初探類神經網路

Week #1

機器學習

Machine Learning

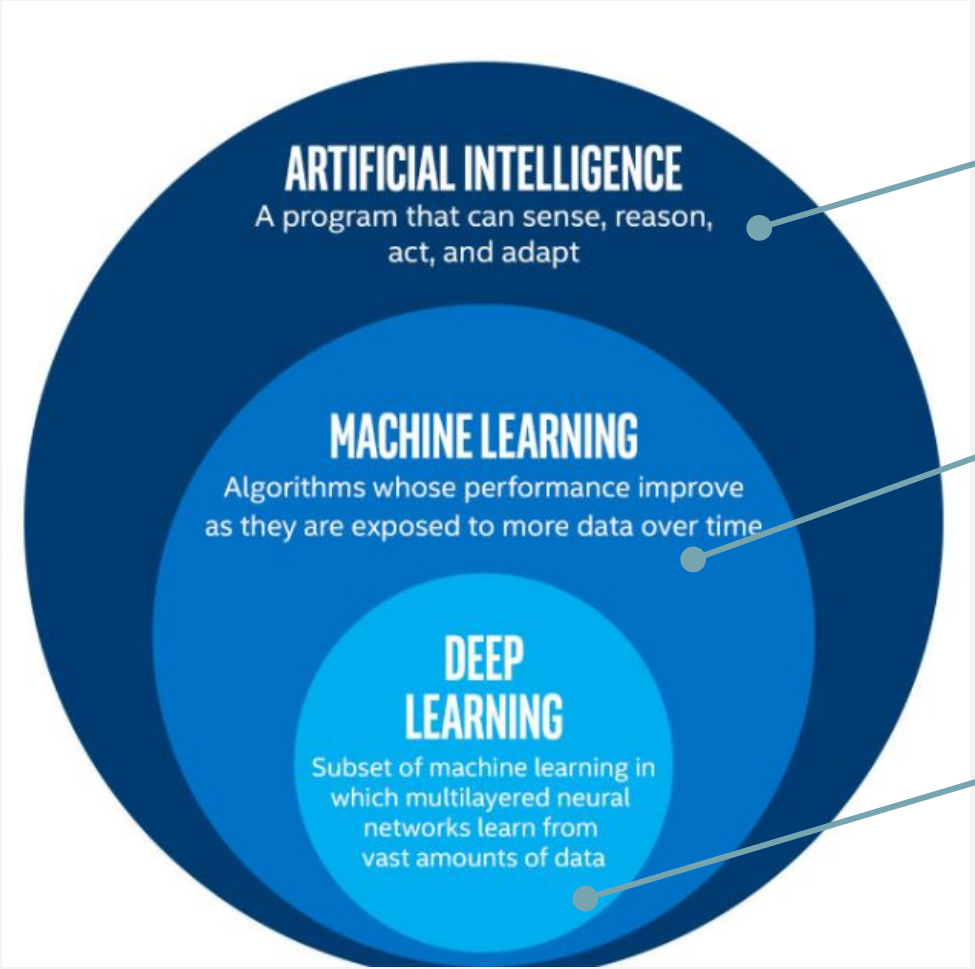
人工智慧

Artificial Intelligence

深度學習

Deep Learning

差在哪裡？



人工智慧(AI):
可以模擬人類感知行為的**程式**。

機器學習(ML):
一種**演算法**，可以透過學習**大量資料**不斷進步。

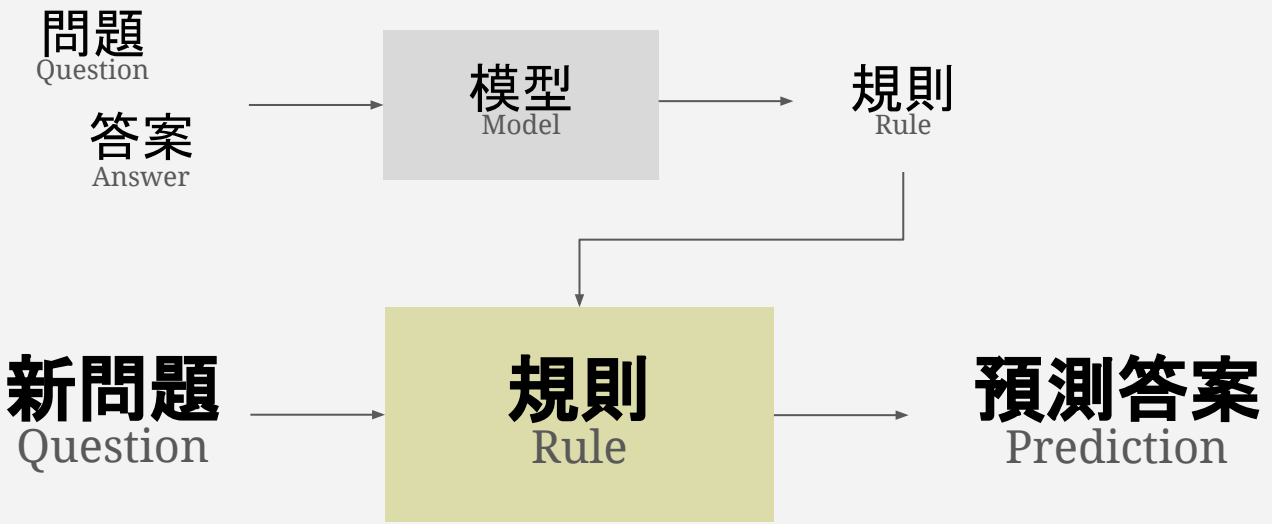
深度學習(DL):
機器學習的一種。透過**多層的結構**進行學習。

What is Machine Learning

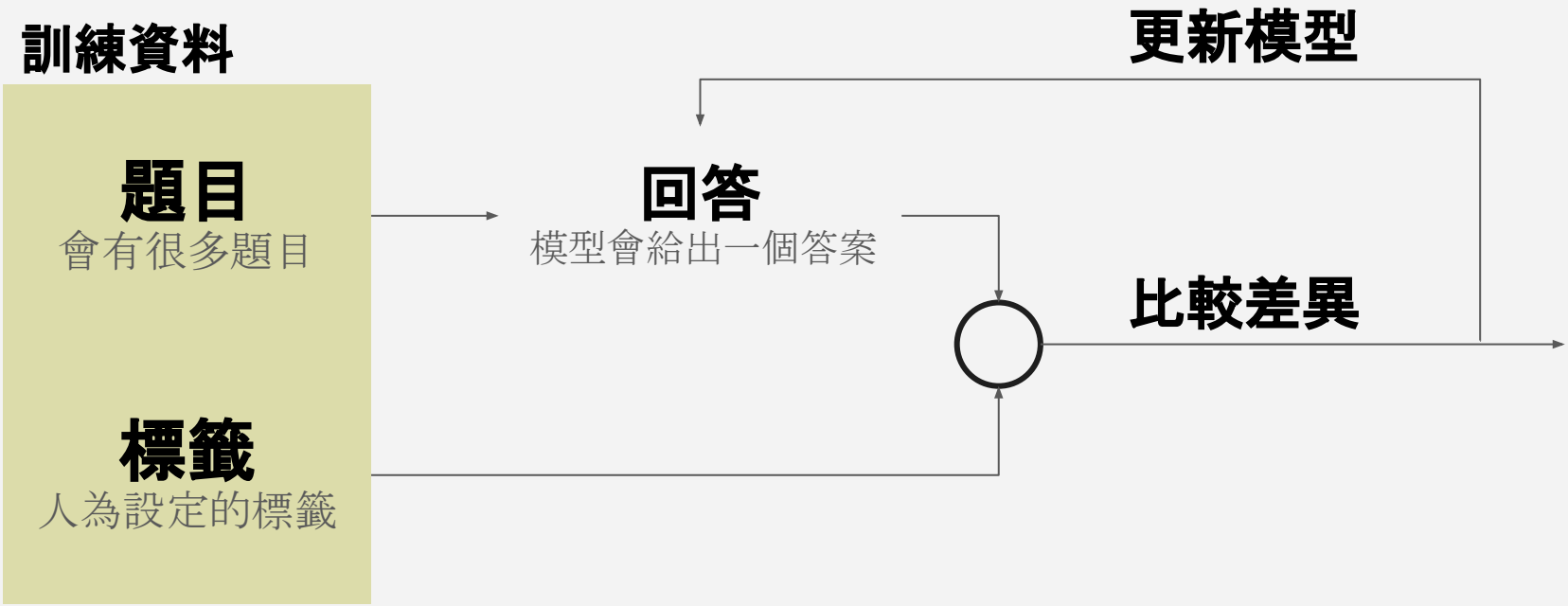
What is Machine Learning

What is Machine Learning

What is Machine Learning



補充:監督式學習 Supervised Learning (Week #2)



Before Machine Learning…

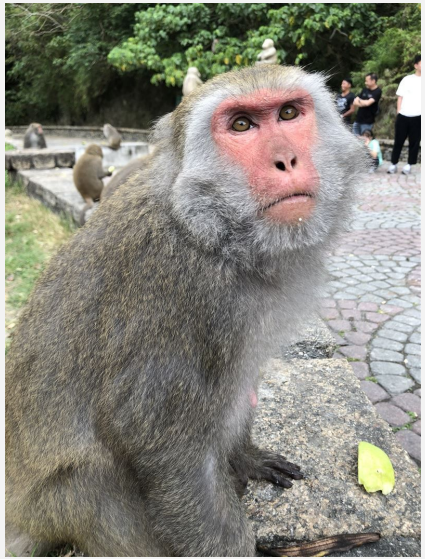
Before Machine Learning…

```
if(有花紋){
    if(){
        ...
    }
}
else{
    if(喵喵叫){
        ...
    }
    else{
        if(){
            ...
        }
    }
}
```

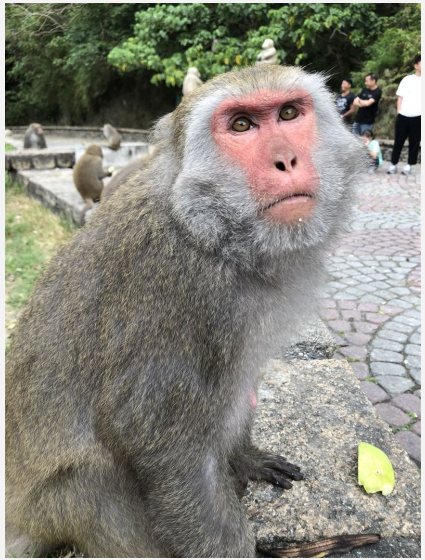
Machine Learning

「機器學習就是一種**找到規則**的過程」

我訓練了一個辨認貓狗的模型，但...



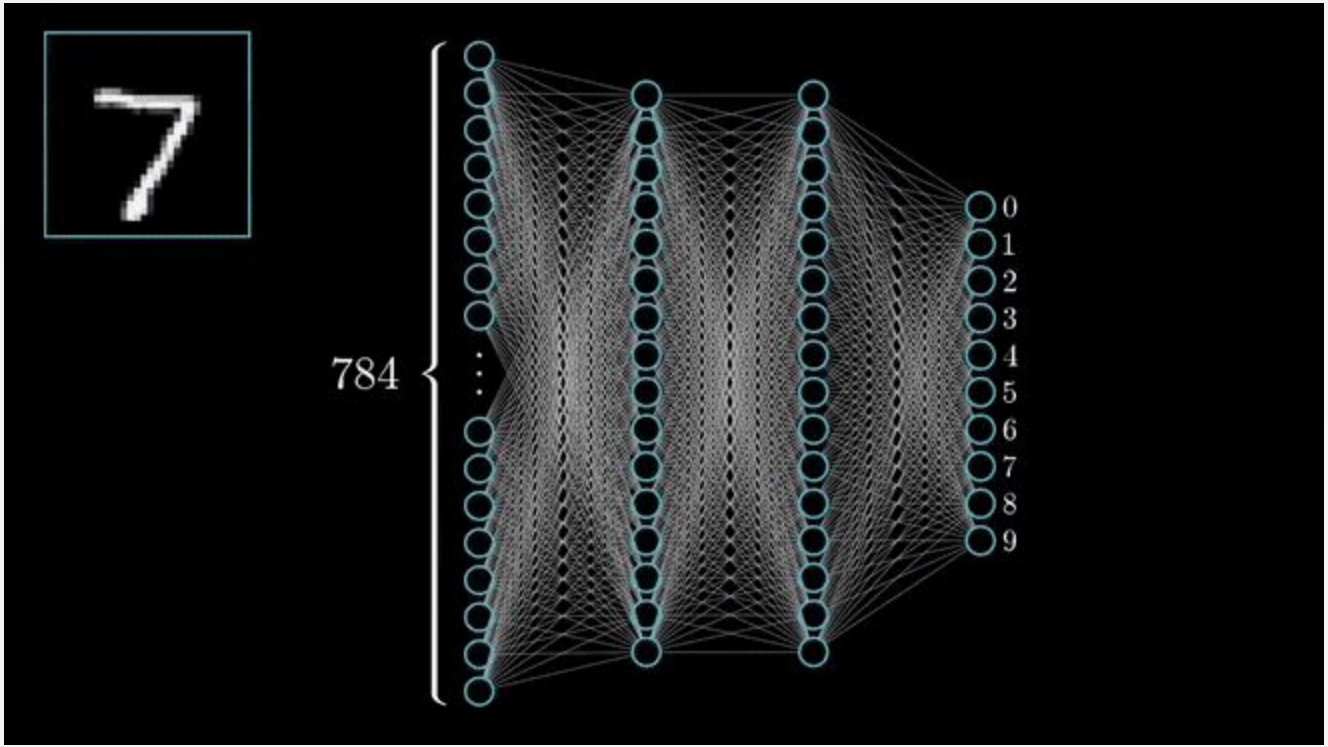
(A) 貓　　(B) 狗　　(C) 猴子

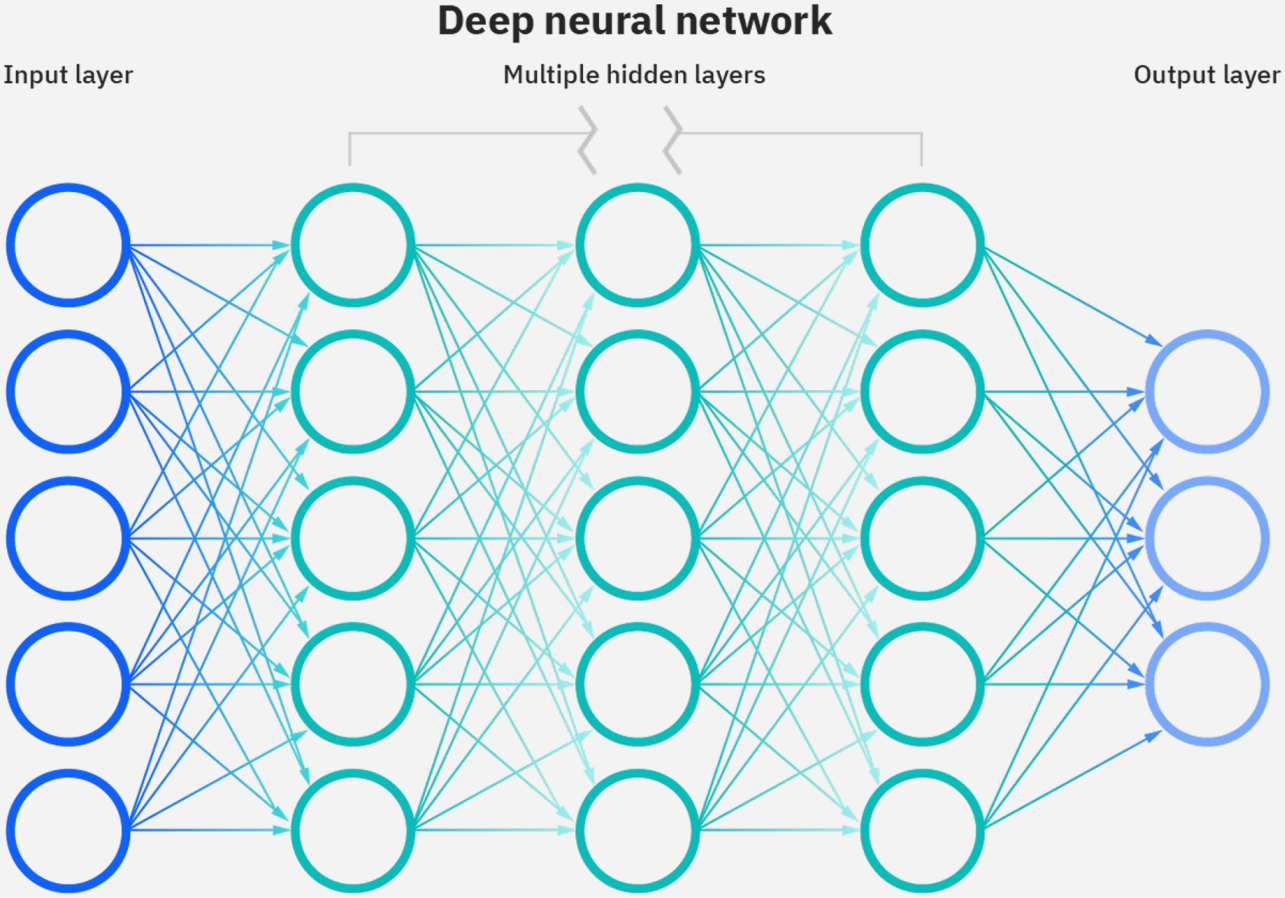


Ans. **貓或是狗**

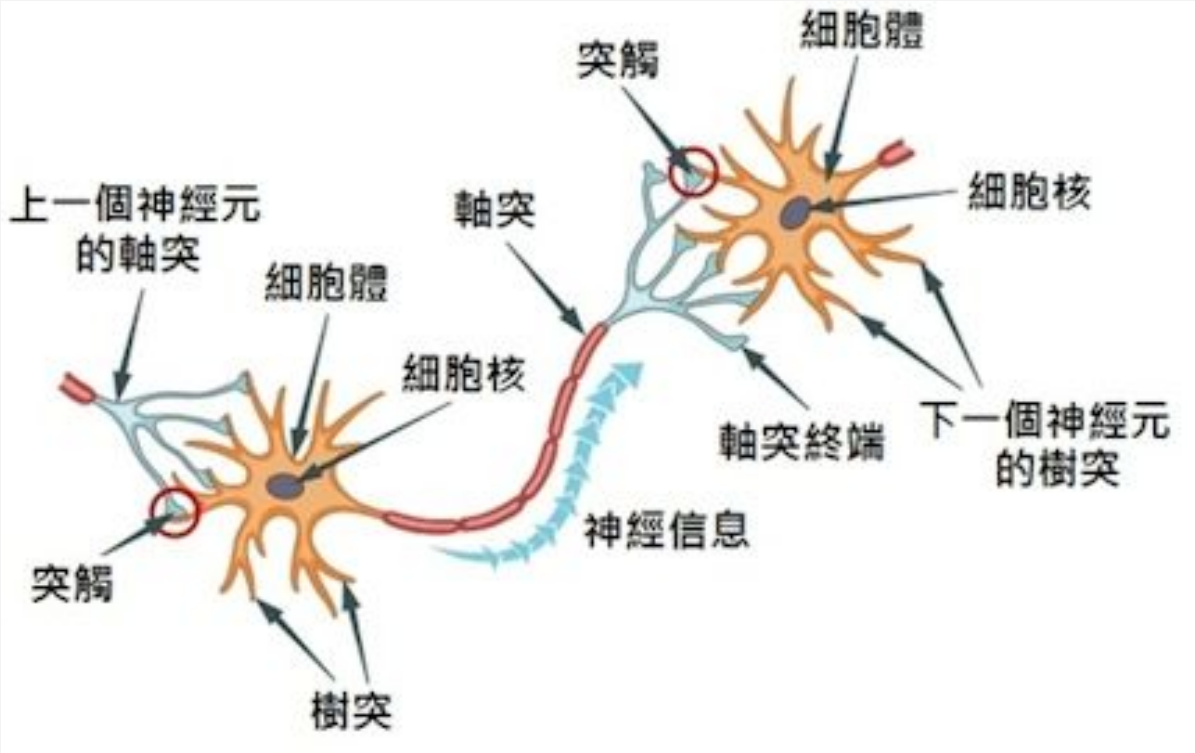
所以，**類神經網路**又是什麼？

(Neural Network, NN)

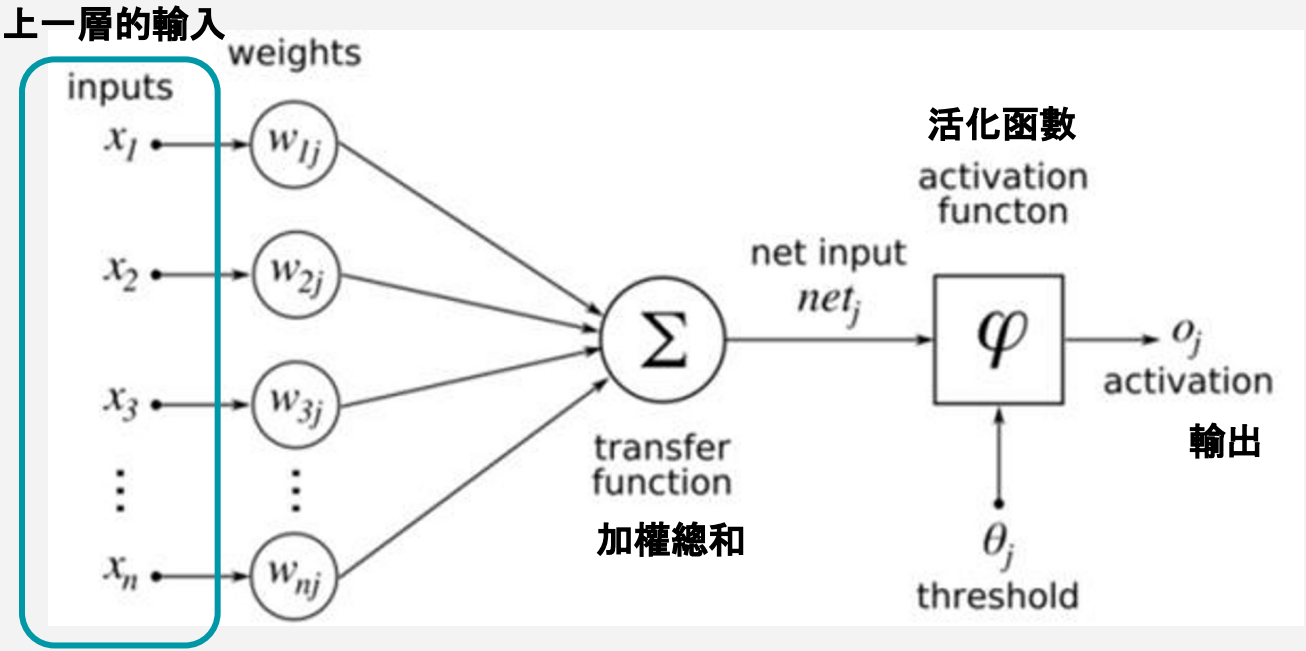

784
0
1
2
3
4
5
6
7
8
9


Deep neural network
Input layer
Multiple hidden layers
Output layer

神經元

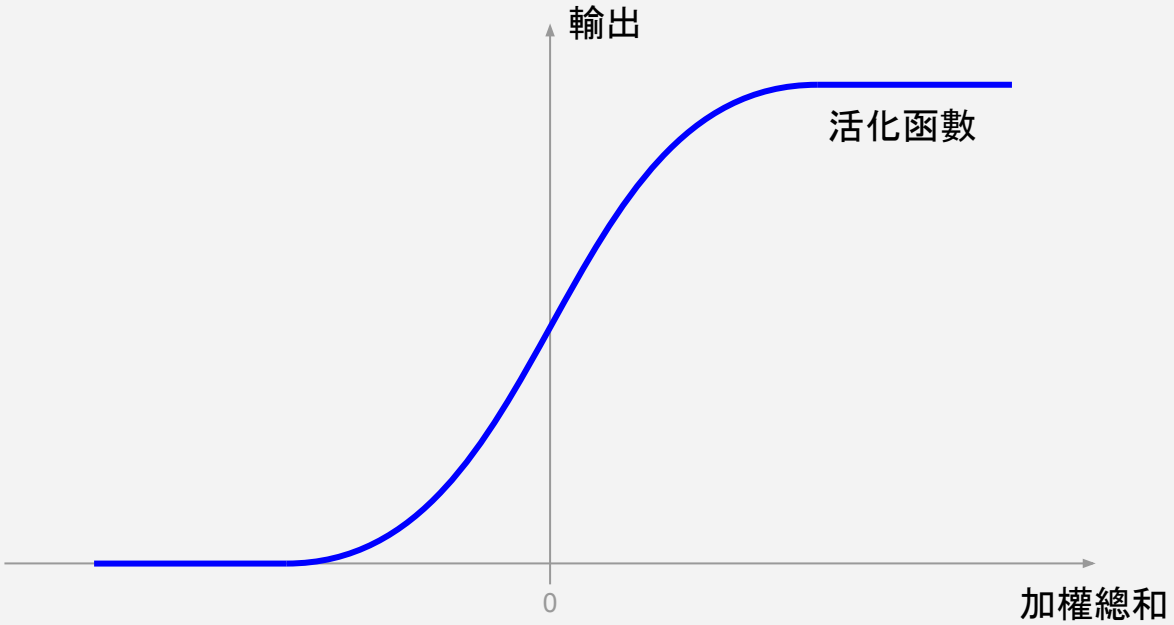


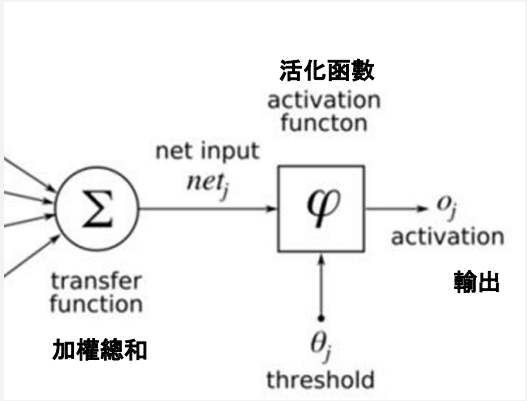
機器學習的神經元


上一層的輸入
weights
inputs
x_1
x_2
x_3
x_n
w_{1j}
w_{2j}
w_{3j}
w_{nj}
Σ
transfer function
加權總和
net input
net_j
活化函數
activation functon
φ
θ_j
threshold
o_j
activation
輸出

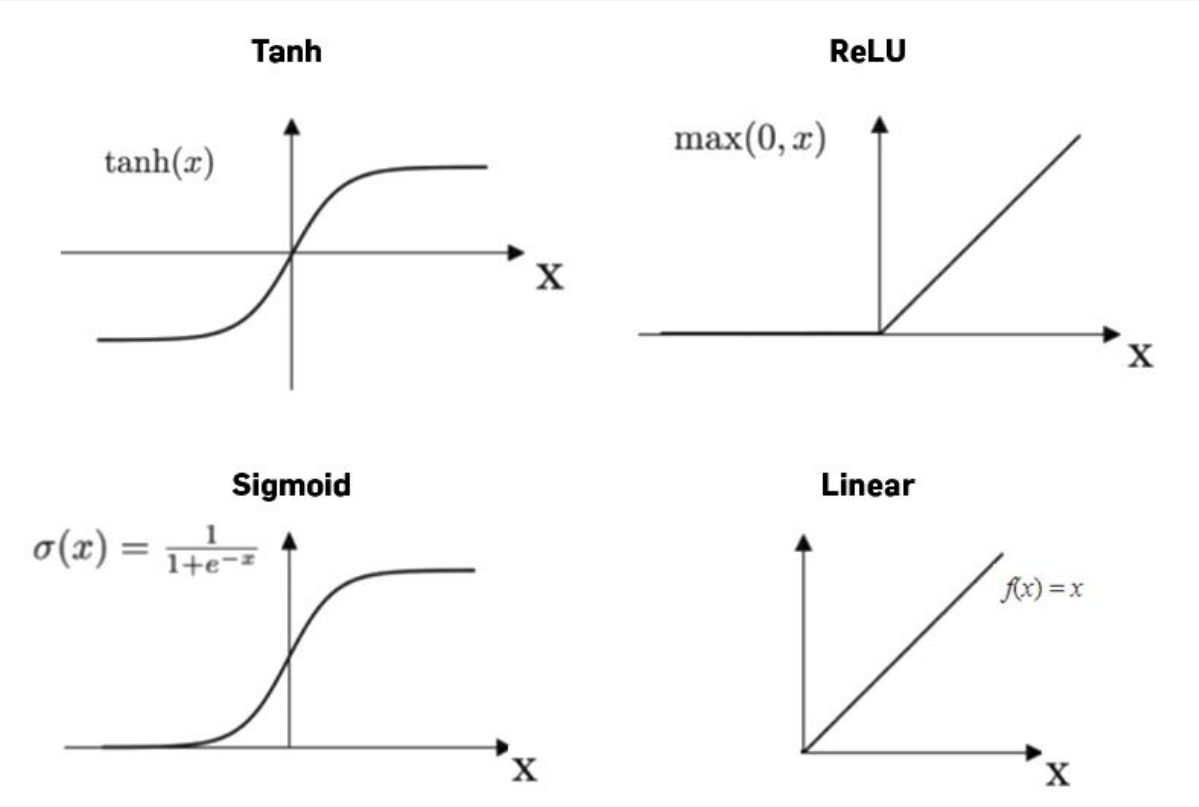
Activation Function 活化函數

可以想像成該神經元的**敏感度**。加總上一層輸入的訊號之後，我們要決定輸出多大的訊號給下一層的神經元。例如：

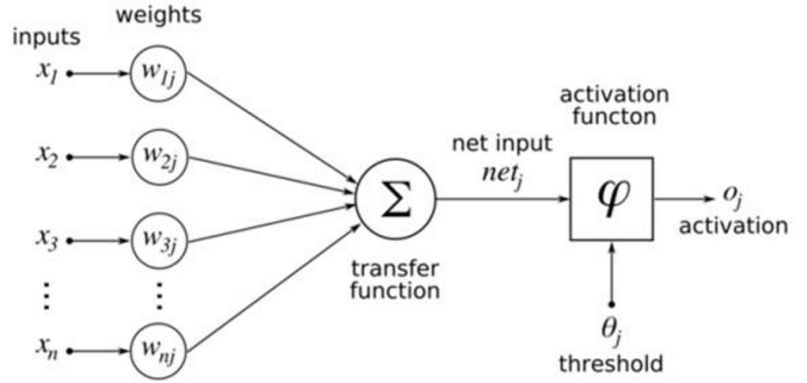




Activation Function 活化函數

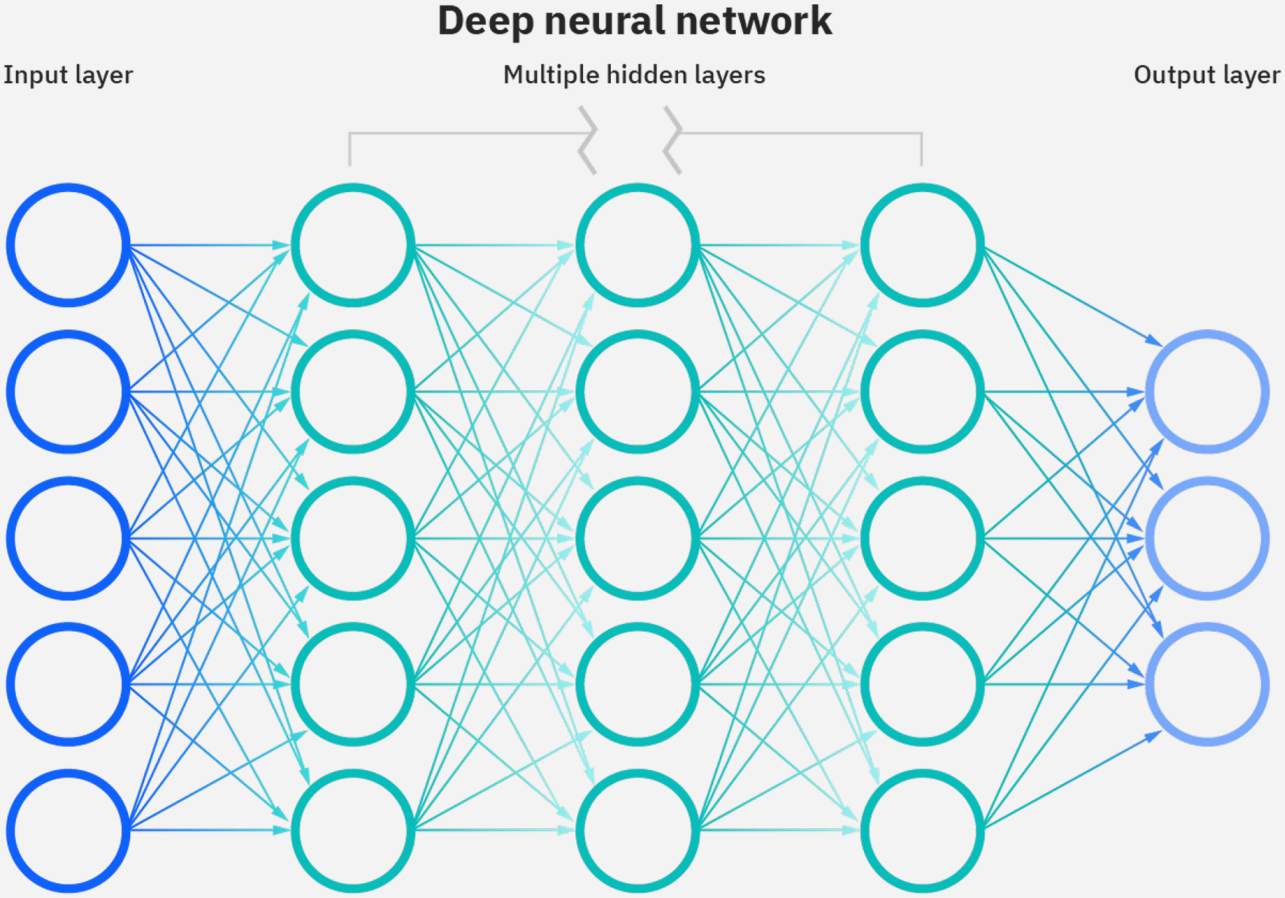


機器學習的神經元 (我們趕快跳過這一頁)



$$net_j = \Sigma w_i x_i + b$$

$$activation\ \boldsymbol{o_j} = f(net_j)$$

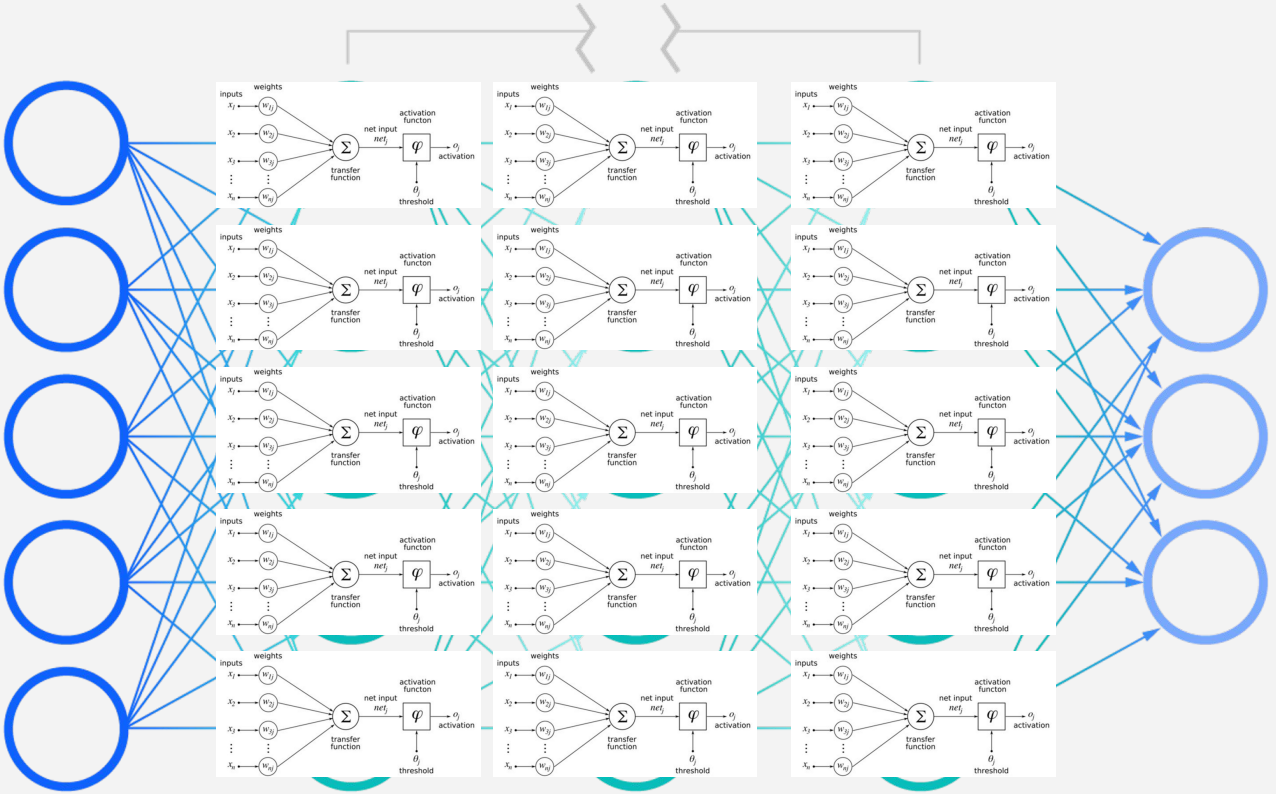

Deep neural network
Input layer
Multiple hidden layers
Output layer

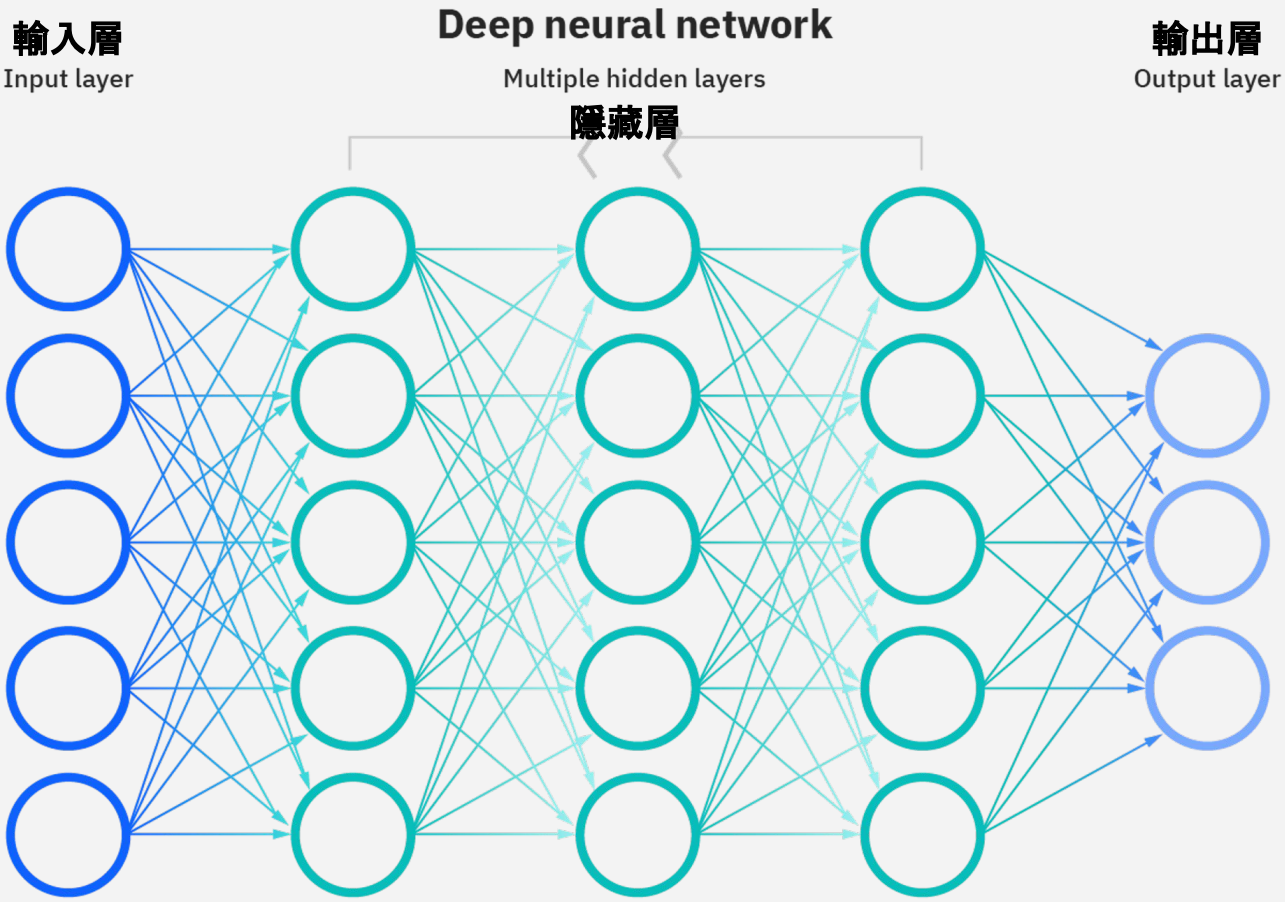
Deep neural network

Input layer

Multiple hidden layers

Output layer

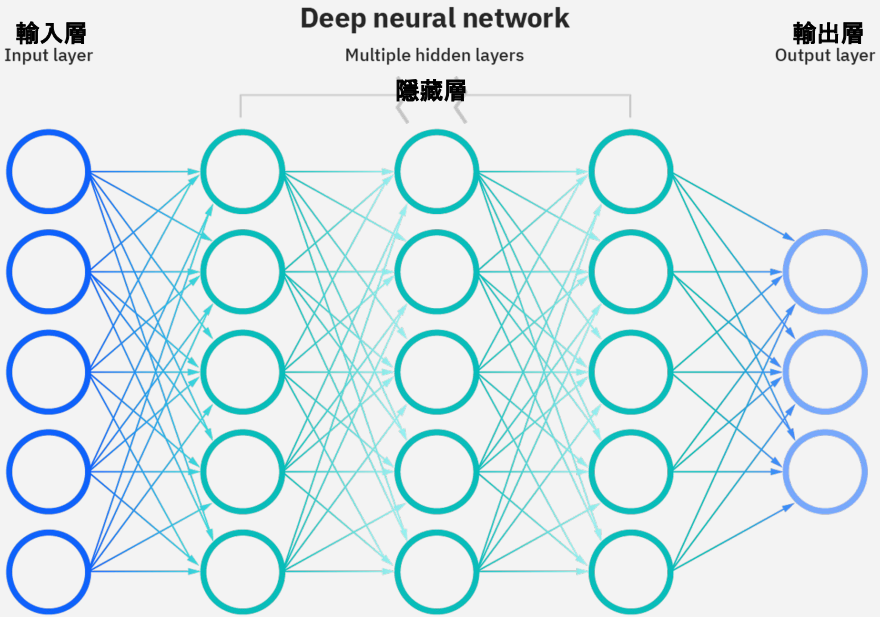



輸入層
Input layer
Deep neural network
Multiple hidden layers
隱藏層
輸出層
Output layer

Neural Network in Deep Learning

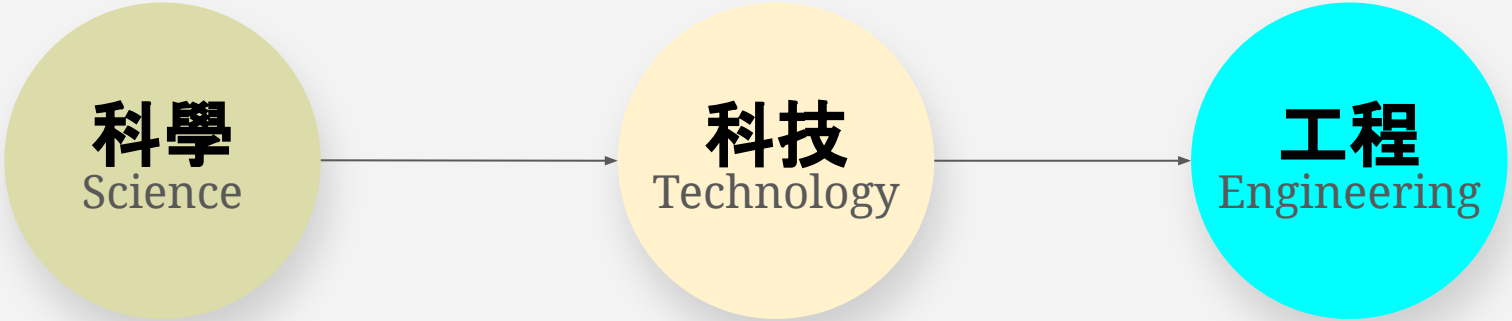
雖然每個神經元只負責很簡單的線性組合還有活化函數的使用，但只要量夠大，還是會有魔法發生！

此外，我們可以用 GPU 進行**平行處理，同時進行很多個神經元的運算**，讓整體運算速度變快。

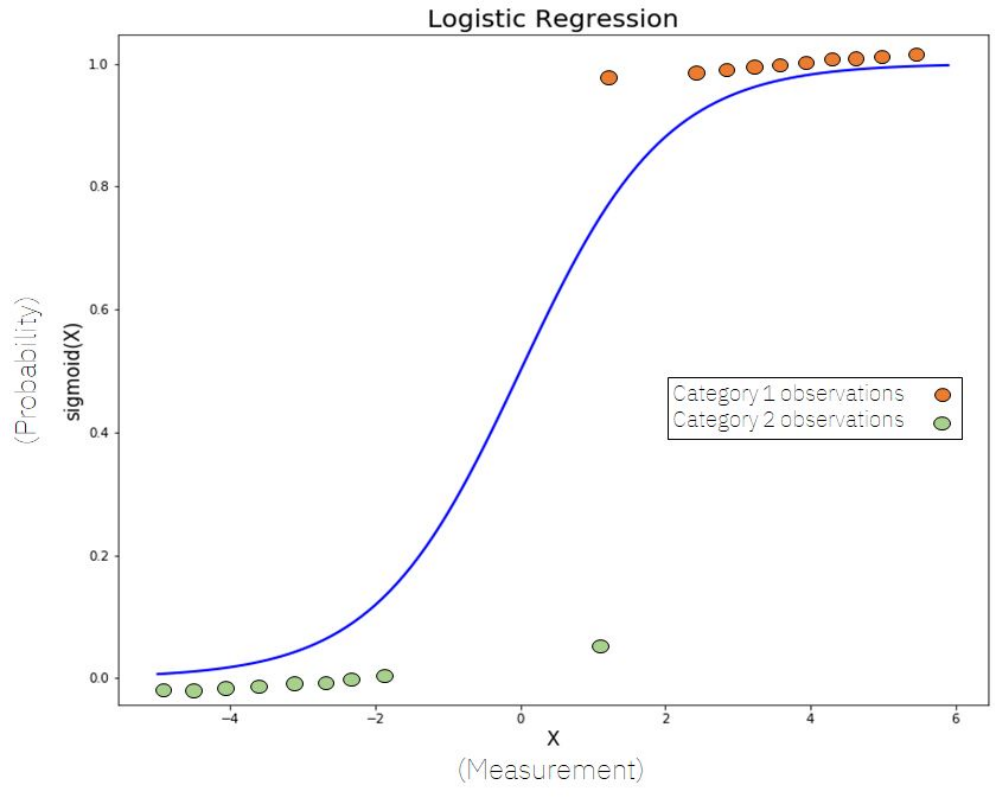



PCI Express 4.0 Host Interface
GigaThread Engine with MIG Control
GPC
TPC
SM
L2 Cache
L2 Cache
Memory Controller
HBM2
High-Speed Hub
NVLink

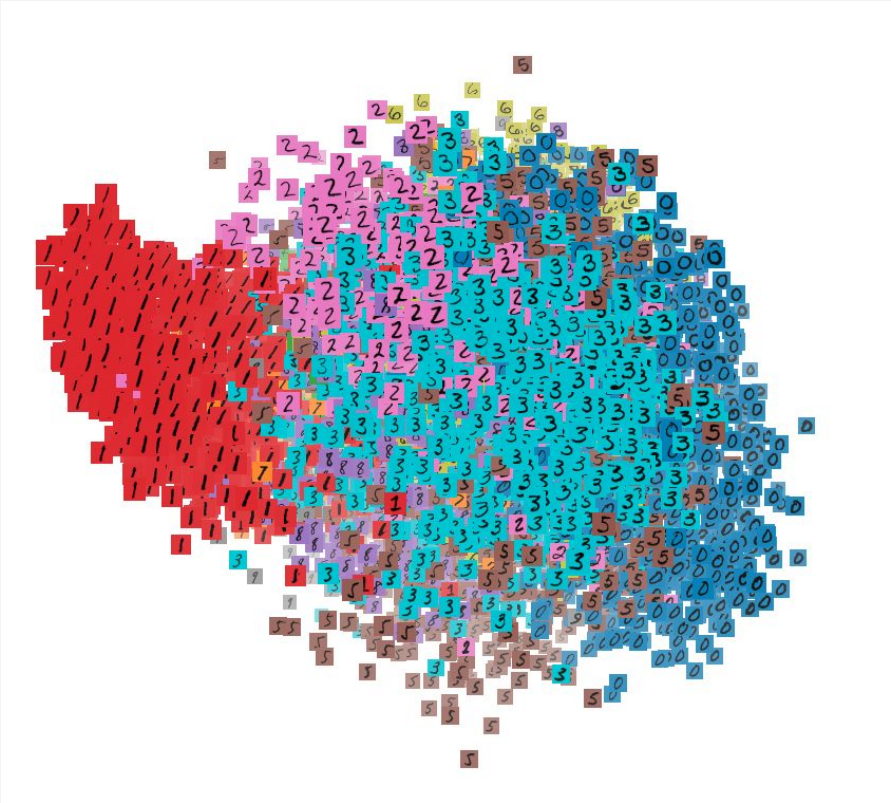
What is Machine Learning



一個神經元能做什麼？


Logistic Regression
1.0
0.8
0.6
0.4
0.2
0.0
(Probability)
sigmoid(X)
−4
−2
0
2
4
6
X
(Measurement)
Category 1 observations
Category 2 observations

很多神經元能做什麼？



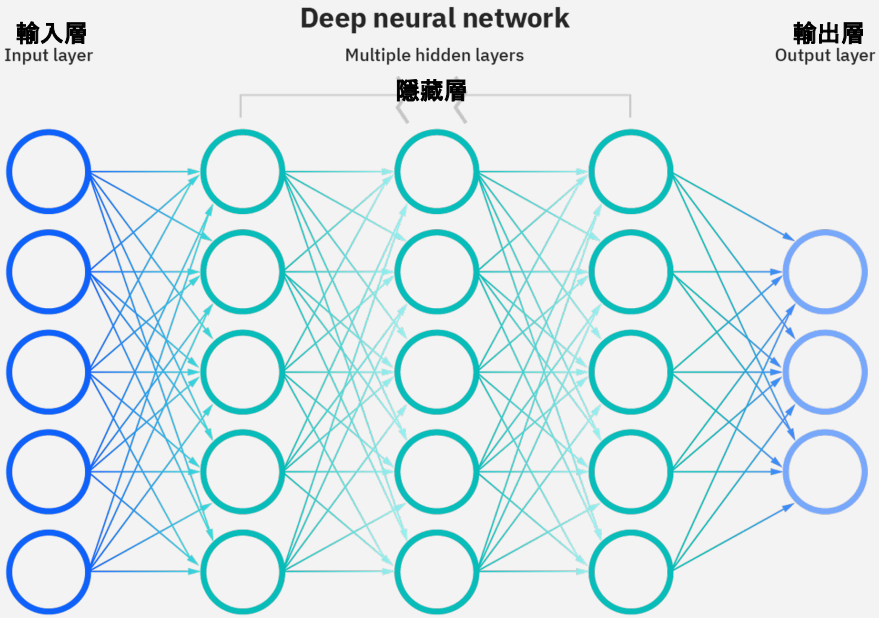
Reference:
http://projector.tensorflow.org/

Neural Network in Deep Learning

輸入層(Input Layer):
人類**輸入資料**的端口，如一張照片的每個像素點。

隱藏層(Hidden Layer):
數層的神經感知器，是高維度的空間，會讓訊號慢慢流向結果。

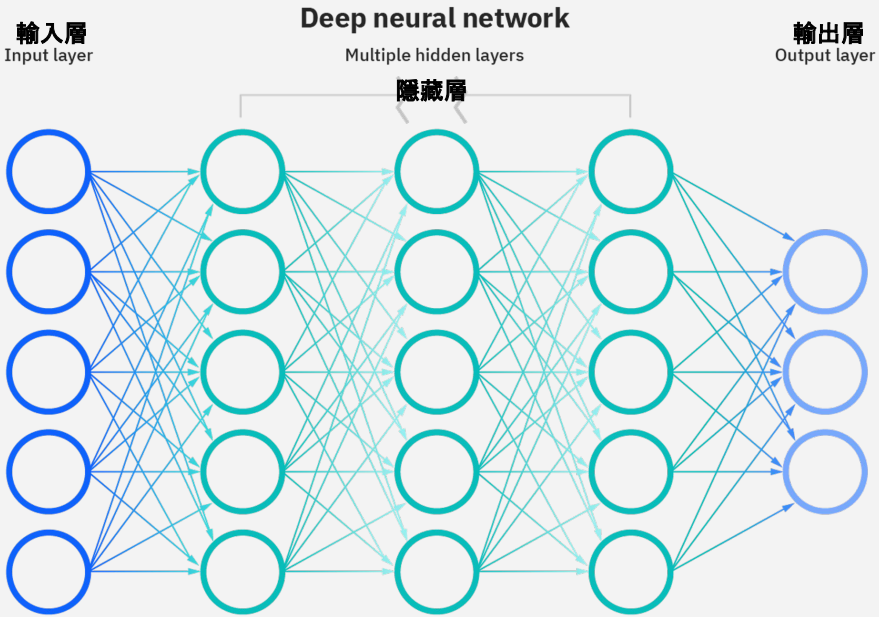
輸出層(Output Layer):
模型認知的**輸出結果**，例如是貓或是狗。



Neural Network in Deep Learning

當神經網路很多很多層時，人們給他一個比較厲害的名字**深度神經網路 (DNN, Deep Neural network)**。

當然，剛剛介紹的只是最單純的神經網路該念。後續延伸了很多種神經網路模型，如適合處理圖像的**CNN卷積神經網路**；最近夯起來的**GAN生成對抗網路**；還有可以進行語音辨識的 **RNN 遞歸神經網路**...等，我們這邊就不去多 說，因為那些數學很可怕的...



類神經網路的應用

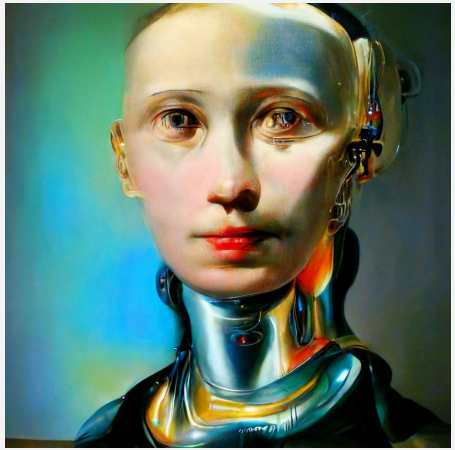
What for ?

機器學習最主要被用來做以下三件事情:

回歸
Regression

分類
Classification

生成
Generating



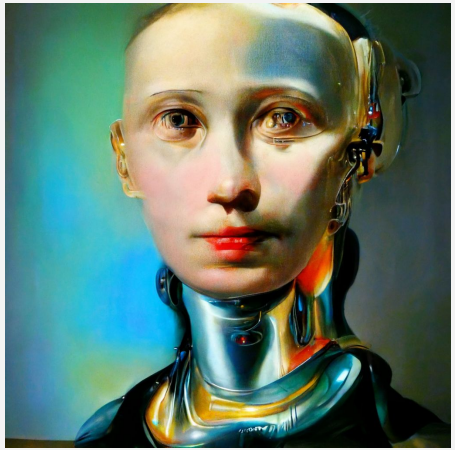
What for ?

這三週，我們會講到回歸和分類，至於生成的部份，就讓大家自己摸索了！

回歸
Regression

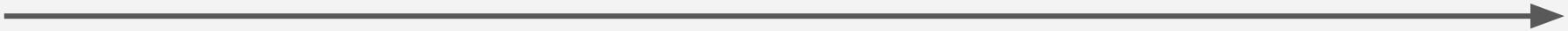
分類
Classification

生成
Generating



How to do?

從零開始架構一個自己的機器學習模型通常會經過一些步驟



資料前處理　選擇模型　訓練　驗證　執行

現在，請想像你是一位老師，眼前的這部機器是你的學生。

資料前處理 選擇模型 訓練 驗證 執行

我們蒐集了一些教材要來讓我們的模型學習，我們要先確保這些教材的**信度和效度是高的**。常見的資料前處理包含影像的縮放、平均、裁切，或是針對缺少或有漏洞的資料進行補齊。

資料前處理通常是決定一個模型好壞的關鍵因素，我們將會在 **Week #3** 時探討這個步驟。

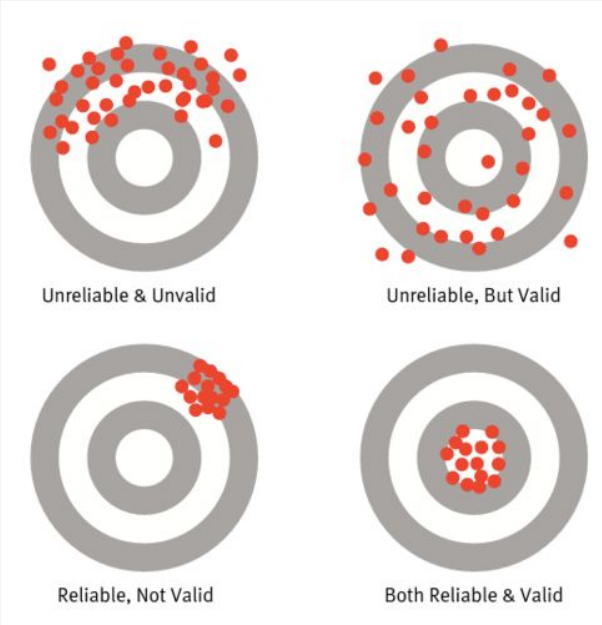
補充：信度和效度 (Week #3)

信度 (Reliability)：試驗結果的**穩定性及可靠性**

準確率時高時低，造成結果較無法被**信任**時，就是信度低。

效度 (Validity)：真實性和**準確性程度**

準確率低，就是效度低。



資料前處理 **選擇模型** 訓練 驗證 執行

就像剛剛說的，光是神經網路模型就有好多種，每種模型的應用都不一樣。**模型選擇正確才能擁有更高的效率。**像是你要中文系的學生算微積分；電機系的學生背四書五經一樣，有成功的可能性但效率不高。

資料前處理　選擇模型　**訓練**　驗證　執行

就像是不斷餵它資料的填鴨式教育過程。假設模型一開始的智商是0，透過一些訓練方法，我們可以逐漸提高它的智商，好比人類社會化的過程，**讓機器理解我們想讓他學習的事情**。

何謂訓練得好的模型，以及怎麼訓練它，我們將會在 **Week #2** 時探討這個步驟。

資料前處理　選擇模型　訓練　**驗證**　執行

訓練完成後，必須通過驗證我們才知道這個模型好不好。像是給它考試，若沒通過這個考試的話就只能重修了。(這時有可能是老師或是學生的問題，如果是老師的問題請重新進行前處理；如果是學生的問題請重新調整模型。)

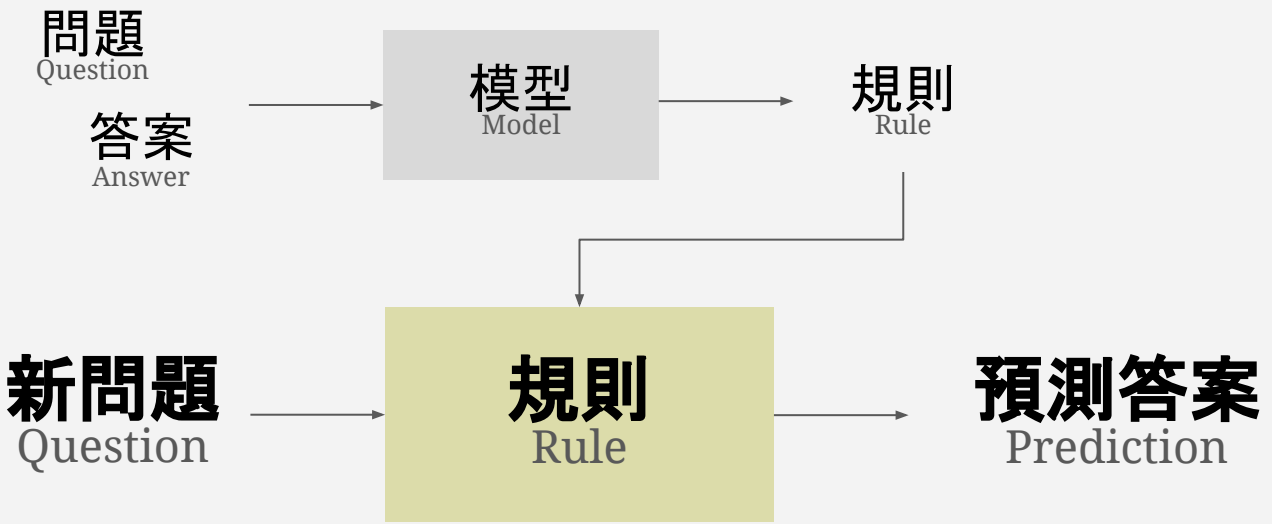
我們將會在 **Week #2** 時探討這個步驟。

資料前處理　選擇模型　訓練　驗證　**執行**

執行最容易了，如果你的模型通過了考試，那就可以發給他畢業證書，出社會工作了。

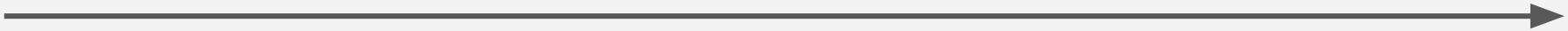
事實上，執行就是做預測，用之前學習到的**規則**推敲出一個沒有答案的結果。

What is Machine Learning



How to Deep Learning

我們今天學會執行 (a.k.a. **預測**) 就好了。



資料前處理　選擇模型　訓練　驗證　**執行**

休息一下！

別緊張，要來寫 Code 了

順便安裝以下套件吧：

```
pip install --upgrade pip setuptools wheel
pip install opencv-python opencv_contrib_python matplotlib
```

OpenCV

OpenCV 是一個開源的**電腦視覺**函式庫，提供影像處理、辨識，或是機器學習等各項功能。

我們今天的目標是直接利用別人訓練好的模型來做人臉識別。

補充：OpenCV v.s. Pillow

u/kigurai • Commented on 6 years ago

Pillow is an *image manipulation/processing* library while OpenCV is a *computer vision* library.

While there is certainly a lot of overlap (i.e. OpenCV contains a fair bit of image processing functionality) they are vastly different in scope.

To make an extreme simplification, you use Pillow if you want to cut and resize images, and maybe do a bit of filtering, and you use OpenCV when you are building a robot that is trying to "see" things.

Personally, as someone who use OpenCV more or less daily, but have never touched Pillow, I would probably use OpenCV for things Pillow does, simply because I am familiar with it. However, unless you have *specific reasons* so to use OpenCV, I would stay away since it can be extremely anoying and over complicated at times.

8 upvotes 1 replies

廢話不多說

若剛才還沒安裝這些東西的話，再給你們一次機會：

```
pip install --upgrade pip setuptools wheel
pip install opencv-python opencv_contrib_python matplotlib
```

OpenCV 顯示圖片

開一個新的 .ipynb 檔案，隨便找一張有臉的圖片和你的程式檔放在同一層目錄下，用 Open CV 開啟。另外，請**先看完下一頁再執行程式**。

```
import cv2

img = cv2.imread('./{你的檔案名稱}')    # 開啟圖片
cv2.imshow('img', img)              # 顯示圖片
cv2.waitKey(0)                      # 等待按下任意鍵
cv2.destroyAllWindows()             # 關閉視窗
```

OpenCV 顯示圖片

執行程式後你會發現，圖片成功開啟了。

請按任意鍵關閉他，**不要按叉叉關閉會當機**。

不小心按到的話重開 VS Code 就可以了。

OpenCV 顯示圖片

因為視窗顯示很麻煩，所以我們用 matplotlib 直接顯示在 notebook 當中。這樣就不會一直當機了

```
from matplotlib import pyplot as plt

img = cv2.imread('./{你的檔案名稱}')    # 開啟圖片
img_transform=img[:,:,::-1]      # BGR to RGB
plt.imshow(img_transform)        # 顯示圖片
plt.show()                       # 顯示圖片
```

OpenCV 顯示圖片

注意，OpenCV 的**色彩空間**是使用 BGR，反正就是 RGB 的反過來，所以如果要用 matplotlib 顯示的話要記得先轉換為大家熟悉的 RGB 色彩空間。

```
from matplotlib import pyplot as plt

img = cv2.imread('./{你的檔案名稱}')    # 開啟圖片
img_transform=img[:,:,::-1]      # BGR to RGB
plt.imshow(img_transform)        # 顯示圖片
plt.show()                       # 顯示圖片
```

OpenCV 顯示圖片

包成一個函式，到時候直接呼叫就很方便了。

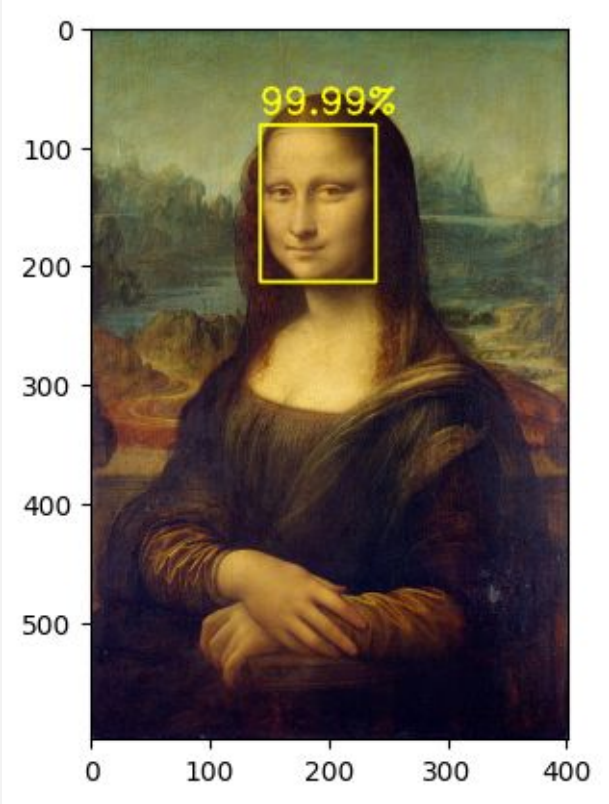
```
# 用 matplotlib 顯示圖片
def showImg(img):
    img_transform=img[:,:,::-1]      # BGR tp RGB
    plt.imshow(img_transform)        # 顯示圖片
    plt.show()                       # 顯示圖片
```

OpenCV 人臉偵測

OpenCV 提供了許多訓練好的特徵模型，下載後就能馬上使用。請下載這些檔案，並且和你的程式放在同一個目錄下：

1. 設定檔
https://github.com/opencv/opencv/blob/master/samples/dnn/face_detector/deploy.prototxt

2. 模型檔
https://github.com/sr6033/face-detection-with-OpenCV-and-DNN/blob/master/res10_300x300_ssd_iter_140000.caffemodel



OpenCV 人臉偵測

這個模型採用 CNN 神經網路，所以非常適合進行圖片辨識。

首先，這個模型訓練時有經過一個叫作「Blob」的**資料前處理**，所以要先將圖片轉換成 Blob。別擔心，等等會讓你們抄 code 。

載入模型之後，我們讓程式開始辨識人臉，並且自動標記出來。

Blob

Blob 分析 (Blob Analysis) 是在影像處理當中一個很重要的知識。背後的理論雖然有點複雜，且牽扯的範圍有點廣泛(如拓撲學、圖論、資料結構等，我也不會，有興趣請點以下連結閱讀[1])。

現在大家只要知道它會**標準化圖片**，且**標示出圖片的各個特徵還有形狀**就好了。

Reference:
[1] https://en.wikipedia.org/wiki/Connected-component_labeling

補充：DNN Face Detector in OpenCV

It is a **Caffe model**[2] which is based on the **Single Shot-Multibox Detector** (SSD) and uses **ResNet-10**[3] architecture as its backbone. It was introduced post OpenCV 3.3 in its deep neural network module.

Convolutional stage	Output size	Layer
Conv1	200×150	7×7, 64, stride 2
Conv2_x	100×75	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}$
Conv3_x	50×38	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}$
Conv4_x	25×19	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}$
Conv5_x	13×10	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}$

Reference:
[2] https://blog.csdn.net/FRD2009041510/article/details/53706735
[3] https://medium.com/...

OpenCV 人臉偵測

先把設定檔和模型載入，並且導入我們要辨識的照片。其中這個 min_confidence 我們等等會用到。

```
prototxt = "./deploy.prototxt"                                    # 設定檔
pretrained_model = "./res10_300x300_ssd_iter_140000.caffemodel"   # 模型檔
min_confidence = 0.5                                              # 最小信心
值
img = cv2.imread('./Mona_Lisa.jpg')                               # 開啟圖片
```

OpenCV 人臉偵測

進行 Blob 前處理，並且餵給模型讓他執行。

```
(h, w) = img.shape[:2]  # 取得圖片的寬高
model = cv2.dnn.readNetFromCaffe(prototxt, pretrained_model)  # 載入模型
# 資料前處理：將圖片 blob
blob = cv2.dnn.blobFromImage(cv2.resize(img, (300, 300)),
   scalefactor=1.0,
   size=(300, 300),
   mean=(104.0, 177.0, 123.0))
model.setInput(blob)  # 給模型輸入資料
detections = model.forward()  # 執行
```

OpenCV 人臉偵測

這個模型會回傳所有有特徵區域，**但這時不一定全部都是人臉**，所以我們要跑遍全部的 detections 逐一排查。我們設定當模型的**信心水準超過 min_confidence，才確定這個 detection 是一張人臉**。

```
# 處理結果
import numpy as np
for i in range(0, detections.shape[2]):
    confidence = detections[0, 0, i, 2]
    if confidence > min_confidence:
        # Do Something...
```

OpenCV 人臉偵測

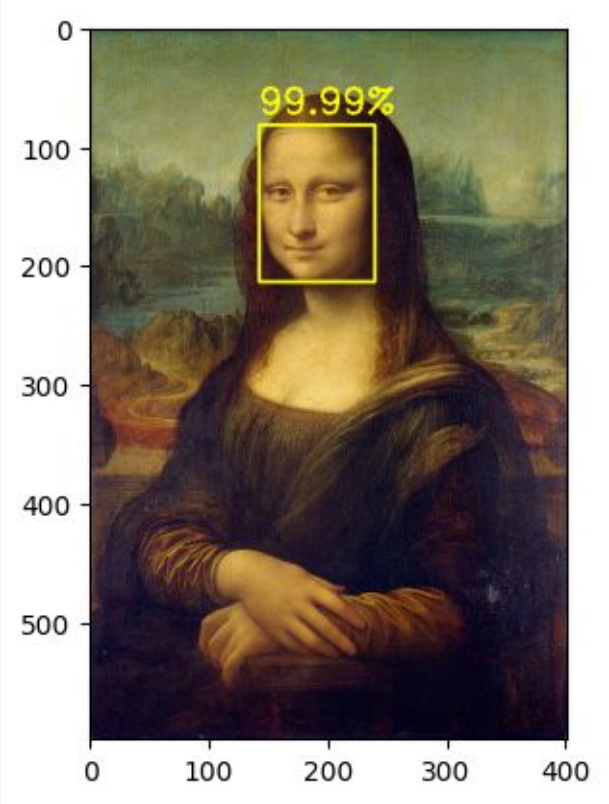
讓 OpenCV 把這張臉匡起來，並且印出他的信心值。

```
if confidence > min_confidence:
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
    (startX, startY, endX, endY) = box.astype("int")
    text = "{:.2f}%".format(confidence * 100)
    y = startY - 10 if startY - 10 > 10 else startY + 10
    cv2.rectangle(img, (startX, startY), (endX, endY),
    (0, 255, 255), 2)
    cv2.putText(img, text, (startX, y),
    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
```

OpenCV 人臉偵測

最後用剛剛寫好的顯示圖片函式進行渲染，大功告成。

```
# 渲染
showImg(img)
```



OpenCV 人臉偵測

等大家的同時想看看：為什麼要進行 Blob，不直接把圖片丟到神經網路裡就好了？

OpenCV 人臉偵測

把剛剛那堆包成一個函式，到時候直接呼叫就很方便了。

```
def faceDetect(img):
    # 略，請參考提供的程式碼
```

OpenCV 人臉偵測

所以, 剛剛的所有程式碼可以簡化成:

```
img = cv2.imread('./Mona_Lisa.jpg')       # 開啟圖片
faceDetect(img)                           # 偵測人臉
showImg(img)                              # 顯示圖片
```

OpenCV 人臉偵測

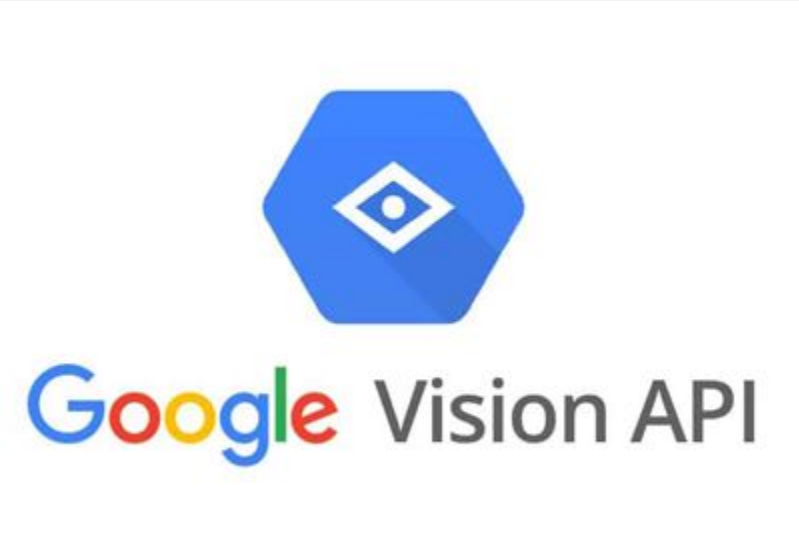
大家可以發現，**模型是說「我有多少%的信心認為這是一張臉」，而不是說「這就是一張臉」**。這個概念很重要，因為在機器學習的世界，結果都是以**機率分佈**呈現的。

OpenCV 人臉偵測

多張人臉也是沒問題的！

```
img = cv2.imread('./img_multi.jpg')
# 自己試看看囉
```

Bonus: 淺談 API



https://cloud.google.com/vision?hl=zh-tw

Bonus: 淺談 API

OpenAI

Google Cloud

IBM

Azure

回家功課

1. 有空的話可以到 https://cloud.google.com/vision?hl=zh-tw 玩看看 Vision API 。
2. 今天做的小專案算是用到機器學習的回歸、分類還是生成？
3. 分享一個你覺得人工智慧**可以**做到的事情和一個**不可能**做到的事情
4. 你認為人工智慧將會如何發展？

歡迎在課程意見回饋表與我們分享你的想法！

補充：今天學到的專有名詞

人工智慧：Artificial Intelligence, AI

深度學習：Deep Learning, DL

機器學習：Machine Learning, ML

深層神經網路：(Deep) Neural Network, (D)NN

神經元：Neuron

權重：Weights

活化函數：Activation Function

補充：今天學到的專有名詞

輸入層：Input layer

隱藏層：Hidden layer

輸出層：Output layer

回歸：Regression

分類：Classification

生成：Generating

補充：今天學到的專有名詞

資料前處理：Data Preprocessing

建模：Modeling

訓練：Training

驗證：Validation

預測：Prediction