Python深度學習實戰-邁向A.I.的第一步 (Part2)

丘祐瑋
David Chiu

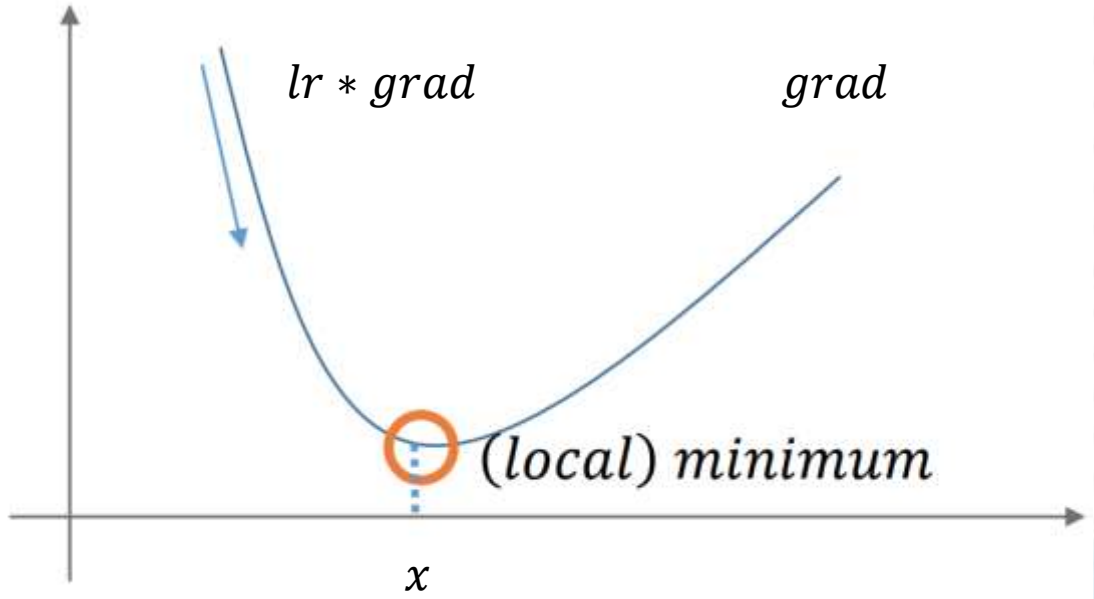
如何選擇優化器

隨機梯度下降 (Stochastic Gradient Descent)

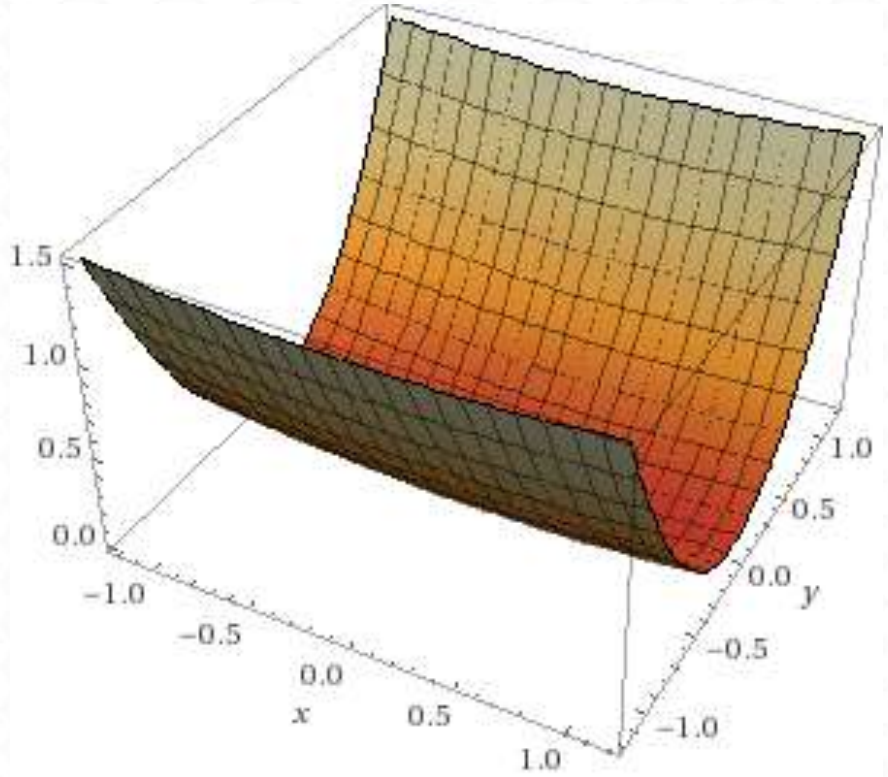
- Gradient Descent Method
  - 找出代價函數的最小值

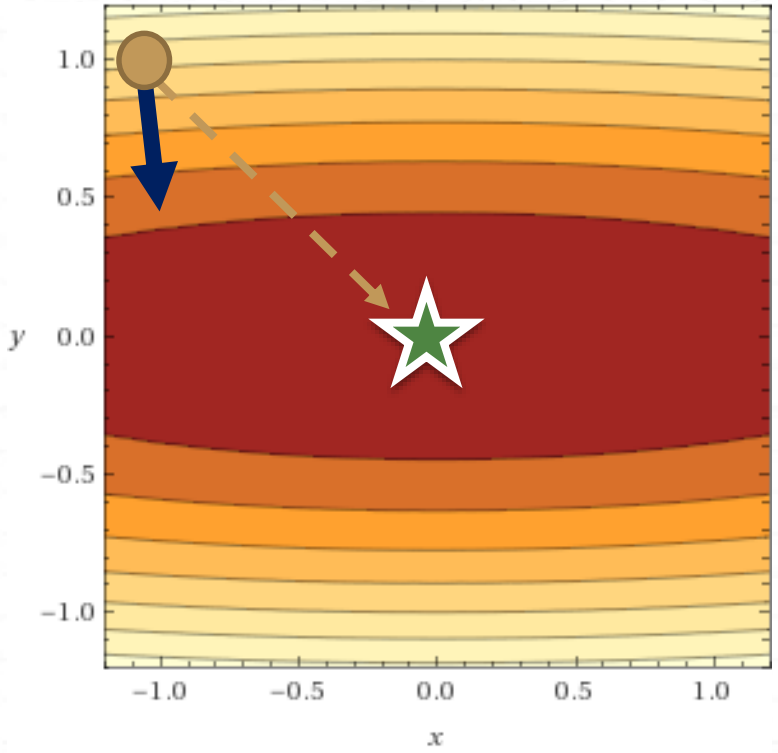
權重更新函數

$$x_i = x_i - lr * grad$$



探索最低點

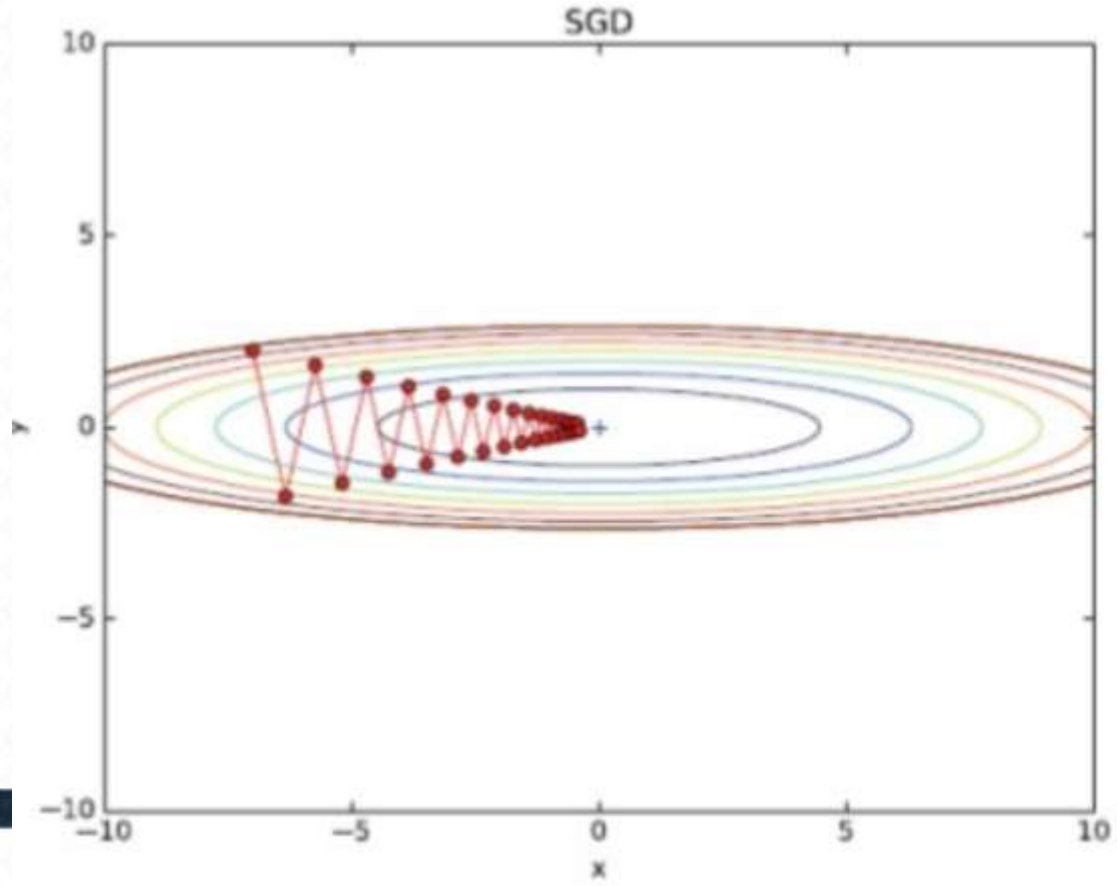




https://www.slideshare.net/juhuipark37/ss-75079799

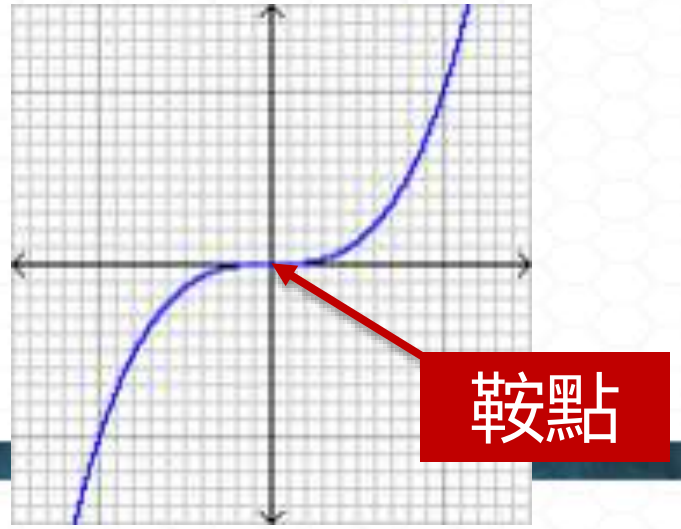
隨機梯度下降的缺點

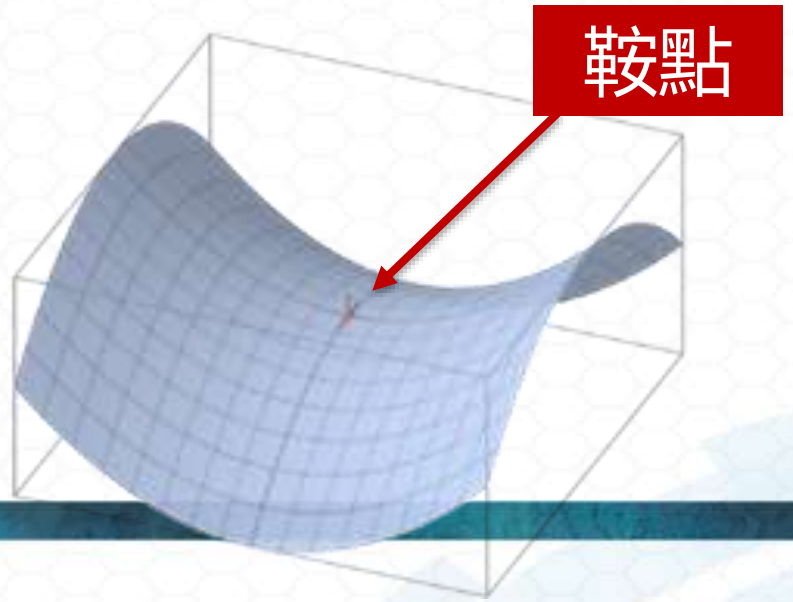
當梯度不指往最小值時, SGD 會以鋸齒狀方式邁向最低點, 相當沒效率



隨機梯度下降的缺點

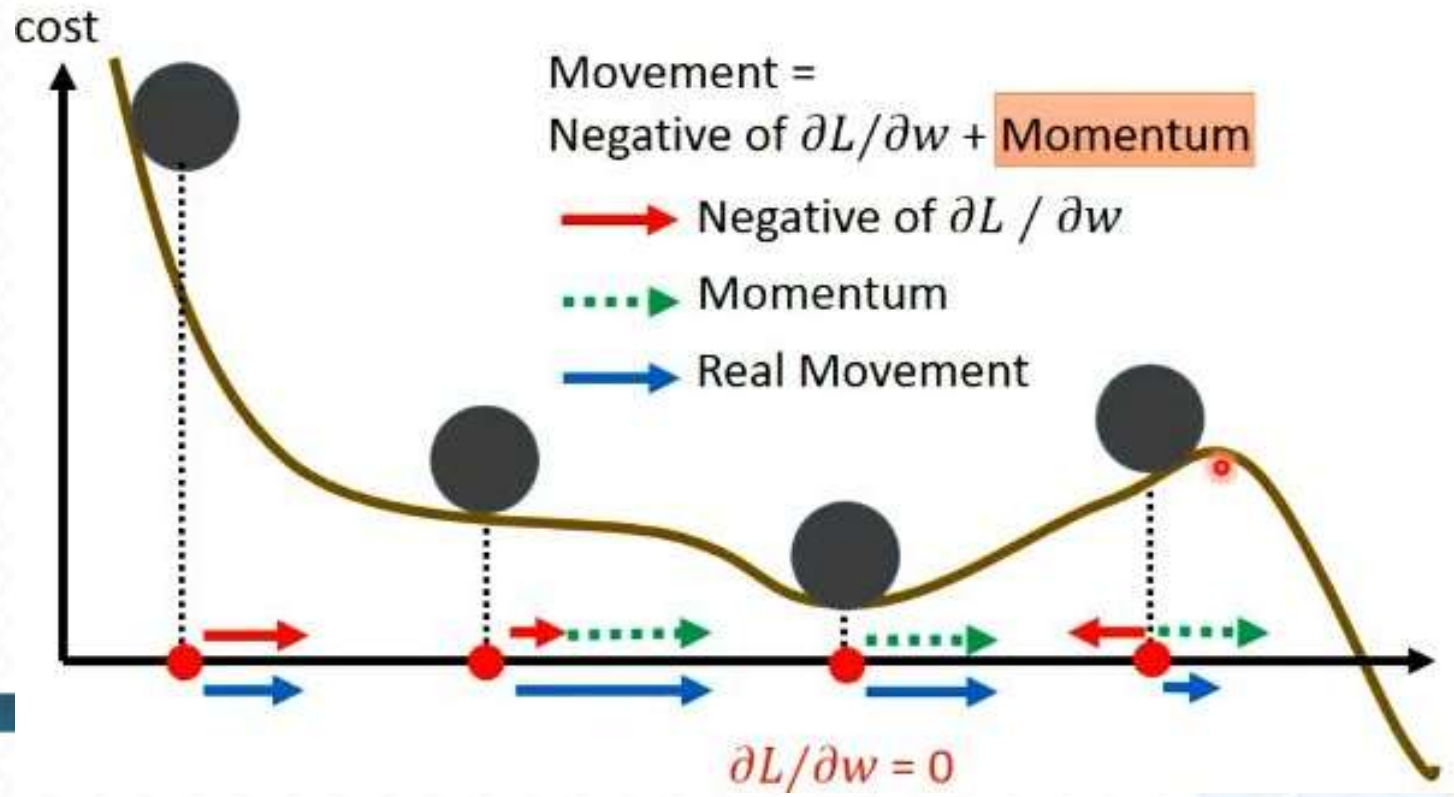
- 1. 很難選擇出合適的學習率,太大太小都不合適
- 2. 相同的學習率不應該適用于所有的參數更新 (e.g. 很少出現的特徵，應使用較大的更新率)
- 3. 隨機梯度下降容易被鞍點困住





動量 (Momentum)

- 模擬物理學中的動量(Momentum)，使用積累的動量替代梯度



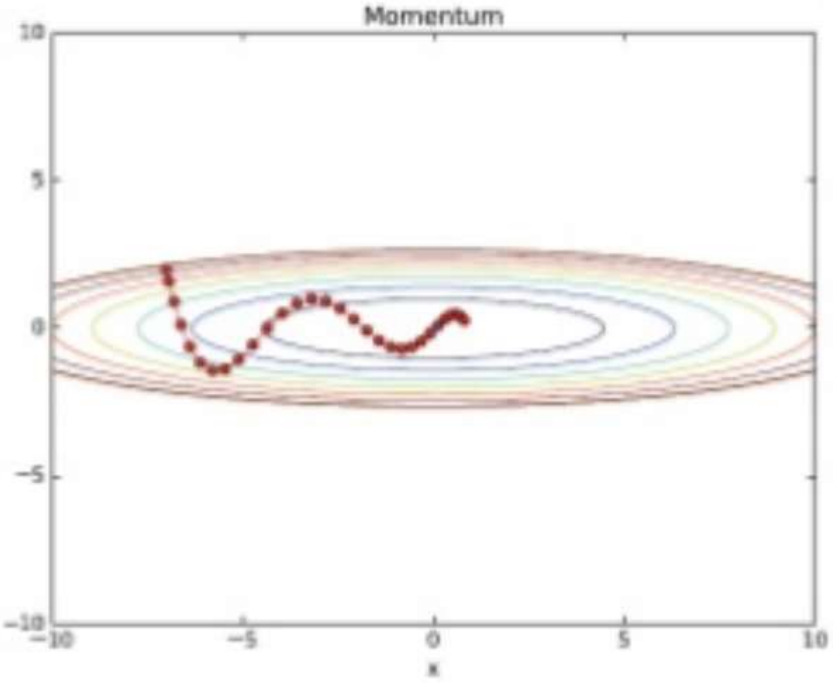
動量 (Momentum)

權重更新函數

$$\mathrm{v} = mv - lr * grad$$

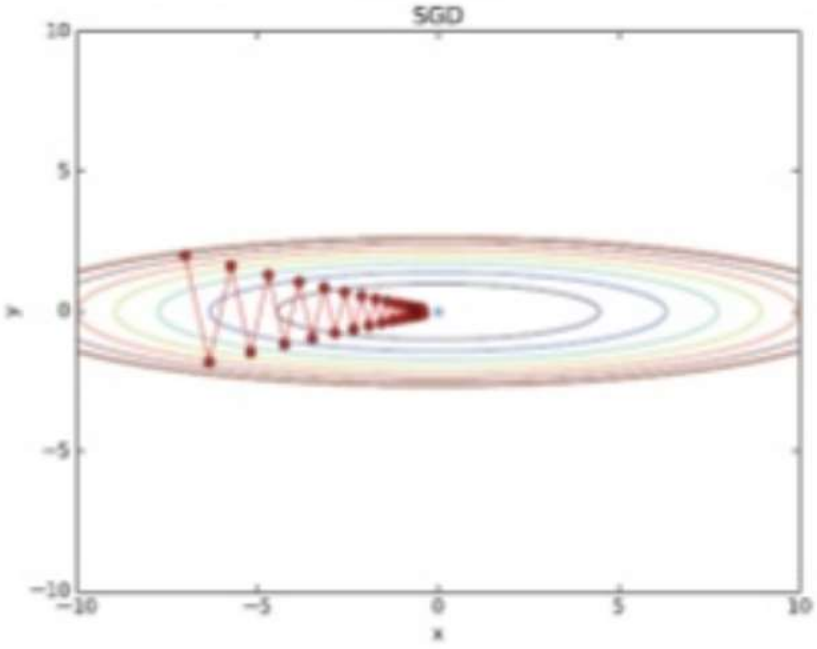
$$x = x + v$$

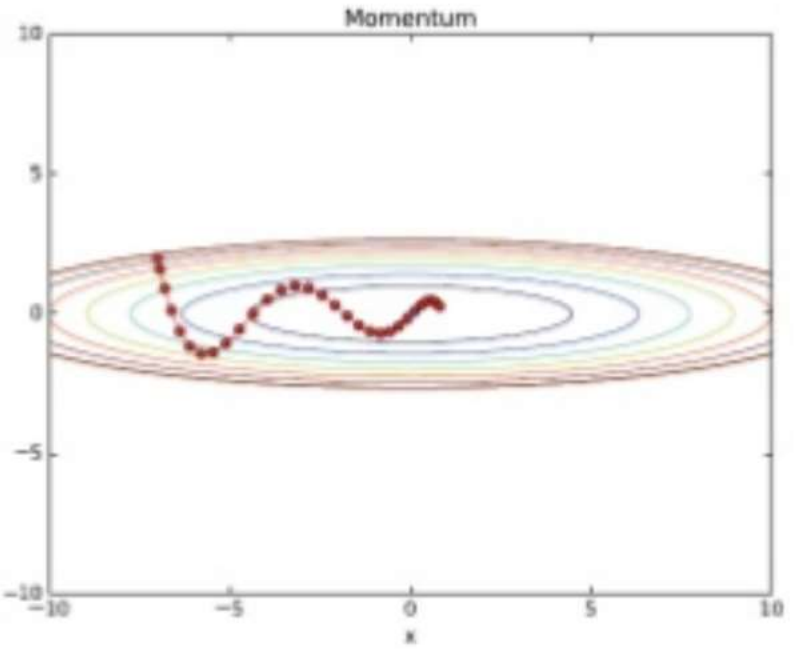
m (momentum)通常為 0.9



動量 (Momentum)

- 動量可視為SGD 的加速版本





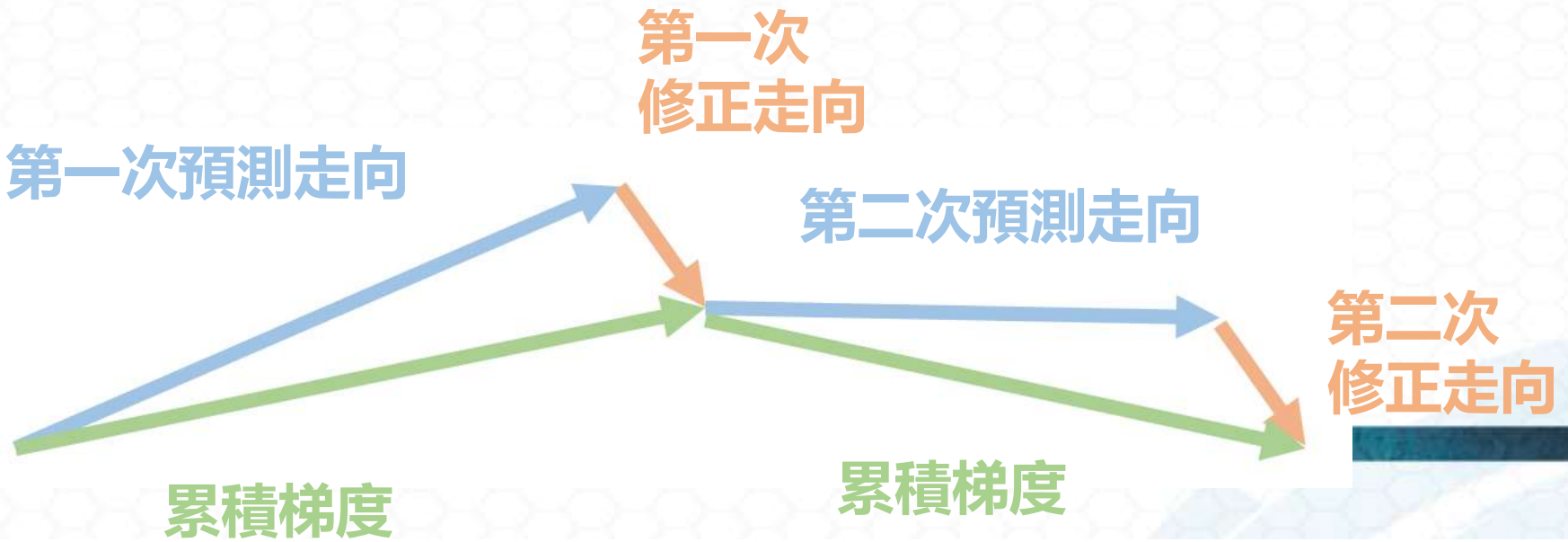
Nesterov accelerated gradient (NAG)

- 梯度更新時，不是盲目追尋最低點, 而是產生一個預測指標, 預測可能的終點, 並且適時減速, 避免超過最低點

權重更新函數

$$v = mv - lr * grad(x + m * v)$$

$$x = x + v$$



自我調整學習率

- 代價函數的梯度大小在不同的層與不同參數可能差異很大,並無法適用同一個學習率, 而自我調整學習率的演算法能為每個參數設置不同的學習率，並能在學習過程中自我調整地改變學習率

- 自我調整學習率的演算法:
  - AdaGrad
  - AdaDelta
  - RMSProp
  - Adam

AdaGrad

- 隨著學習過程縮小學習率 (learning rate), 又稱為學習率遞減 (Learning Rate Decay)
- 對於經常變動的參數，學習率會逐漸變小
- 無限學習 量會降到 0

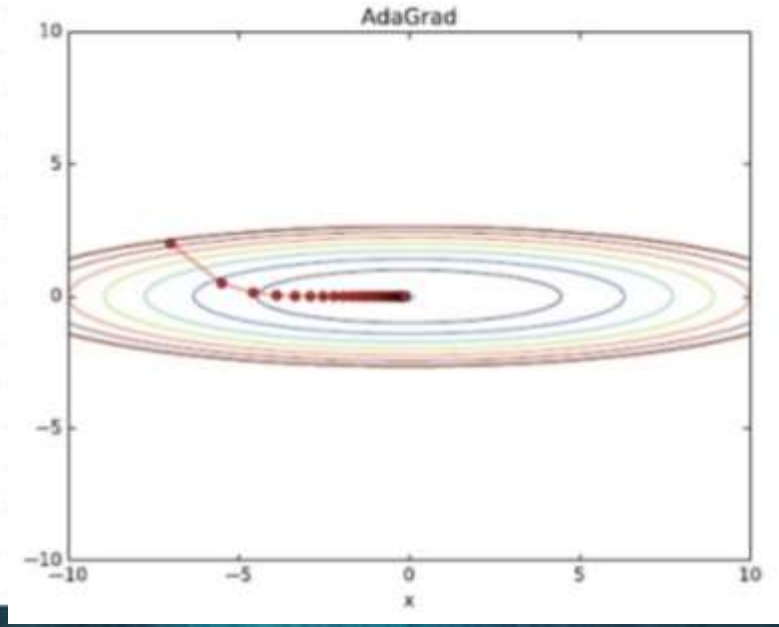
權重更新函數

紀錄過程中所有梯度值的平方和

$$h = h + grad(x) \odot grad(x)$$

$$x = x - lr\frac{1}{\sqrt{h}}grad$$

調整學習率



AdaDelta

- Adagrad
  - 紀錄過程中的所有梯度值的平方和
  - 無限學習下去，更新量會變為 0

- Adadelta
  - 只記錄固定區間（Window）**權重更新函數** 和

$$E|grad^2|_t = \gamma E|grad^2|_{t-1} + (1-\gamma)grad_t{}^2$$

$$x = x - lr\frac{1}{\sqrt{E|grad^2|_t}}grad$$

γ 類似Momentum

AdaDelta 改良版

- AdaDelta 甚至不需要設學習率

權重更新函數

$$x = x - lr\frac{1}{\sqrt{E|grad^2|_t}}grad$$

$$x = x - \frac{RMS|x|_{t-1}}{RMS|grad|_t}grad$$

RMSprop

- 與AdaDelta 幾乎同時被提出，其γ為一常數，可以當作 Adadelta的一個特例

權重更新函數

$$E|grad^2|_t = 0.9E|grad^2|_{t-1} + 0.1grad_t{}^2$$

$$x = x - lr\frac{1}{\sqrt{E|grad^2|_t}}grad$$

γ 類似Momentum, Hinton 建議設為0.9

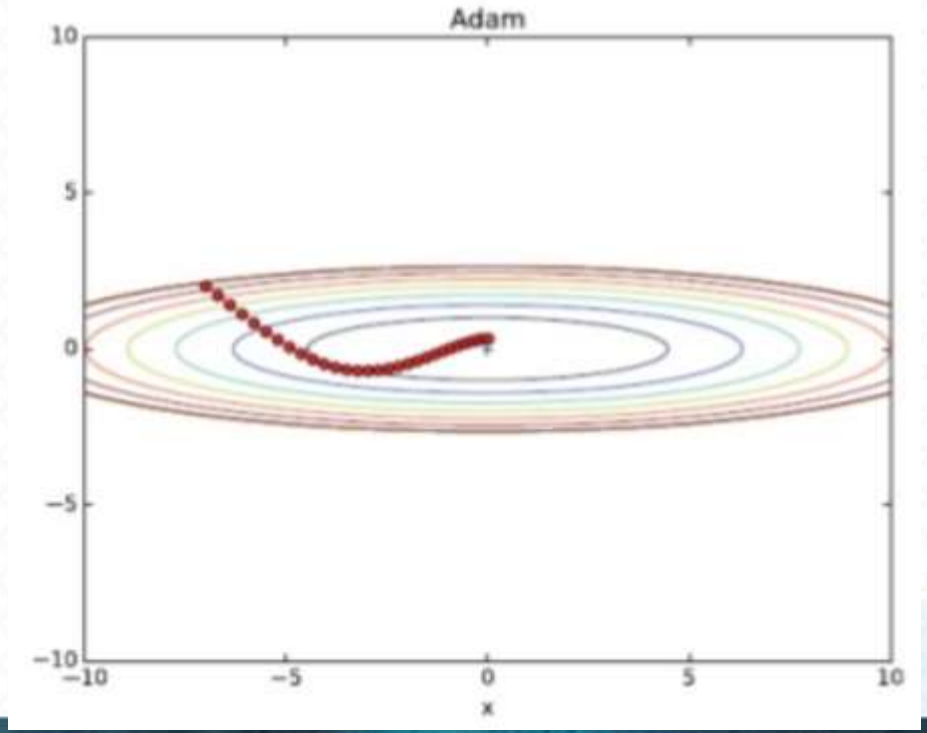
Adam

- Momentum 與 AdaGrad 的綜合體
  - 透過Momentum 加速收斂
  - 透過學習率衰減自動調整學習率

權重更新函數

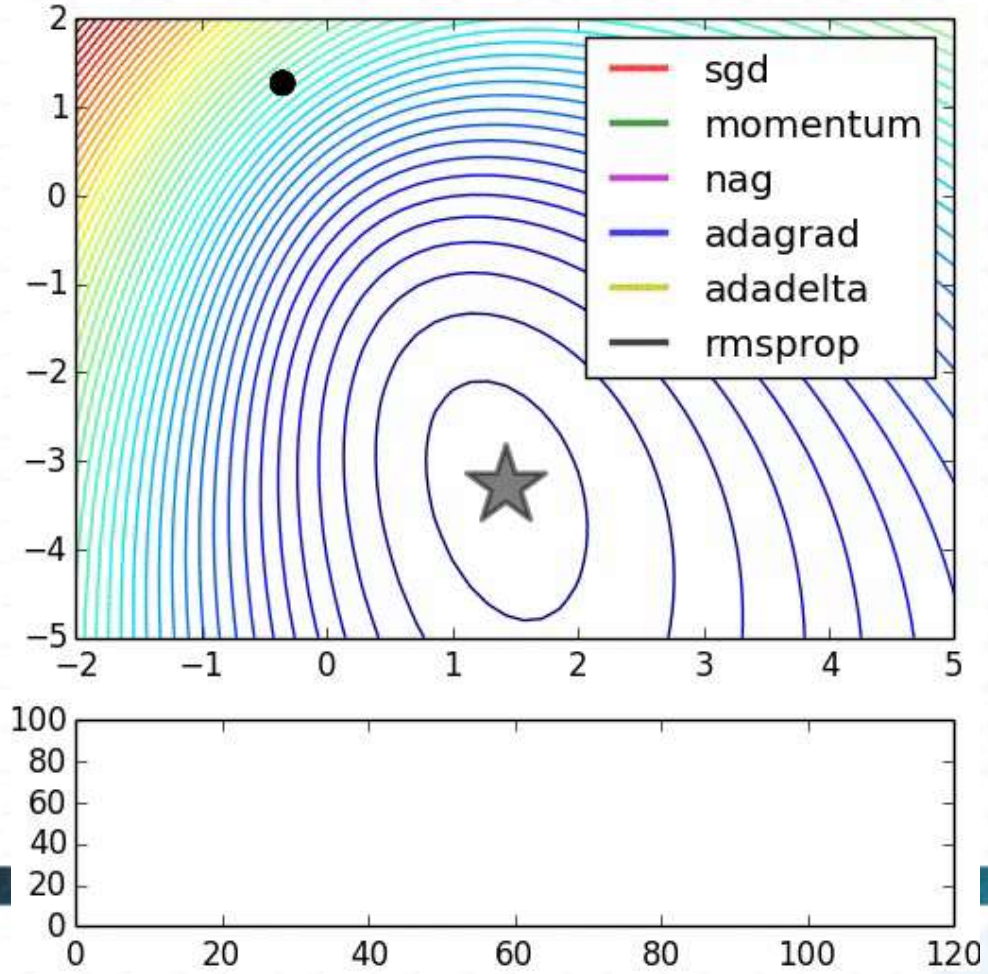
$$x = x - \frac{lr}{\sqrt{\hat{v}_t}}\hat{m}_t$$

$\hat{m}_t$: 指數衰減平均值
$\hat{v}_t$: 平方梯度的指數衰減平均值



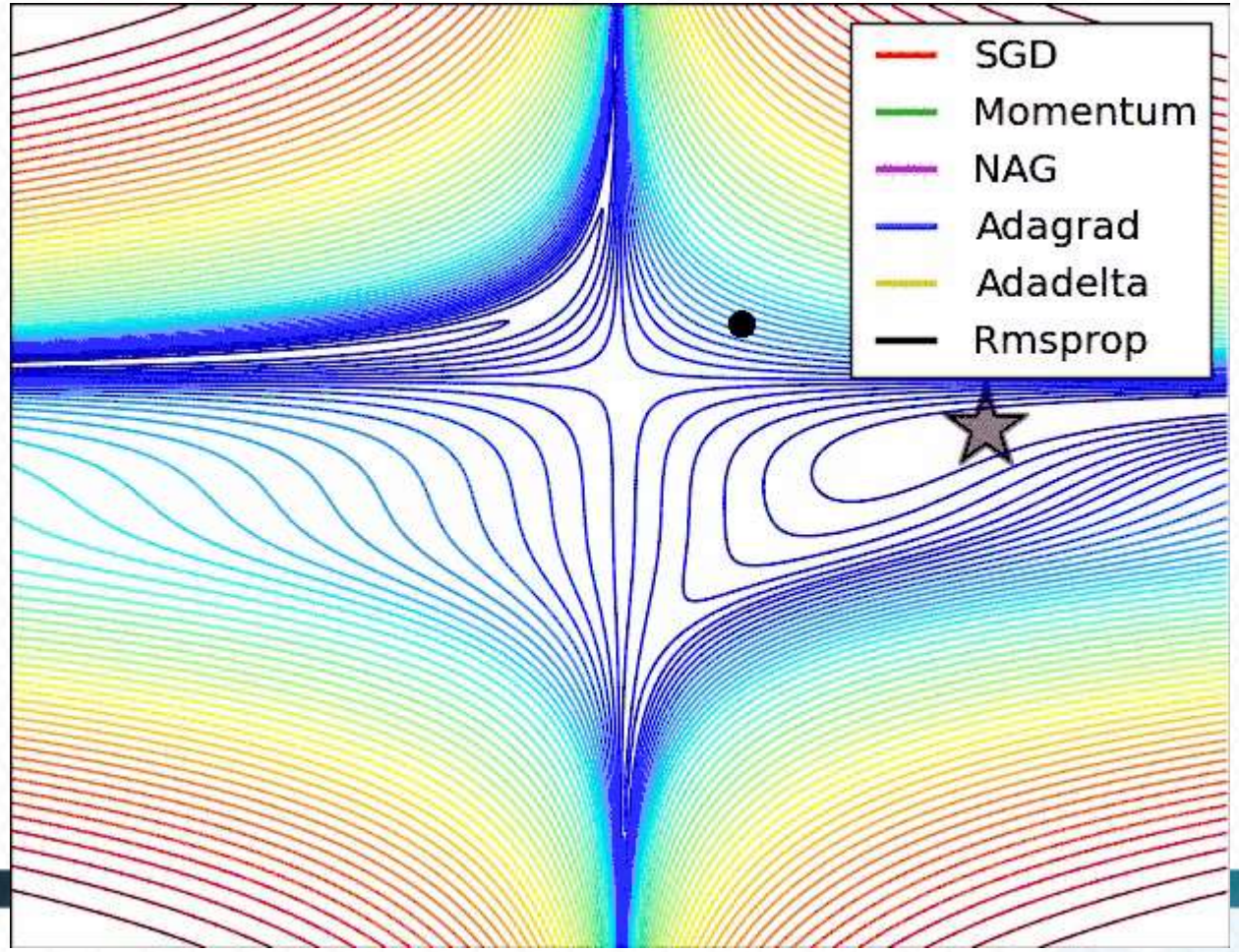
優化演算法動畫

- By Alec Radford
  - https://bit.ly/2u87PsD



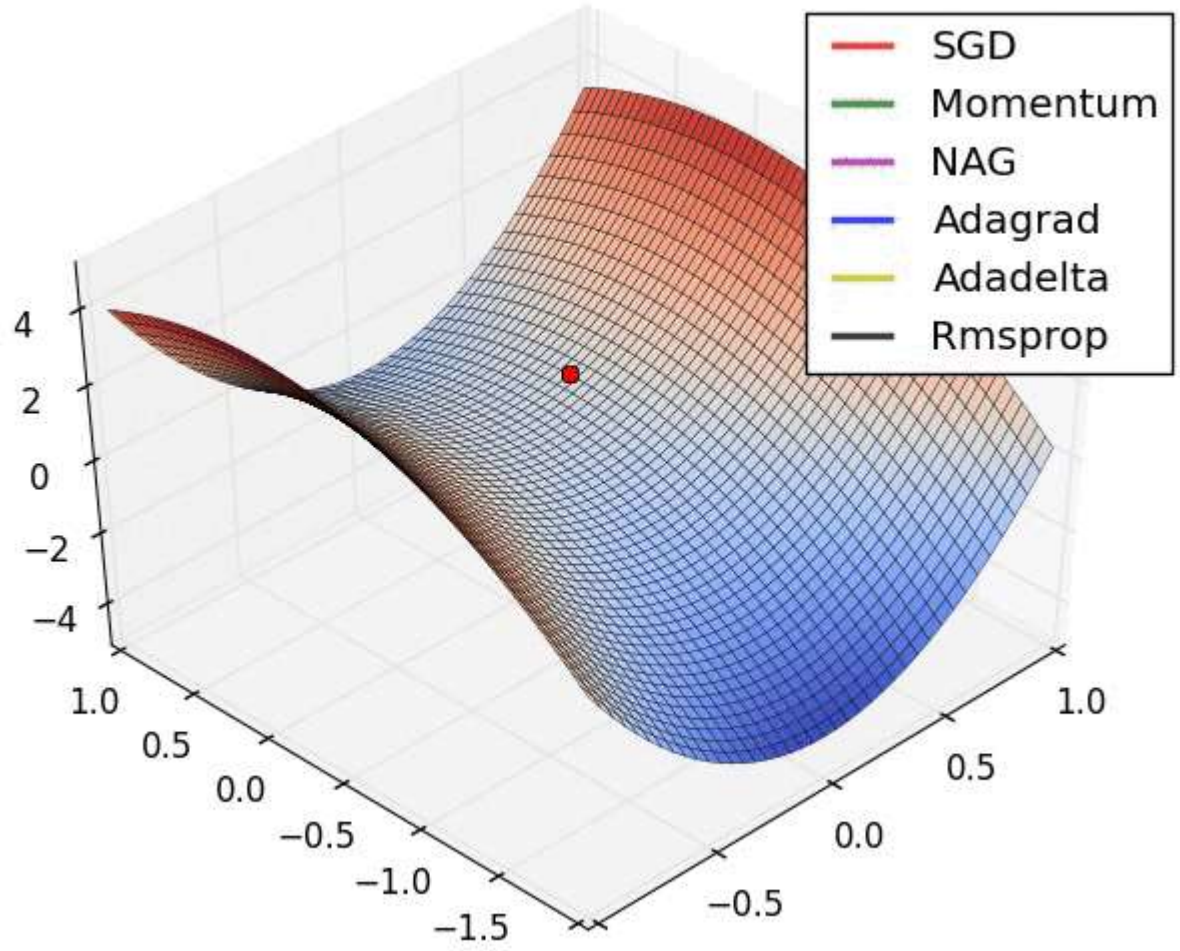
優化演算法動畫 (二)

- By Alec Radford
  - https://bit.ly/2u87PsD



優化演算法動畫 (三)

- By Alec Radford
  - https://bit.ly/2u8



該使用哪種優化器

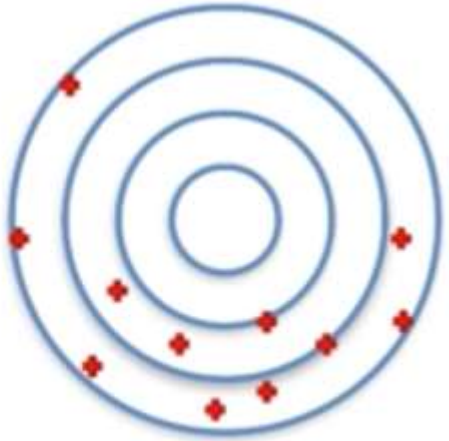
- 對於稀疏資料集，應該使用某種自我調整學習率的方法
  - 自我調整學習率較適用於稀疏資料集
  - 不用選擇學習率
- AdaGrad, RMSprop, AdaDelta 以及Adam 在多數的表現上非常類似,不過Kingma的比較論文中, Adam 的表現優於 RMSprop
- Adam在實際應用中效果最好
  - 不過多數的論文都使用SGD 做優化器!?

Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint

評估、調參、優化人工神經網路

Bias (偏差) 和 Variance (方差)

高 Bias (偏差) 低 Variance (方差)

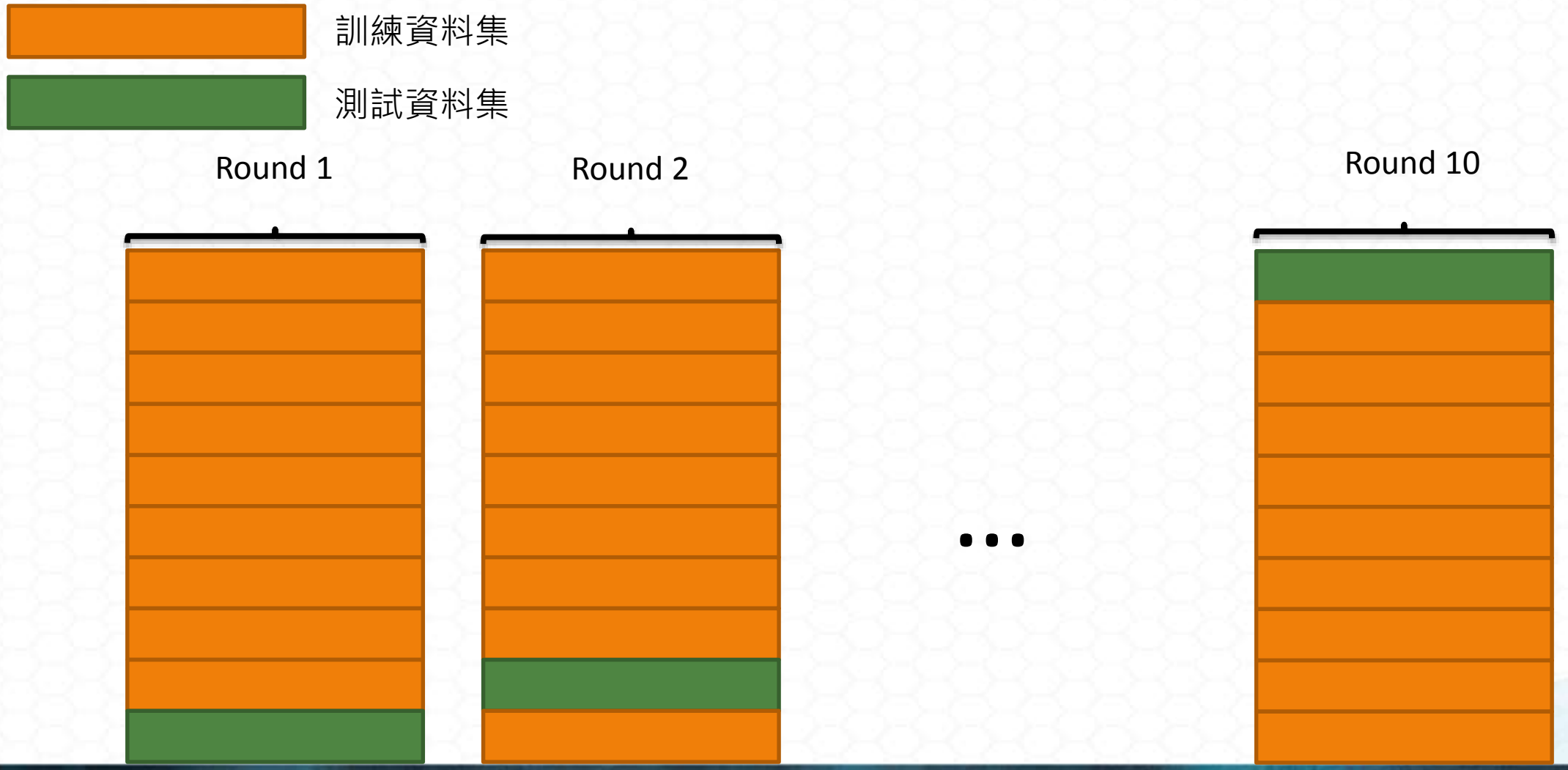


高 Bias (偏差) 高 Variance (方差)

低 Bias (偏差) 低 Variance (方差)

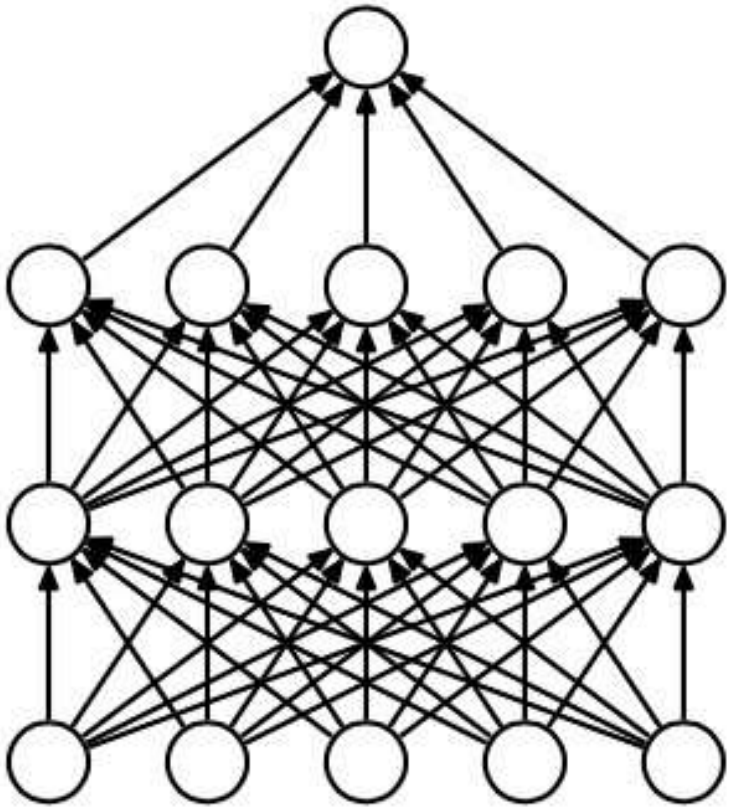
低 Bias (偏差) 高 Variance (方差)

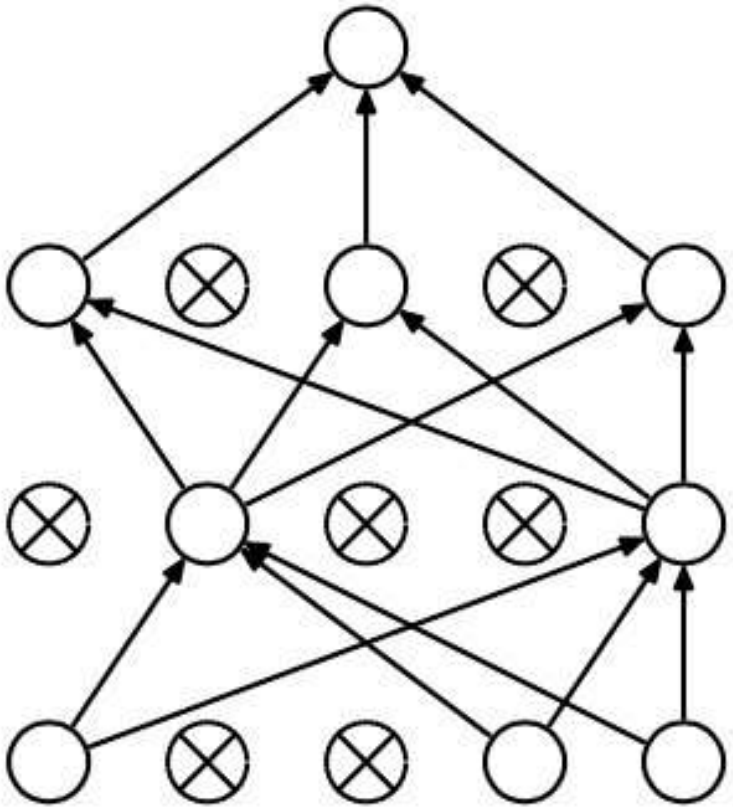
K-fold 交叉驗證



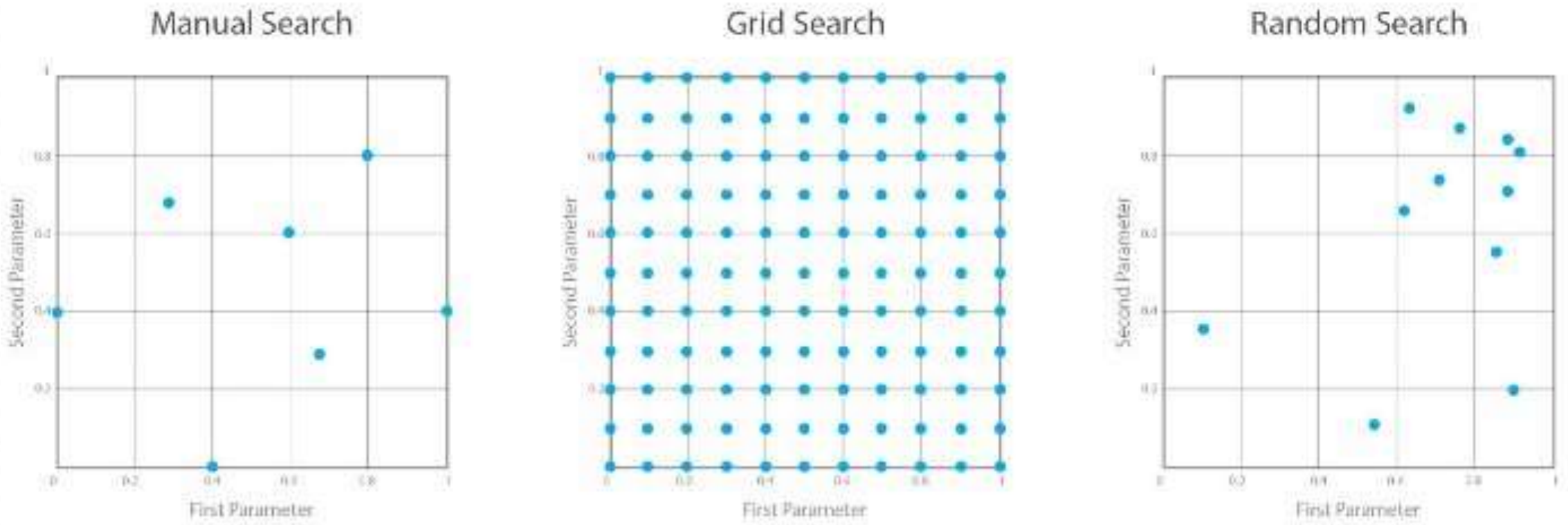
Dropout

- 隨機剔除神經元解決過擬和問題



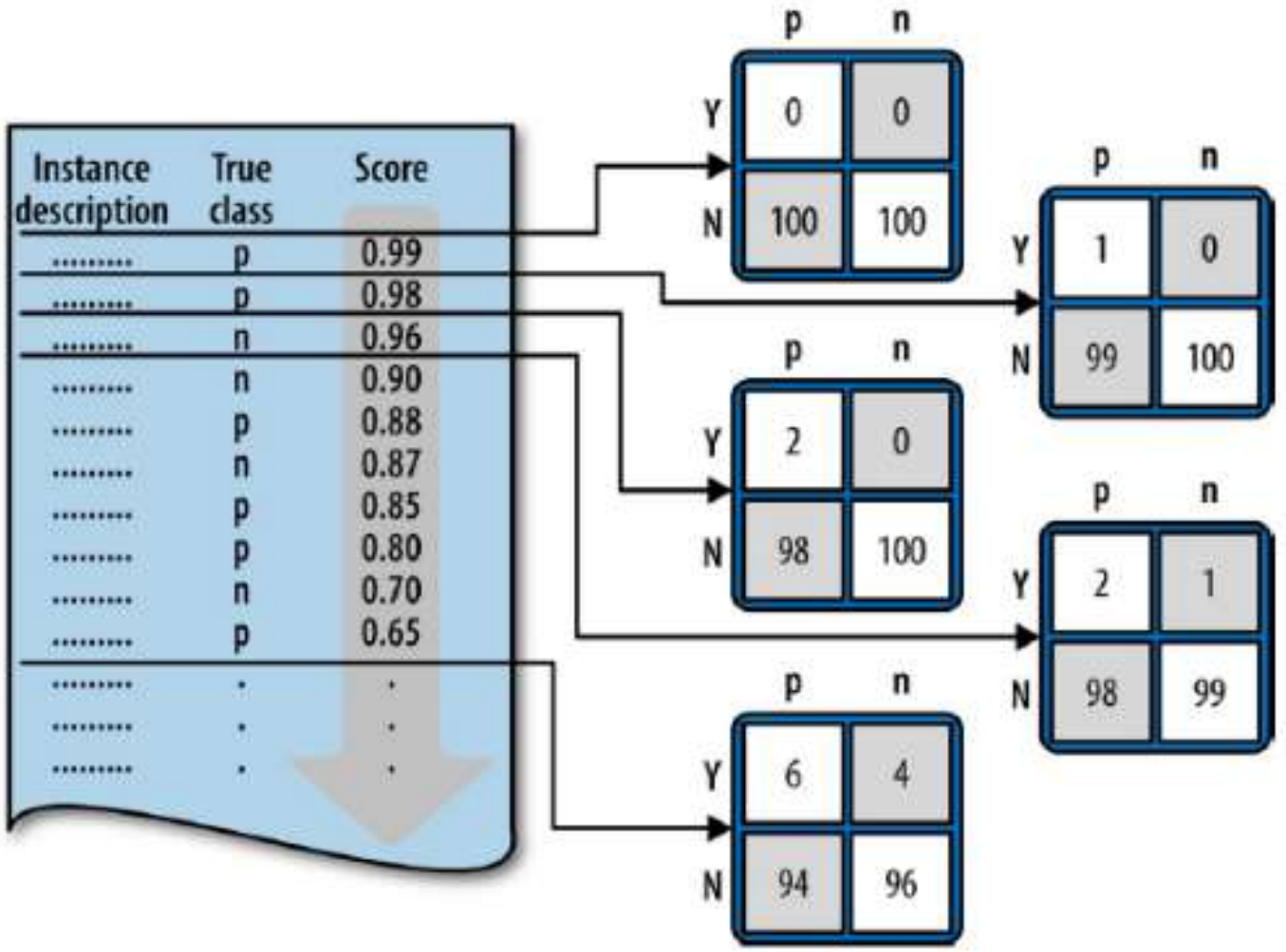


使用Grid Search 調整超參數



比較人工神經網路與其他機器學習模型

考慮不同成本下的混淆矩陣

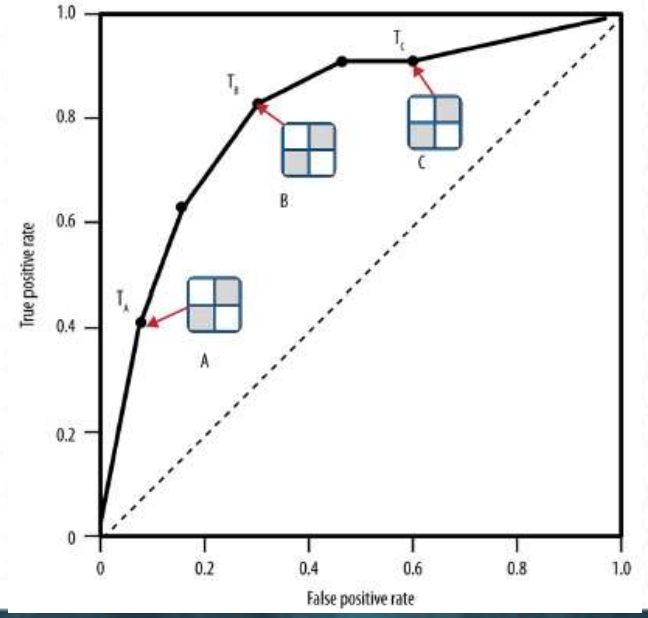


ROC 曲線

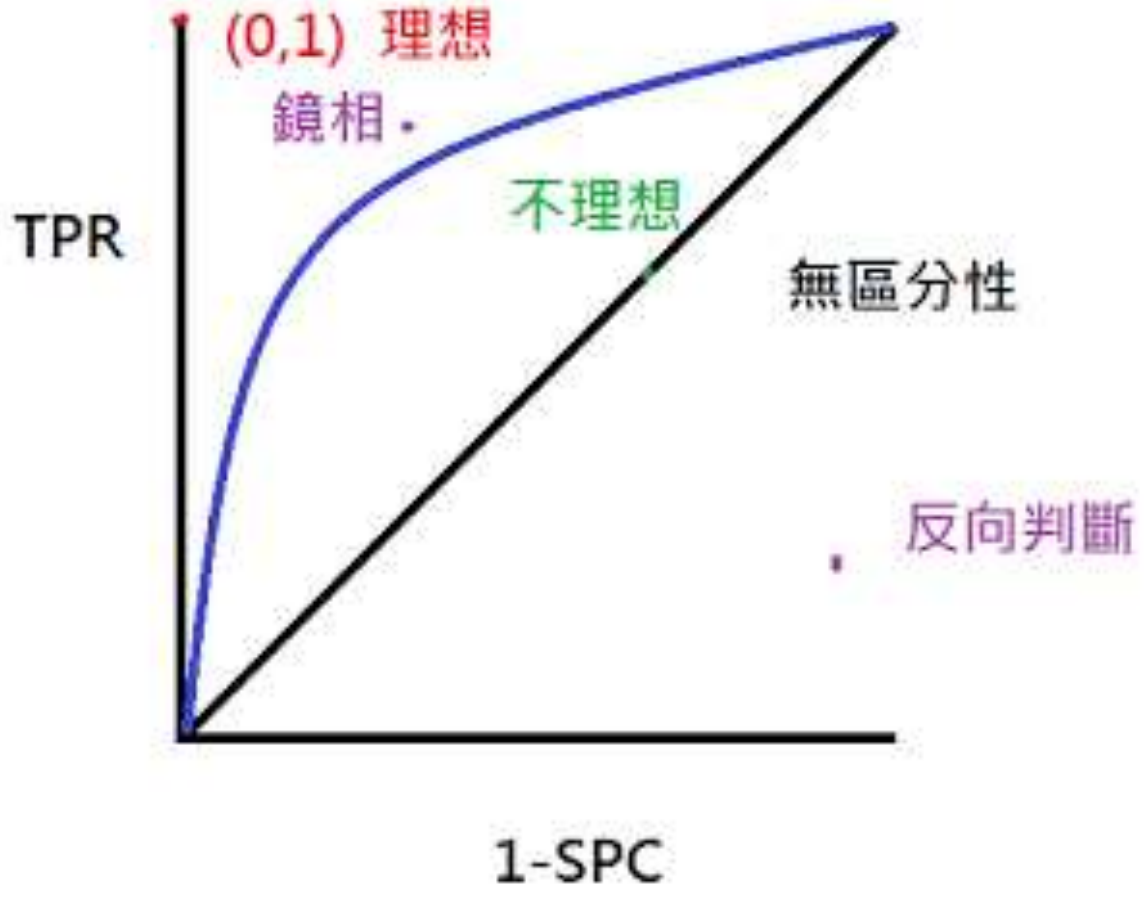
- 接收者操作特徵(receiver operating characteristic, ROC curve)

  1.以假陽性率(False Positive Rate, FPR)為X軸，代表在所有陰性相本中，被判斷為陽性(假陽性)的機率，又寫為(1-特異性)。

  2.以真陽性率(True Positive Rate, TPR)為Y軸，代表在所有陽性樣本中，被判斷為陽性(真陽性)的機率，又稱為敏感性



评估 ROC 曲线



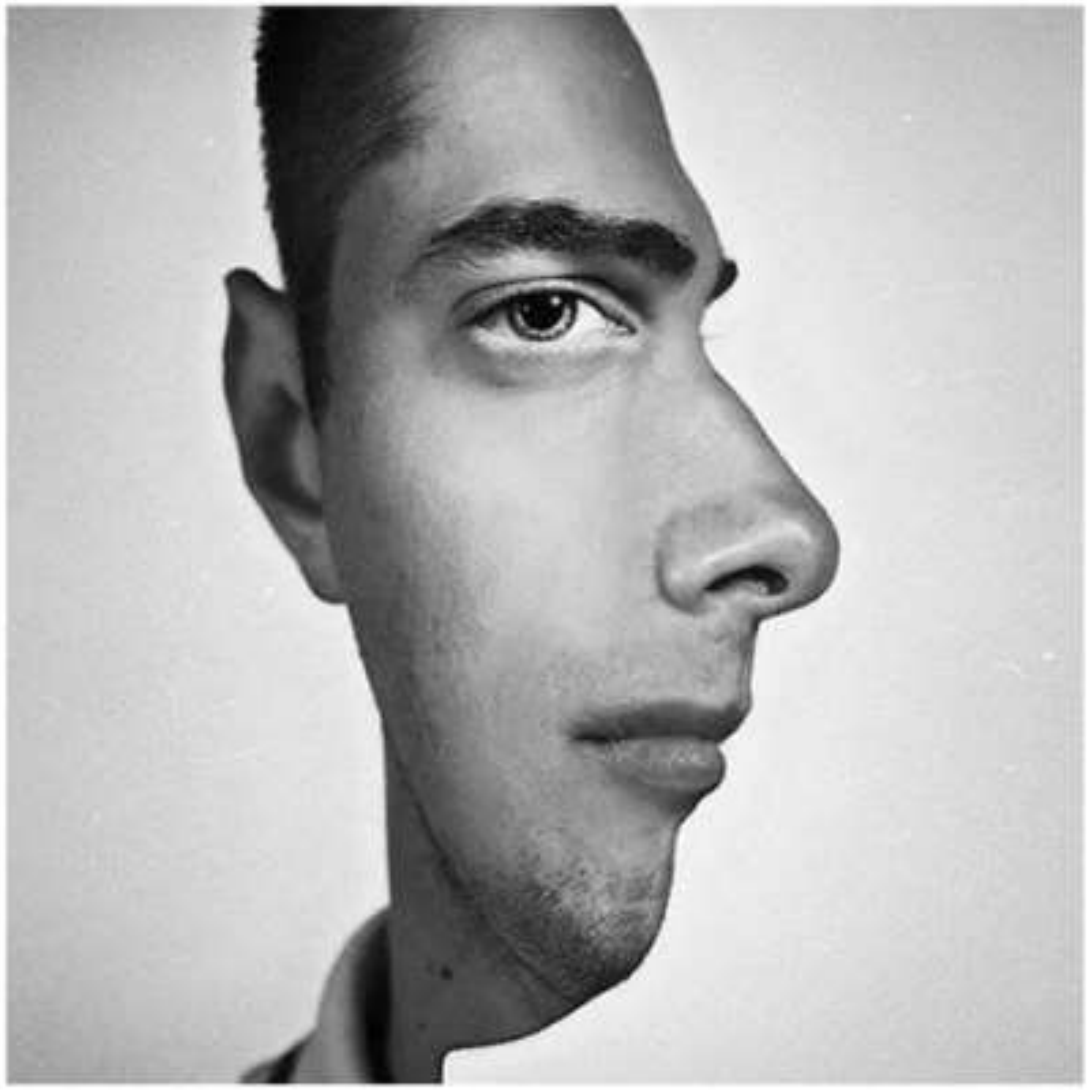
AUC

曲線下面積(Area Under Curve, AUC)為此篩檢方式性能優劣之指標，AUC越接近1，代表此篩檢方式效能越佳。指標可參考以下條件。

AUC数值	解释
1	完美分类器，无论cut-off point如何设定都可正确预测。通常不存在
0.5<AUC<1	优于随机，妥善设定可有预测价值
0.5	同随机，预测讯息没有价值

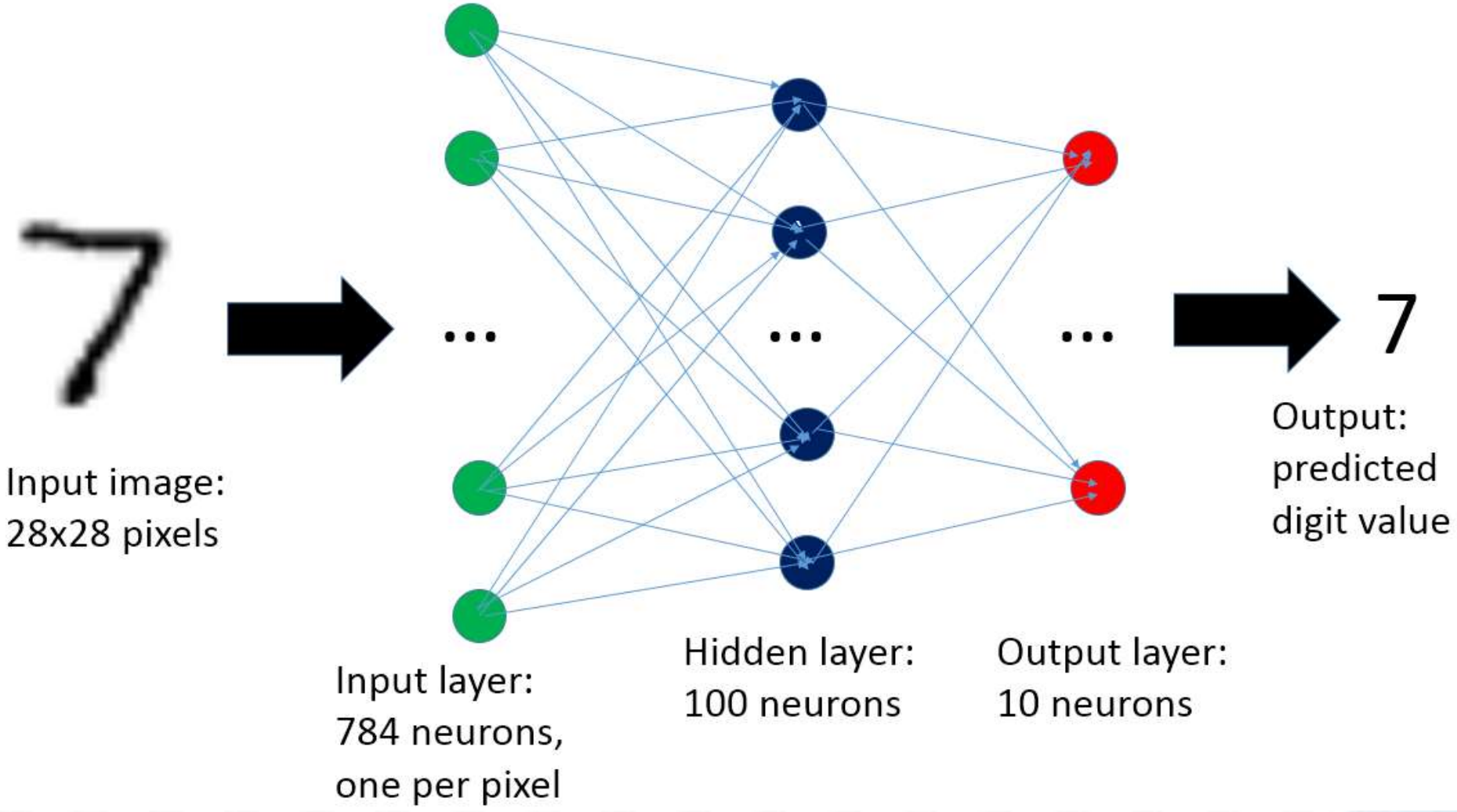
卷積神經網路
(Convolution Neural Network)

側臉或正面?



老太太或妙齡女子?

如何辨識手寫數字？


Input image:
28x28 pixels
Input layer:
784 neurons,
one per pixel
Hidden layer:
100 neurons
Output layer:
10 neurons
7
Output:
predicted
digit value

ANN在圖形識別時的缺點

- **參數數量龐大**

  輸入28* 28圖元的圖片，輸入層有784節點。假設第一個隱藏層有100個節點，一層就有(784+1)*100=78,500個參數

- **沒有利用圖像特徵去學習**

  每個圖元只與周邊的圖元有關, 識別出圖像的特徵, 可以大幅加速學習過程, 提高學習準確率.

- **網路層數限制**

  當節點太多,自然無法建立出較多層數的神經網路

Yann LeCun

- http://yann.lecun.com/

根據特徵做學習?

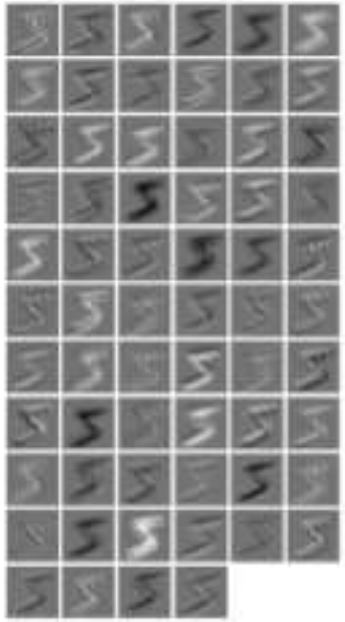


Figure 1: The 64 feature maps after first layer of first block (spatial convolution)

Figure 2: The 64 feature maps after second layer of first block (tanh)

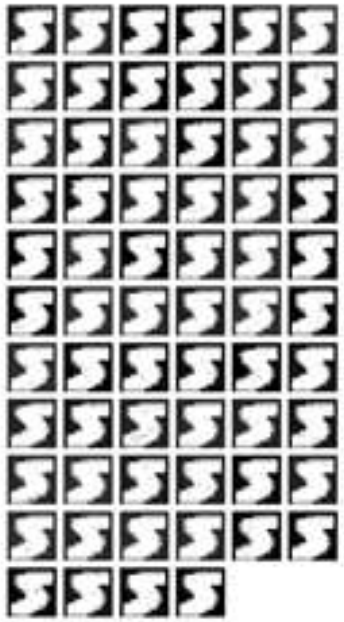
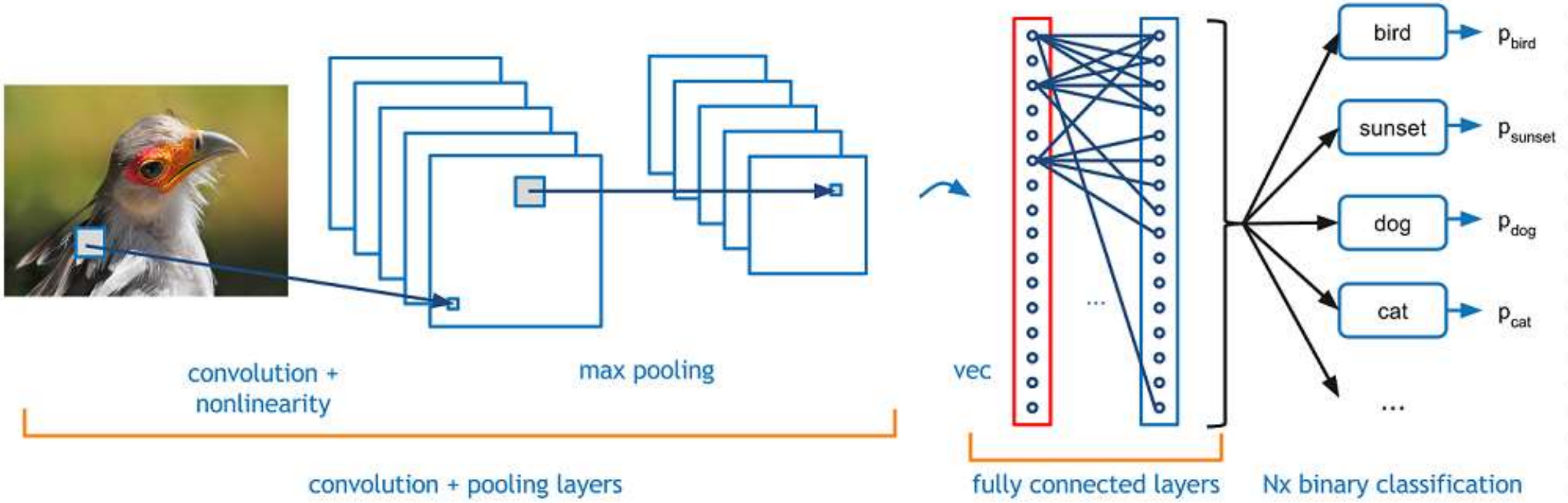


Figure 3: The 64 feature maps after third layer of first block (L2 Pooling)

Figure 4: The 64 feature maps after fourth layer of first block (substractive normalization)

卷積神經網路架構

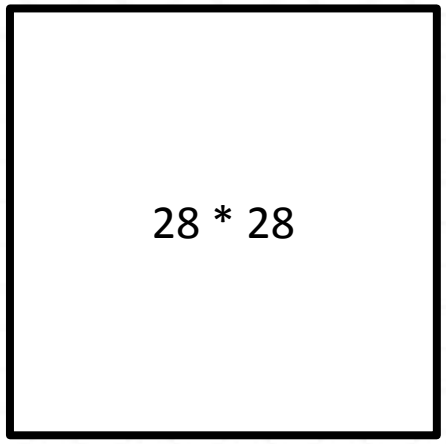

convolution +
nonlinearity
max pooling
vec
bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + pooling layers
fully connected layers
Nx binary classification

卷積神經網路方法

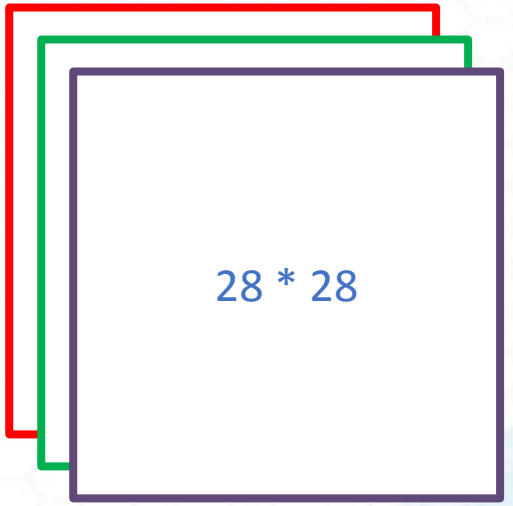
卷積特徵提取

將圖片變為數學矩陣

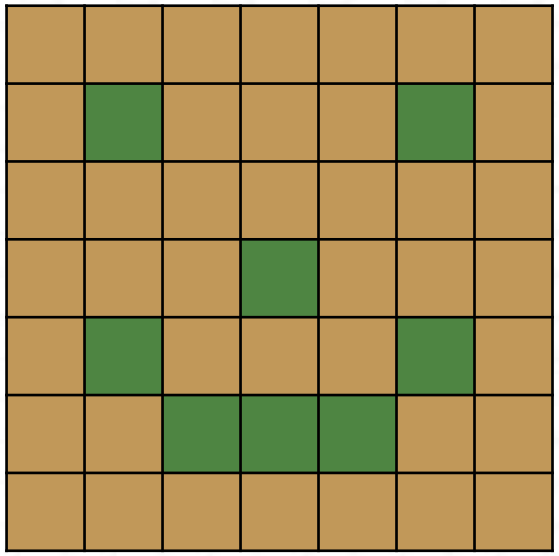
2d Array



3d Array

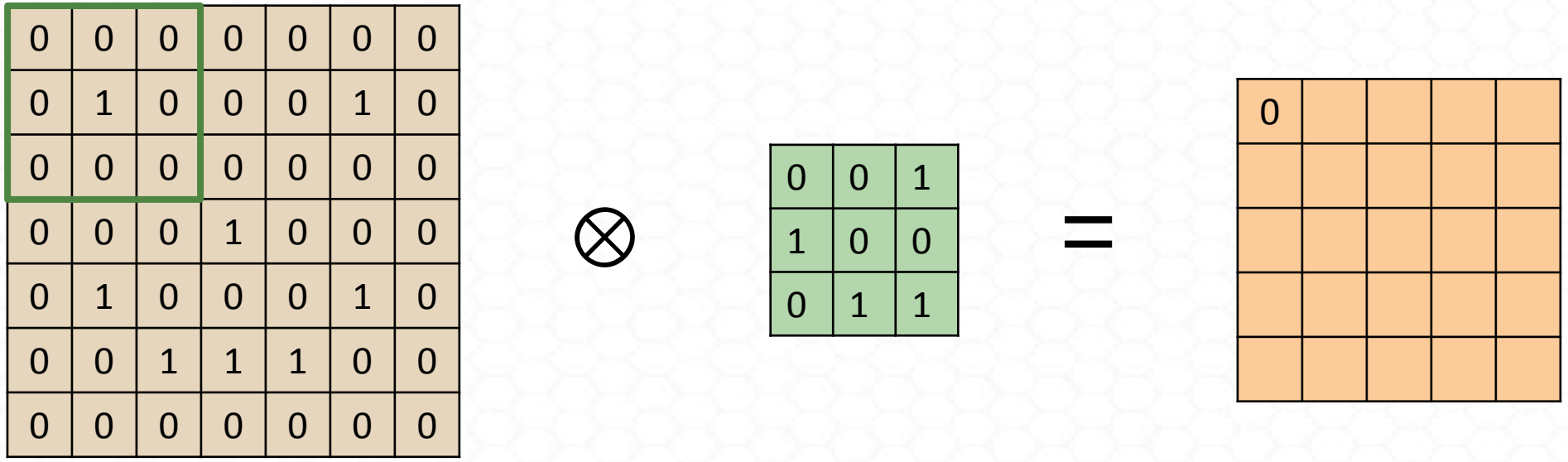


將圖片變為數學矩陣



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

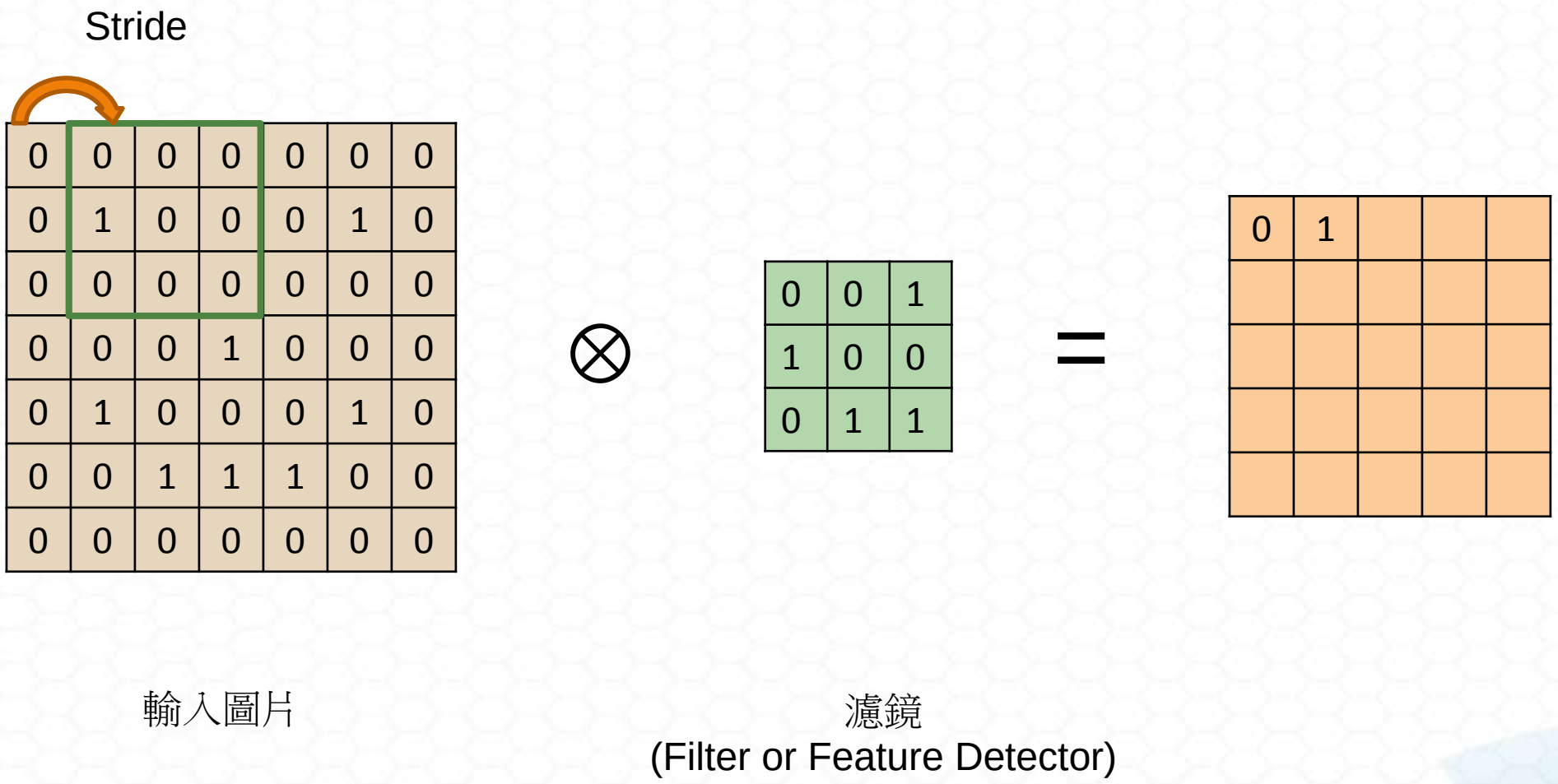
卷積特徵提取



輸入圖片

濾鏡
(Filter or Feature Detector)

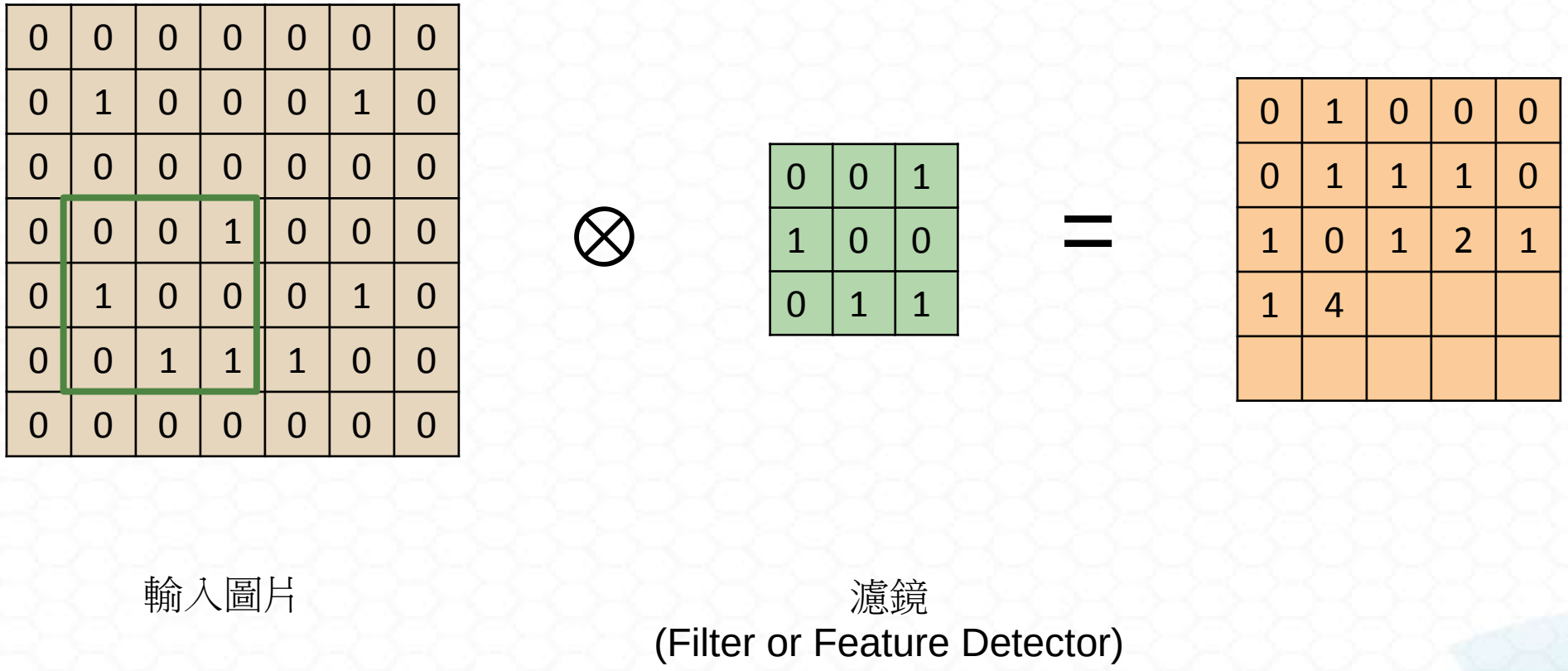
卷積特徵提取



輸入圖片

濾鏡
(Filter or Feature Detector)

卷積特徵提取



輸入圖片

濾鏡
(Filter or Feature Detector)

卷積特徵提取

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

輸入圖片

$\otimes$

0	0	1
1	0	0
0	1	1

濾鏡
(Filter or Feature Detector)

$=$

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Padding

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

輸入圖片

⊗

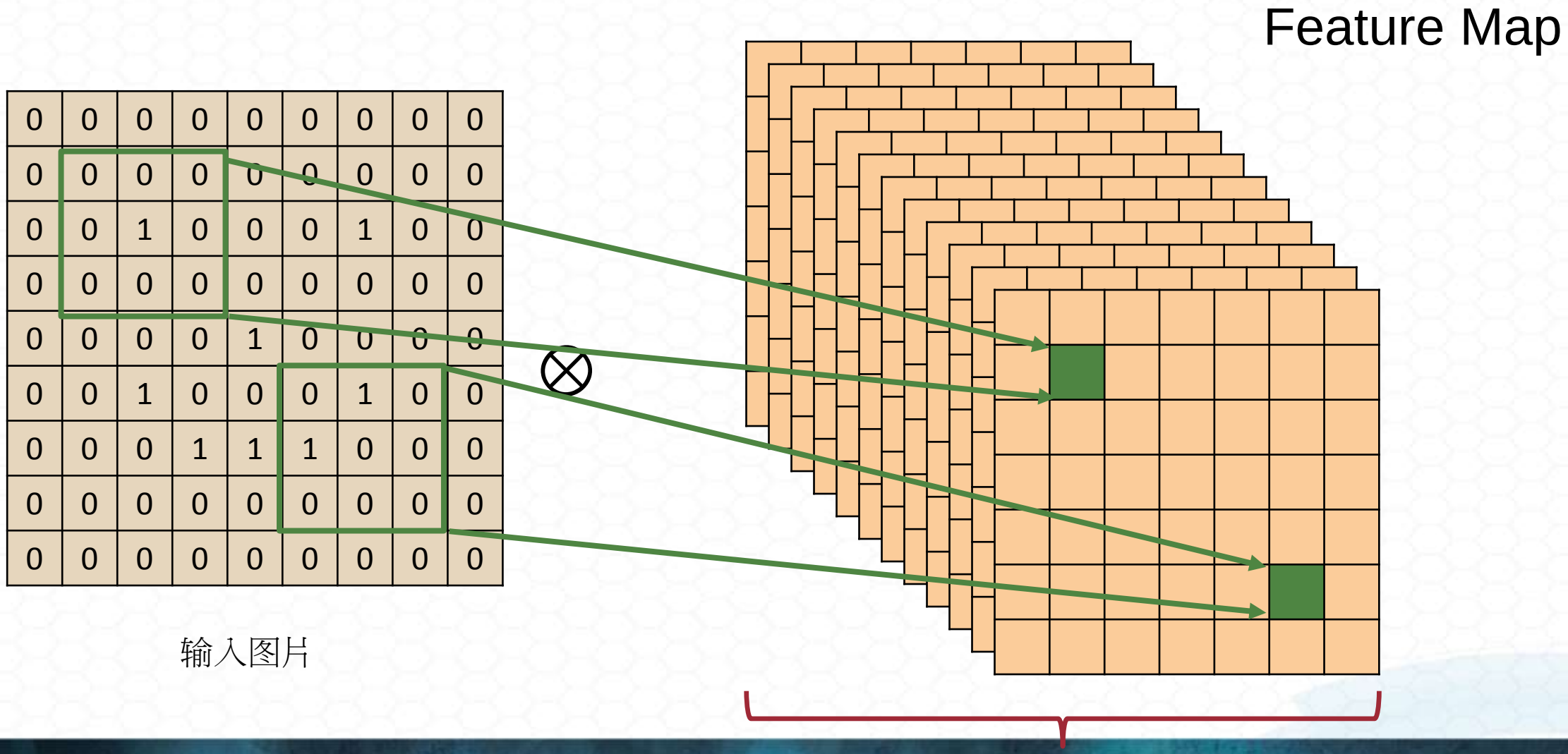
0	0	1
1	0	0
0	1	1

濾鏡
(Filter or Feature Detector)

=

1	1	0	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	0	0
1	1	0	1	2	1	0
0	1	4	2	1	0	1
1	0	0	1	2	1	0
0	0	1	1	1	0	0

Convolution Layer



卷積矩陣

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

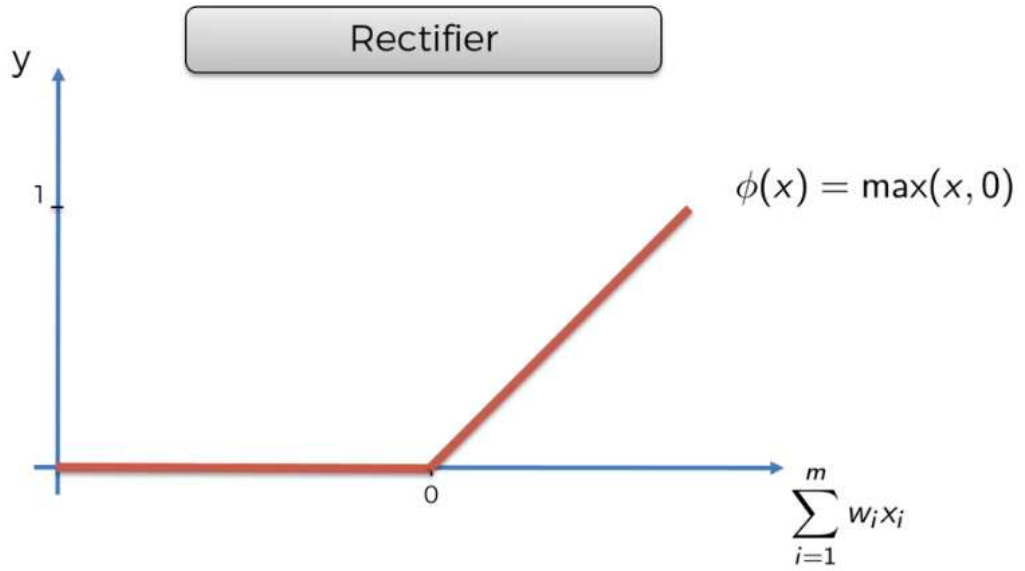
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

RELU層

ReLU 函数

```
def relu_function(x):
    return np.maximum(0,x)

x = np.array([-1,1,2])
relu_function(x)
```

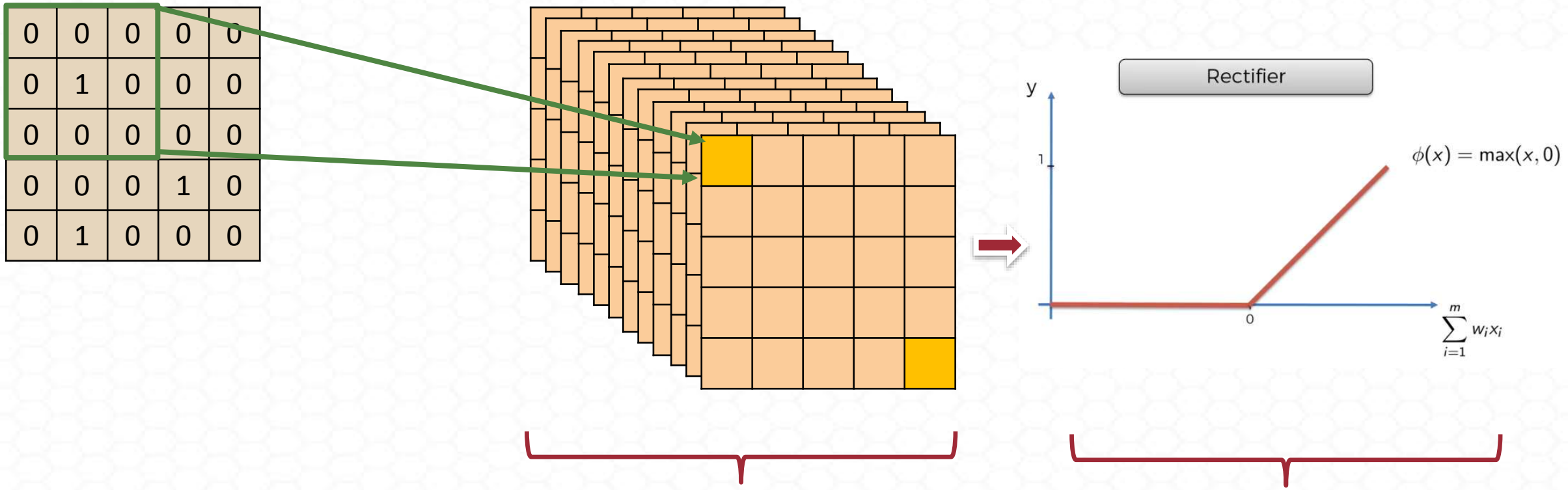


优点:
- 快速收敛
- 解决梯度消失问题

缺点:
- 神经元死亡问题

增加ReLU层

Feature Map



Convolution Layer

ReLU Layer

ReLU效果

Original:

Feature Detector:

ReLU:

池化層

如何用不同角度辨認老鷹?

如何用不同角度辨認老鷹?






池化層 (Pooling layer)

1. 保留重要的特徵
2. 具有抗干擾的作用
3. 減少過擬合(Over-Fitting)

池化過程

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

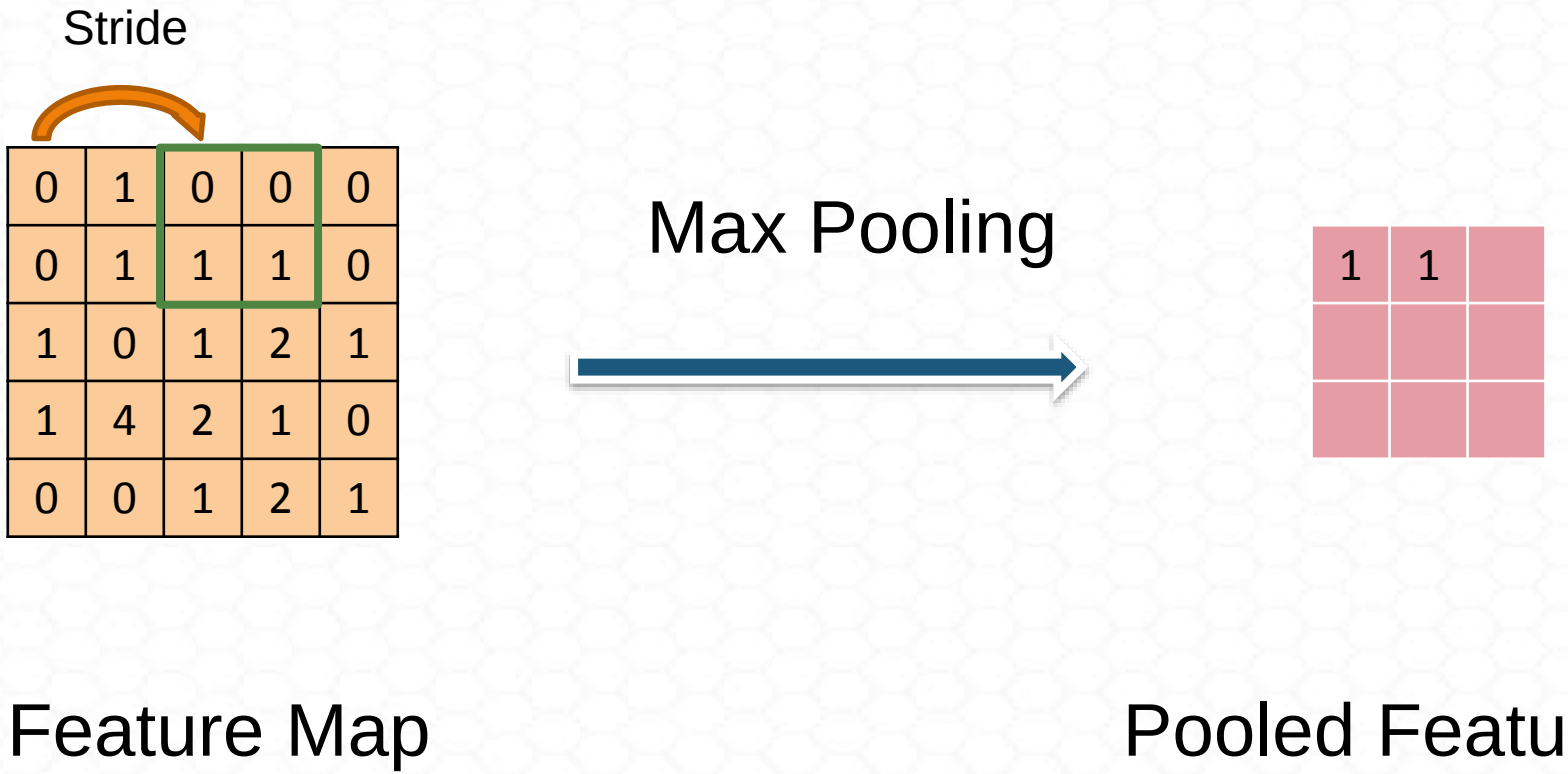
Max Pooling

1		

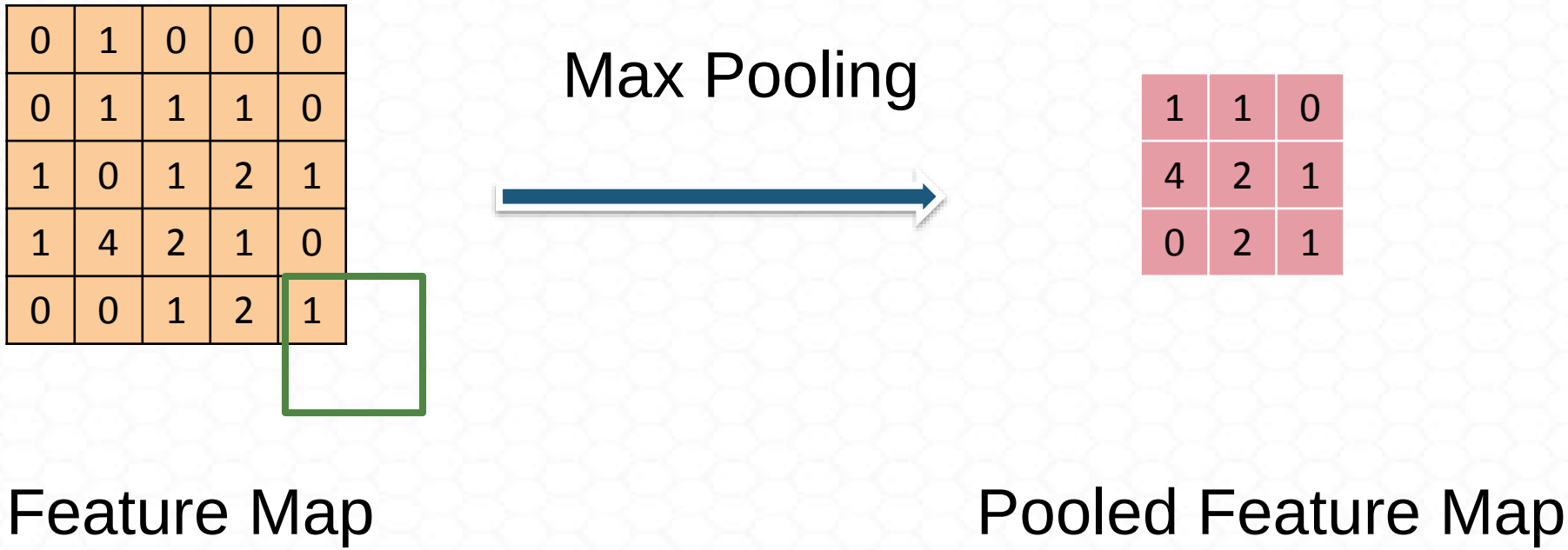
Feature Map

Pooled Feature Map

池化過程


Stride
0 1 0 0 0
0 1 1 1 0
1 0 1 2 1
1 4 2 1 0
0 0 1 2 1
Max Pooling
1 1
Feature Map
Pooled Feature Map

池化過程


0 1 0 0 0
0 1 1 1 0
1 0 1 2 1
1 4 2 1 0
0 0 1 2 1
Max Pooling
1 1 0
4 2 1
0 2 1
Feature Map
Pooled Feature Map

平化 (FLATTENING)

平化 (Flattening)

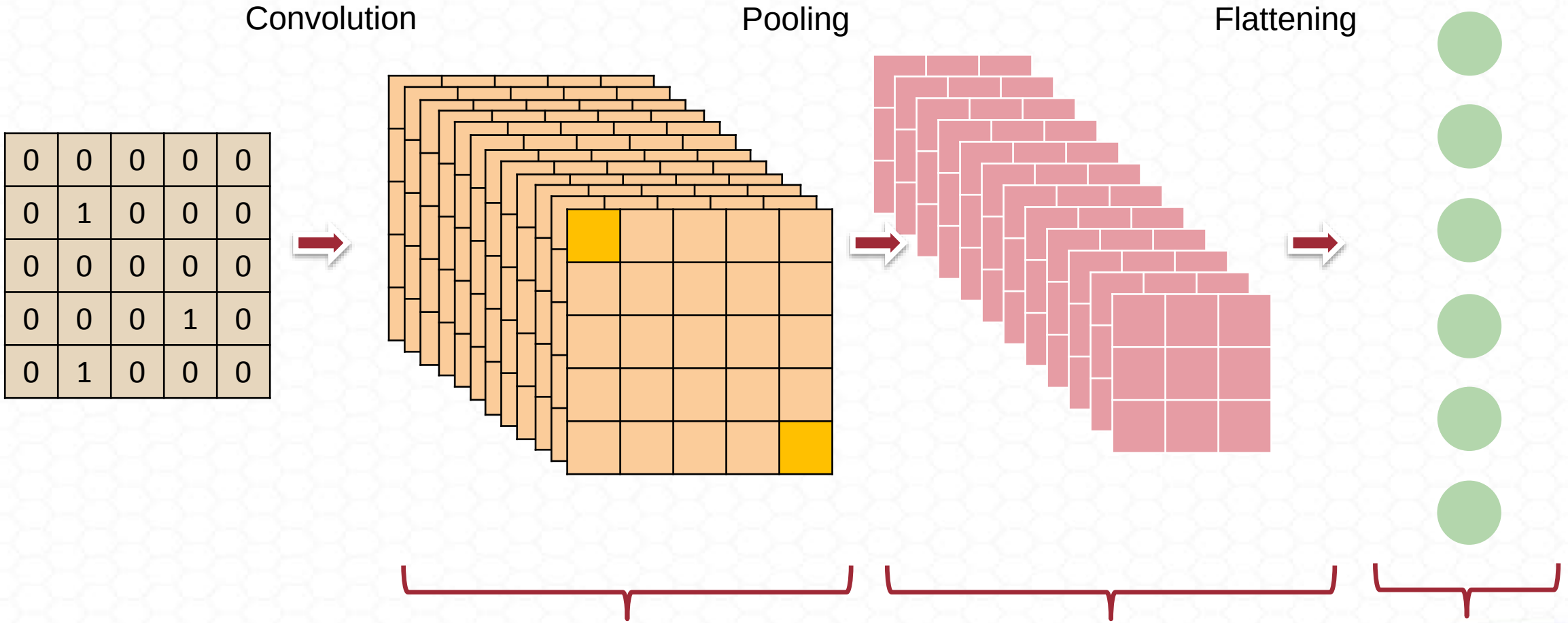
1	1	0
4	2	1
0	2	1

Flattening

1
1
0
4
2
1
0
2
1

Pooled Feature Map

平化 (Flattening)

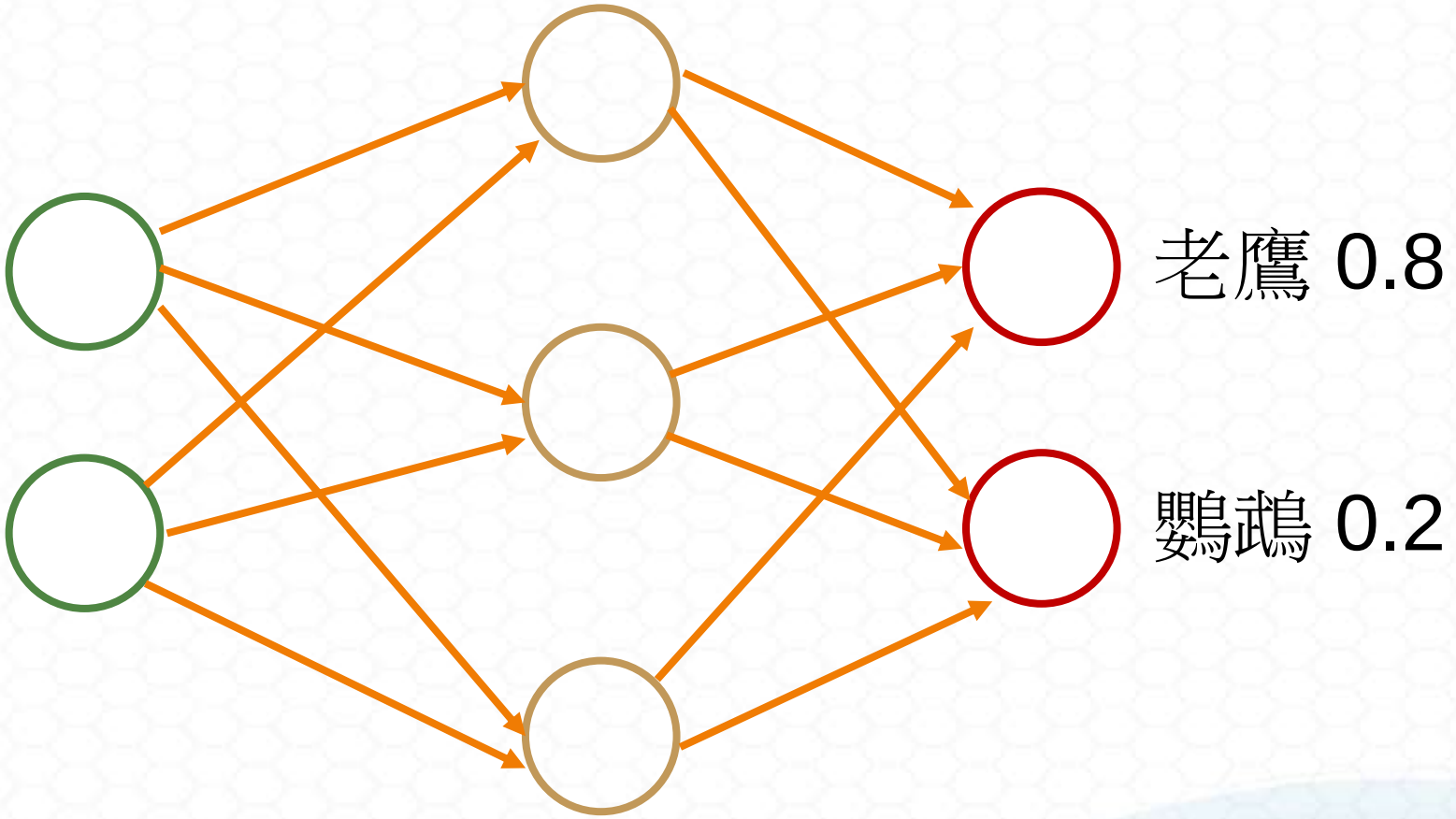


建立卷積神經網路

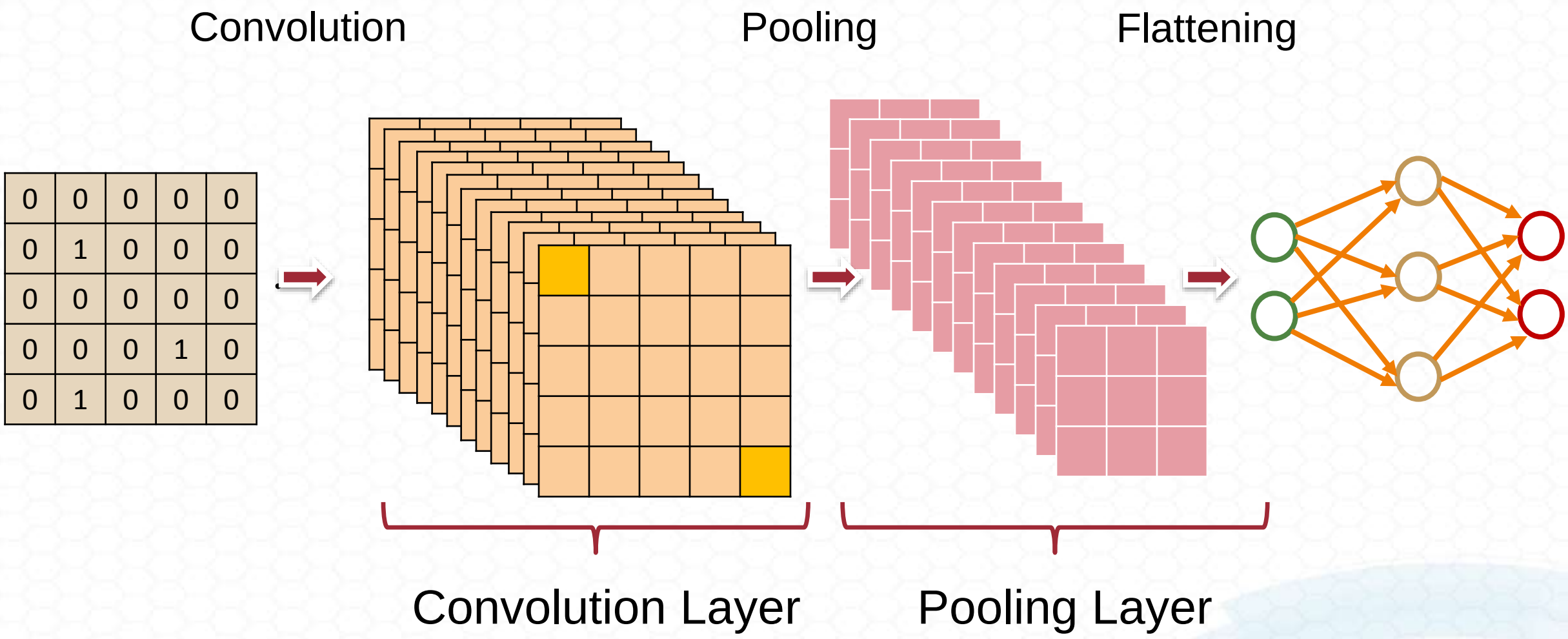
卷積神經網路方法

建立卷積神經網路

Flattening



建立卷積神經網路


Convolution
Pooling
Flattening
0 0 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 1 0
0 1 0 0 0
Convolution Layer
Pooling Layer

[實例] 利用卷積神經網路辨識圖片

利用卷積神經網路辨識明星圖片

使用網路爬蟲抓取明星圖片

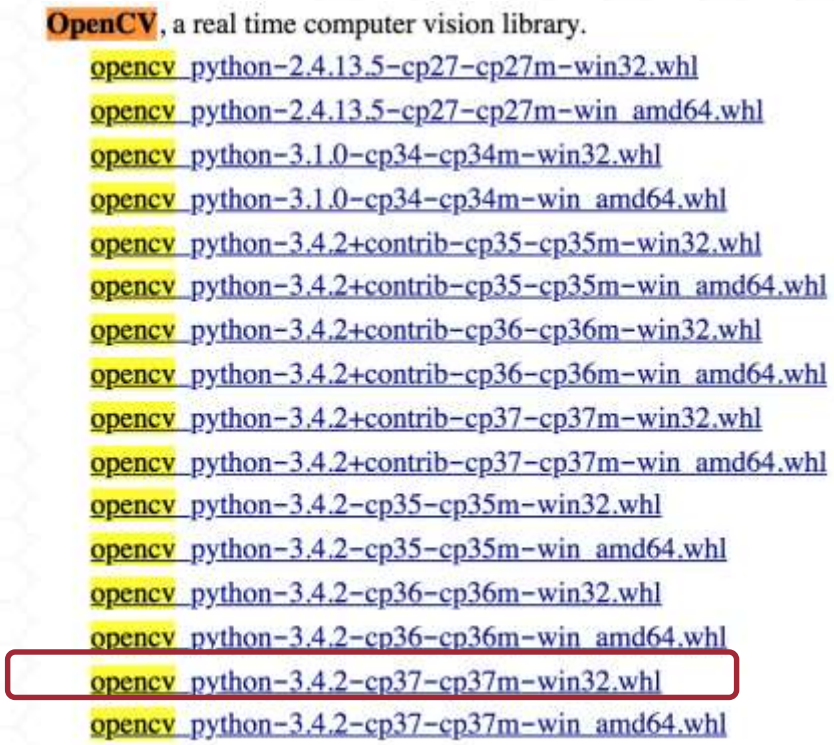
使用OpenCV 擷取人臉

Mac 安装OpenCV

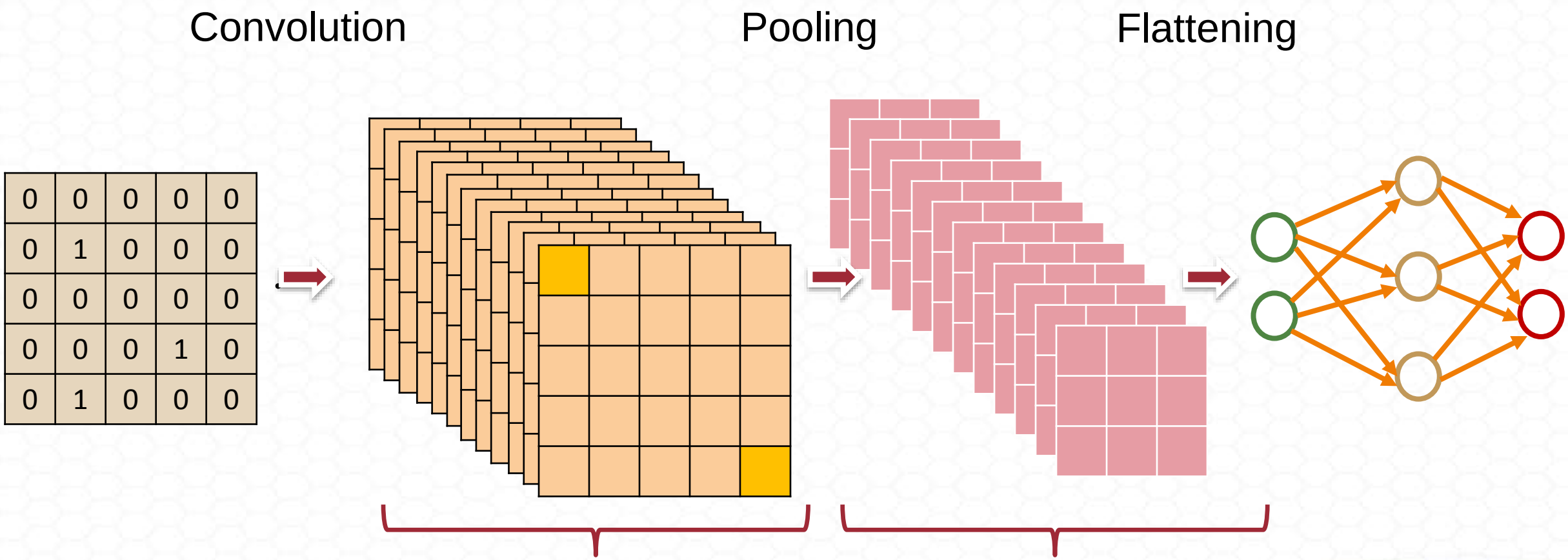
```
# add opencv
brew tap homebrew/science

# install opencv
brew install opencv

# install opencv3
brew install opencv3
```

Windows 安裝OpenCV

- 下載OpenCV
  - https://www.lfd.uci.edu/~gohlke/pythonlibs/

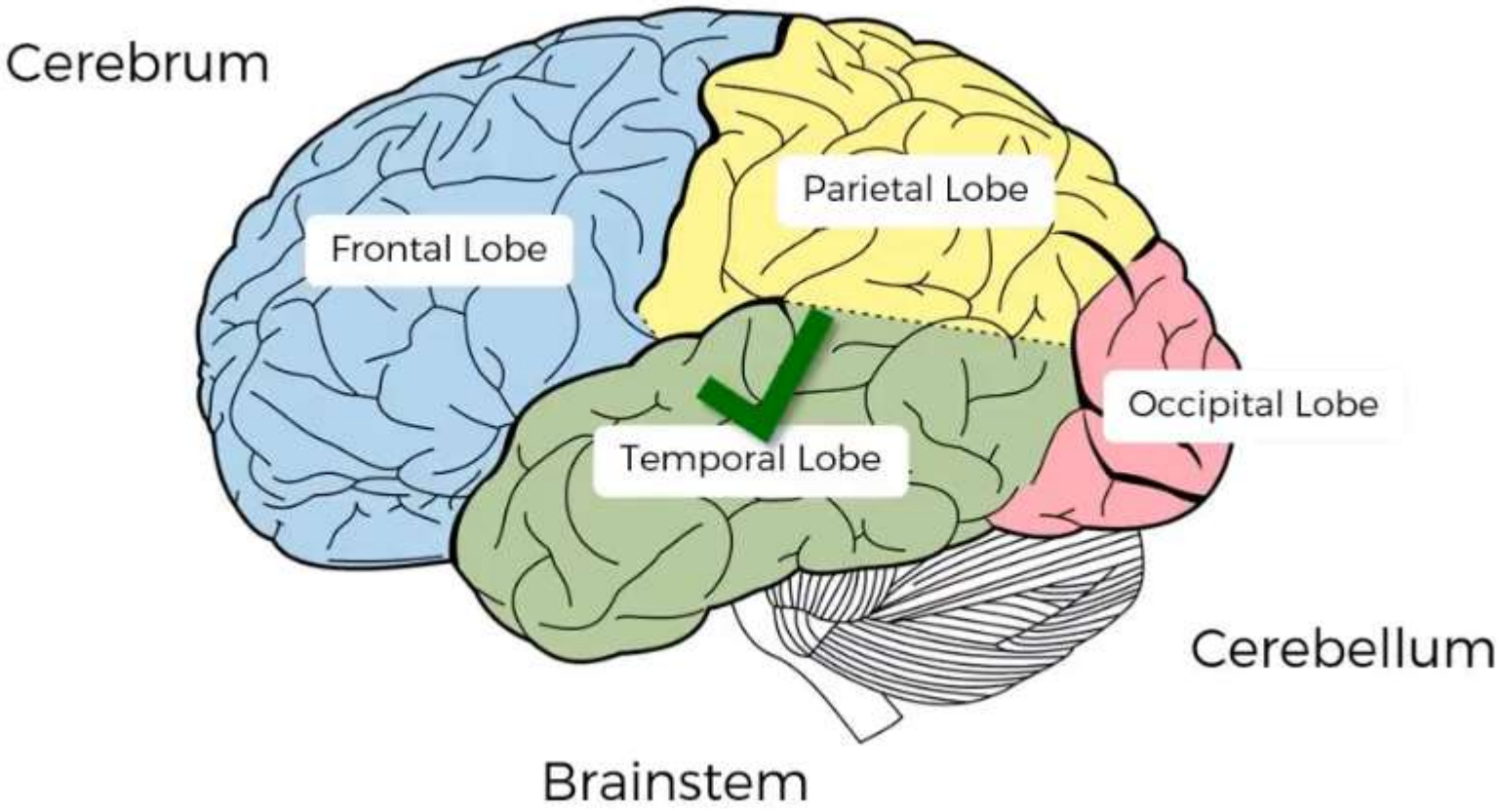


建立卷積神經網路

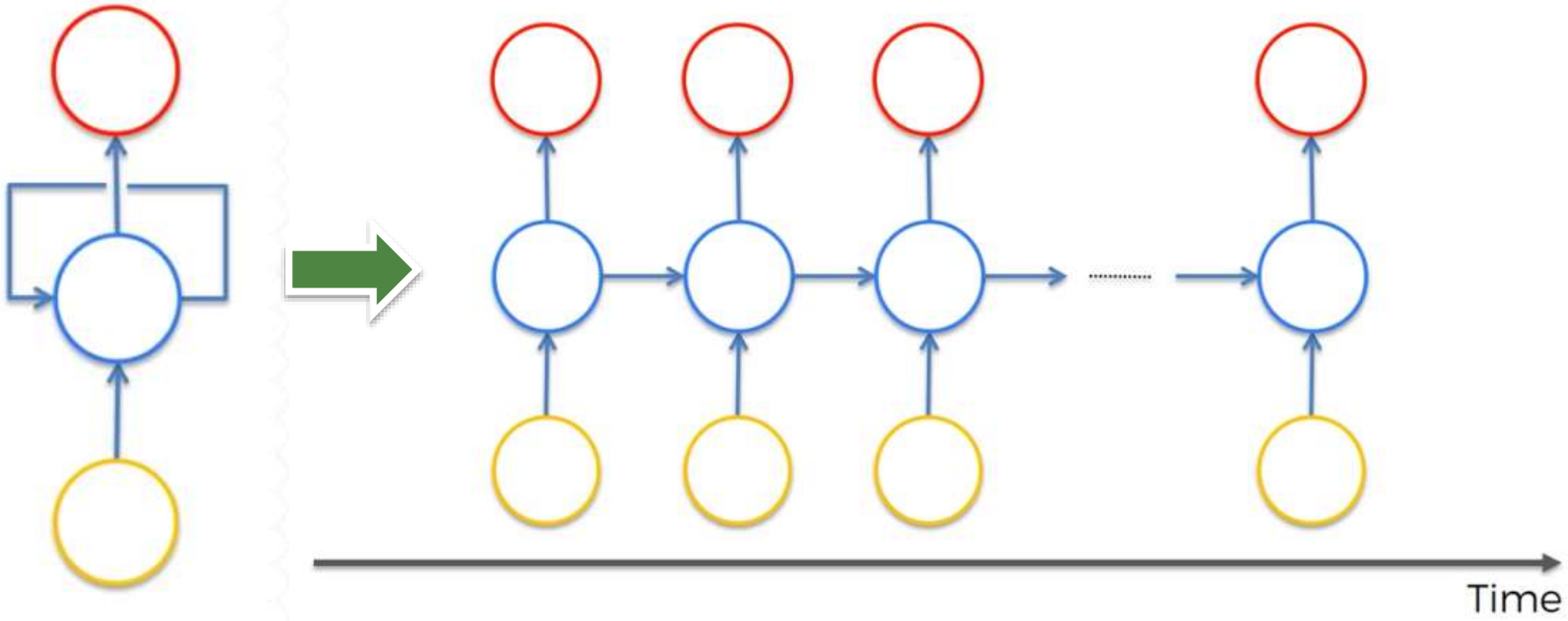

Convolution
Pooling
Flattening
0 0 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 1 0
0 1 0 0 0
Convolution Layer
Pooling Layer

循環神經網路
(Recurrent Neural Networks)

腦的運作方式


Cerebrum
Frontal Lobe
Parietal Lobe
Occipital Lobe
Temporal Lobe
Cerebellum
Brainstem

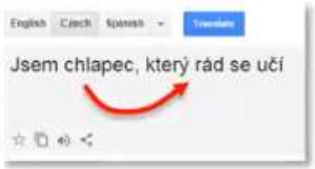
循環神經網路 (Recurrent Neural Networks)

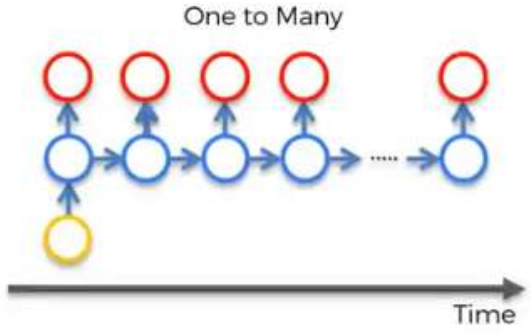


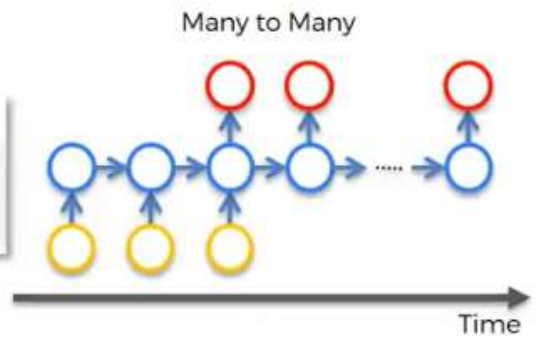
循環神經網路的應用

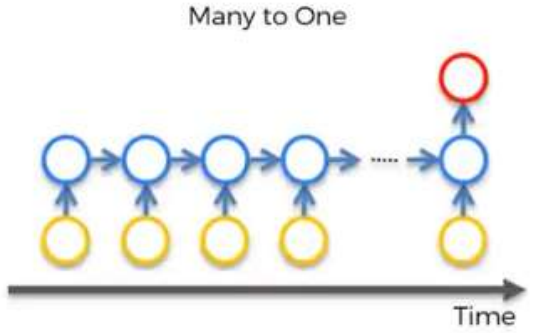
"black and white dog jumps over bar."

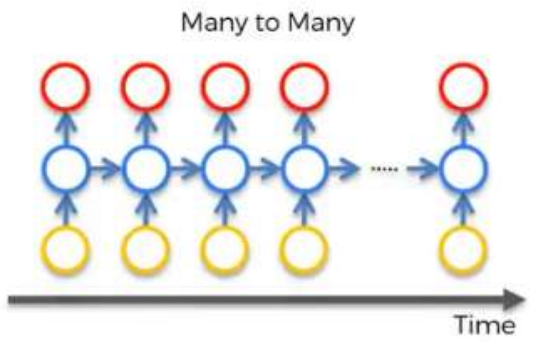
karpathy.github.io



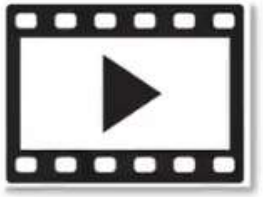




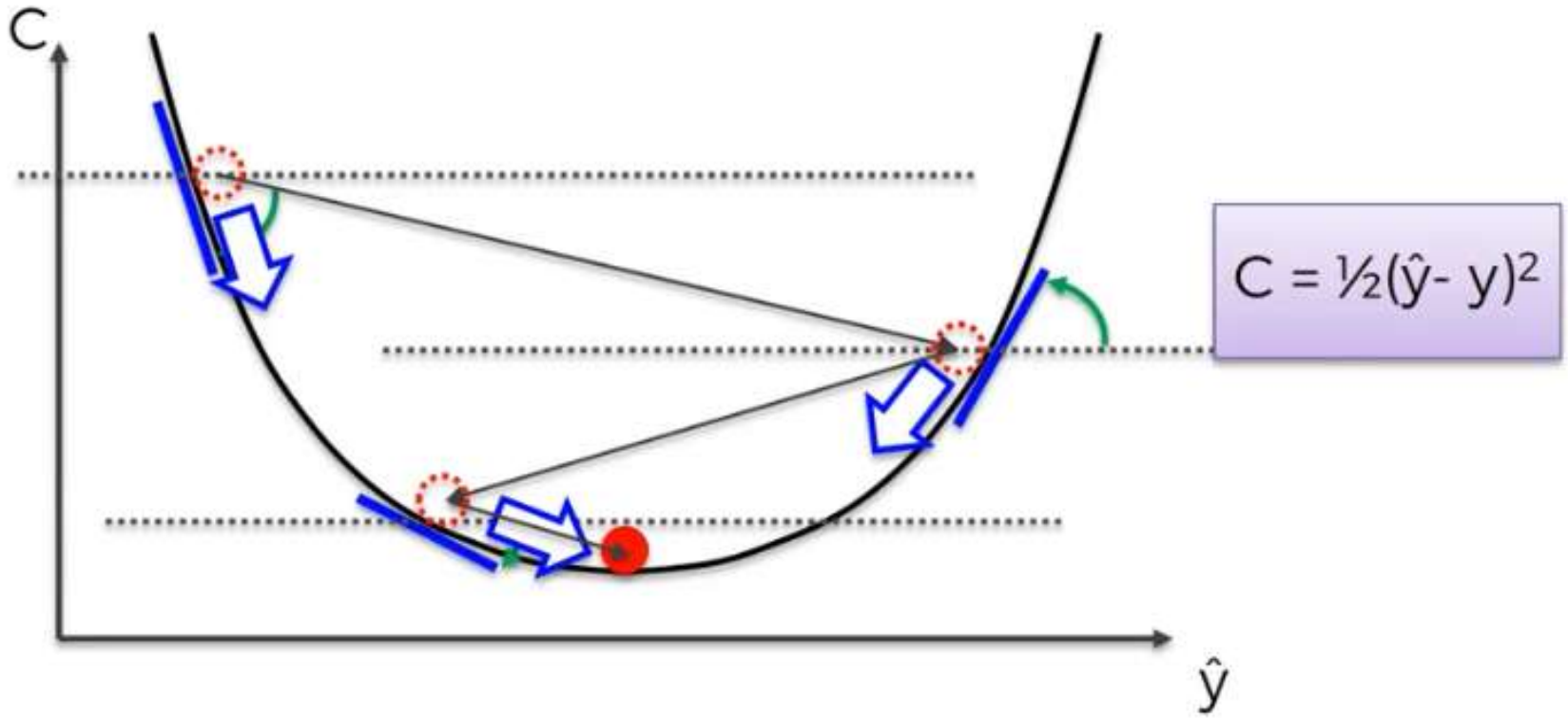




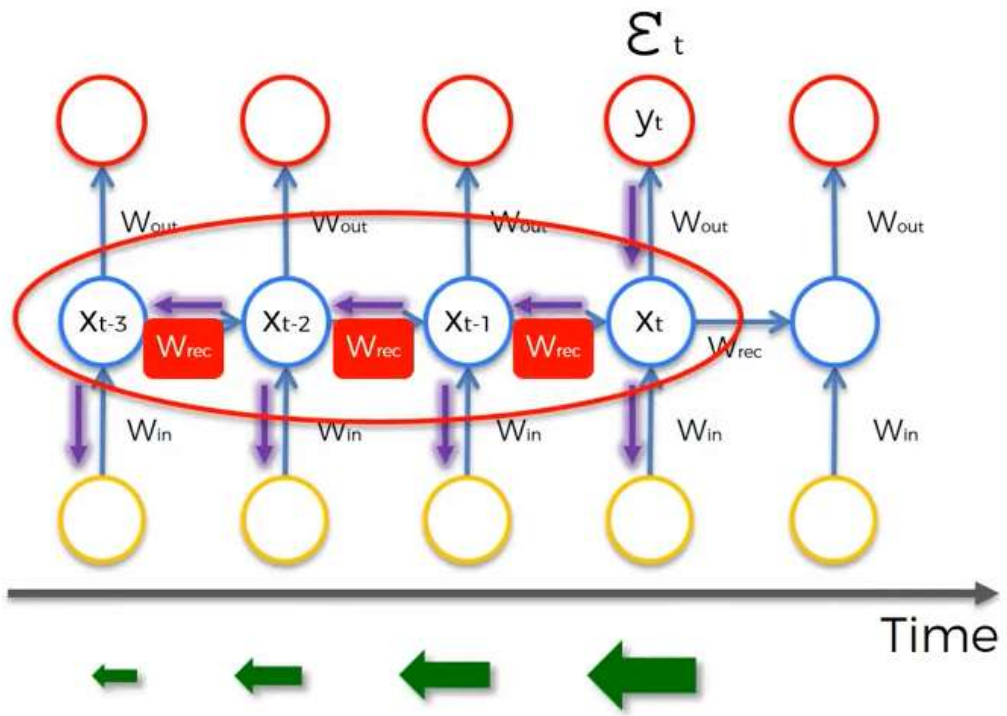
dev.havenondemand.com



梯度下降問題



梯度消失問題



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

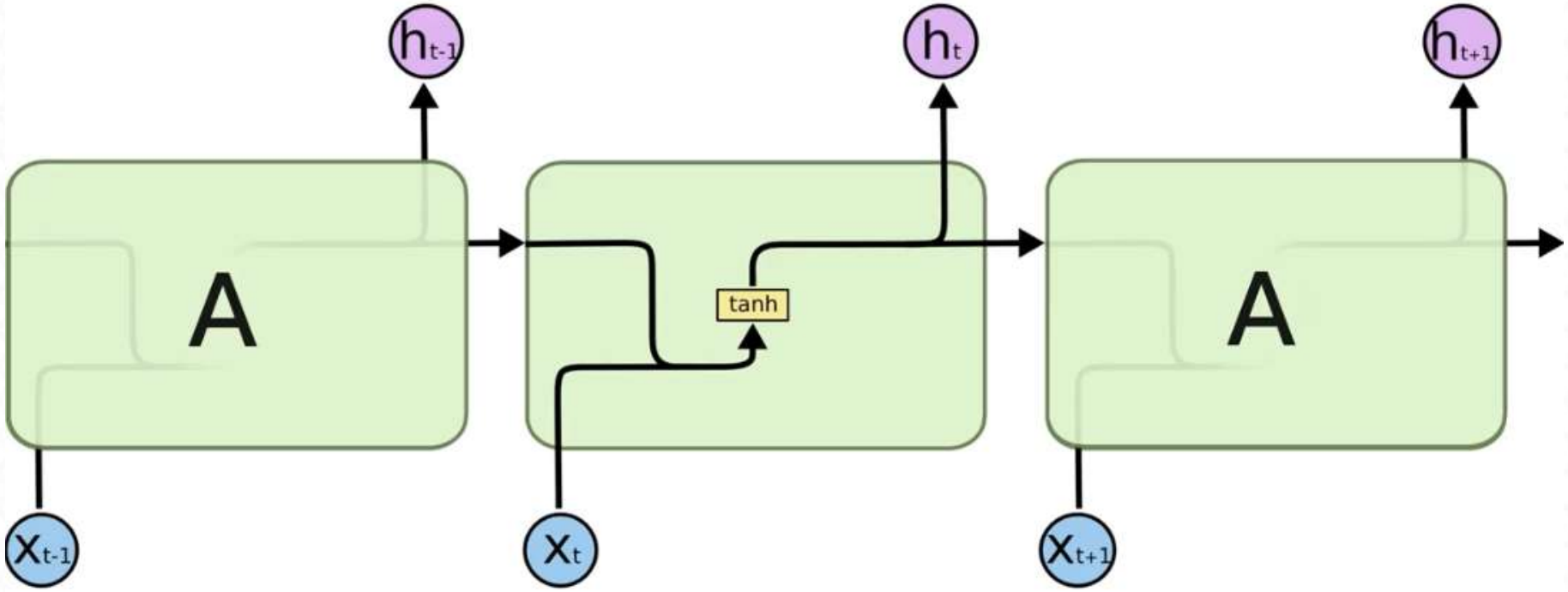
$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

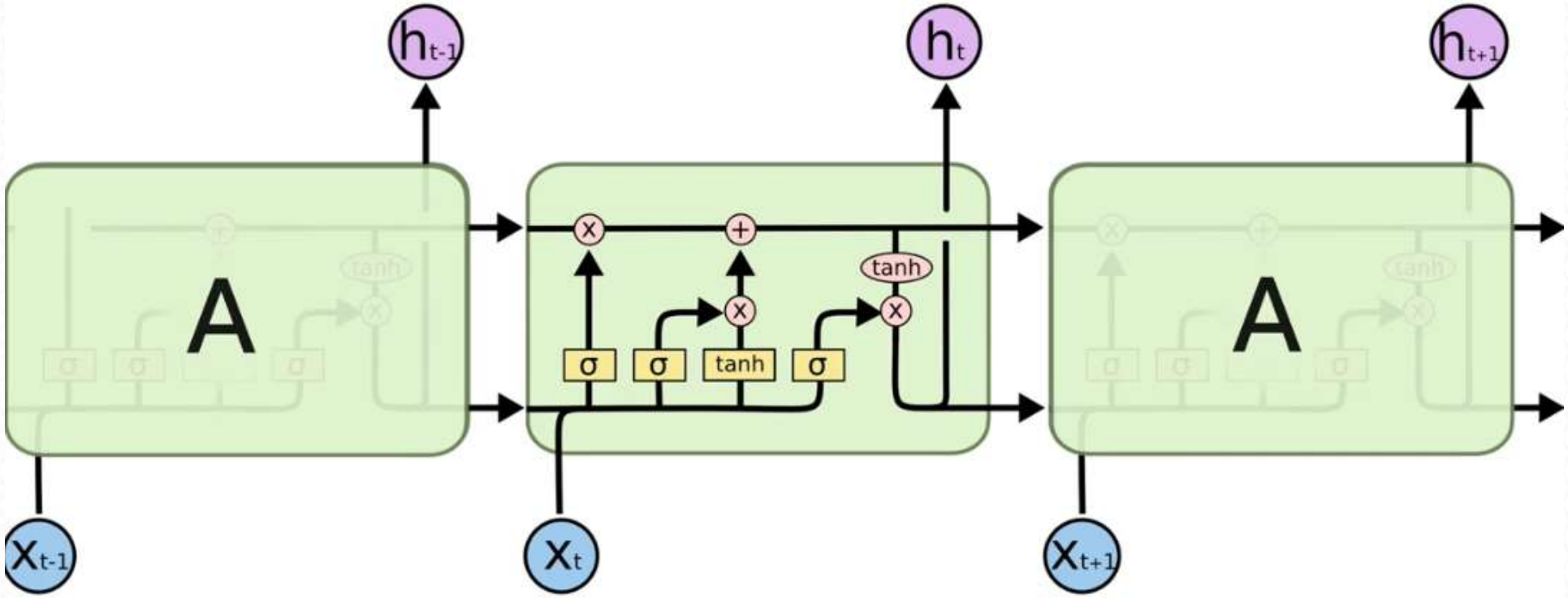
W_{rec} ~ small ⇨ Vanishing

W_{rec} ~ large ⇨ Exploding

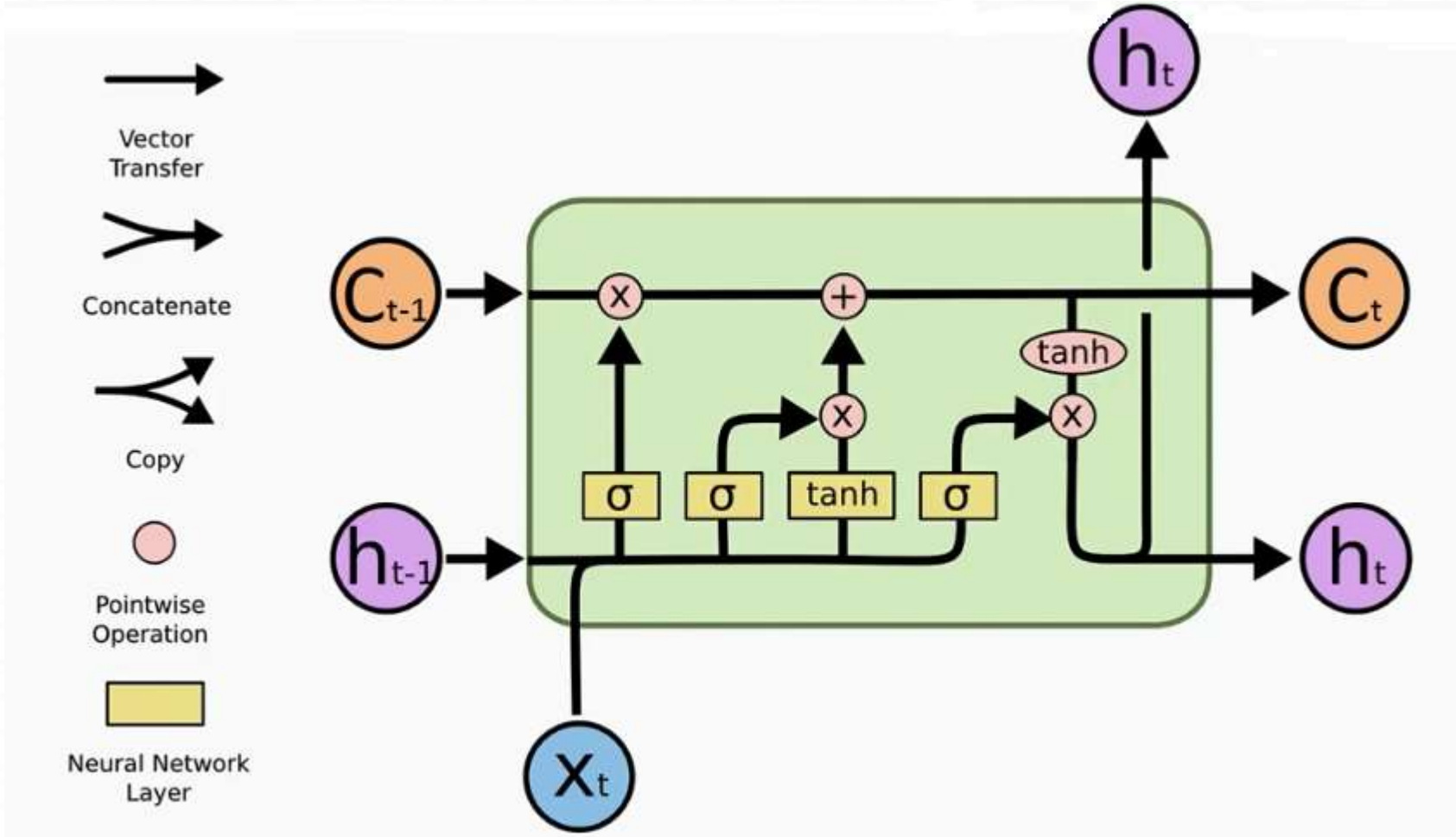
Long Short-Term Memory



Long Short-Term Memory

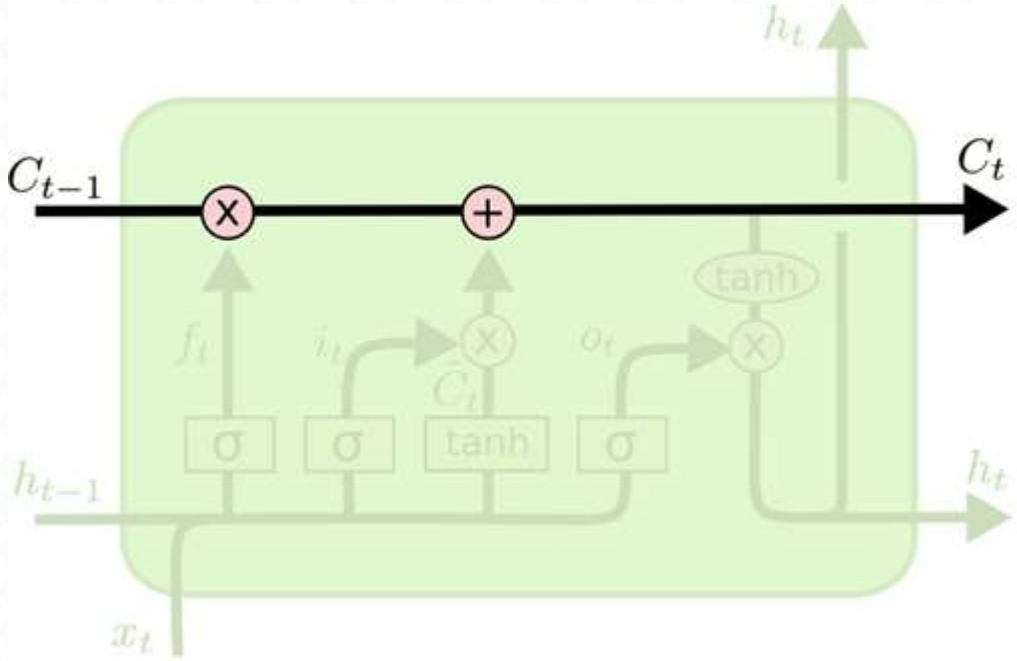


Long Short-Term Memory

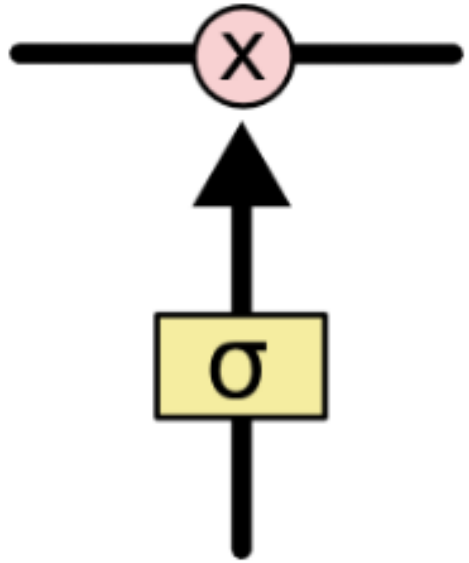


Long Short-Term Memory 詳解

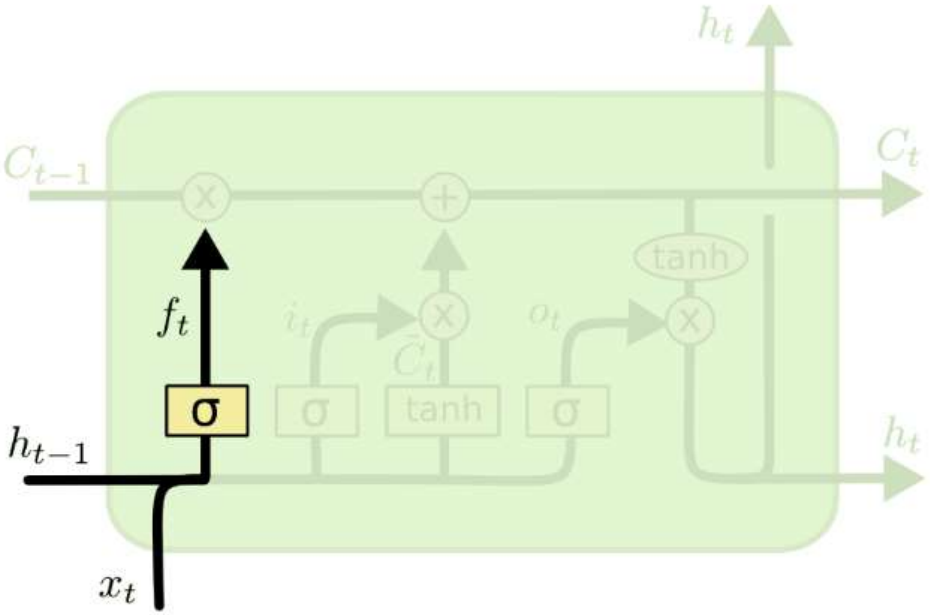
RNN 傳遞過程



Sigmoid 閘門

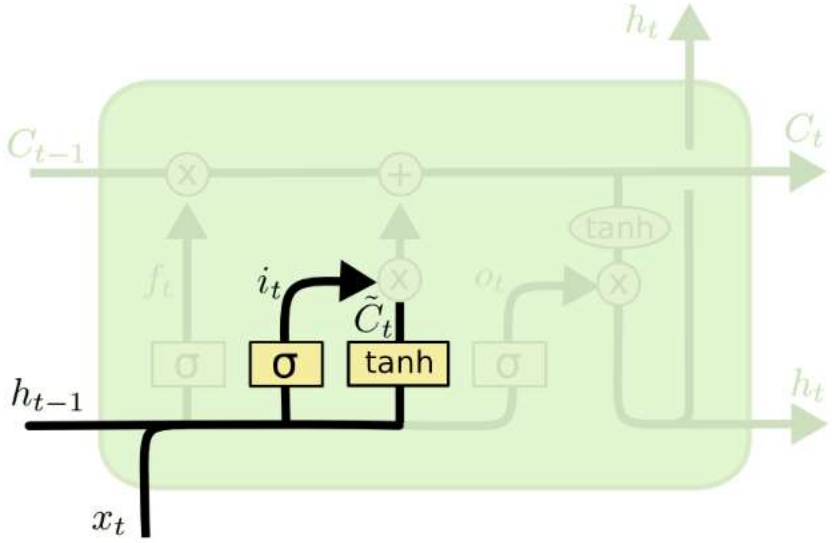


遺忘門 (Forget Gate)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

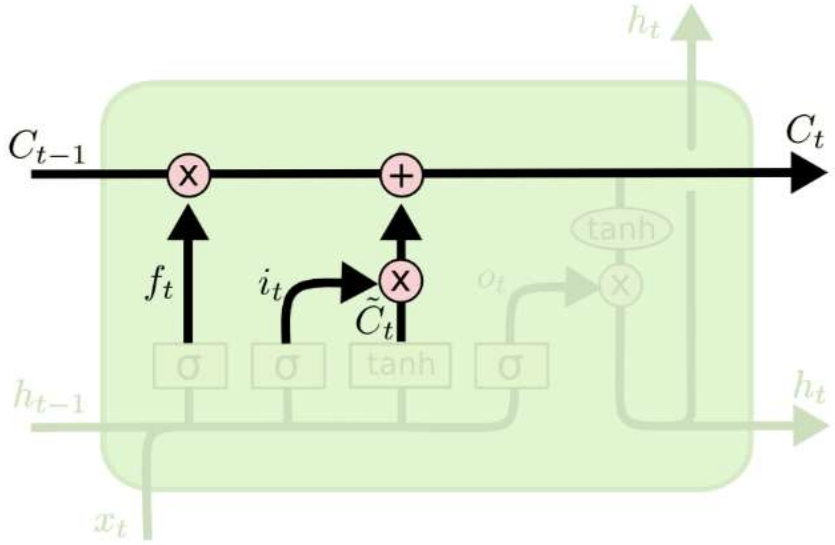
輸入門 (Input Gate)



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

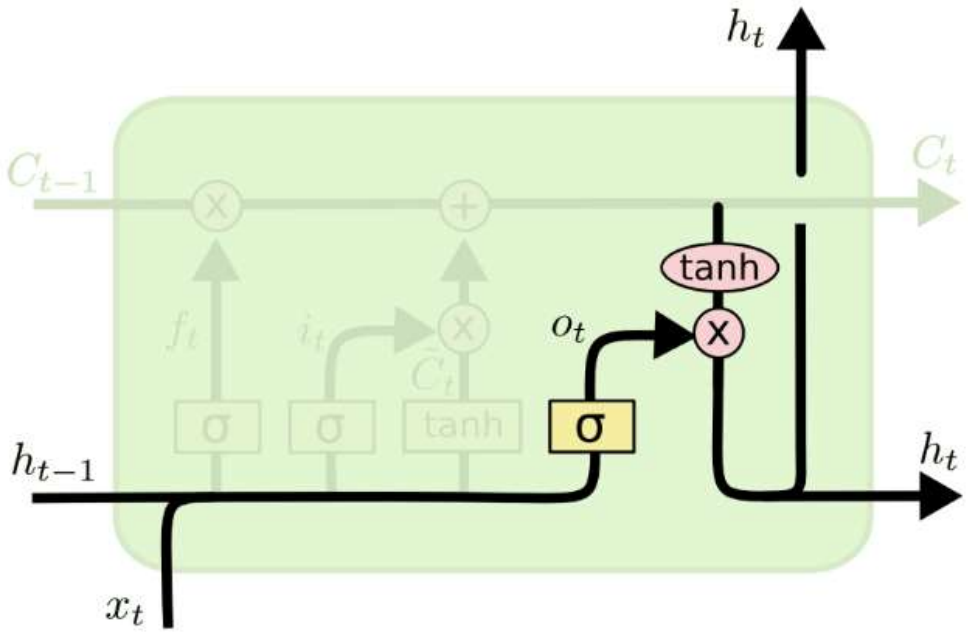
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

更新細胞狀態 (Cell State)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

輸出門 (Output Gate)



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

[實例] 股價預測

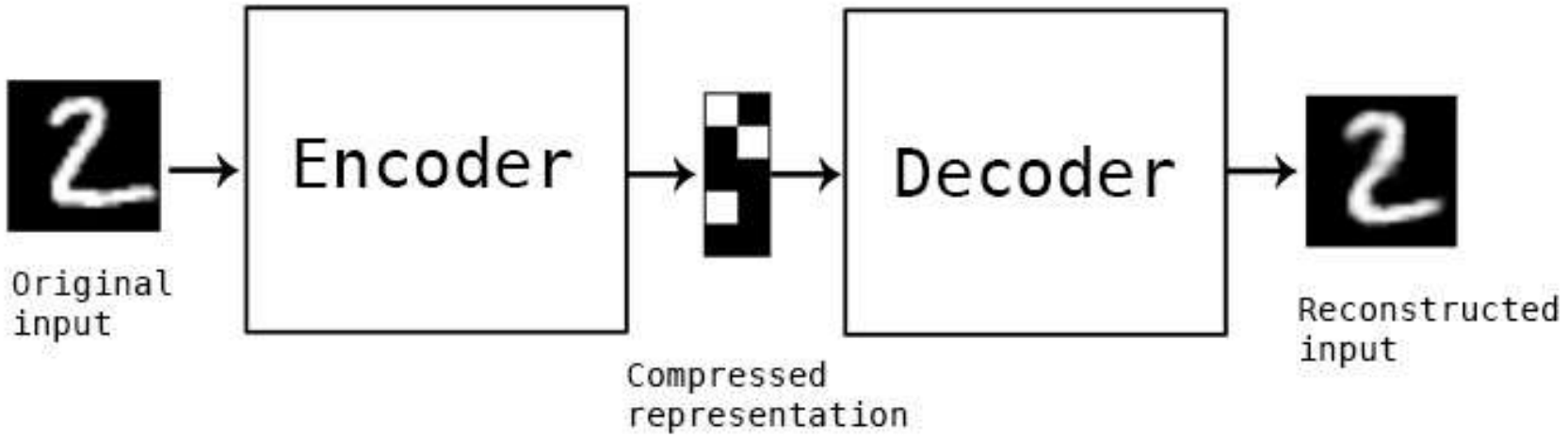
股價預測


GOOG (Google, Inc.) Nasdaq GS
29-Oct-2007 11:52am
© StockCharts.com
Open 677.77 High 679.51 Low 672.09 Last 676.19 Volume 1.6M Chg +1.59 (+0.24%)
GOOG (Daily) 676.19
MA(50) 571.62
MA(20) 631.33
Price Gap 1
Price Gap 2
Price Gap 3
680
660
640
620
600
580
560
540
520
500
480
Jun 11 18 25 Jul 9 16 23 Aug 6 13 20 27 Sep 10 17 24 Oct 8 15 22 29
Volume 1.57M, EMA(60) 5.44M
15M
10M
5M
Jun 11 18 25 Jul 9 16 23 Aug 6 13 20 27 Sep 10 17 24 Oct 8 15 22 29

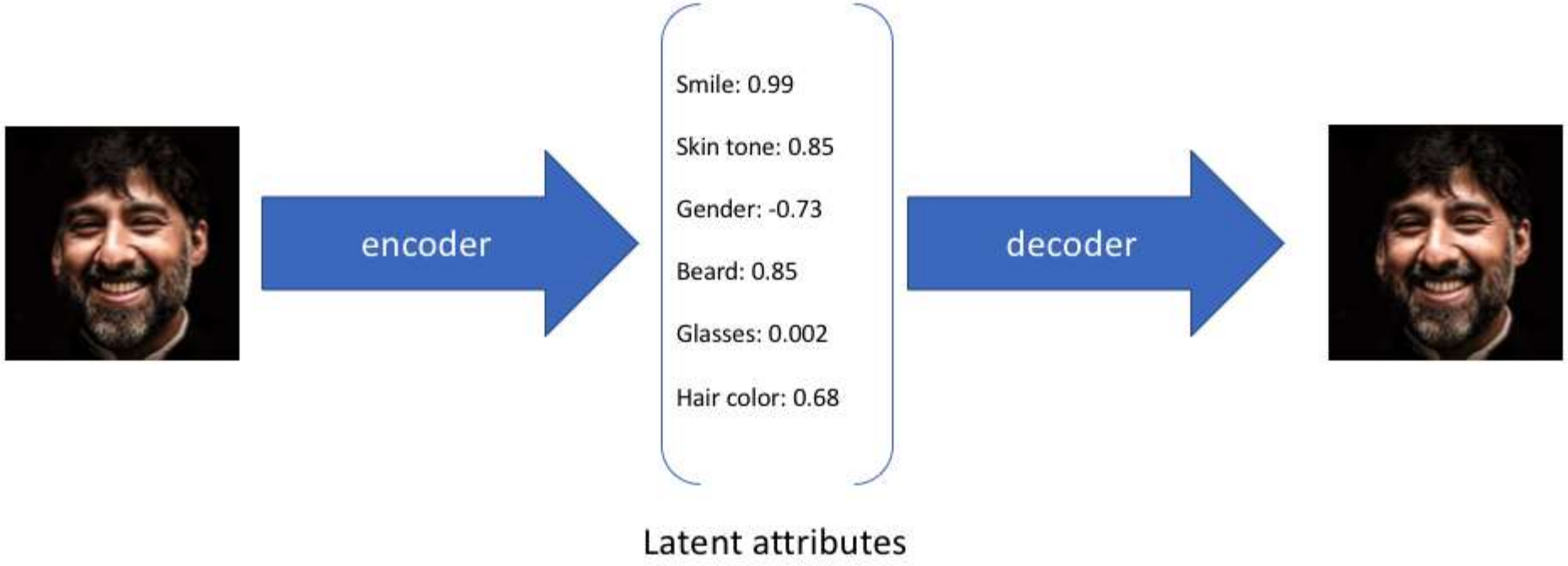
自編碼器
(Autoencoder)

什么是自编码器

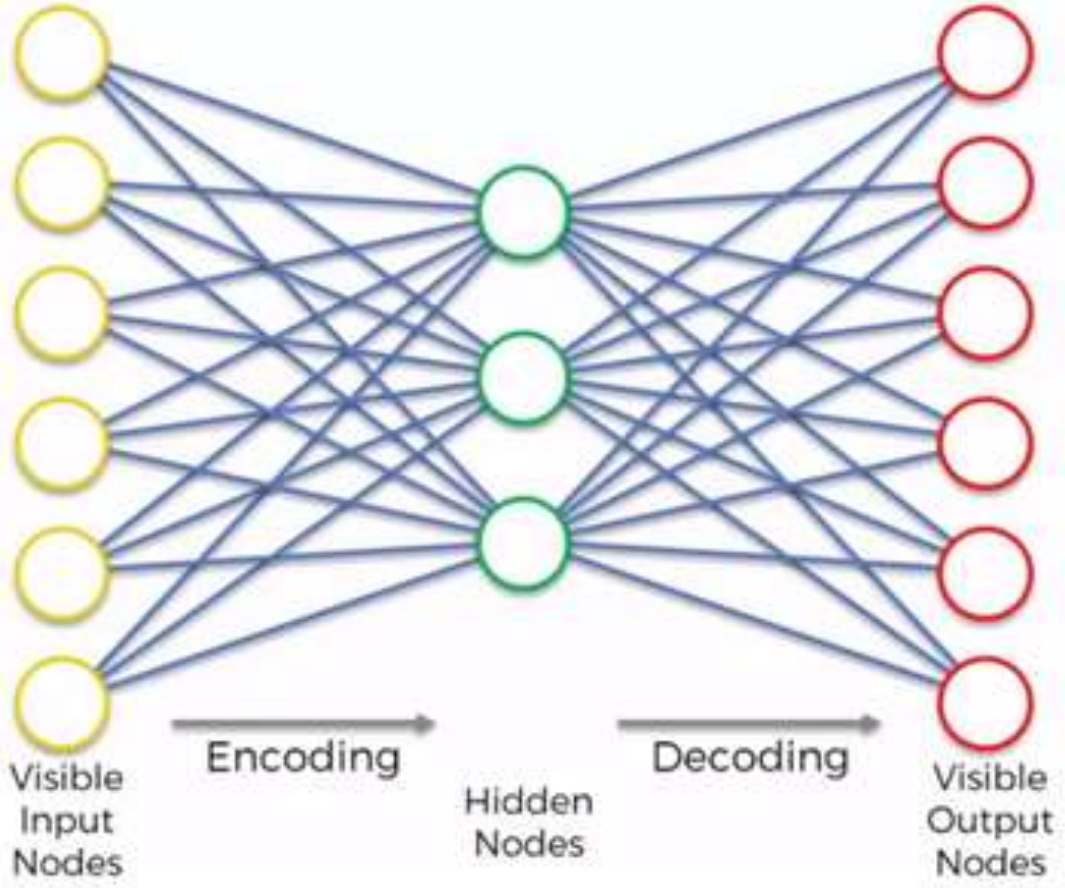
- 自编码器是一种压缩(compression)算法，包含以下特性：
  - 数据相关 (Data-specific)
  - 有损的 (Lossy)
  - 从样本中自动学习(Learned automatically from examples)



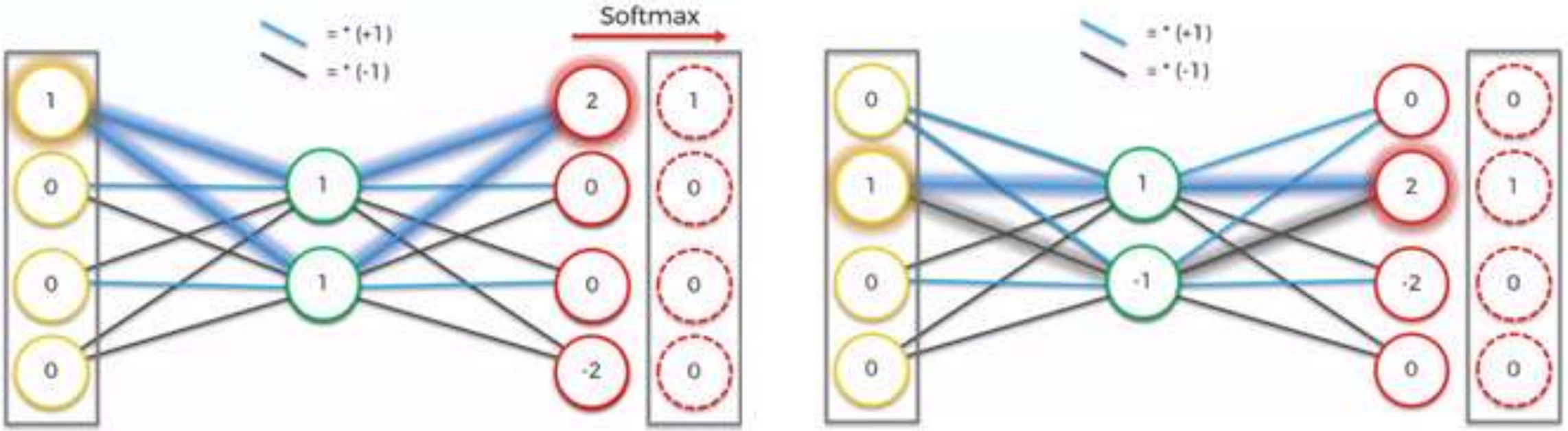
Autoencoders



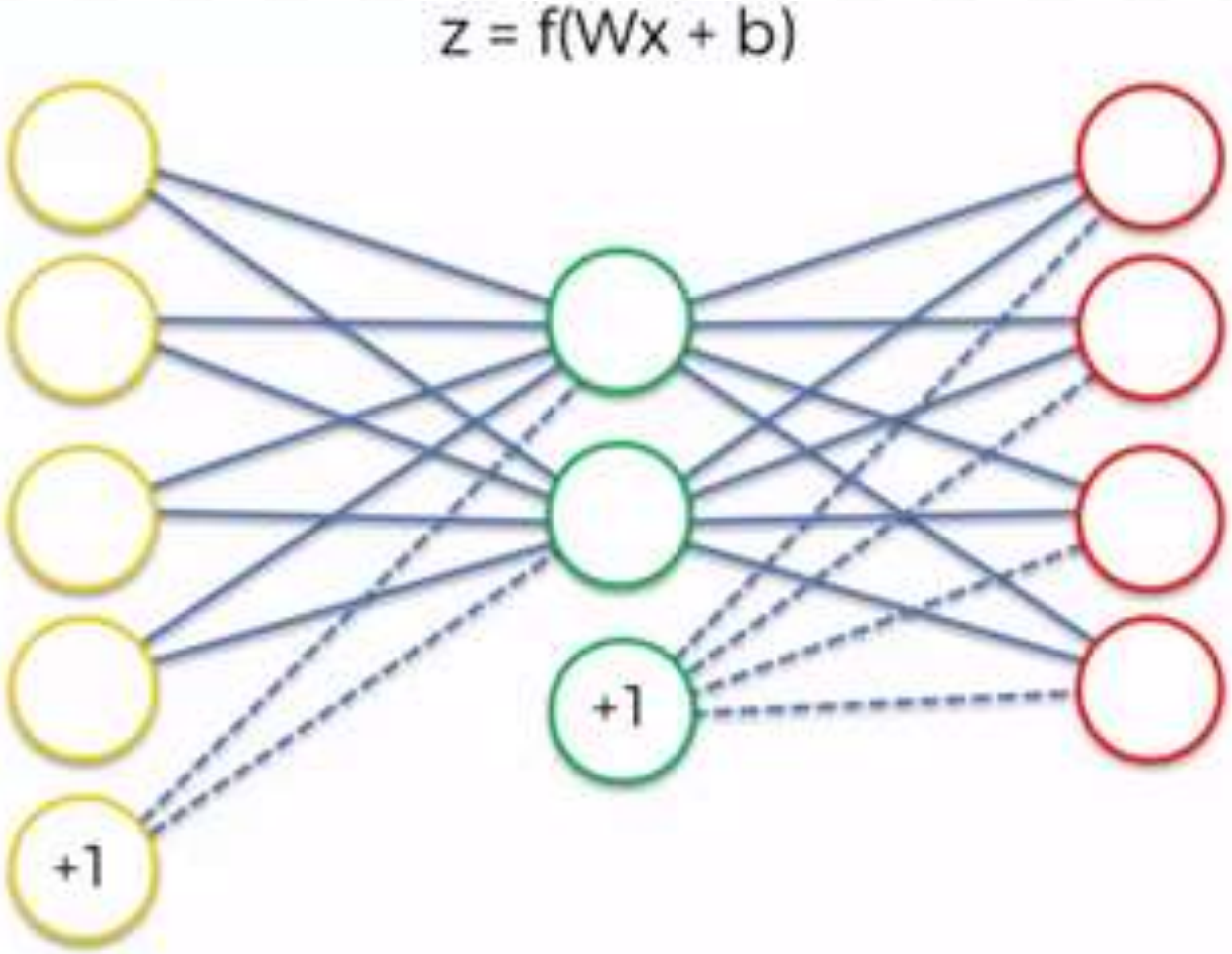
Autoencoder



Autoencoder



增加偏倚 (Bias)



訓練自編碼器(Autoencoder)

STEP 1

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

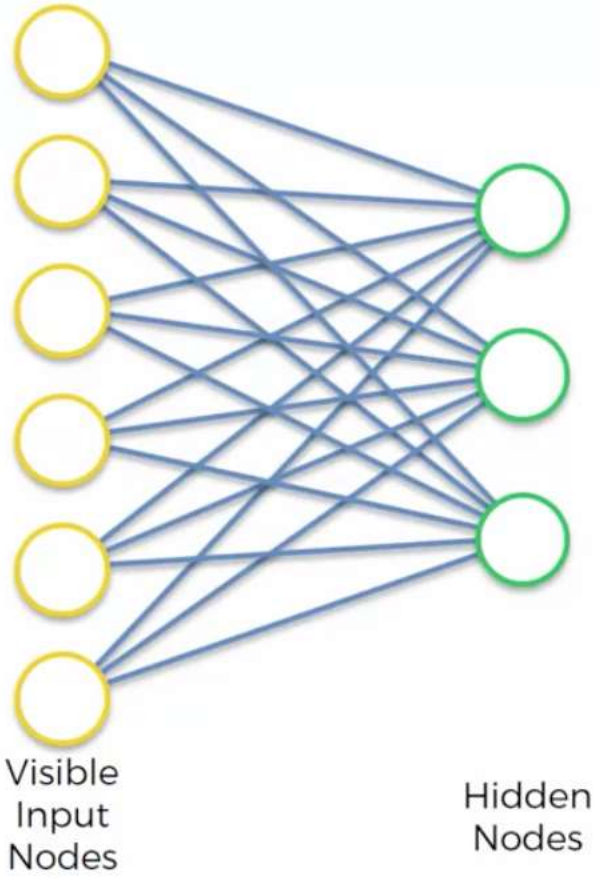
STEP 2

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

訓練自編碼器(Autoencoder)

STEP 3

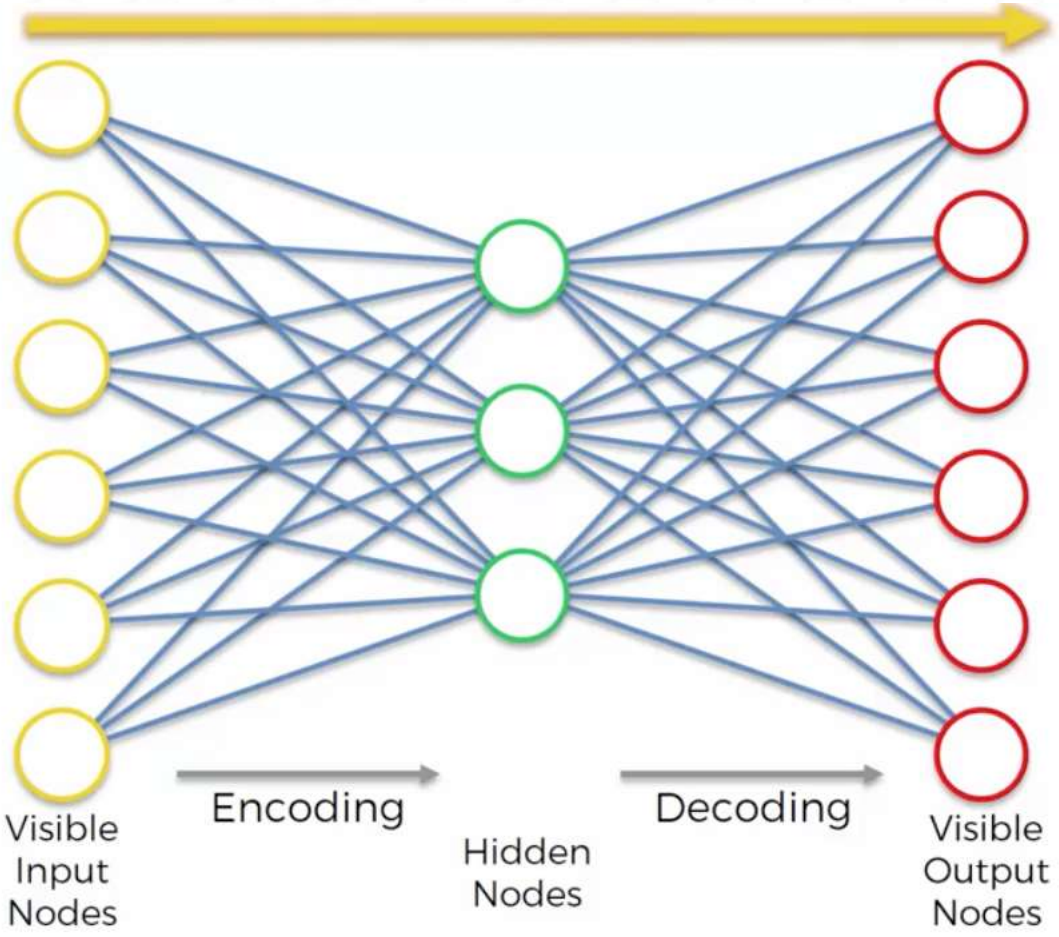
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



訓練自編碼器(Autoencoder)

STEP 4

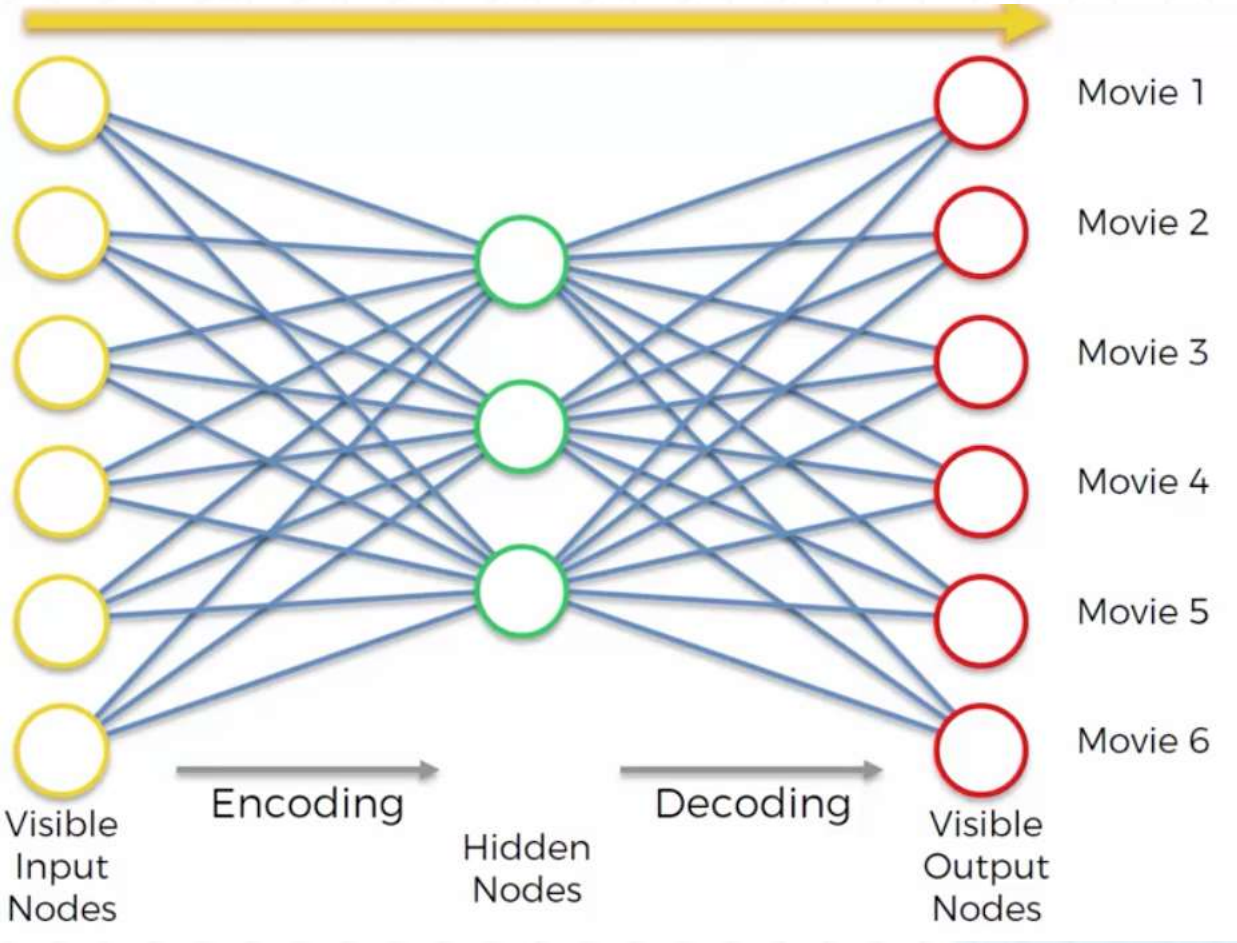
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



訓練自編碼器(Autoencoder)

STEP 5

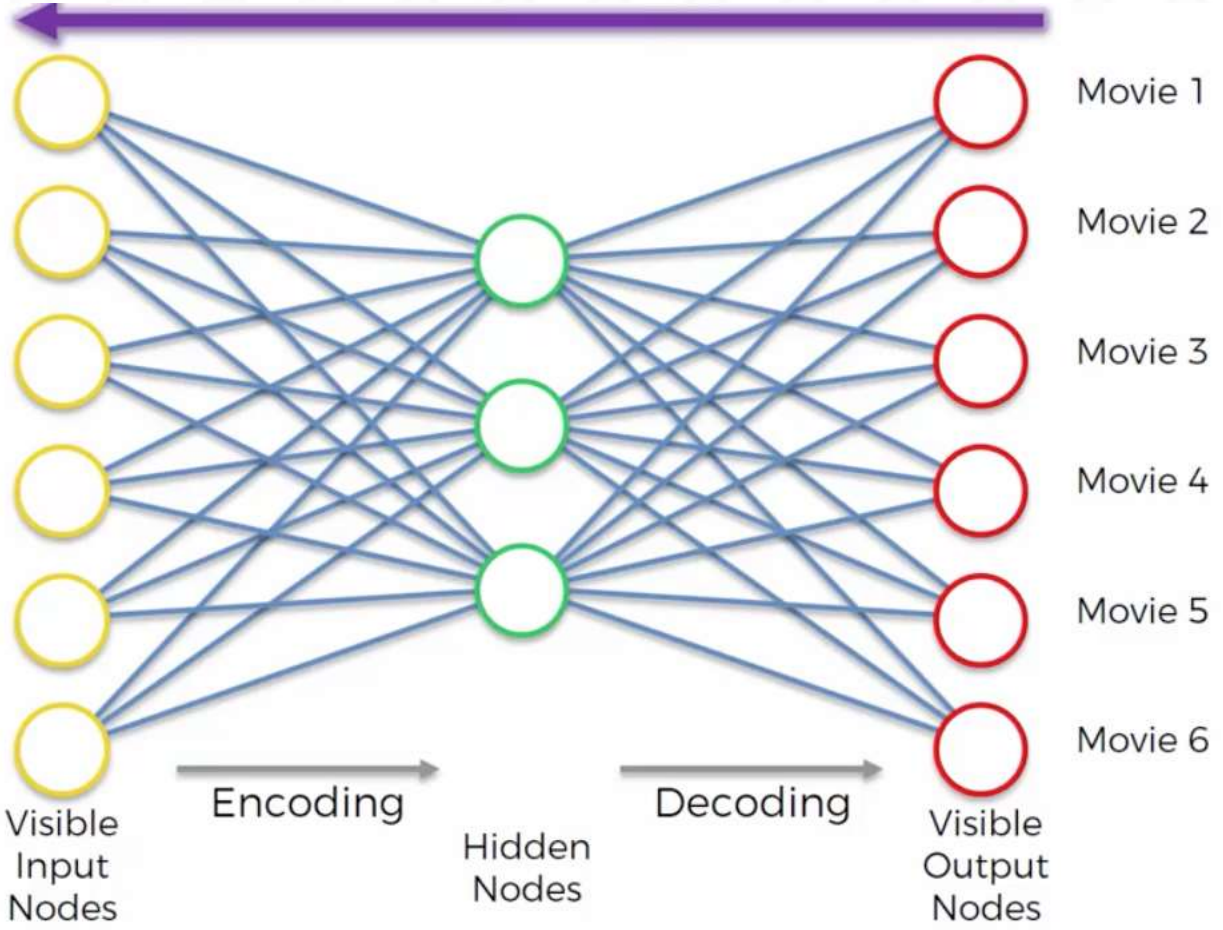
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



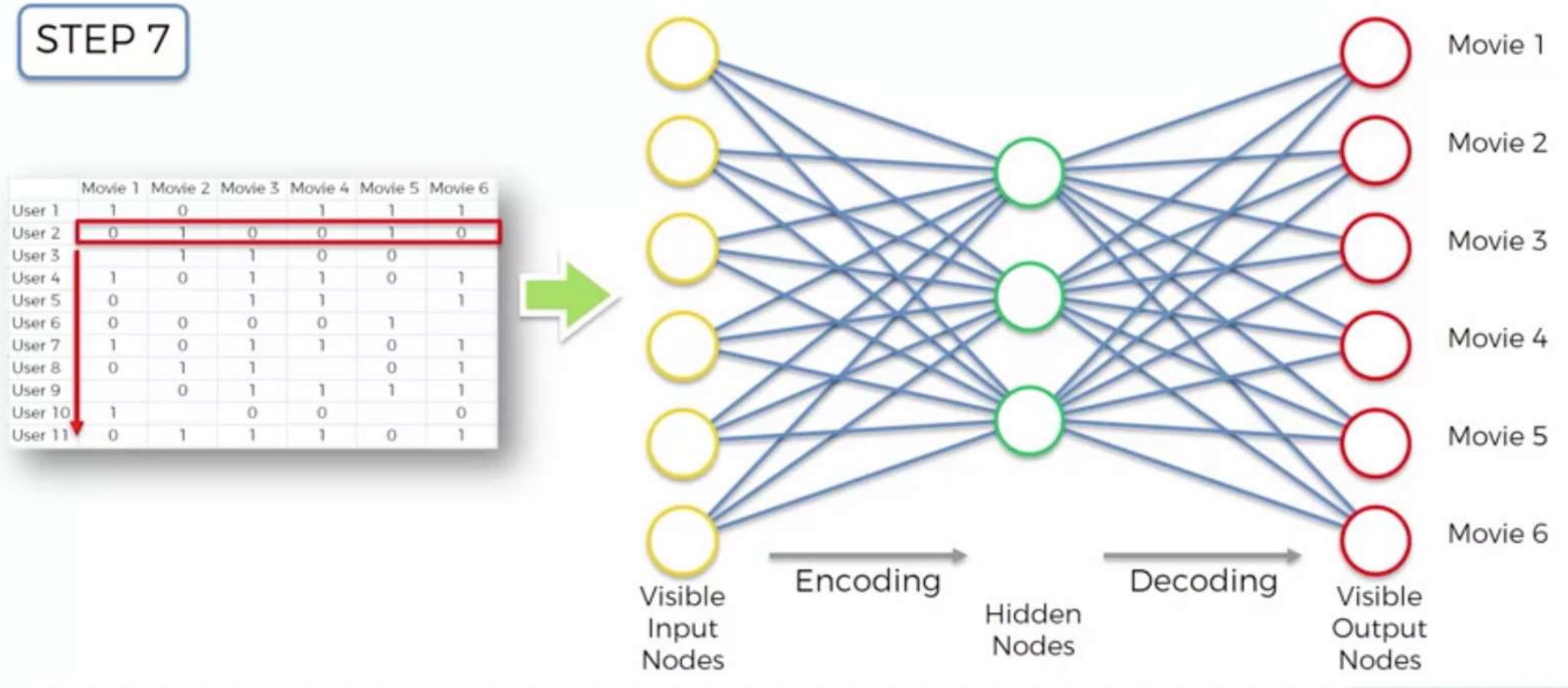
訓練自編碼器(Autoencoder)

STEP 6

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



訓練自編碼器(Autoencoder)

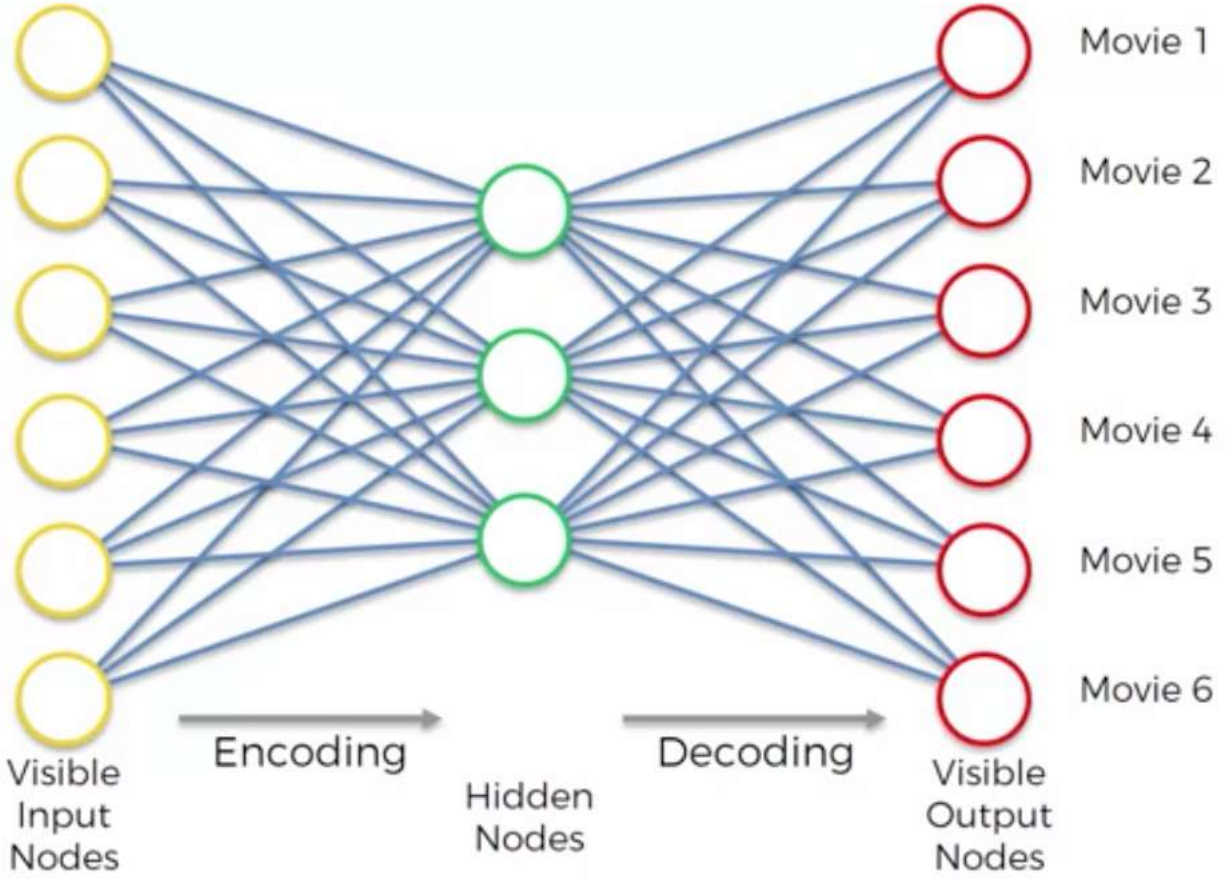


訓練自編碼器(Autoencoder)

STEP 8

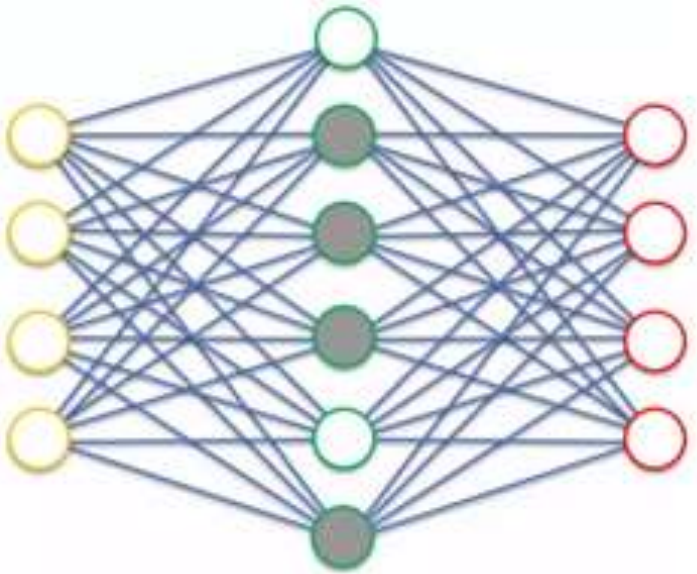
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

Repeat



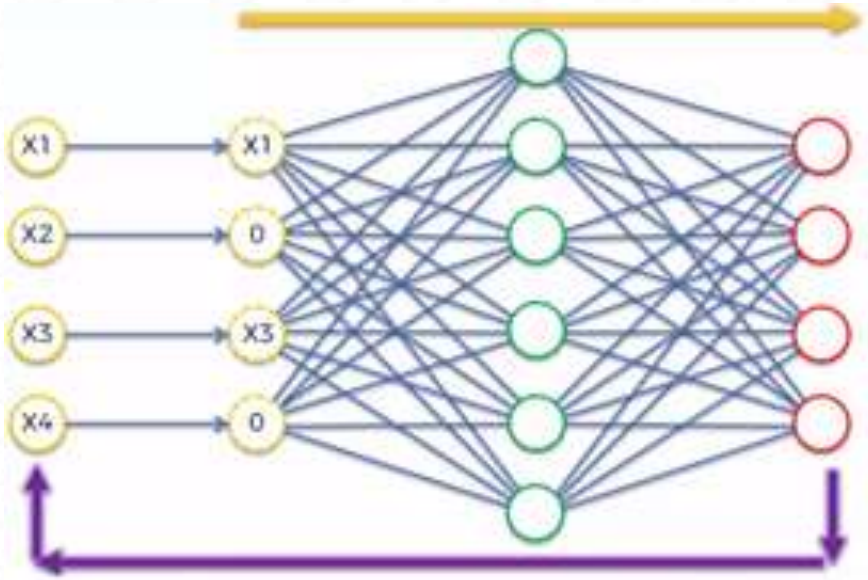
Sparse Autoencoders

- 稀疏自編碼器的隱藏層神經元大於輸入層神經元



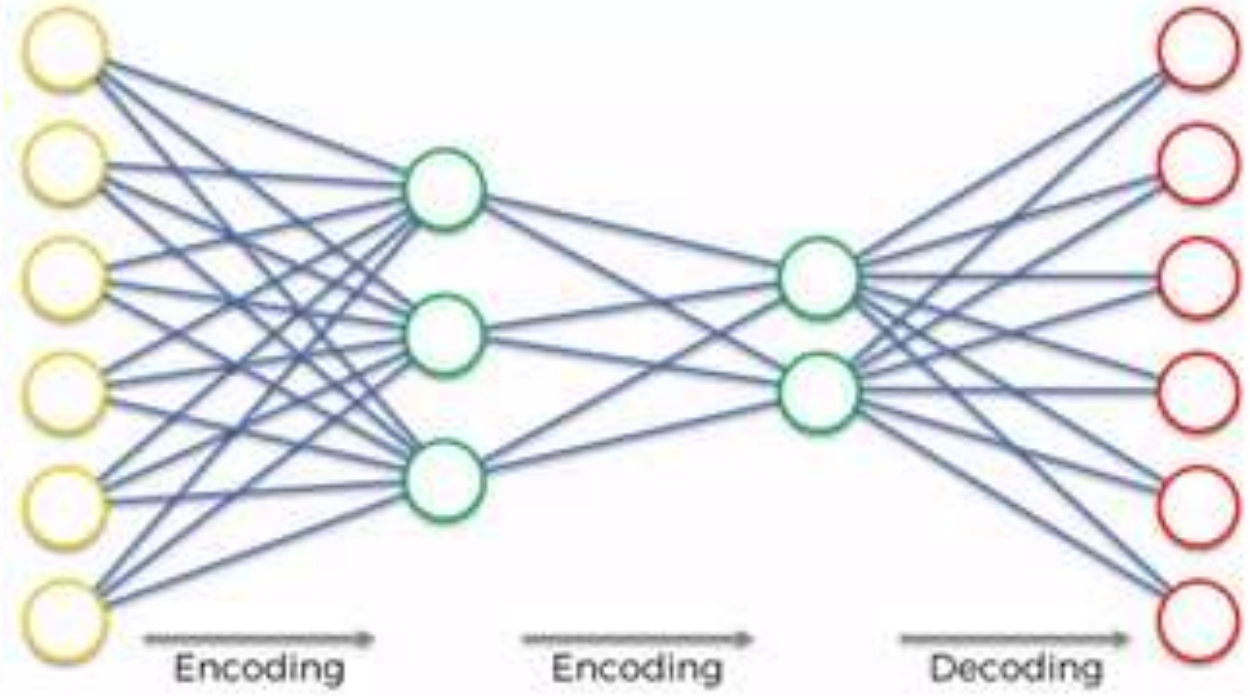
Denoising Autoencoders

- 將輸入層替代為更新後的值 ,會隨機將部分的值變為 0



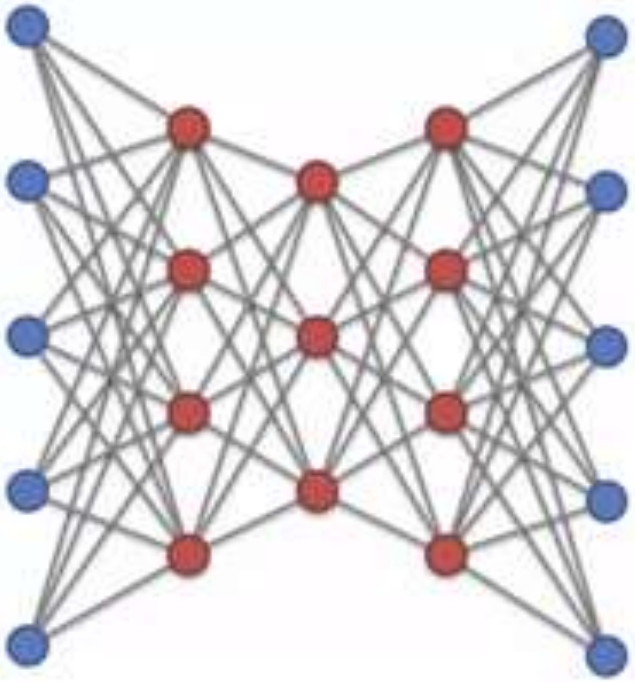
Stacked Autoencoders

- 有一層以上隱藏層的autoencoders



Deep autoencoders

- 使用預先訓練好的層堆疊而成的RBM



[實例] 推薦問題

推薦系統適用於

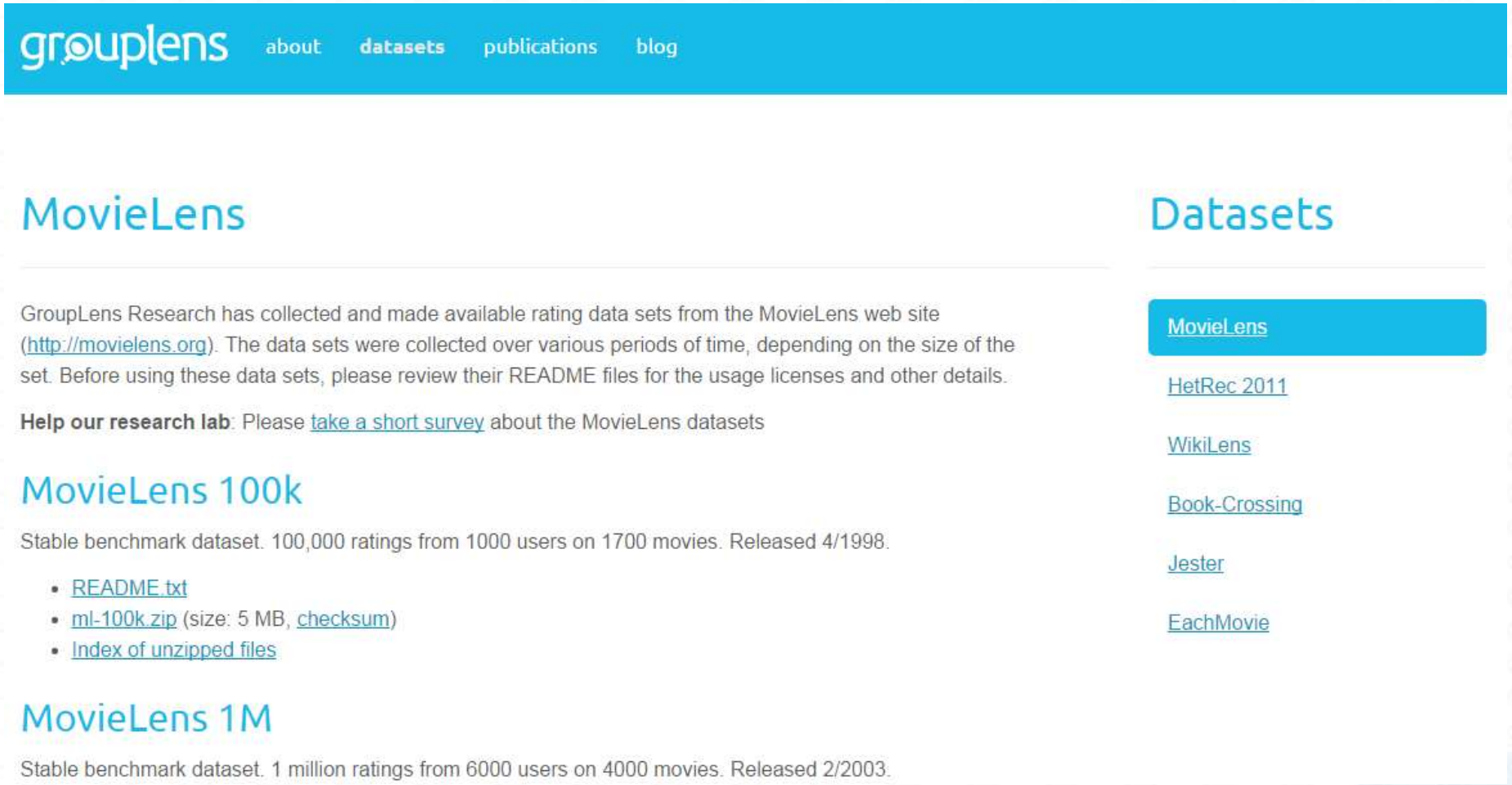
- 使用者需在眾多的商品選項做取捨:
  - 使用者難以從眾多選項中找他們所想要的，雖然商品搜尋可能可以滿足需求，但如何提供使用者未知卻需要的商品？
- 依個人行為與品味做推薦
  - 如果可以找到偏好相似的使用者，便可透過偏好的相似性進行商品推薦

推薦模型

- 基於相似內容過濾
  - 基於商品內容的推薦 (文章標題、文章內容、影片特徵)
  - 可以針對使用者檔案所列的喜好做推薦

- 協同過濾法
  - 使用”群眾智慧”找出使用者偏好
  - 基於使用者的協同過濾：使用者對類似商品的偏好可推得他們有相同偏好
  - 基於商品的協同過濾：商品被類似使用者推薦時，代表性質類似
  - 類似最近鄰居法(Nearest Neighbor)

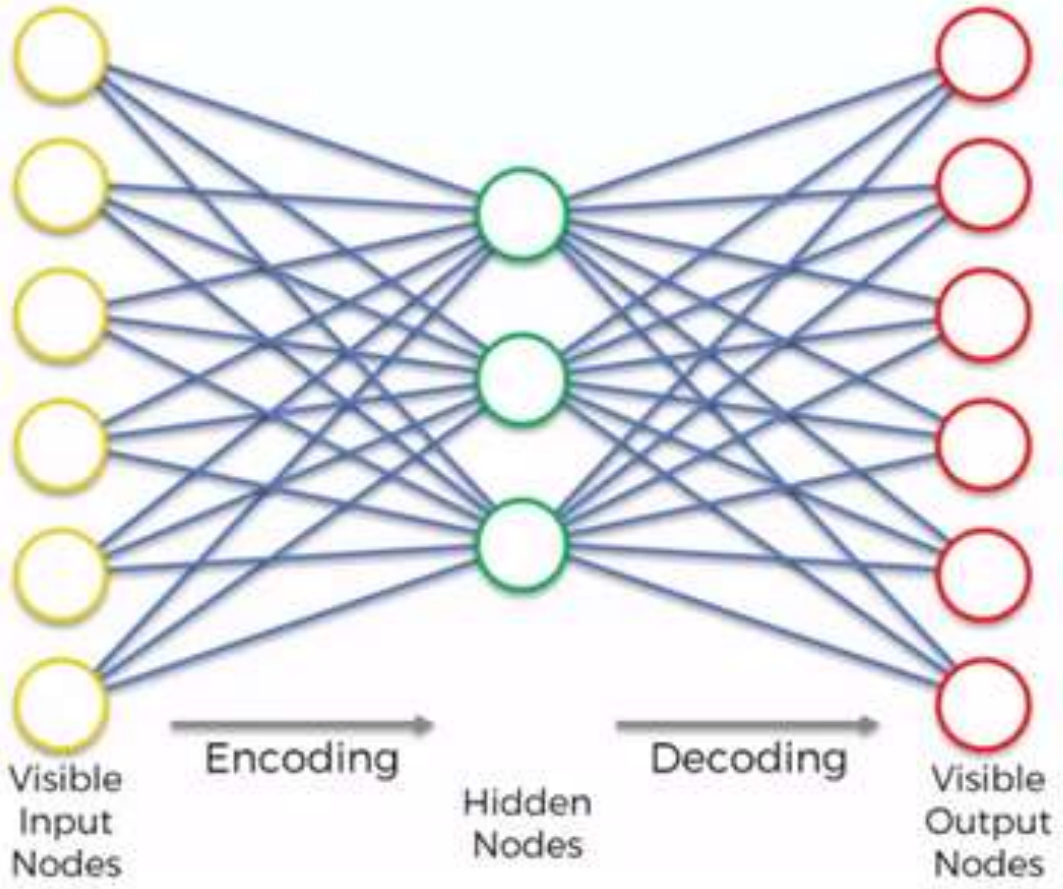
下載MovieLens 資料



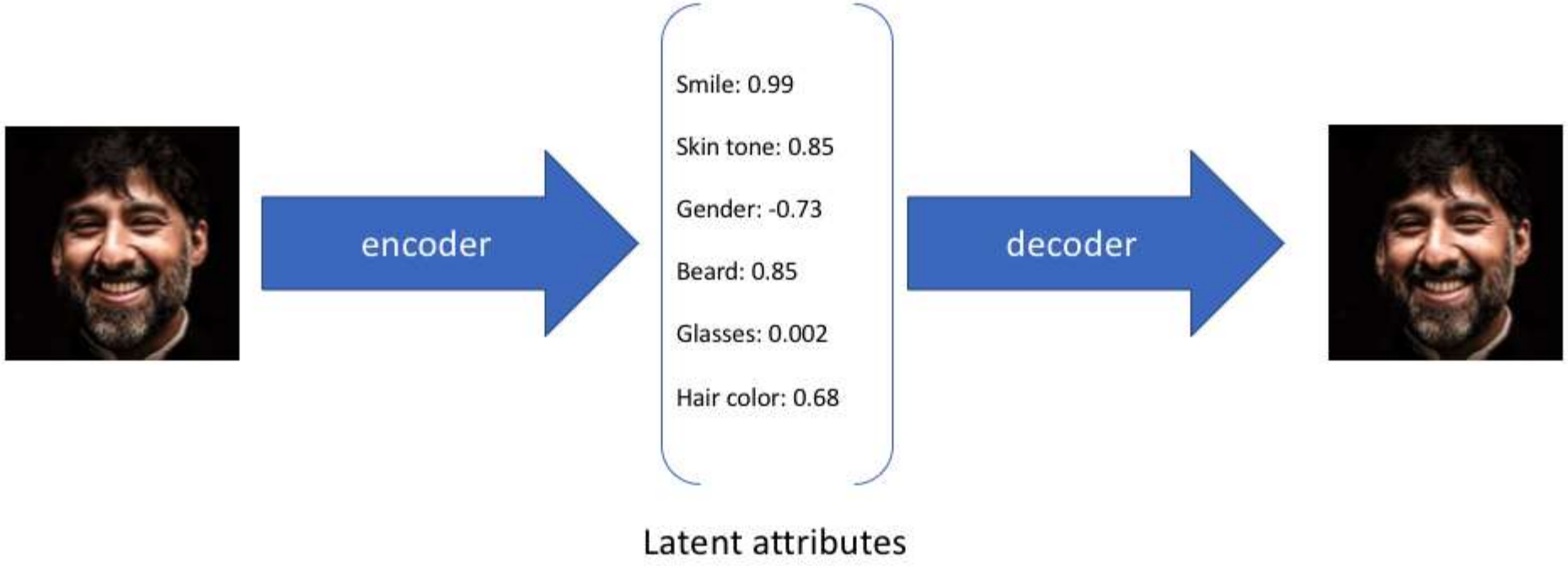
變分自編碼器
(Variational Autoencoder)

Autoencoder

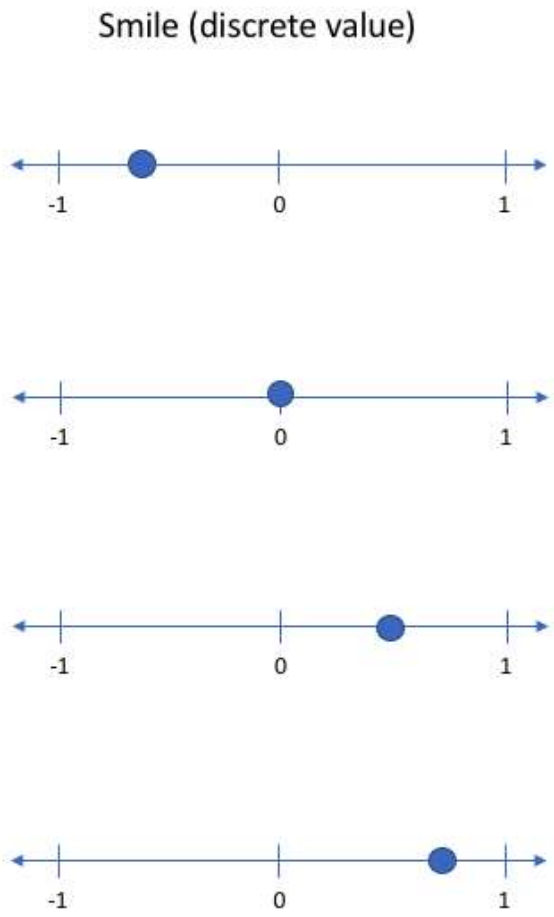
- Autoencoder 的隱藏層比輸入層的神經元來的少
- 隱藏層可以被稱為隱變數（Latent Variable Models）



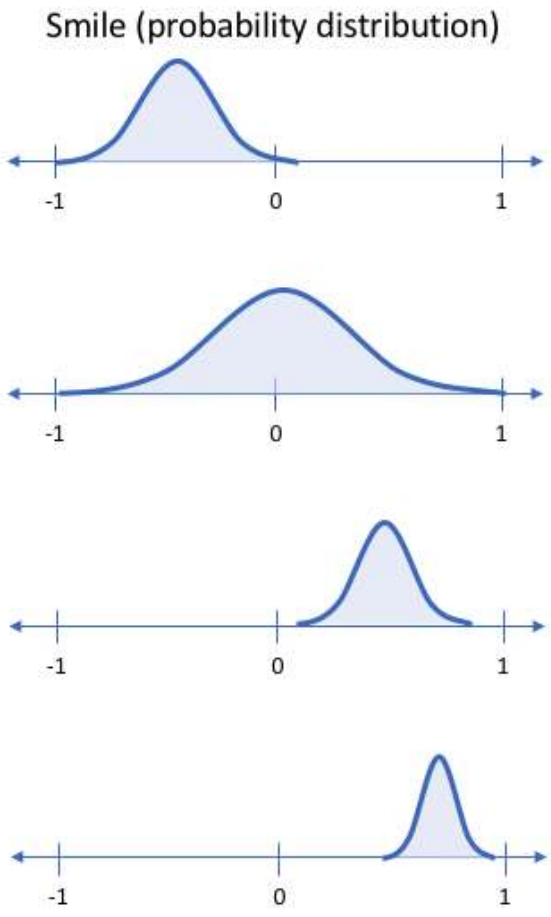
Autoencoders



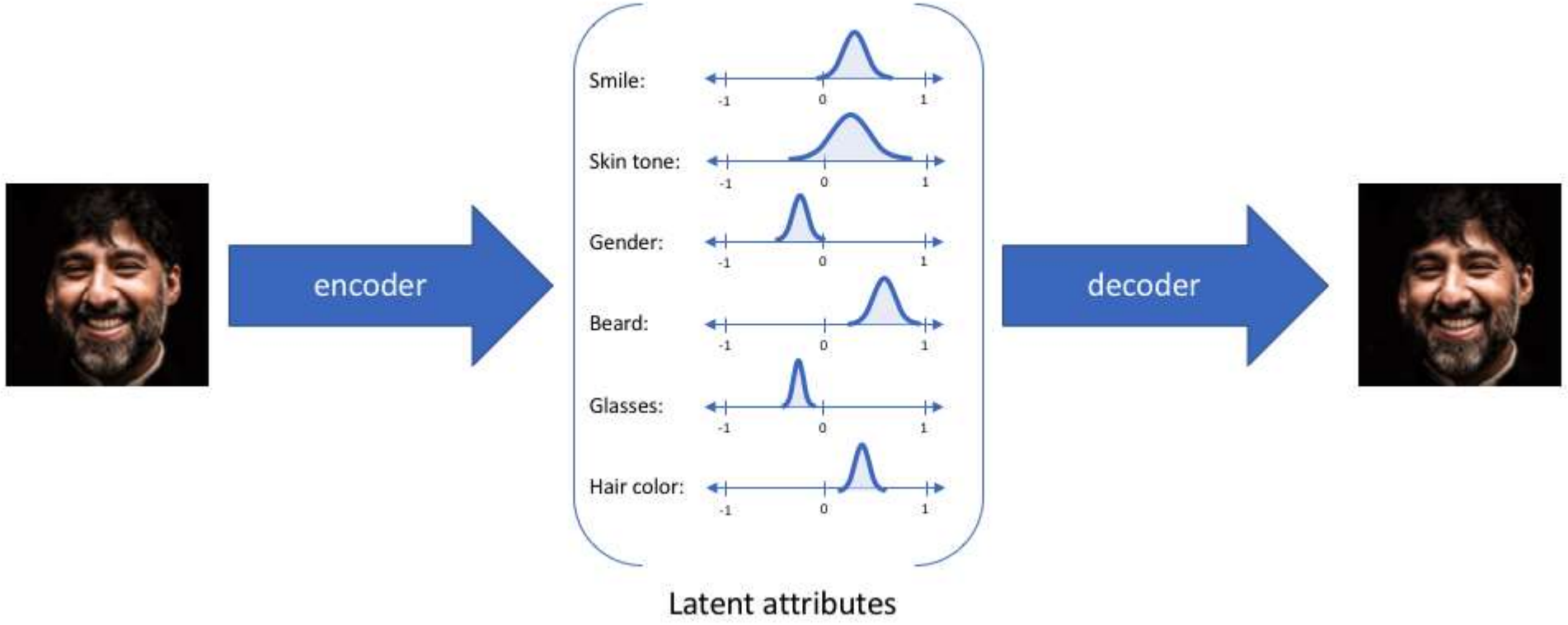
使用機率描述隱藏層



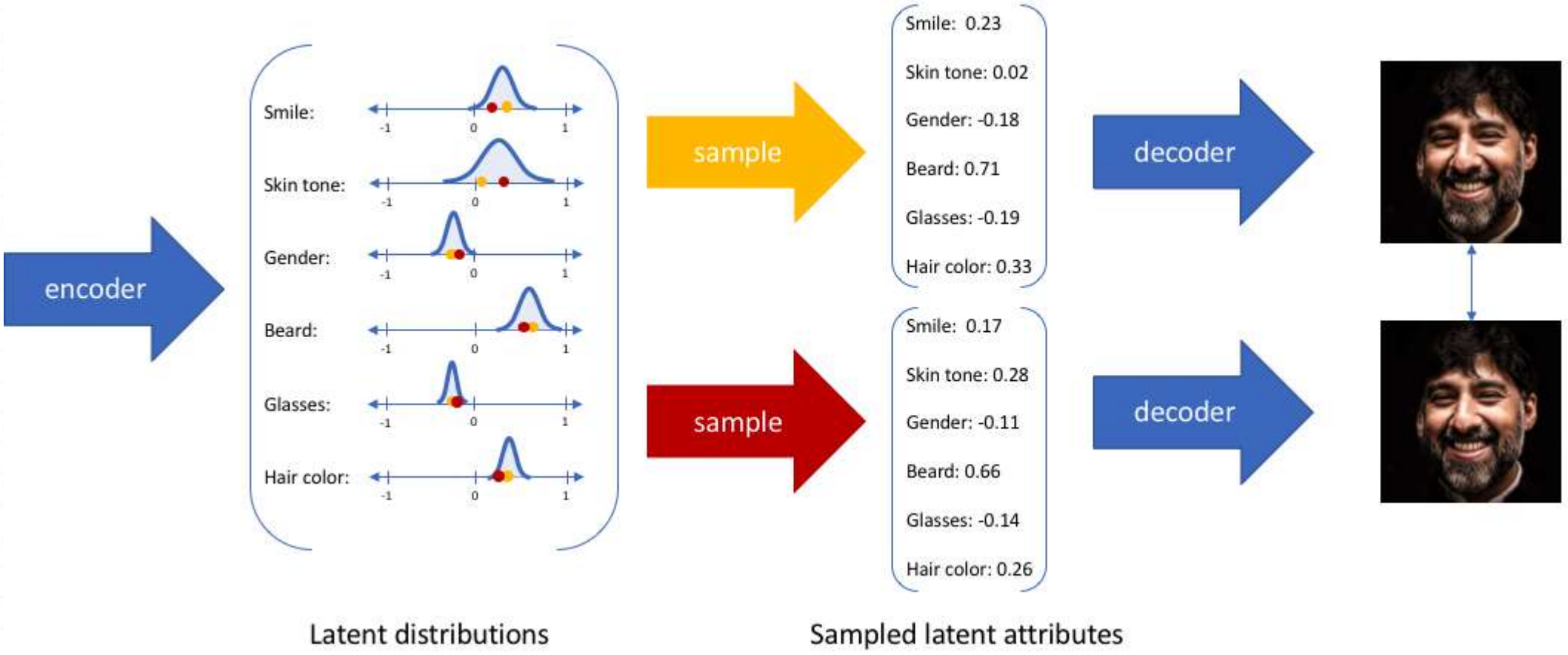
vs.



使用隱藏層的機率產生圖片



希望從隱藏層能重建出類似的圖片



Variation Inference

$$p(z|x) = \frac{p(x|z)\,p(z)}{p(x)}$$

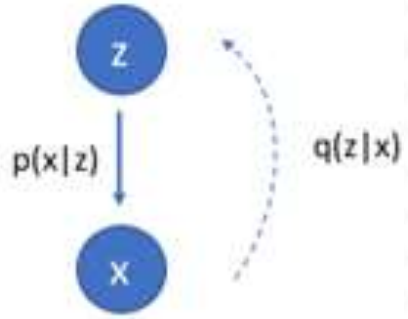
$$p(x) = \int p(x|z)\,p(z)\,dz$$

隱藏變數Z 可以產生X
但當我們只看見 X 時怎麼推估Z?

Minimize KL Divergence

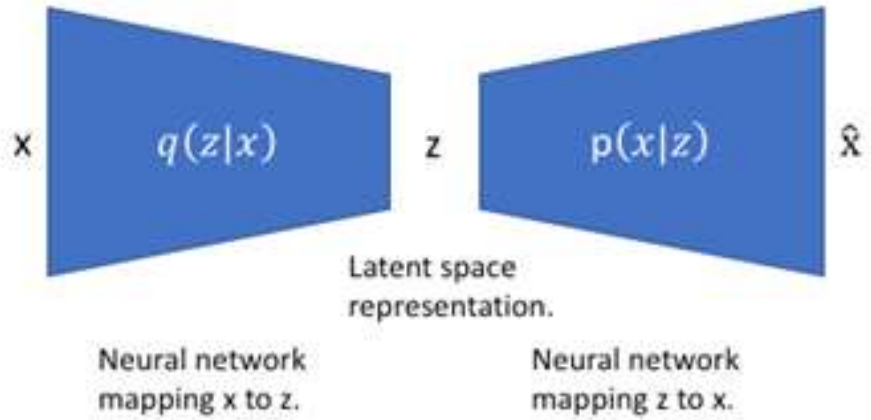
- 讓q(Z|X) 與 P(Z|X) 相似

$$\min KL\left(q\left(z|x\right)||p\left(z|x\right)\right)$$

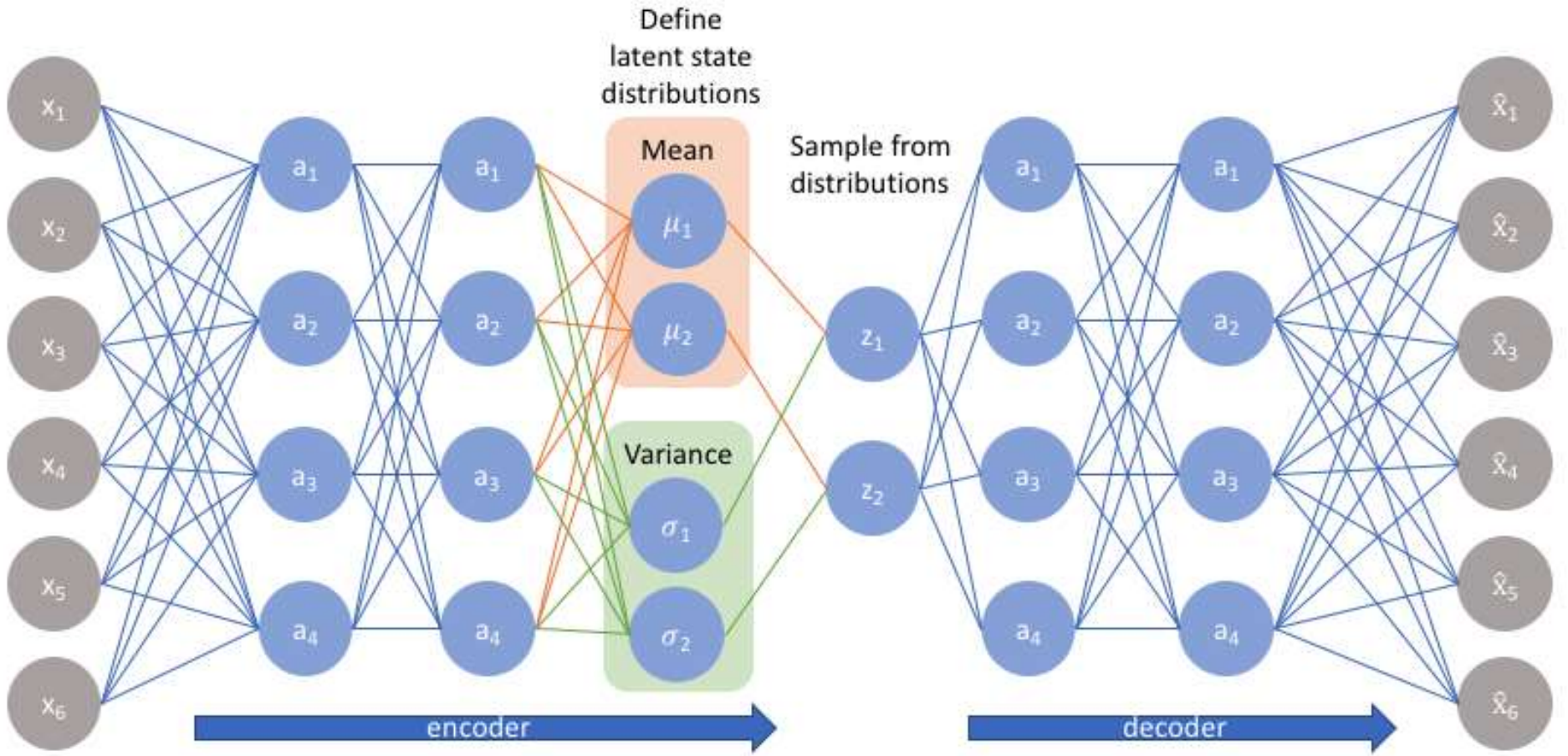


- 讓學習到的分布q 與 分布 p 相近

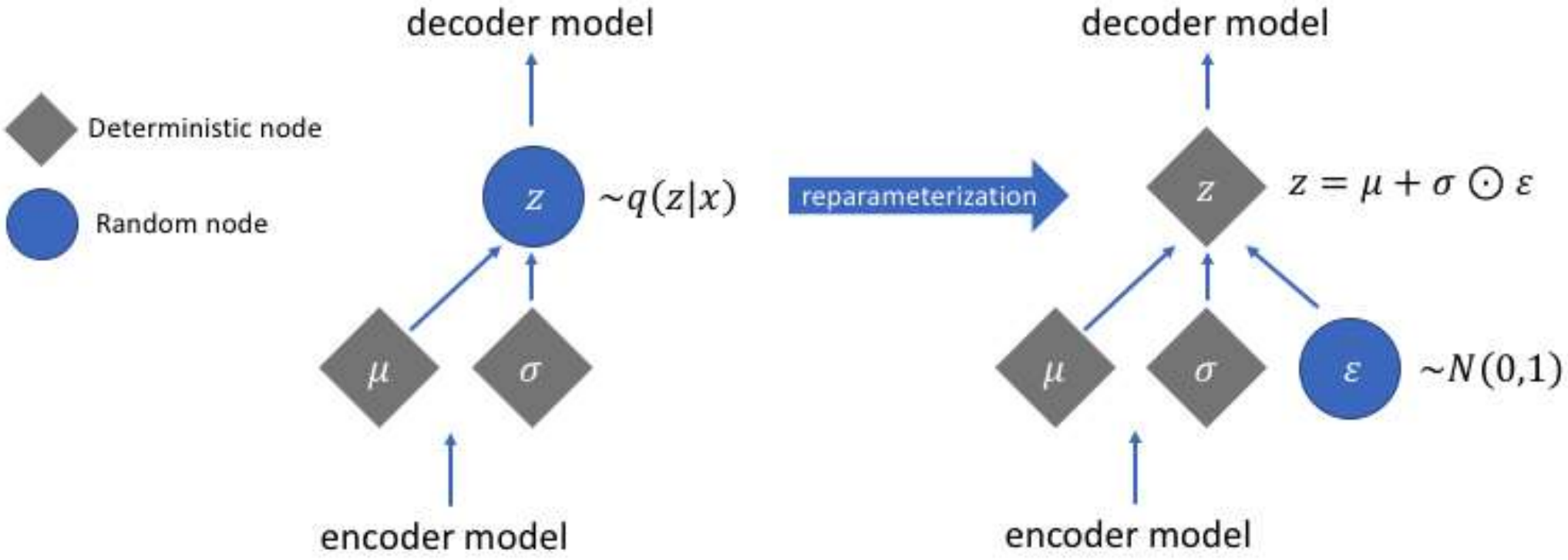
$$E_{q(z|x)}\log p\left(x|z\right) - KL\left(q\left(z|x\right)||p\left(z\right)\right)$$



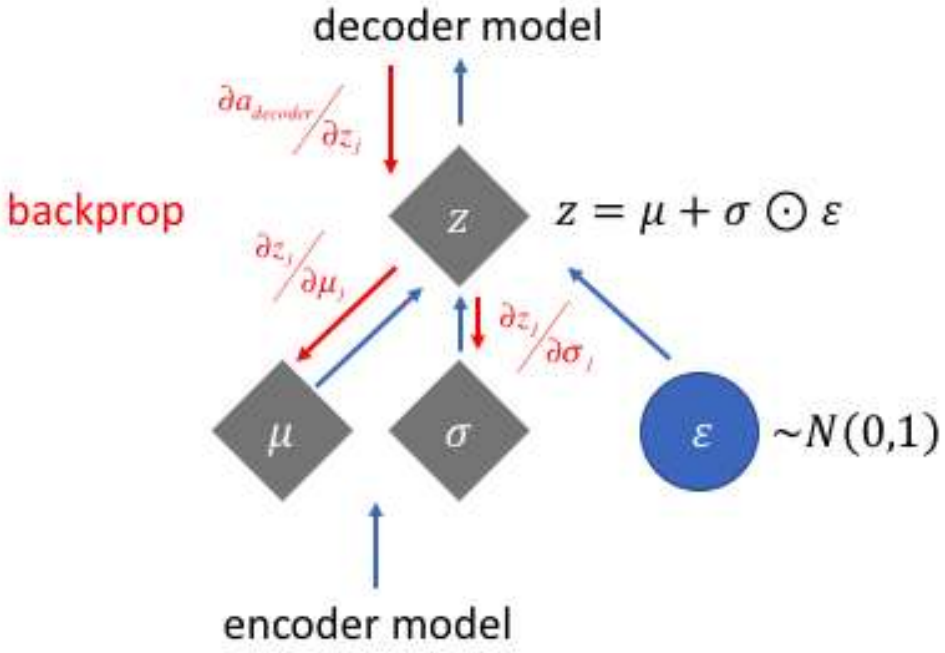
VAE 架構


Define
latent state
distributions
Mean
Variance
Sample from
distributions
encoder
decoder

reparameterization trick

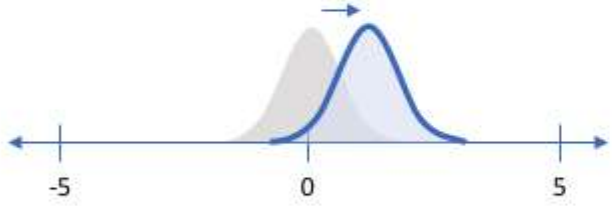


向後傳播


decoder model
backprop
$z = \mu + \sigma \odot \varepsilon$
$\sim N(0,1)$
encoder model

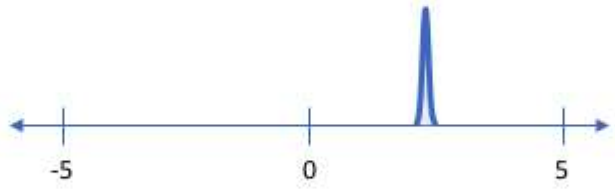
VAE 逼近過程

Penalizing reconstruction loss encourages the distribution to describe the input



Our distribution deviates from the prior to describe some characteristic of the data

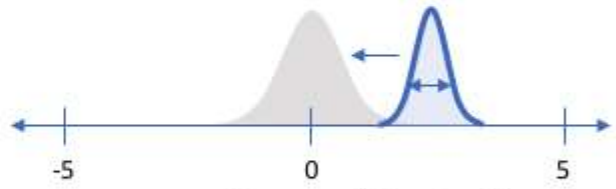
Without regularization, our network can "cheat" by learning narrow distributions



With a small enough variance, this distribution is effectively only representing a single value

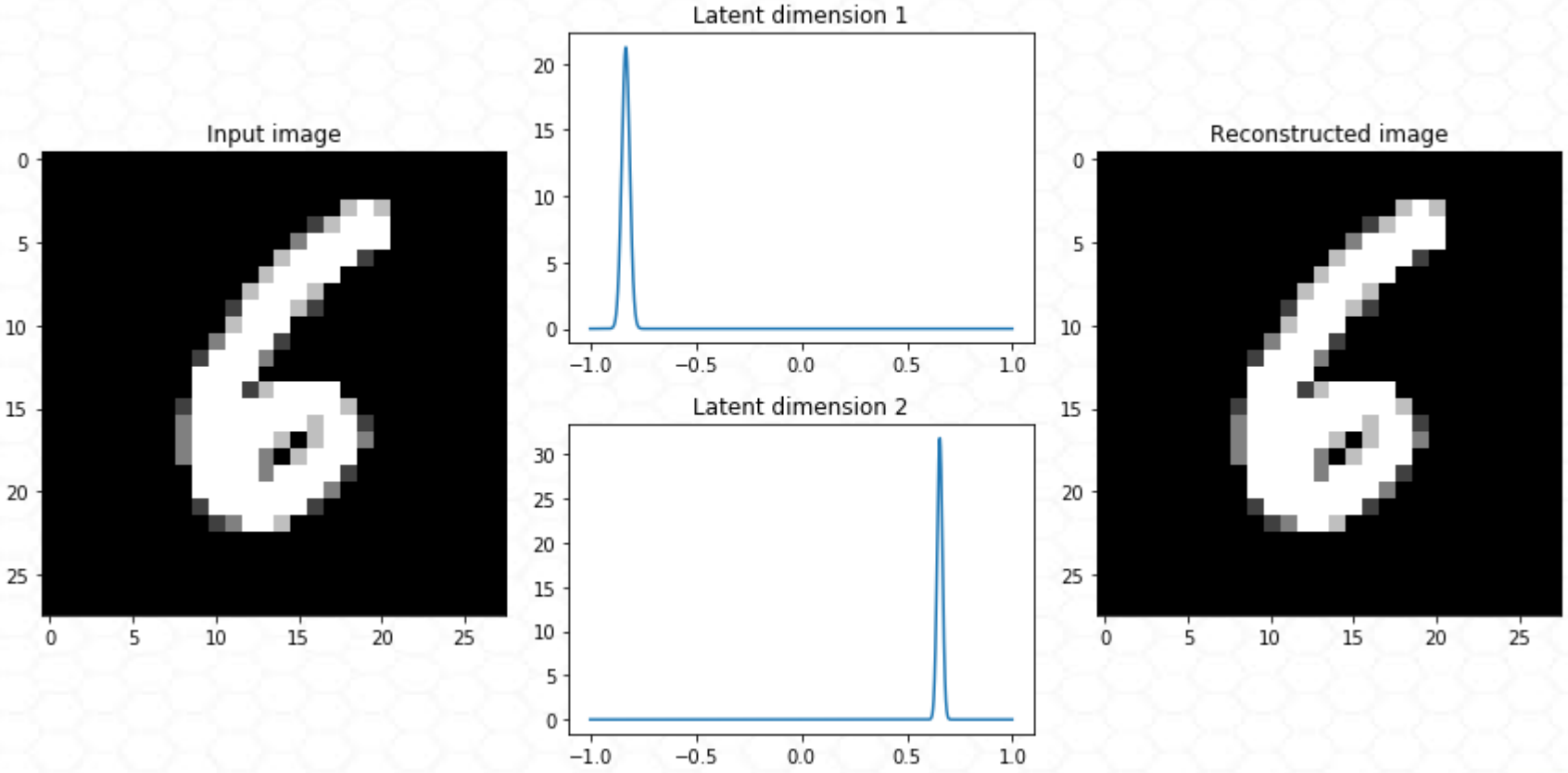
Penalizing KL divergence acts as a regularizing force

Attract distribution to have zero mean



Ensure sufficient variance to yield a smooth latent space

重建影像


Input image
Latent dimension 1
Latent dimension 2
Reconstructed image

生成式對抗網絡
(GAN)

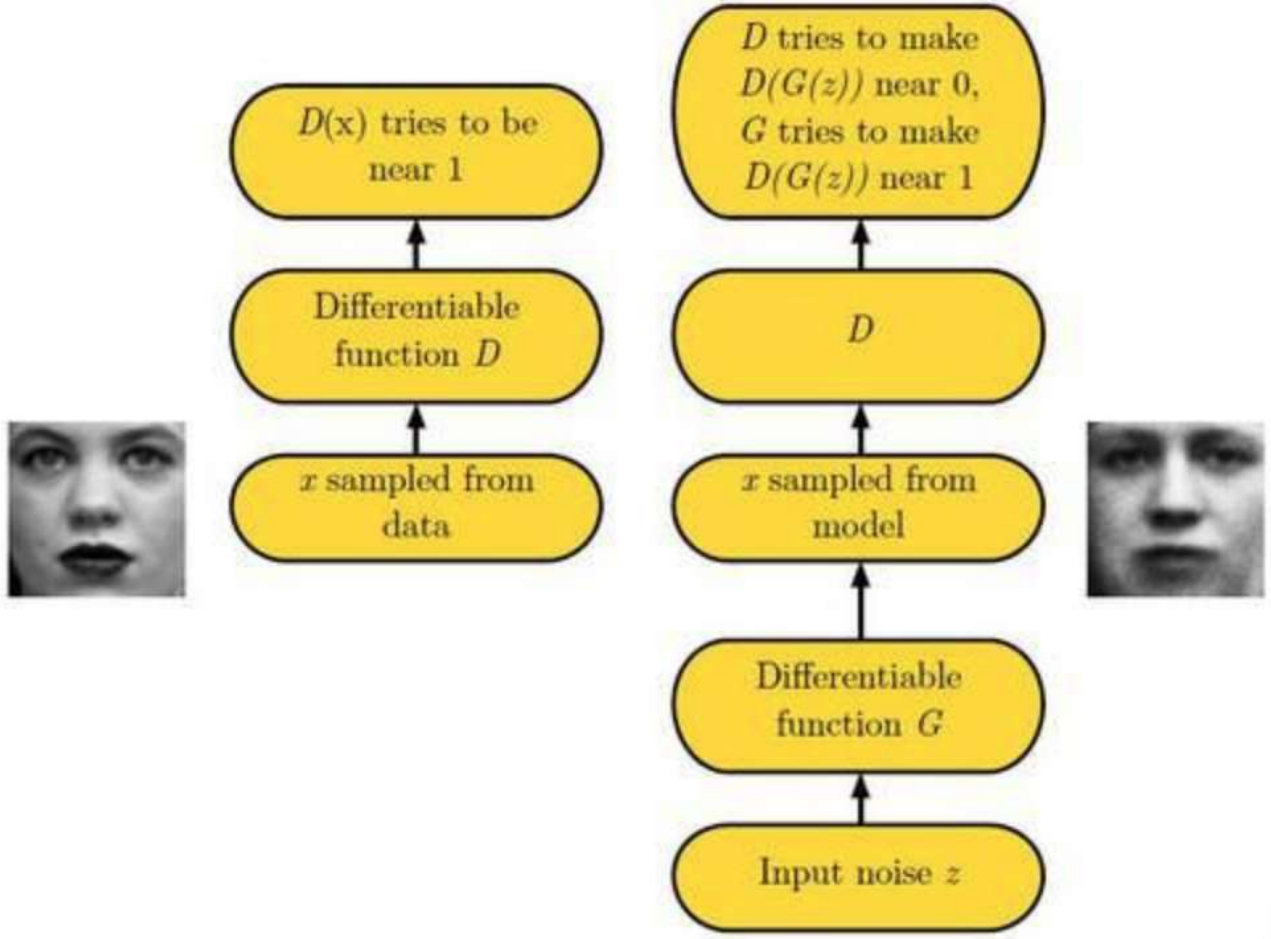
男女朋友間的GAN 話

男：你看這張拍的好不好？
女：這是在拍誰，是拍我還是在拍景物？
男：哦
……
男：那這張呢？
女：不行，拍得太胖了
男：哦
……
男：這張總算好點了吧？
女：角度再高一點，看起來比較好看
……
男：這樣呢？
女：Good 我待會發IG

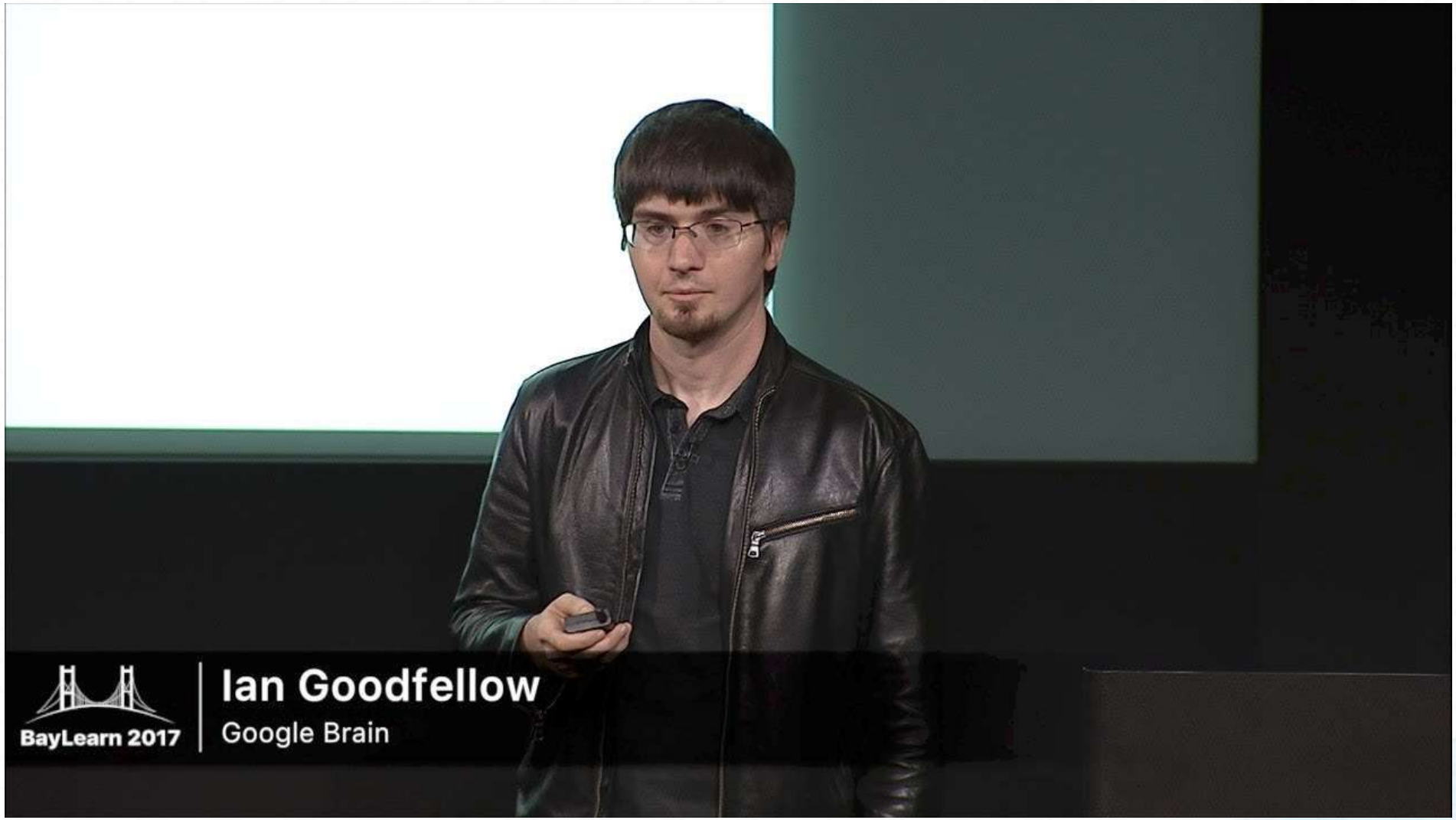
Generative Adversarial Network (GAN)

- Generator和Discriminator
  - Generator 是一個生成圖片的網路，它接收一個隨機的雜訊z，通過這個雜訊生成圖片，記做G(z)
  - Discriminator 是一個判別網路，判別一張圖片是不是真實的。它的輸入參數是x，x代表一張圖片，輸出D（x）代表x為真實圖片的概率，如果為1，就代表100%是真實的圖片，而輸出為0，就代表不可能是真實的圖片
- 在訓練過程中，Generator的目標就是儘量生成真實的圖片去欺騙Discriminator。而Discriminator的目標就是儘量把Generator生成的圖片和真實的圖片分別開來

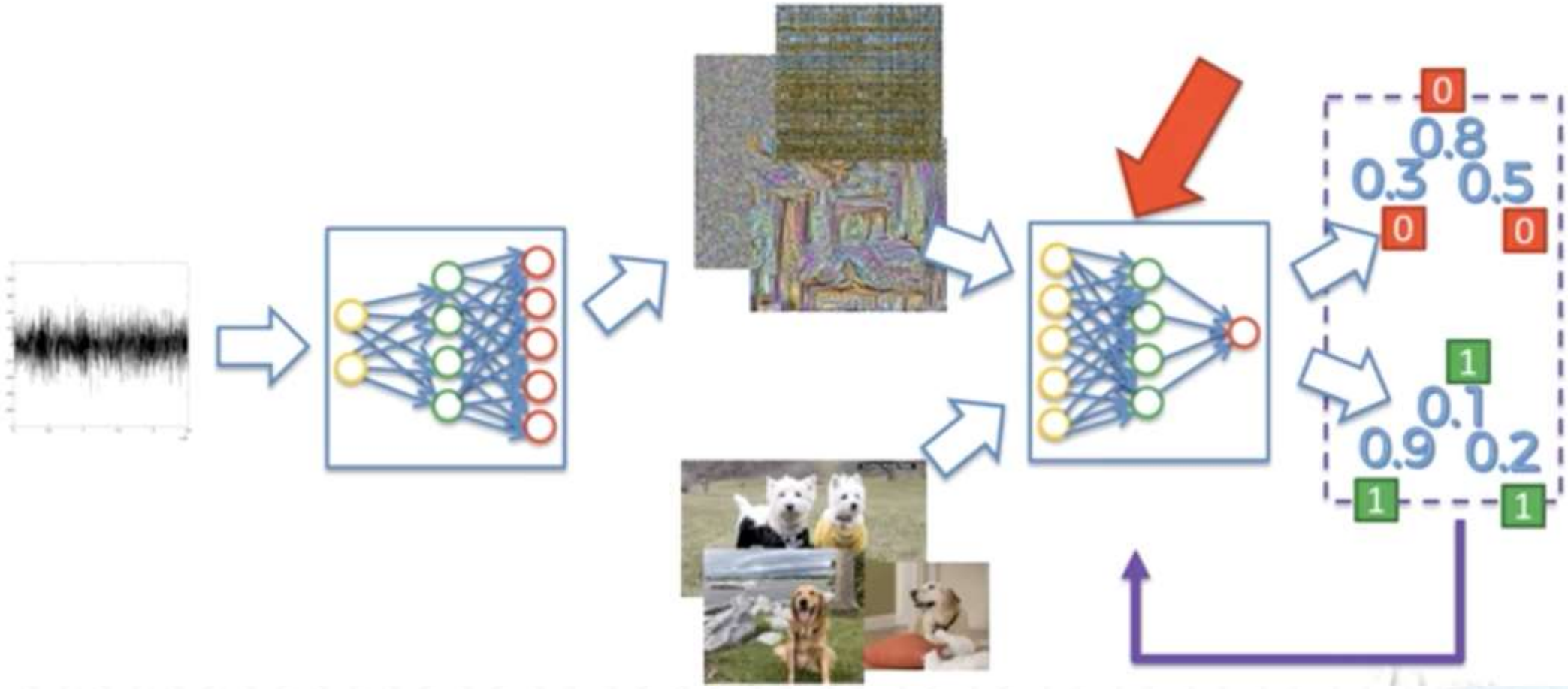
Generative Adversarial Network (GAN)



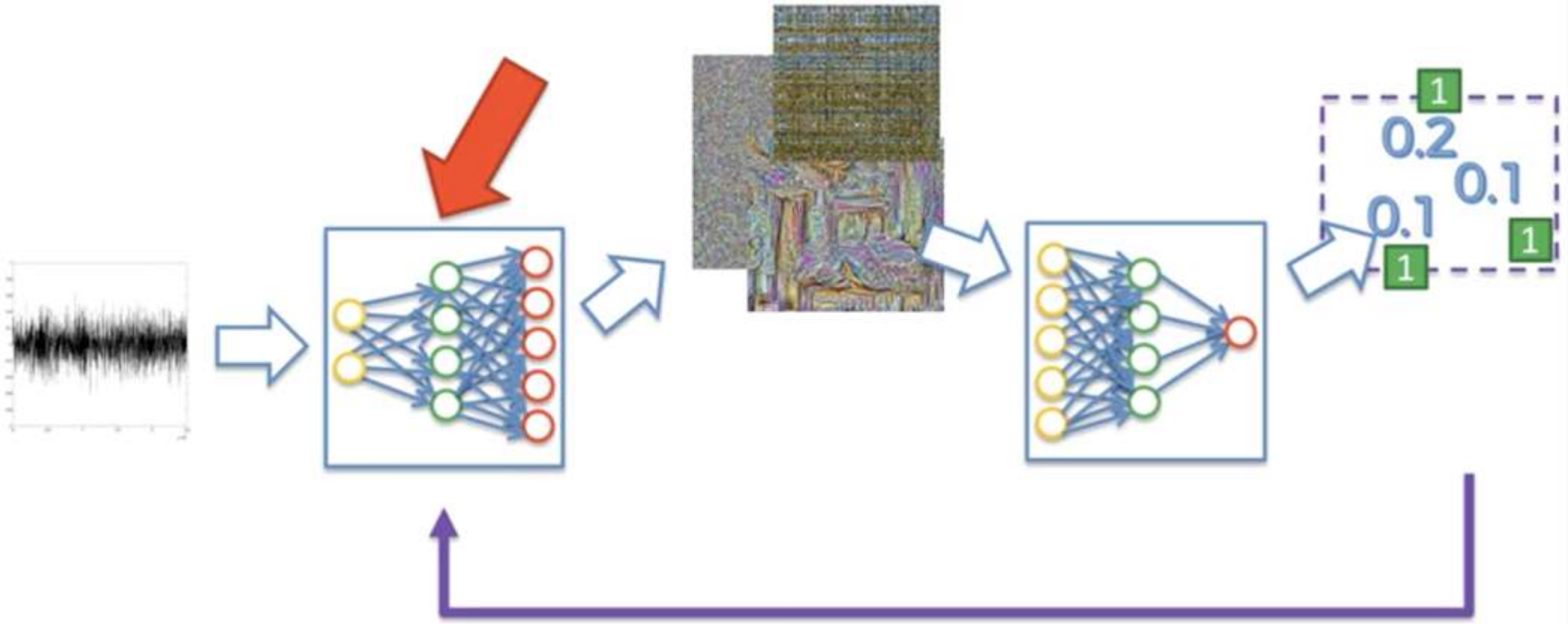
Ian Goodfellow



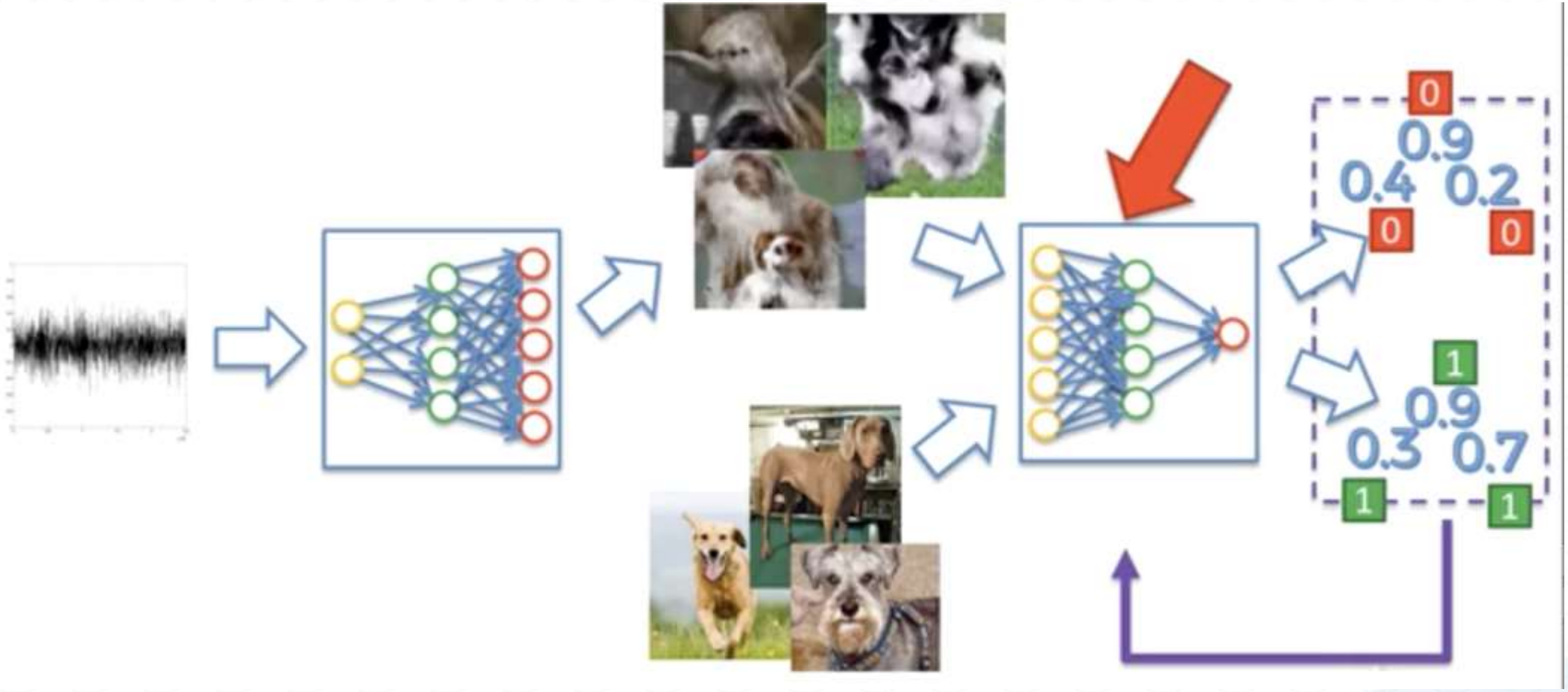
第一步: 訓練 Discriminator



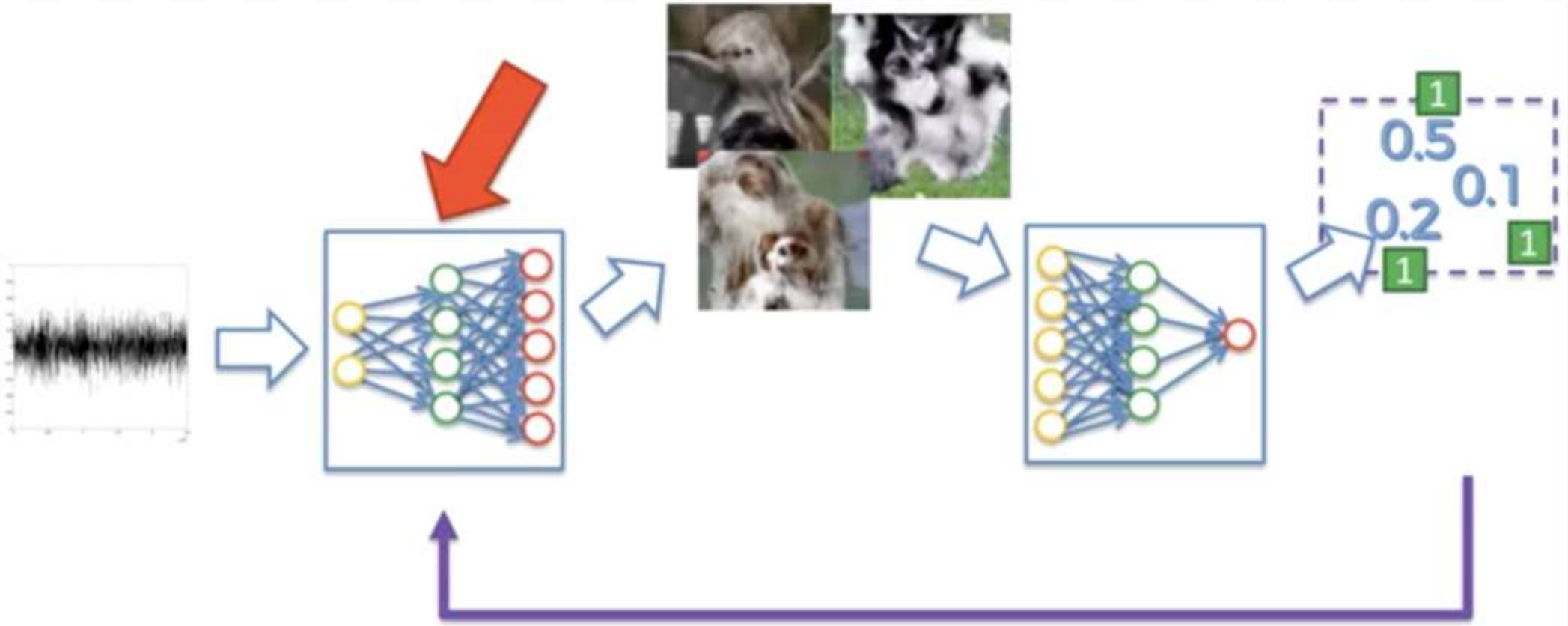
第一步: 訓練 Generator



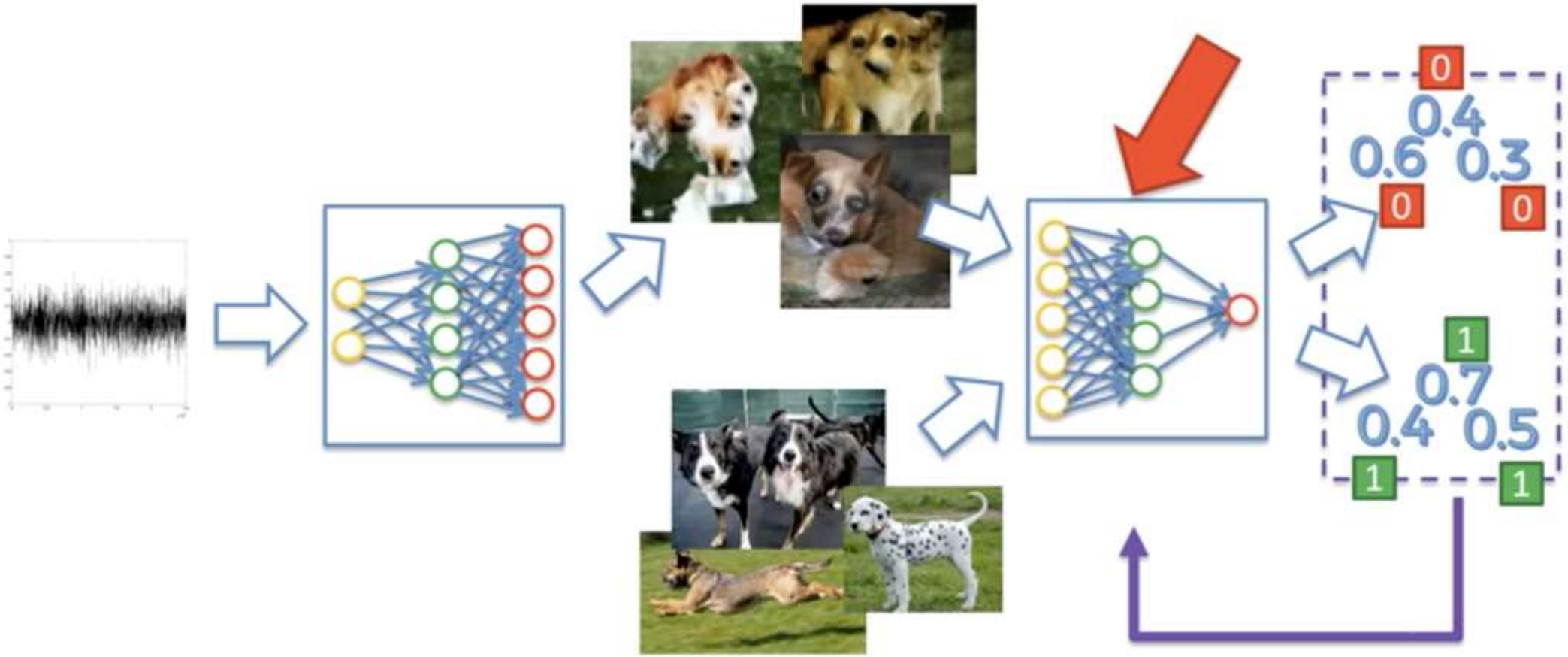
第二步: 訓練 Discriminator



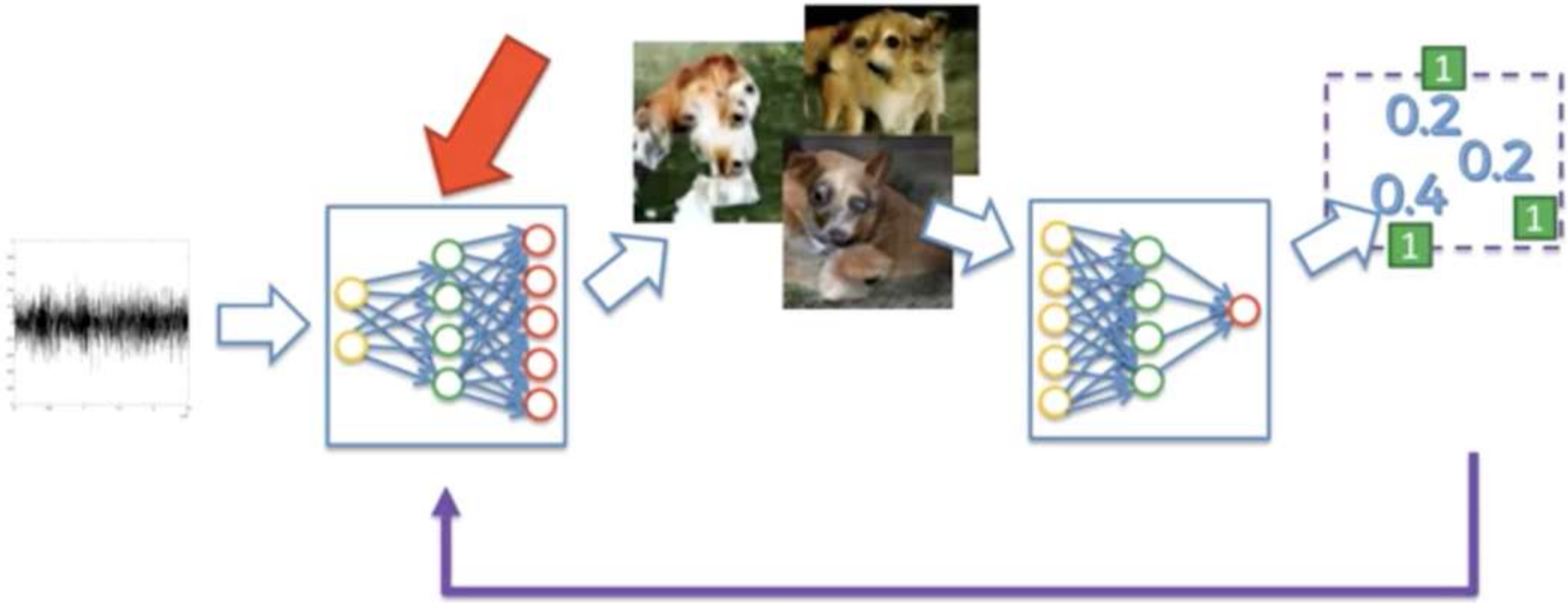
第二步: 訓練 Generator



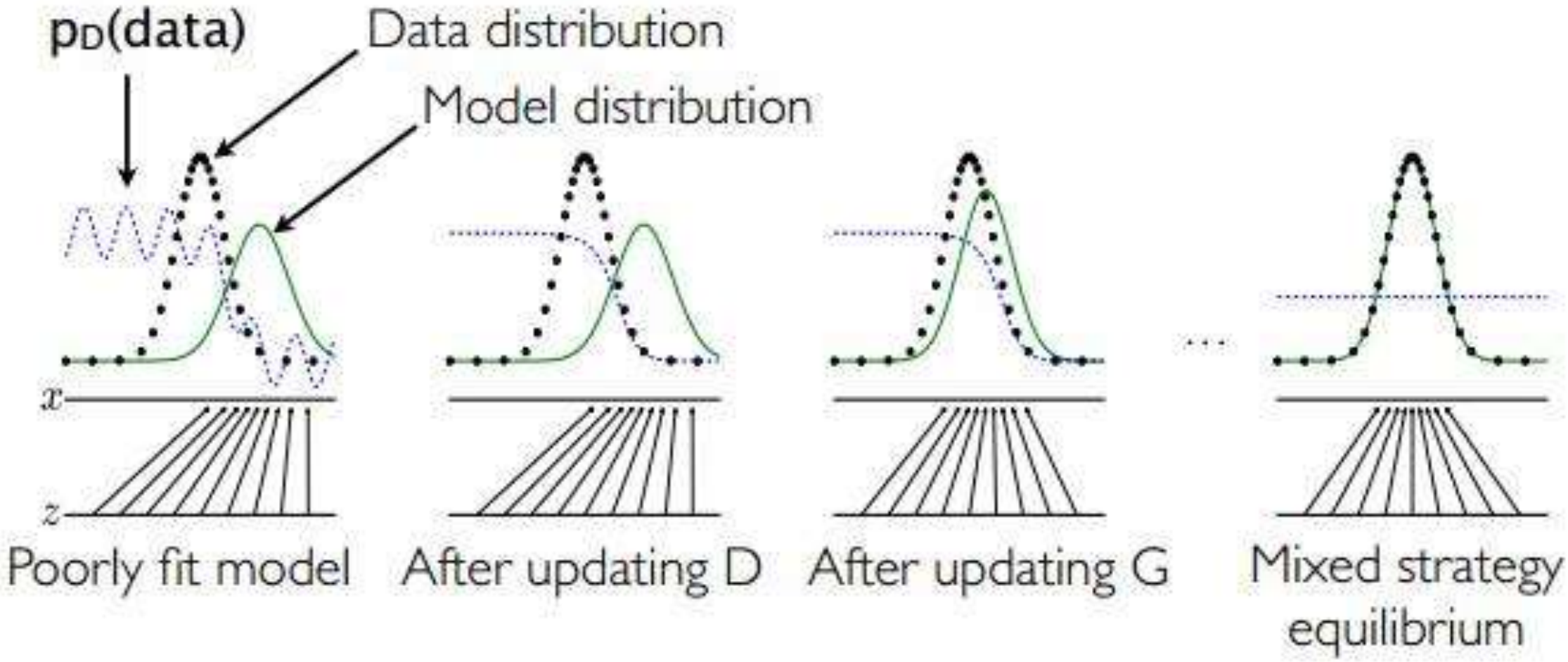
第三步: 訓練 Discriminator



第三步: 訓練 Generator



GAN 生成概念


p_D(data)
Data distribution
Model distribution
x
z
Poorly fit model
After updating D
After updating G
Mixed strategy equilibrium

GAN 的特點

- 相比較傳統的模型，他存在兩個不同的網路，而不是單一的網路，並且訓練方式採用的是對抗訓練方式
- GAN中G的梯度更新資訊來自判別器D，而不是來自資料樣本

GAN的優點

- GAN是一種生成式模型，相比較其他生成模型(如玻茲曼機) 只用到了反向傳播,而不需要複雜的馬可夫鏈(Markov Chain)
- GAN採用的是一種無監督的學習方式訓練，可以被廣泛用在無監督學習和半監督學習領域
- GANs可以產生更加清晰，真實的樣本，而且生成的圖片是一致的,但是VAE是有偏差的
- GAN 不用設計損失函數，只要有一個的基準，便可交給對抗訓練了

GAN 的缺點

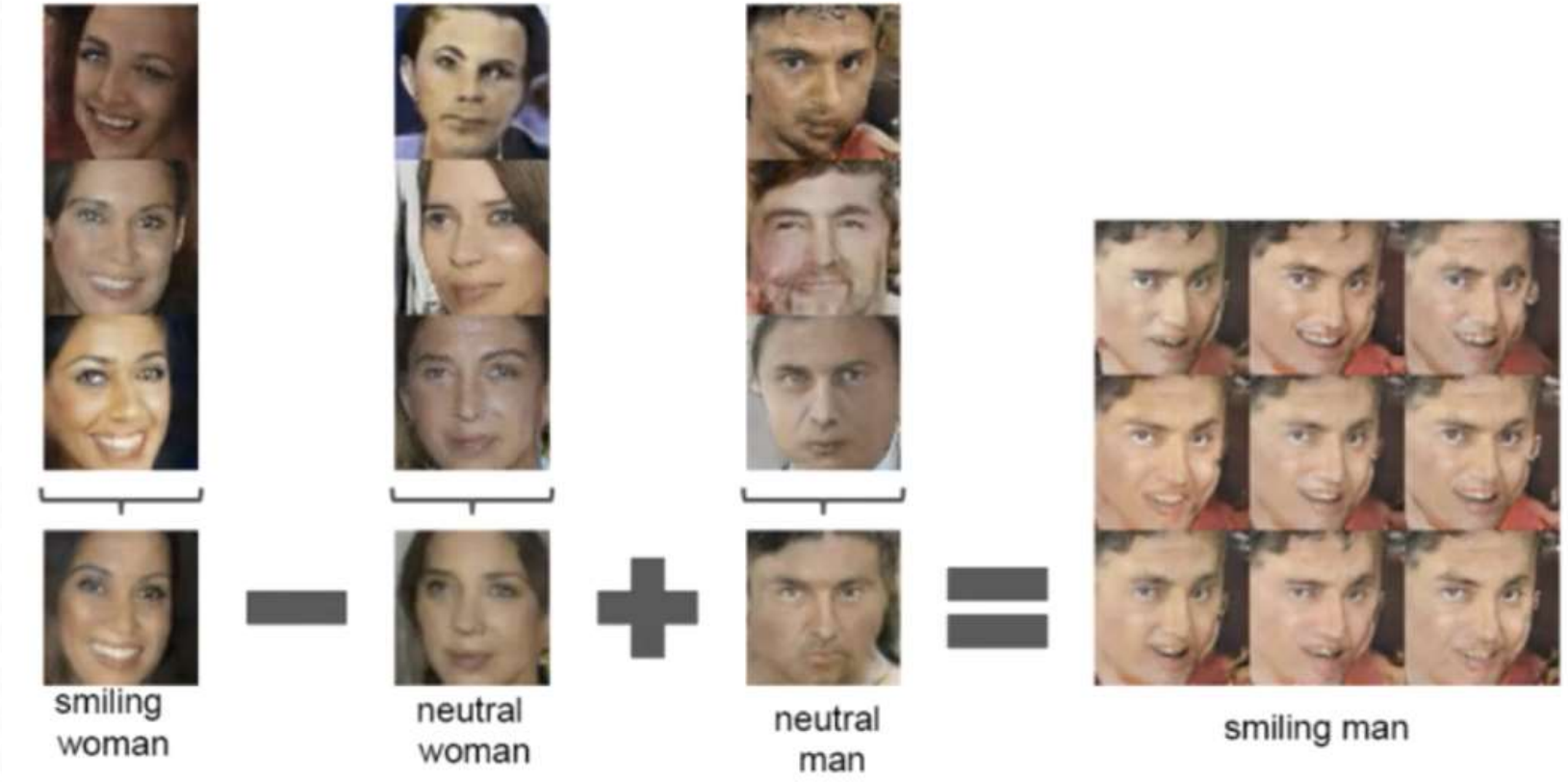
- 訓練GAN需要達到Nash Equilibrium,有時候可以用梯度下降法做到,但有時候做不到,所以訓練GAN相比VAE是不穩定的
- GAN不適合處理離散形式的資料，比如文本資料

圖片生成 (Image Generation)

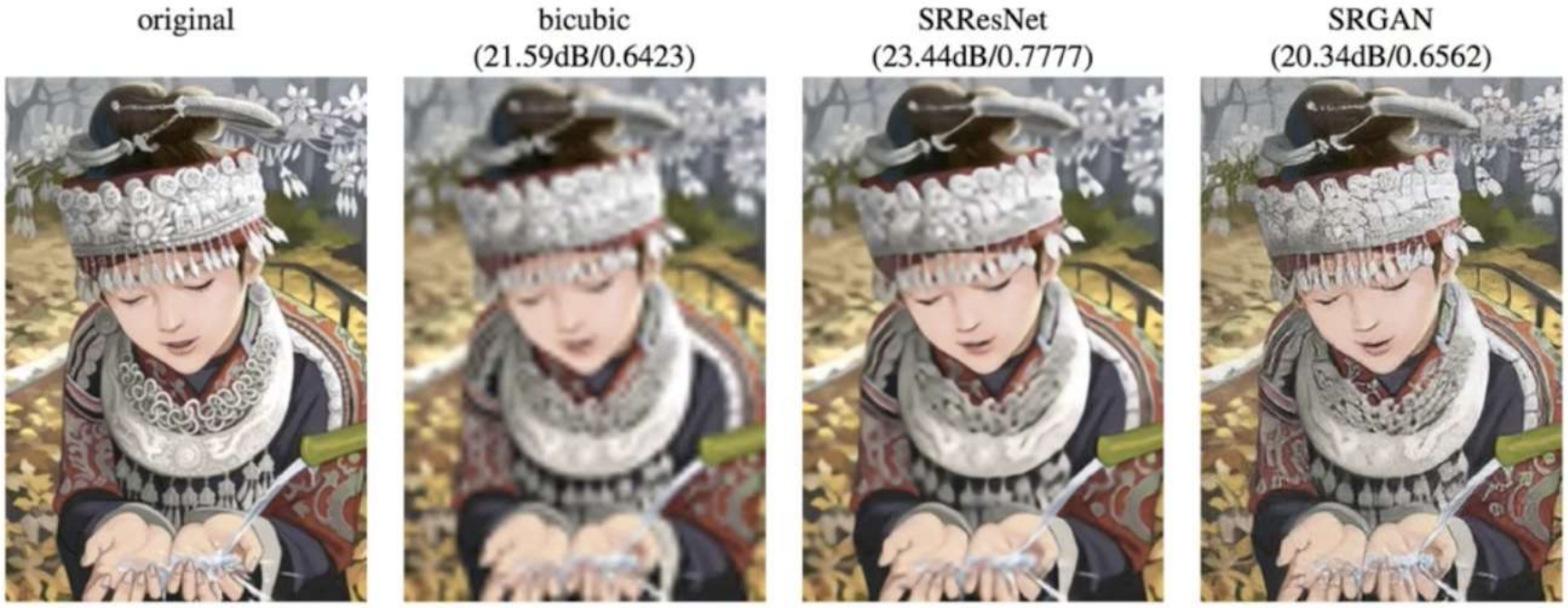
原本圖片

生成圖片

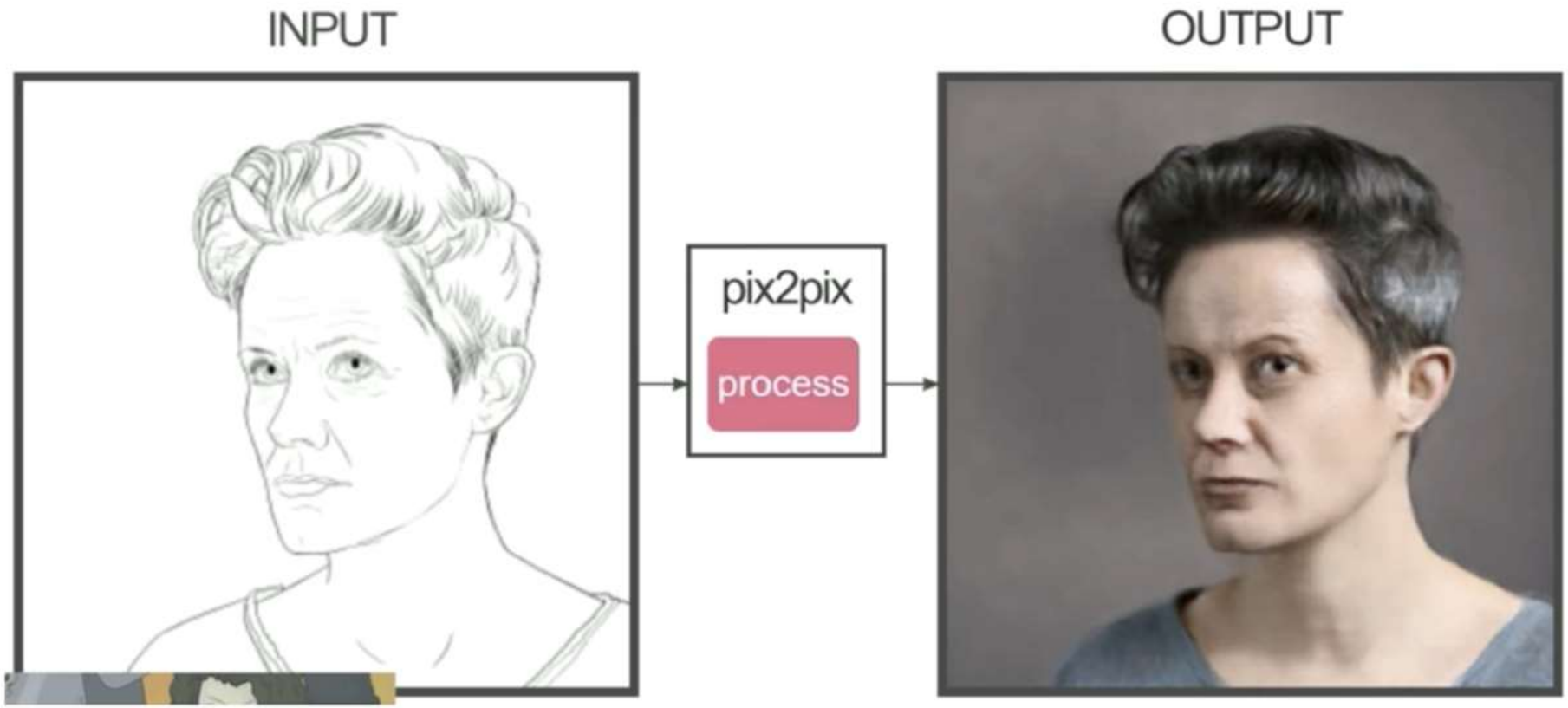
修改影像 (Image Modification)



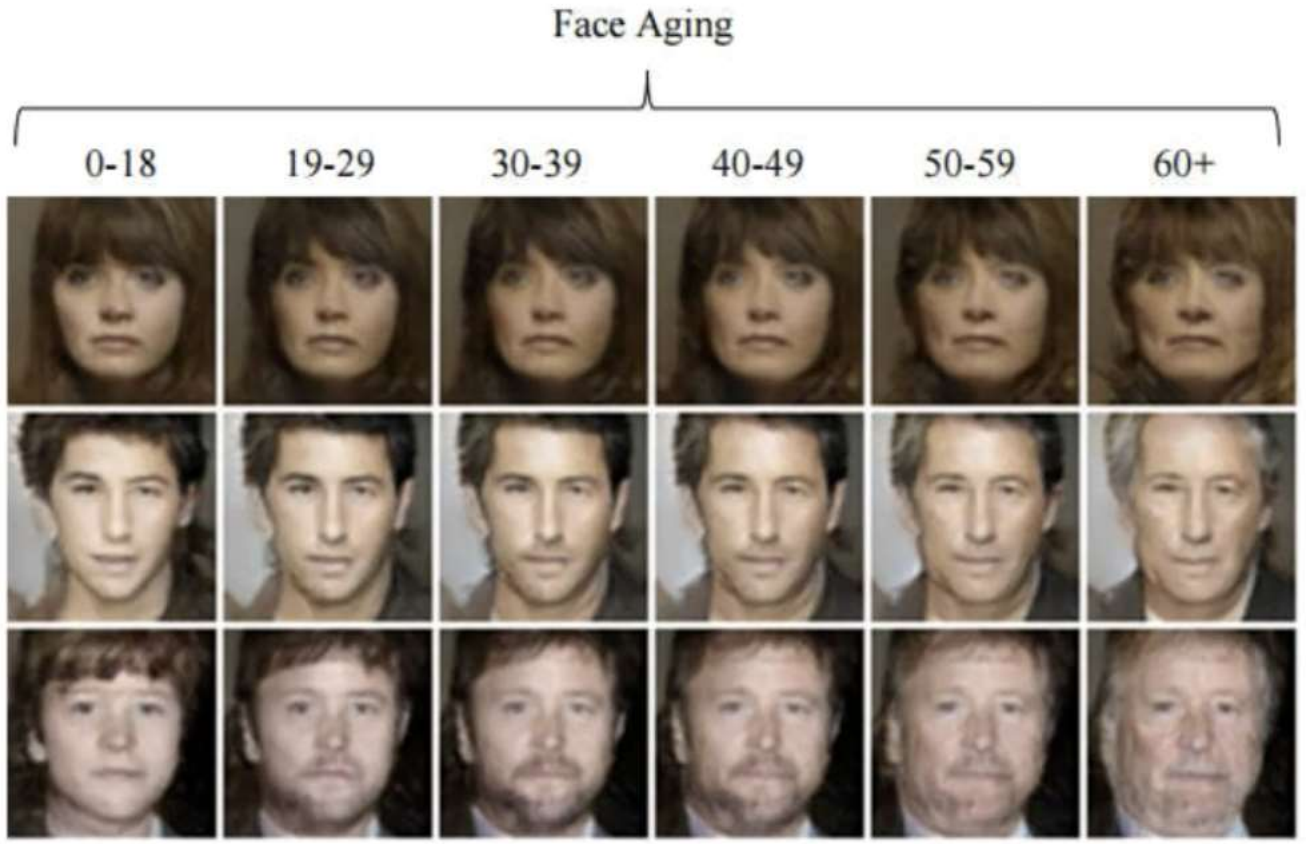
增強解析度 (Super Resolution)



製作出如相片般的影像 (Photo Realistic Image)



改變臉部年紀 (Face Aging)



Keras-GAN

- https://github.com/eriklindernoren/Keras-GAN

Keras-GAN

Collection of Keras implementations of Generative Adversarial Networks (GANs) suggested in research papers. These models are in some cases simplified versions of the ones ultimately described in the papers, but I have chosen to focus on getting the core ideas covered instead of getting every layer configuration right. Contributions and suggestions of GAN varieties to implement are very welcomed.

See also: PyTorch-GAN

THANK YOU