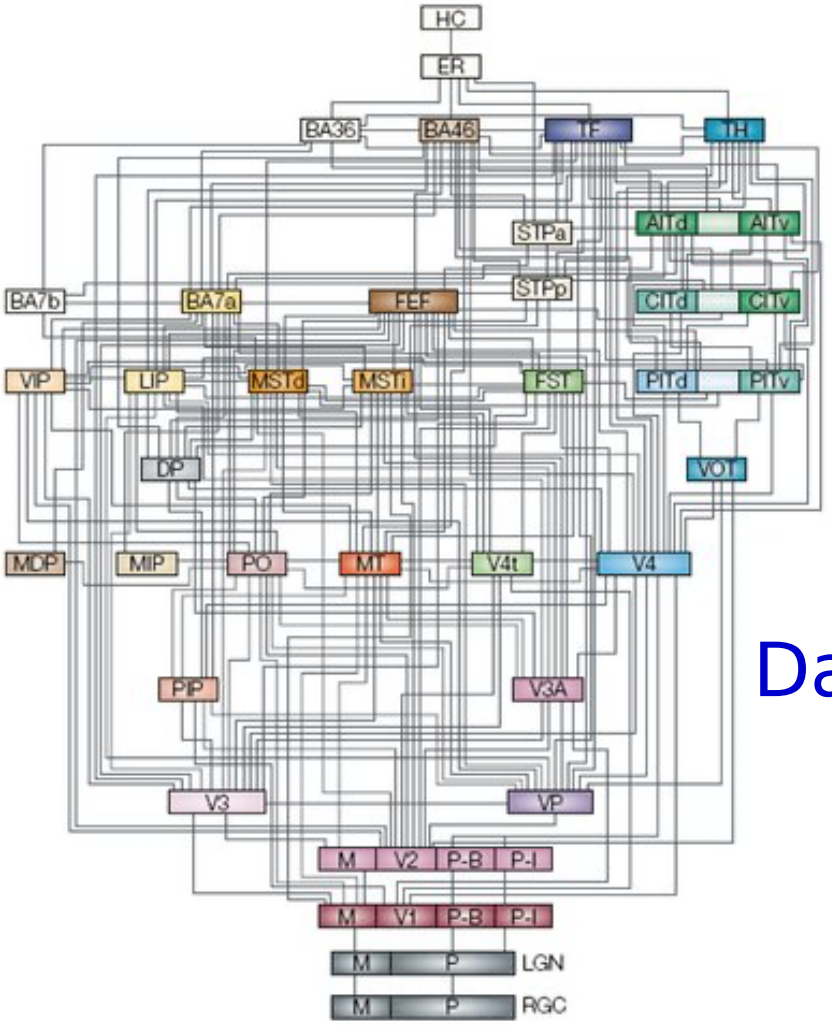
$\vec{\boldsymbol{a}}$ The original image
$\vec{\boldsymbol{x}}$ The generated image
$\boldsymbol{L}$ Total number of all layers
$\boldsymbol{w_l}$ Weighting factors of the contribution of each layer to the total loss
$\boldsymbol{E_l}$ loss in layer l

$\boldsymbol{N_l}$ number of distinct feature maps
$\boldsymbol{M_l}$ the height times the width of the feature map
$\boldsymbol{G_{ij}^{l}}$ pairwise feature maps i and j in the style representation of $\vec{x}$ in $l$
$\boldsymbol{A_{ij}^{l}}$ pairwise feature maps i and j in the style representation of $\vec{a}$ in $l$

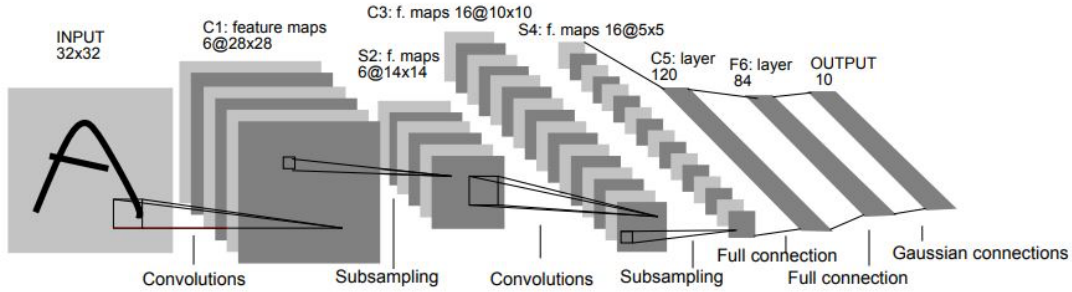

Reinforcement Learning
Generative Models
Closing Remarks

# 生物神經與類神經的三種關係

## 預測性模型也可被解釋或幫助解釋




Architectural Metaphors

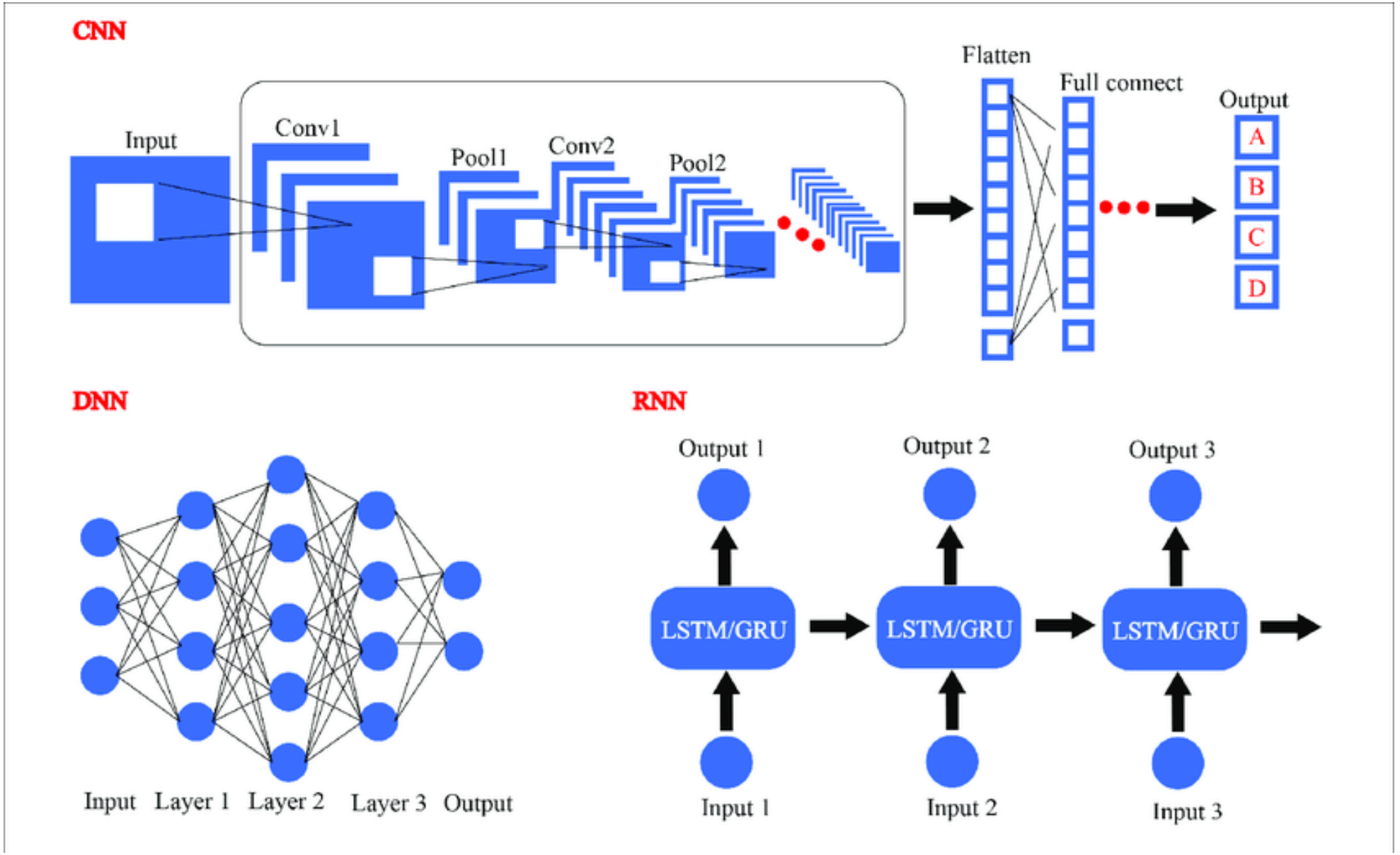



Data Modeling

Functional Metaphors

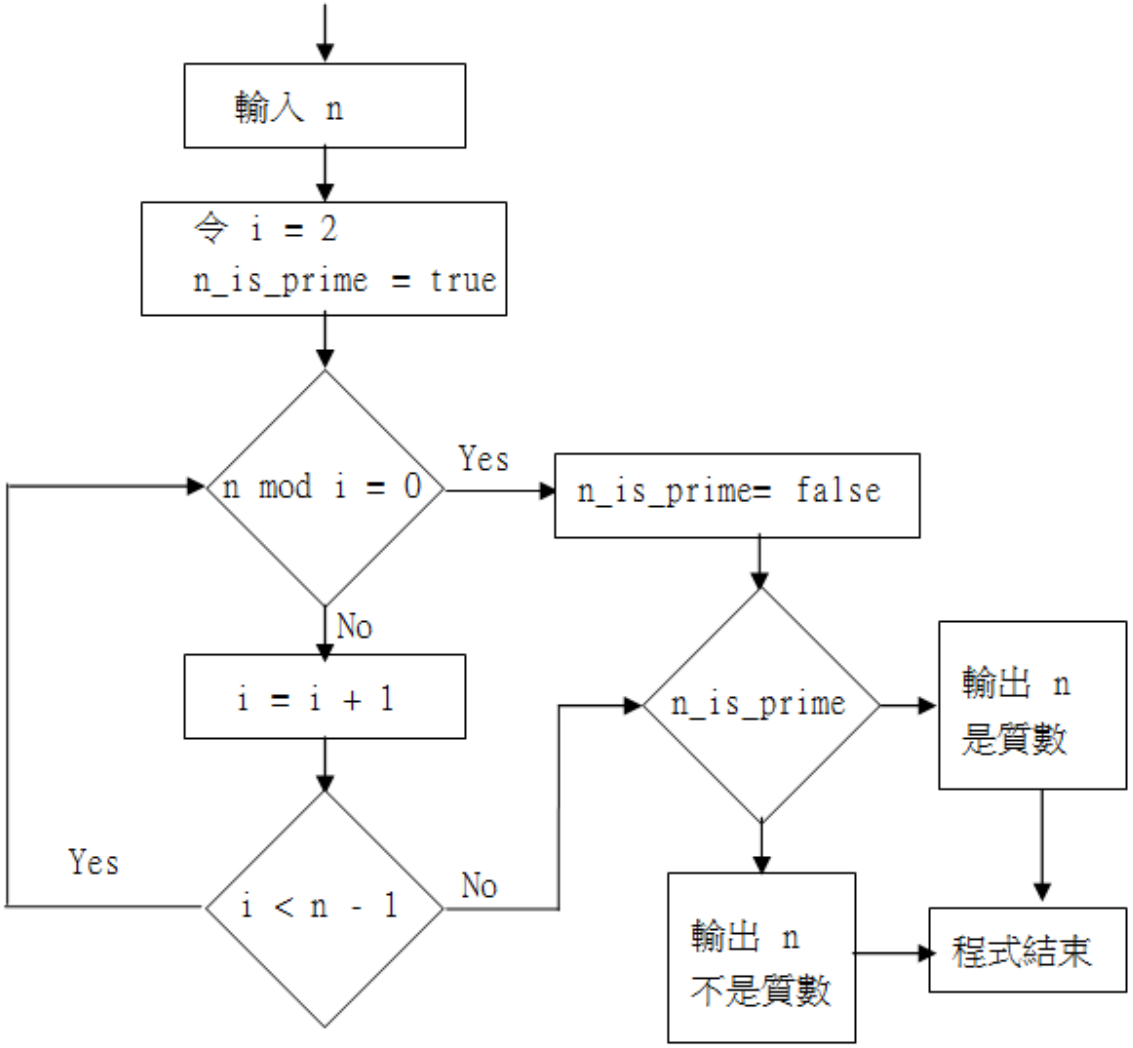
# Deep Neural Networks 的分類

CNN 通常處理影像資料；RNN 通常處理語言資料

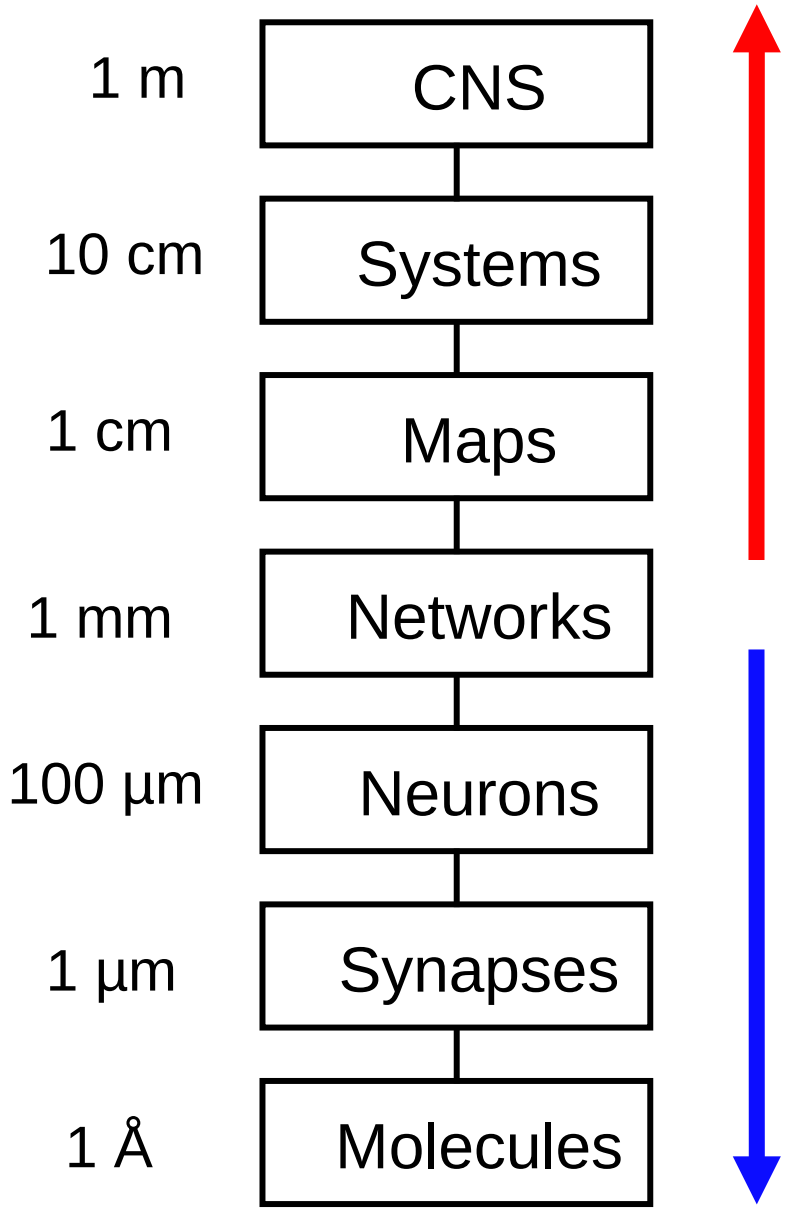


# Limits of Deep Learning

整體架構還是歸納學習與內插法預測



# Research Scales




**Computational Cognitive Science**

Bayesian perception
Diffusion models

**Computational Cognitive Neuroscience**

Connectivity analysis
Neural networks

**Computational Neuroscience**

Neural coding
Neuronal models

# Game Over