第 7 週 - 機器學習-圖片辨識

大 綱
Outlines

01 Convolutional Neural Network (CNN)、loss function

02 使用 keras 搭建 CNN

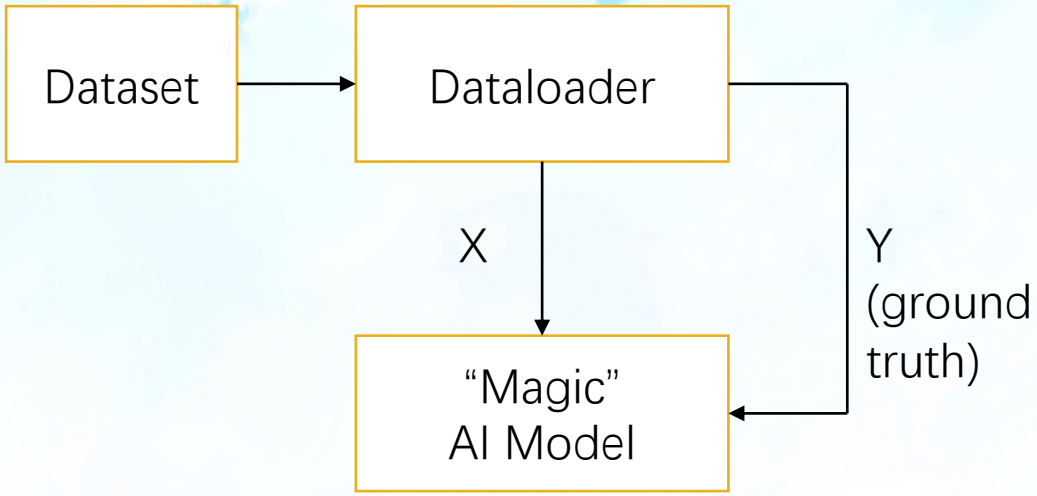
03 實作：建置資料集

01

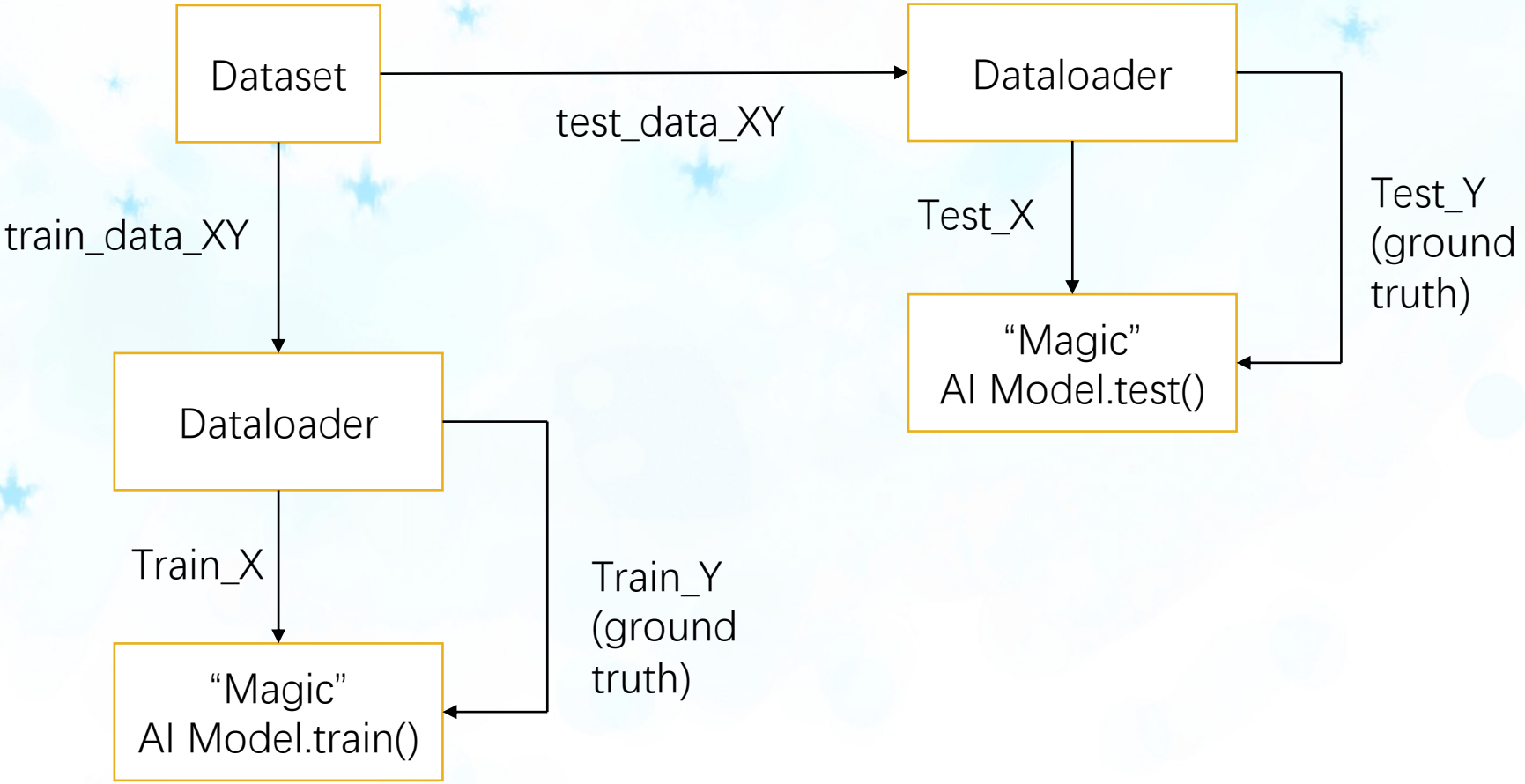
Convolutional Neural Network (CNN)

來源：[魔法陣系列] Convolutional Neural Network（CNN）之術式解析

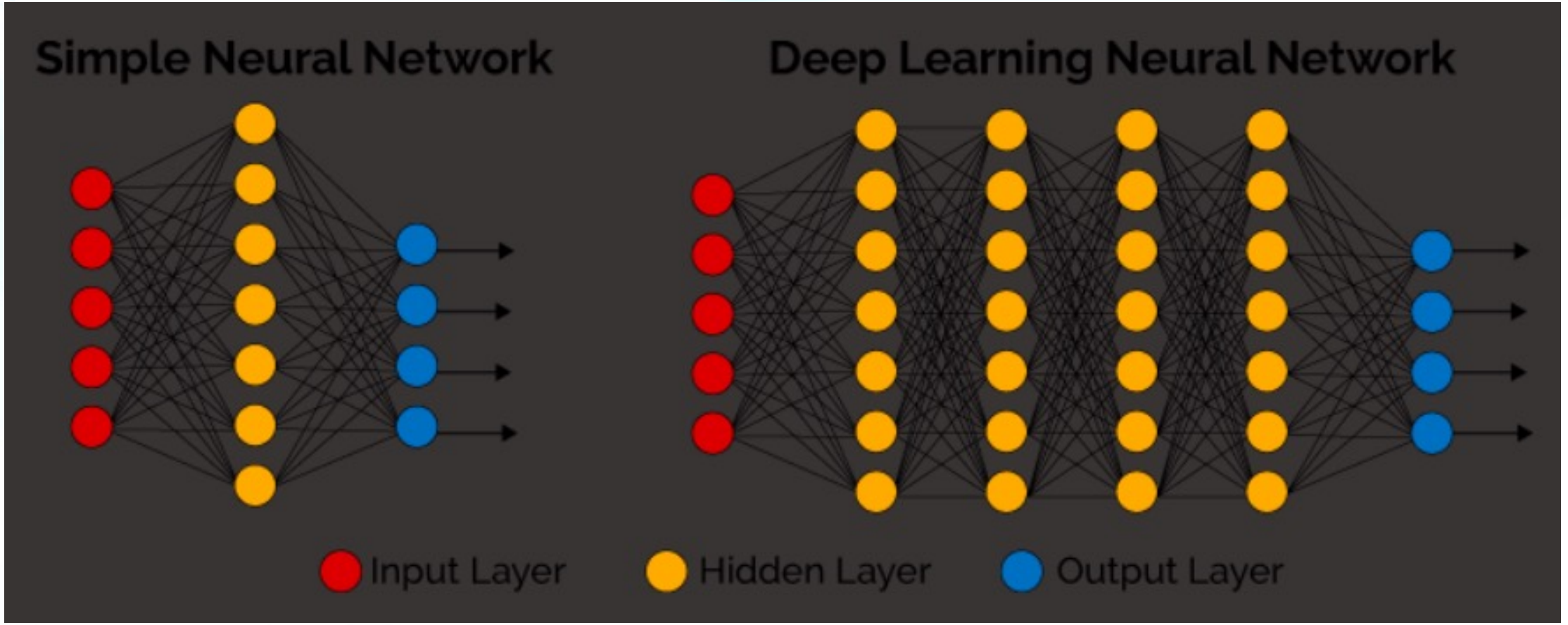
Training process



Training process



Convolutional Neural Network



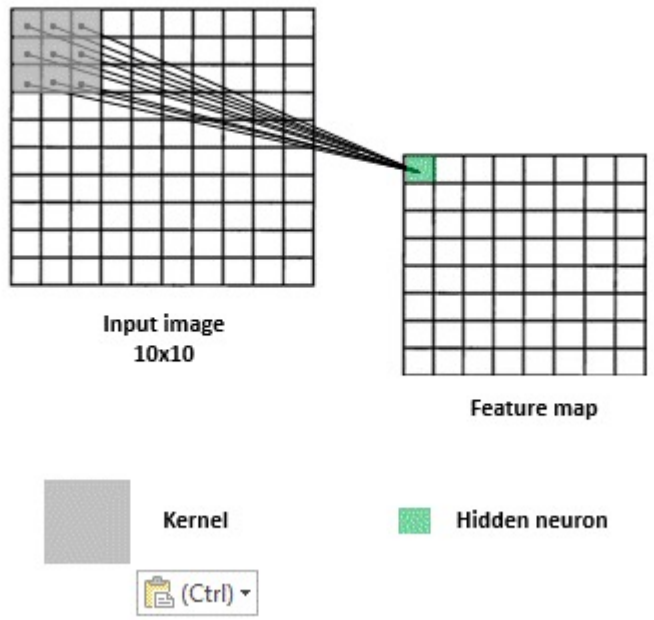
Convolutional Neural Network

- CNN 主要由下列概念所組成：

1. Convolution Operation
2. Pooling
3. Fully Connected Networks
4. Activation

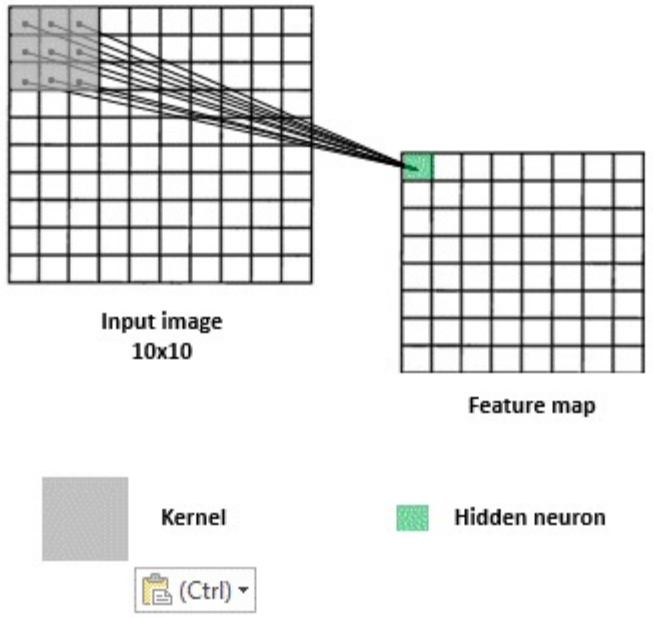
1. Convolution Operation

- 神經網絡隨機生成 Kernel（又稱 Filter）來抓取不同的特徵，將特徵存入 feature maps 中，再透過訓練決定哪些類型是重要的。Filter 在過程中可將圖片變小，使得圖片能夠更容易以及快速被神經網絡處理。



在這過程中是否會遺失掉訊息？

- 答案：會的。
- 不過就如同前面探討的圖片，實際上我們在看圖時不會每個 pixel 都看，例如可能只看眼睛、鼻子等特徵（feature），而這些特徵被保存在 feature map 中



Filter 抓取特徵

Sharpen

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

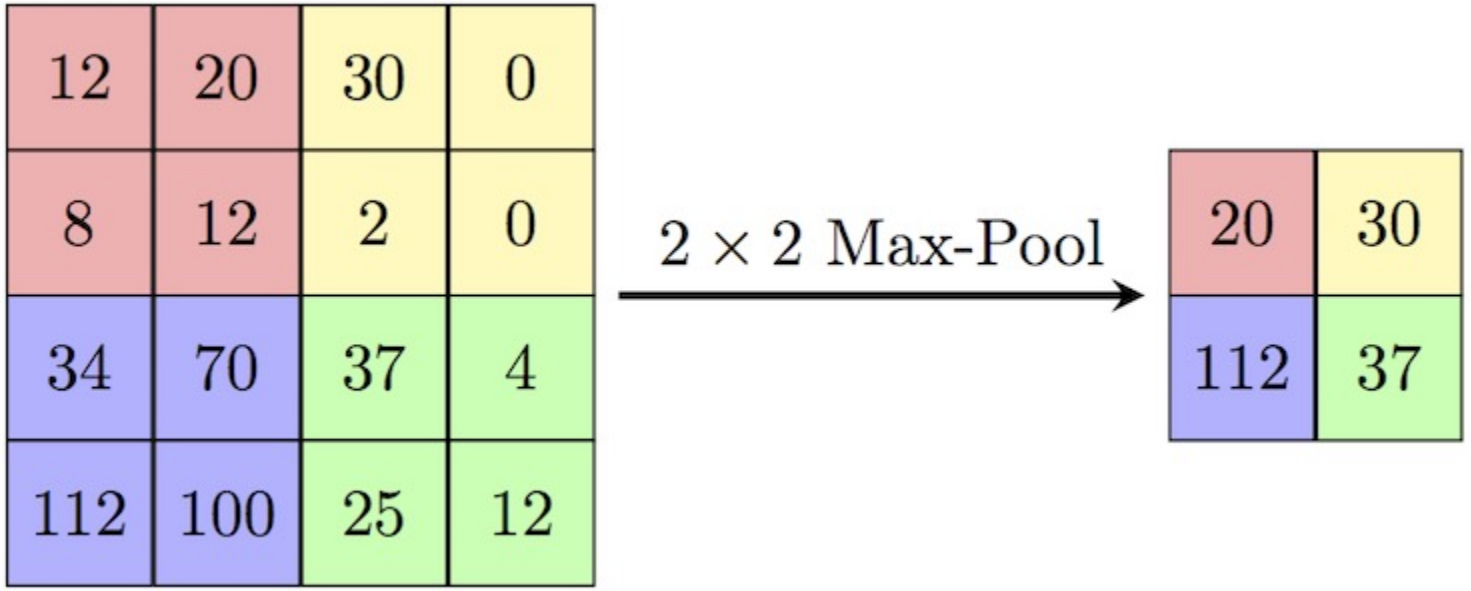
Blur

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

2. Pooling

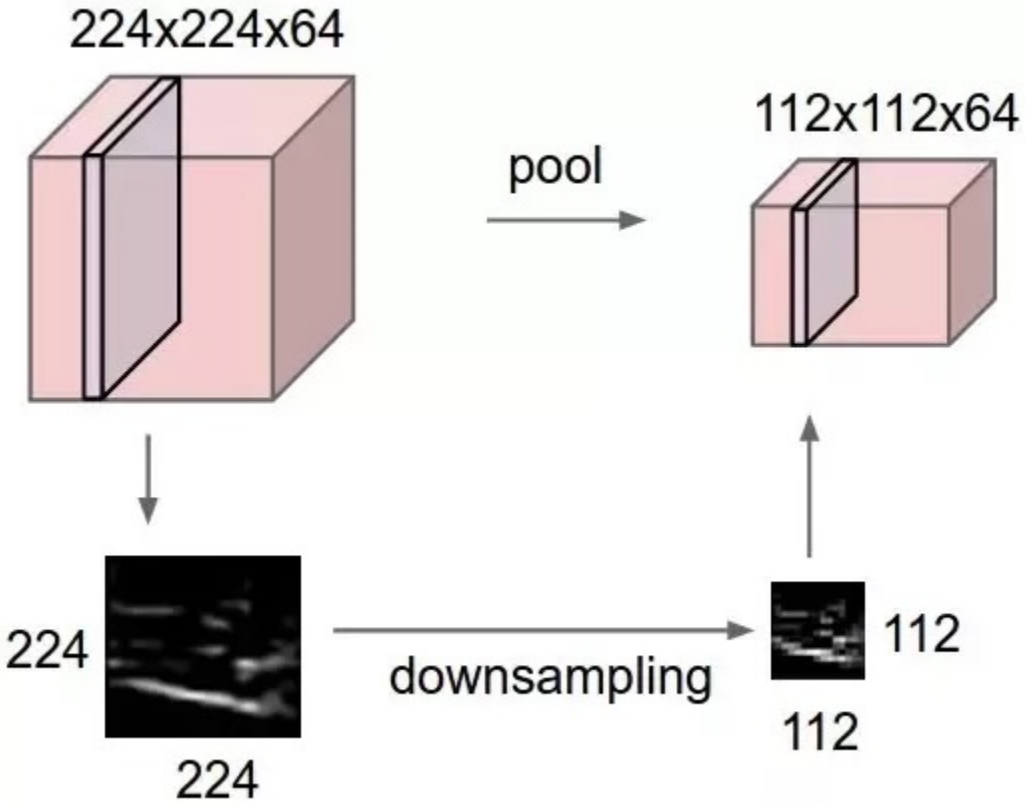
- 電腦不如人類聰明，同一張圖片在不同的角度、光線、質地、部位會使電腦判斷成是完全不同的特徵，如何讓神經網絡能夠聰明一點，彈性的處理這些特徵？這就是 Pooling 的功能了，它有不同種形式：

1. Mean pooling
2. Max pooling
3. Sum pooling



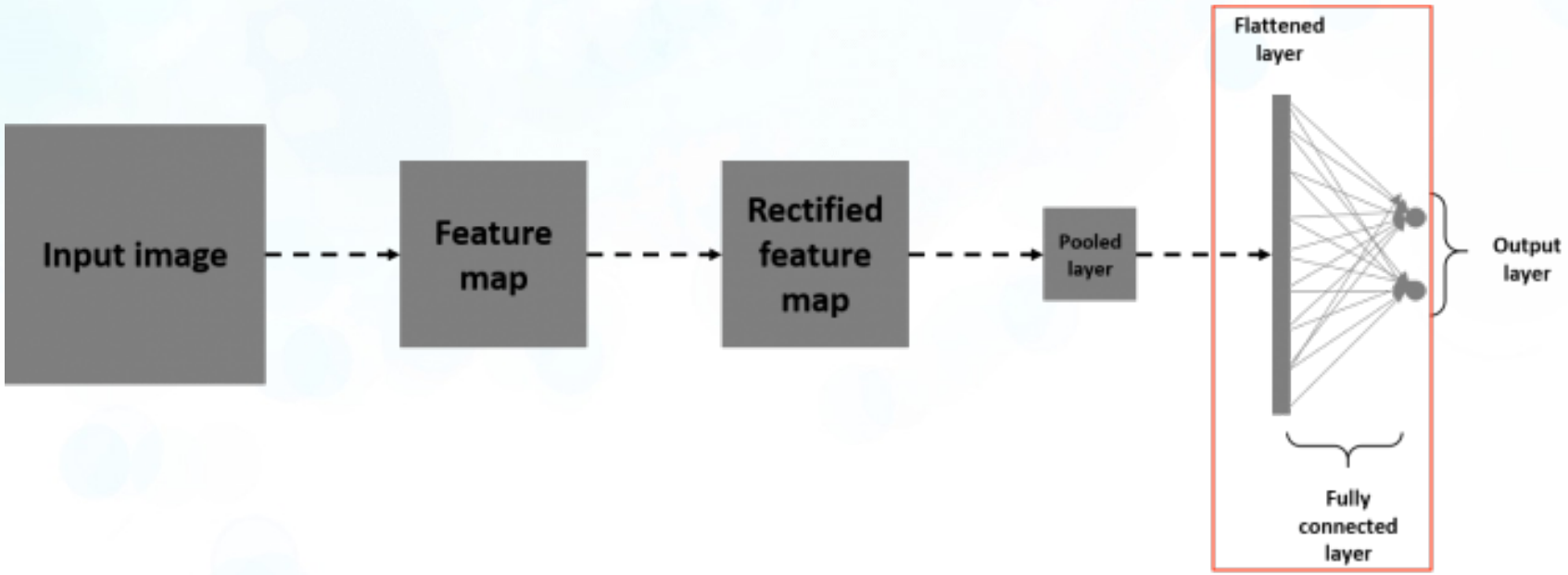
Max pooling

- 如圖所示，抓出矩陣中的最大值。Max pooling 擁有去雜訊的功能，且當圖片平移的話也不會影響電腦的判斷。



3. Fully Connected Networks

- 這邊的全連接層將前一層的輸出做平坦化（Flatten）攤平合併到一個大向量內。



4. Activation

- 激活函數（Activation functions）對於人工神經網絡模型去學習、理解非常複雜和非線性的函數來說具有十分重要的作用。
- 它們將非線性特性引入到我們的NN中，使網路能理解更為複雜的事情。

- 常見的activation

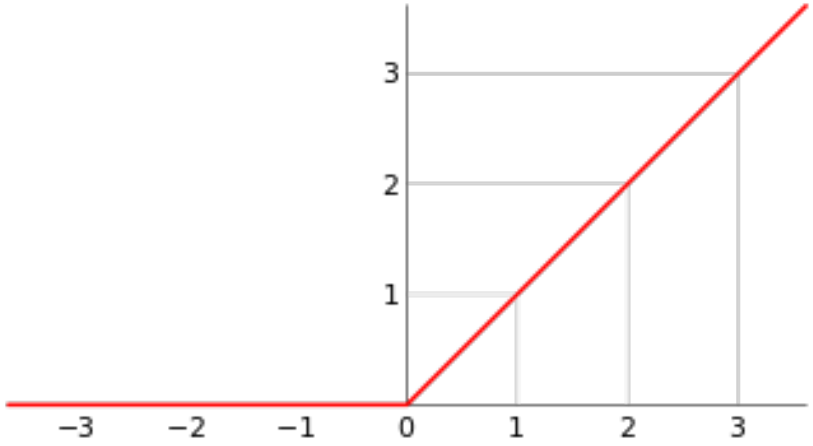
1. ReLU
2. Sigmoid
3. Softmax

1. ReLU

- 優點
  - 避免了梯度爆炸和梯度消失問題
  - 簡化計算過程

- 常應用於隱藏層

$$f(x) = \max(0, x)$$

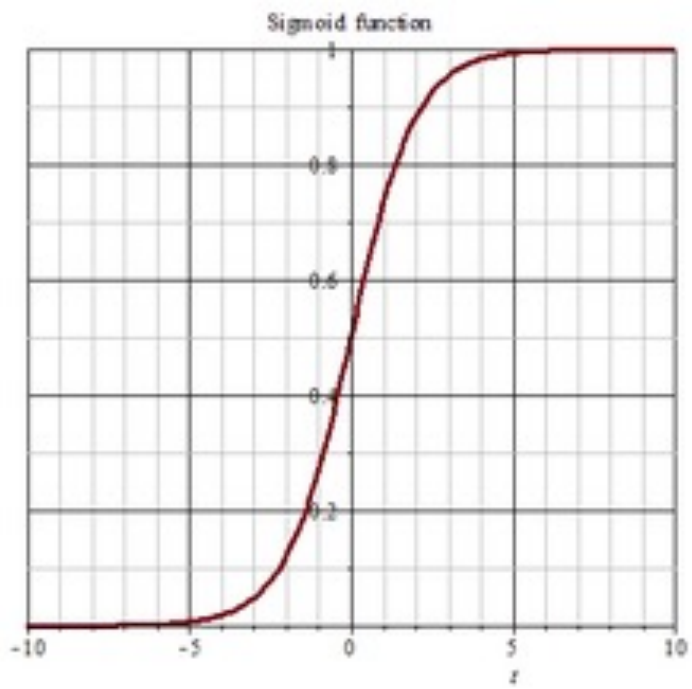


2. Sigmoid

- 特性
  - 將輸出結果壓縮至0~1間

- 常應用於二元分類的輸出層

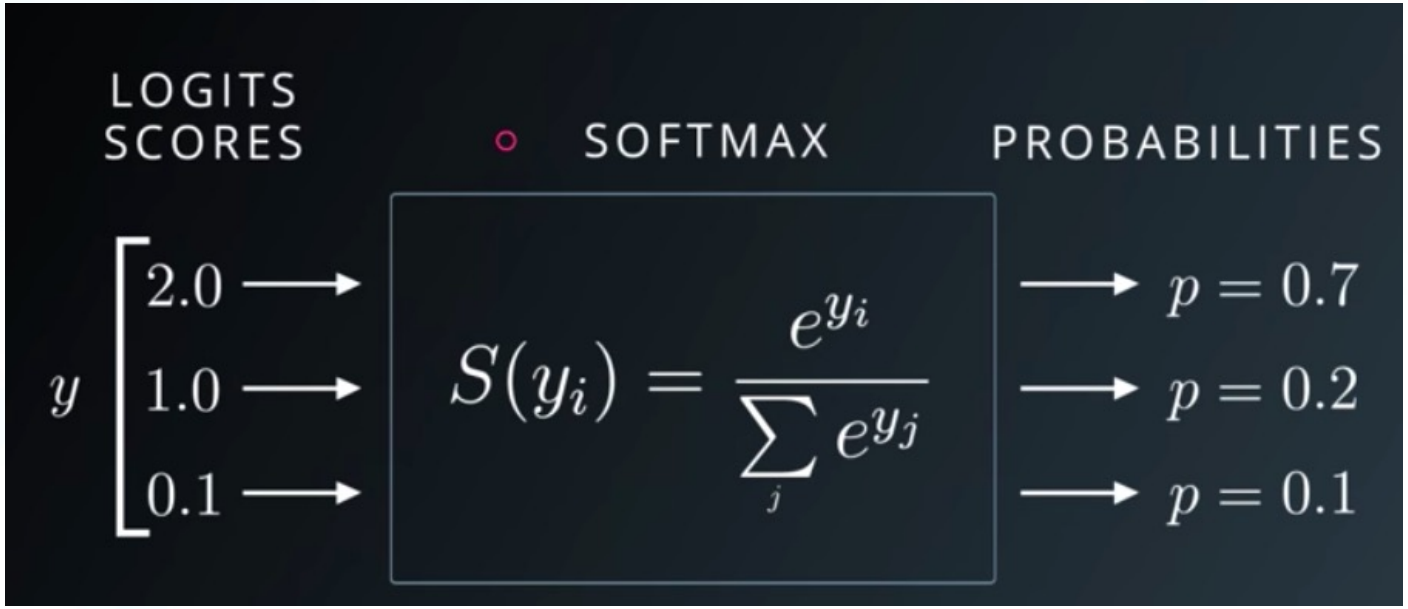
$$S(t) = \frac{1}{1 + e^{-t}}.$$



3. Softmax

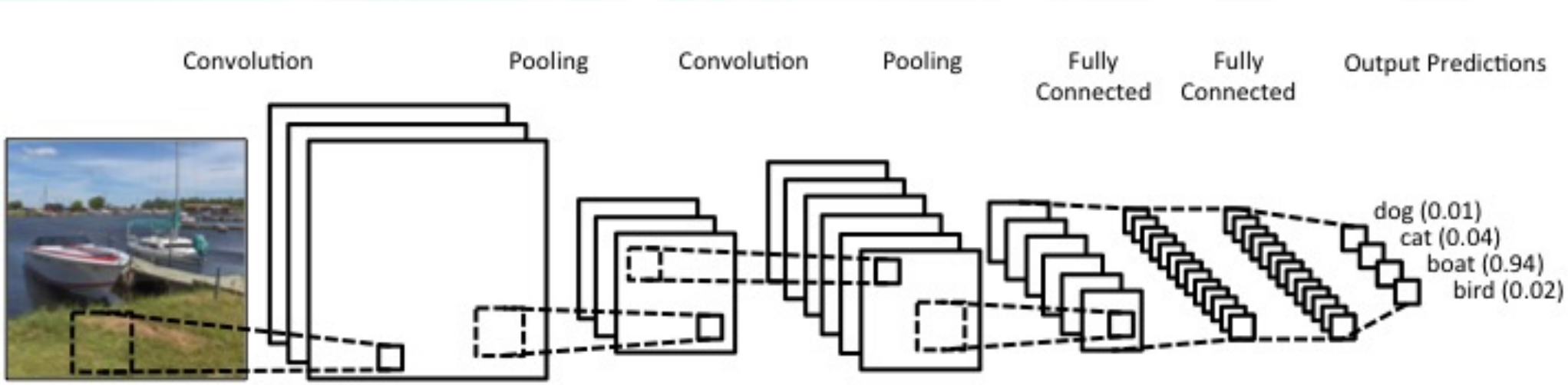
- 特性
  - 將輸出結果壓縮至0~1間，且所有結果相加為1

- 常應用於多元分類的輸出層



Convolutional Neural Network

- 全連接層（Fully Connected Layer）起到分類任務的作用，而前面的層充當特徵提取器，整個 CNN 的流程如下圖：



Math inference

- Softmax/Argmax
- Shannon’ s entropy

$$H(\mathrm{x}) = \mathbb{E}_{\mathrm{x}\sim P}[I(x)] = -\mathbb{E}_{\mathrm{x}\sim P}[\log P(x)]$$

$$H(p) = E_{x\sim p}[-\ln p(x)]$$

$$\sum_i -p_i \ln p_i$$

Math inference

- Softmax/Argmax
- Shannon’ s entropy
- Cross entropy loss

$$H(p, q) = E_{x \sim p}[-\ln q(x)]$$

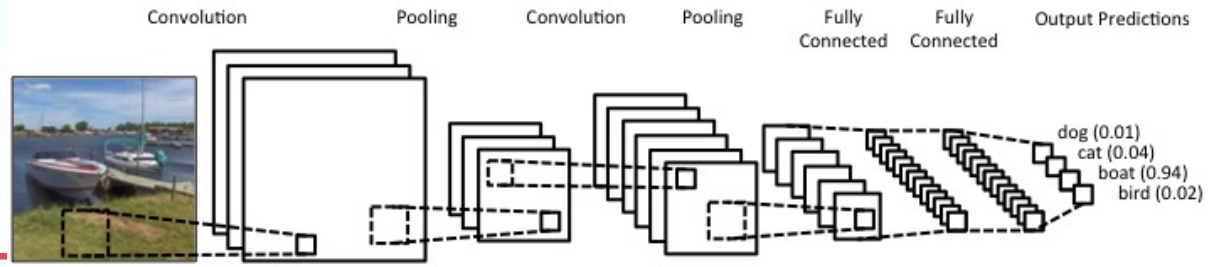
$$\text{Loss} = -\frac{1}{\substack{\text{output} \\ \text{size}}} \sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

02

使用 keras 搭建 CNN

來源：[實戰系列] 使用 Keras 搭建一個 CNN 魔法陣（模型）

搭建CNN模型



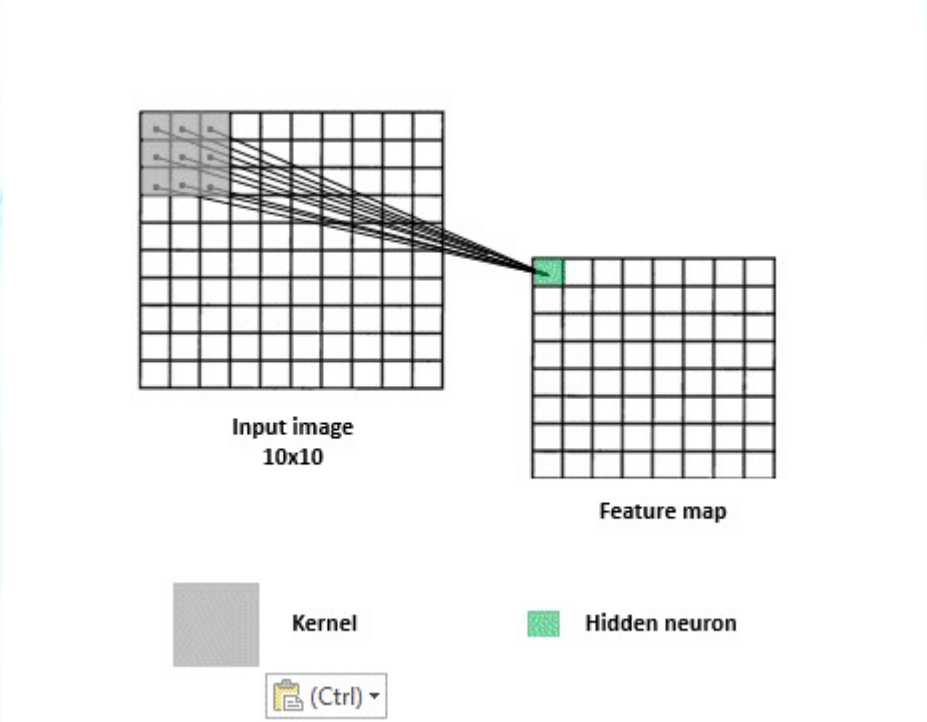
[4]:

```
from keras.models import Sequential
from keras.layers import InputLayer, Conv2D, MaxPool2D, Flatten, Dense

model = Sequential()
model.add(InputLayer(input_shape = (128, 128, 3)))
model.add(Conv2D(32, 3, activation = 'relu'))
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
```

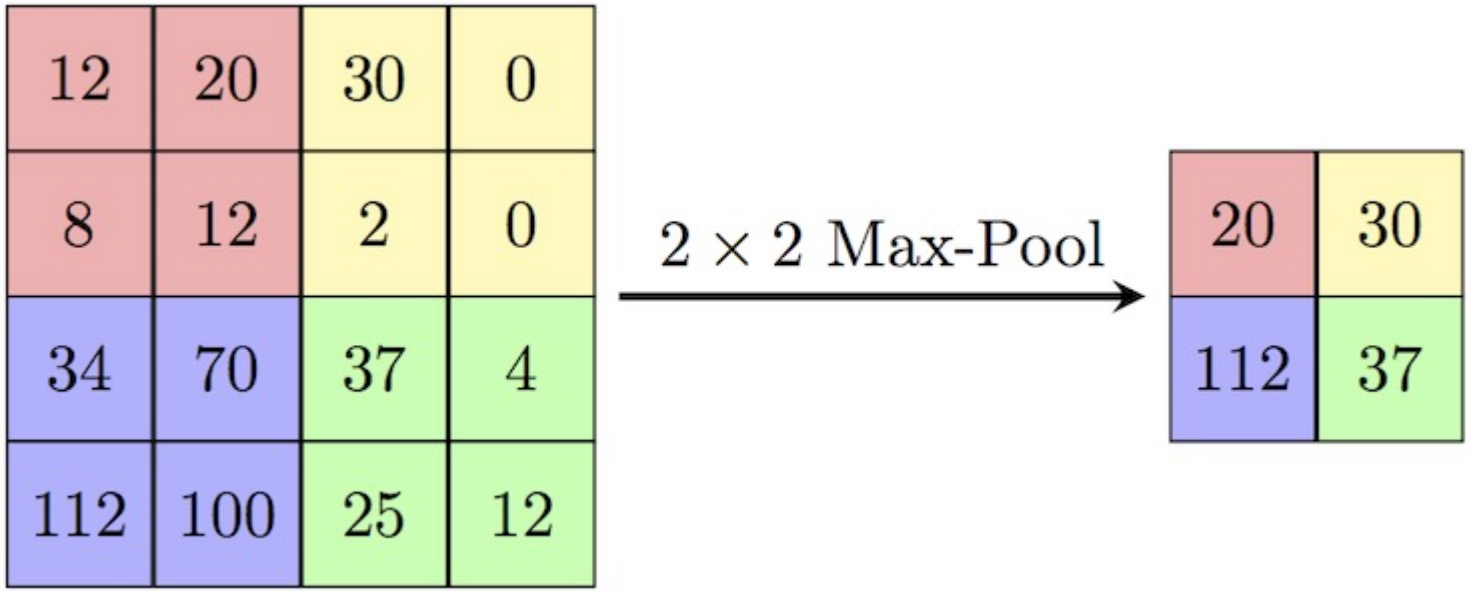
+ Code + Markdown

Conv2D



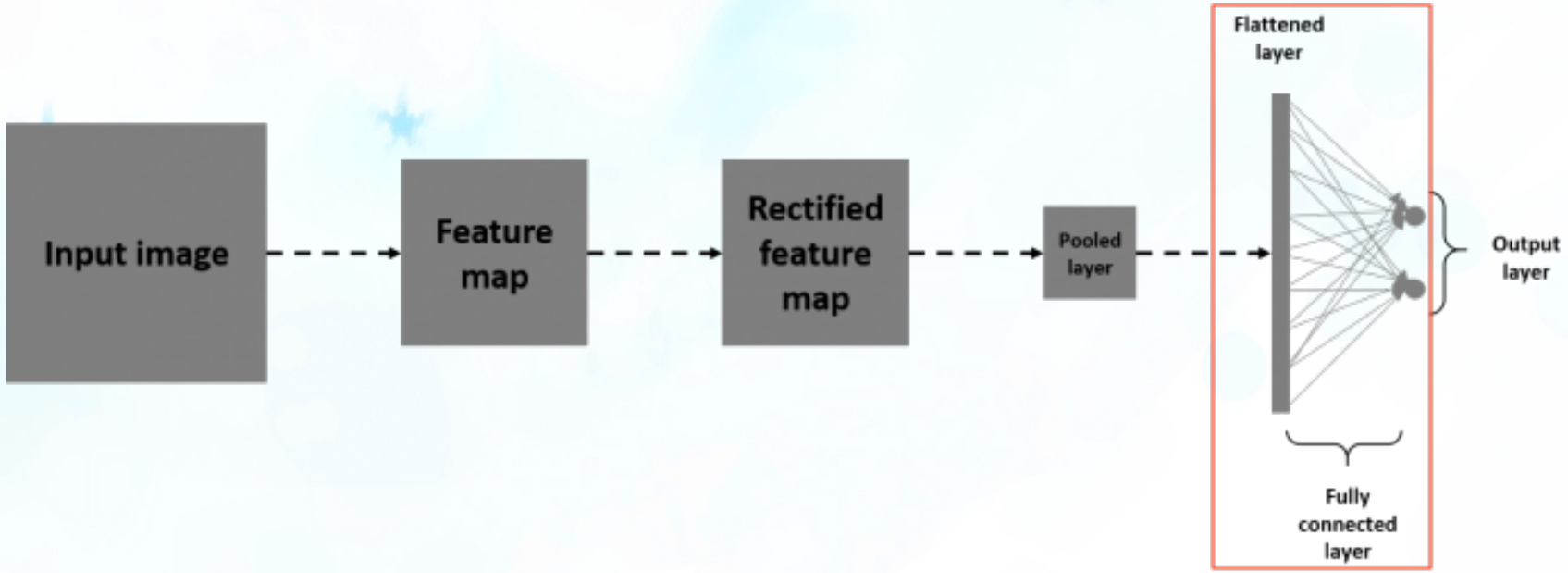

```
model.add(Conv2D(32, 3, activation = 'relu'))
```

MaxPool2D




```
model.add(MaxPool2D(pool_size = (2, 2)))
```

Flatten & Dense




```
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
```

編譯、訓練、應用CNN模型

```
model.compile(optimizer="rmsprop",loss='categorical_crossentropy', metrics=['acc', 'mse'])
model.fit( x=None,y=None,batch_size=None,epochs=1,callbacks=None,
          validation_split=0.0,validation_data=None,shuffle=True)
model.evaluate(x=None,y=None)
result = model.predict(x)
```

03

實作：建構CNN模型

Customized Dataset

- Colab: https://colab.research.google.com/drive/1rake9kGLVHNRfA3dDBhuZ9RI3k_fJrA7
- Google drive: https://drive.google.com/file/d/1rake9kGLVHNRfA3dDBhuZ9RI3k_fJrA7/view?usp=sharing