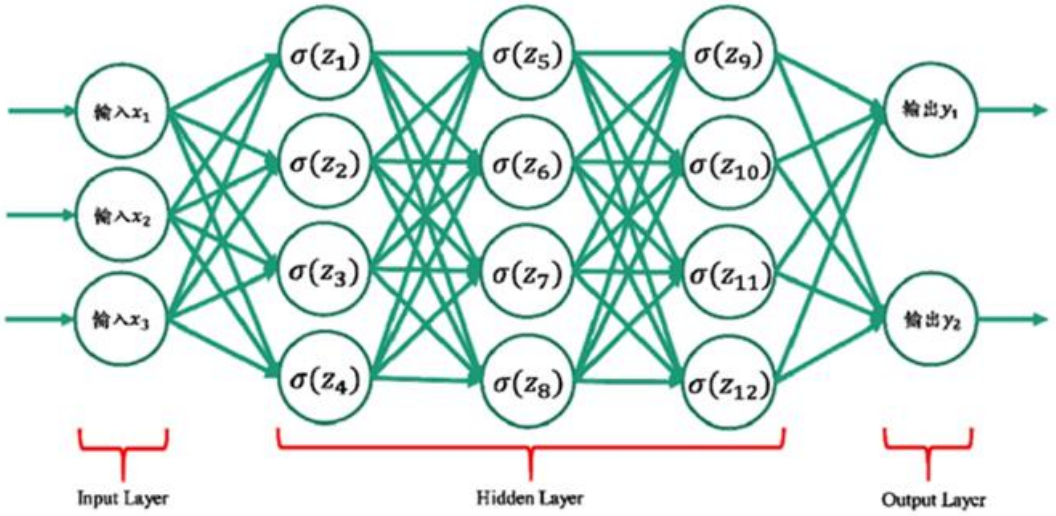
卷積神經網路
Convolutional Neural Networks

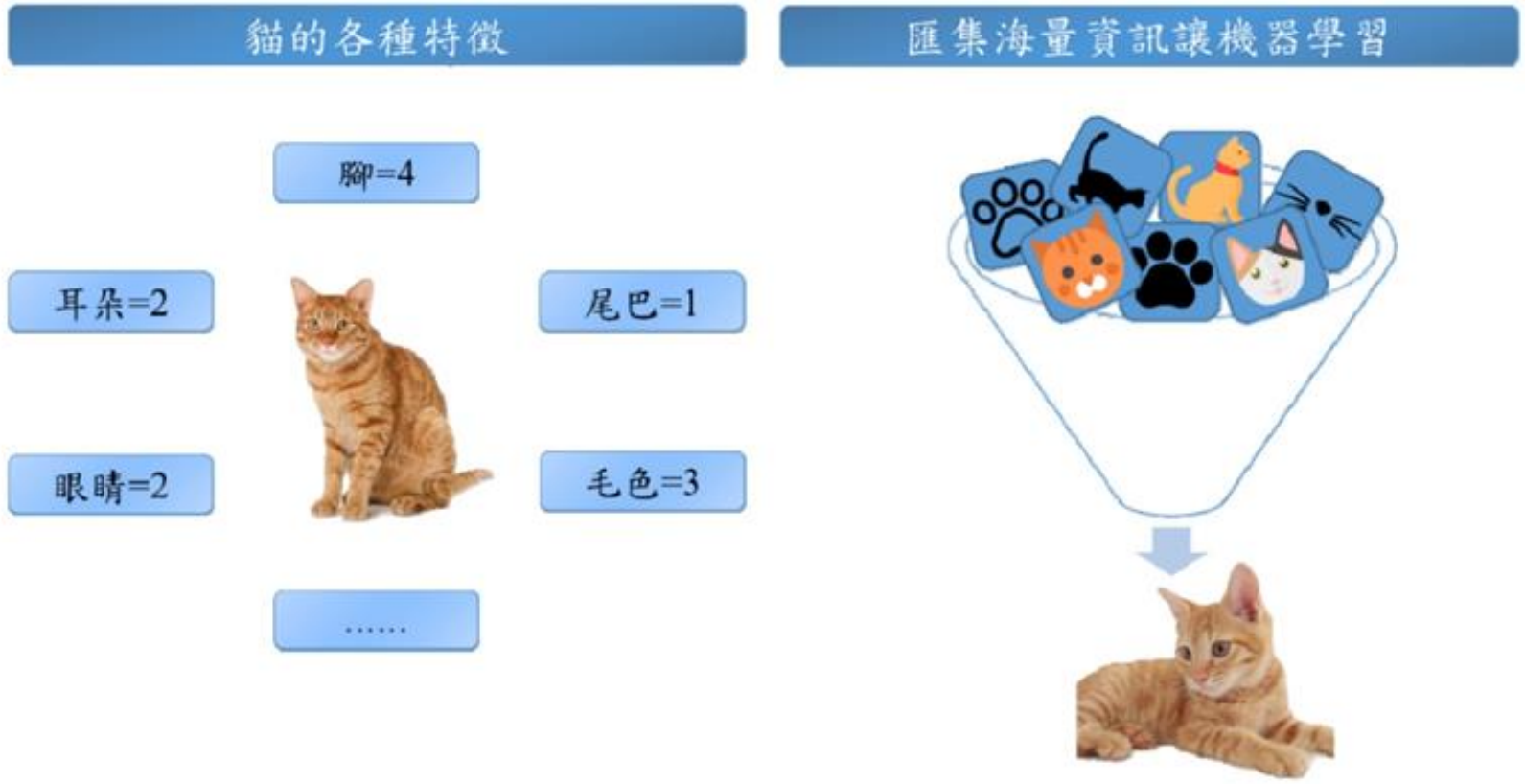
類神經網路的架構

- **輸入層**(Input Layer)是最左邊的第一層，其輸入資料的個數決定了該層神經元的個數。
- **輸出層**(Output Layer)是最右邊的最後一層，用來輸出分類或預測的結果。
- **隱藏層**(Hidden Layer)則介於輸入層和輸出層中間。輸入層接收資料後，傳給隱藏層層的每個神經元進行非線性的運算；其中，上一層的每一個神經元所產出的輸出，都會做為下一層的每個神經元的輸入。



神經網路架構

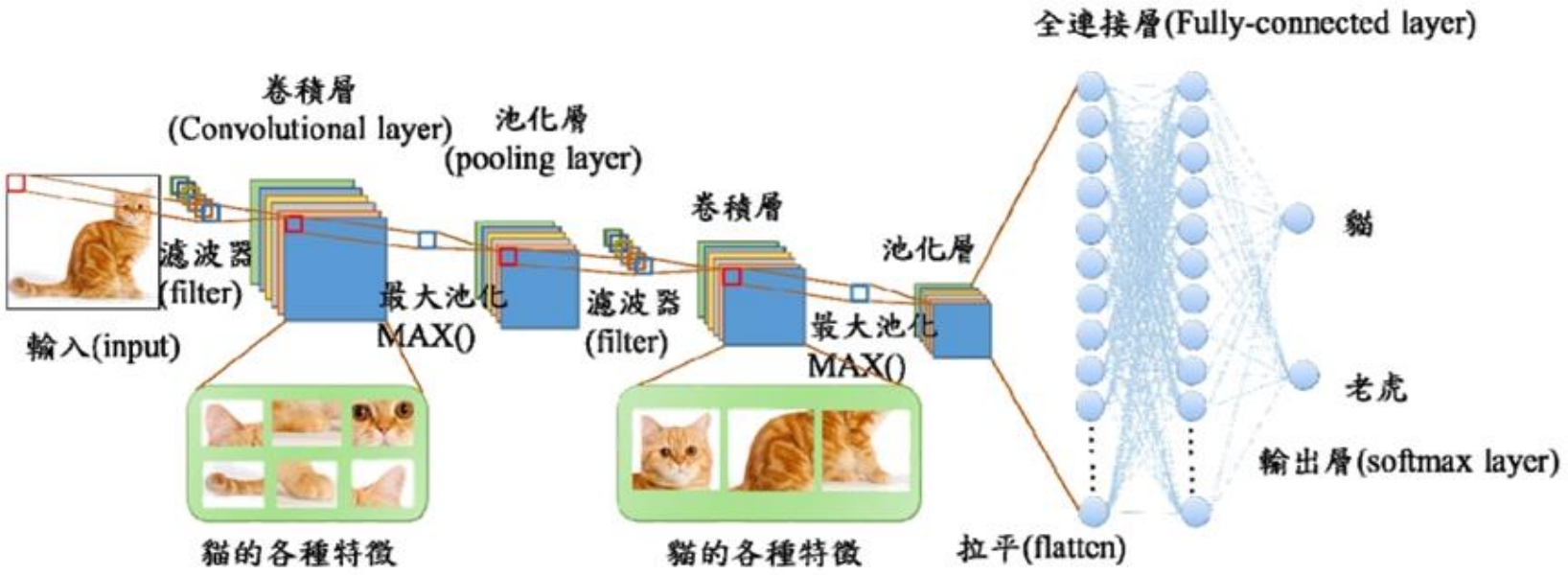
卷積神經網路-CNN



機器如何辨識貓

CNN架構

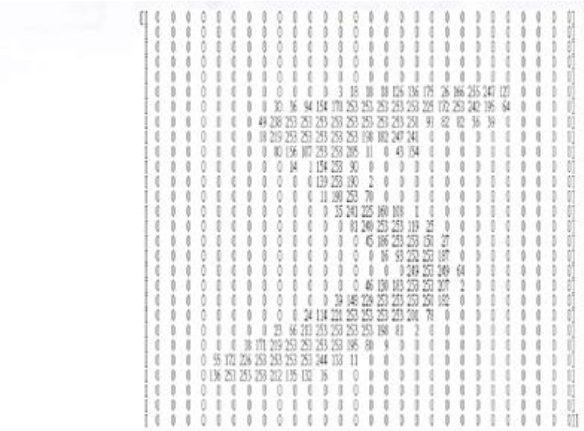
- 輸入層
- 隱藏層
  - 卷積層
  - 池化層
  - 全連接層
- 輸出層



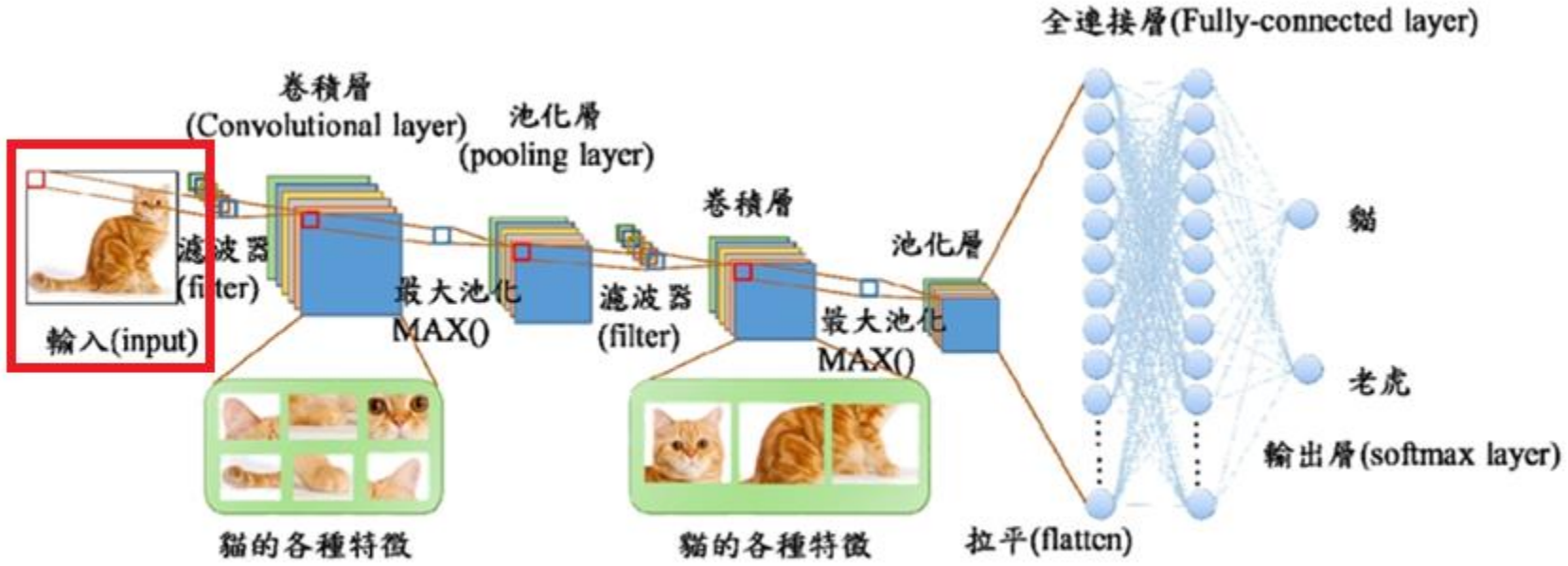
卷積神經網路架構圖

CNN架構

- 輸入層（Input Layer）



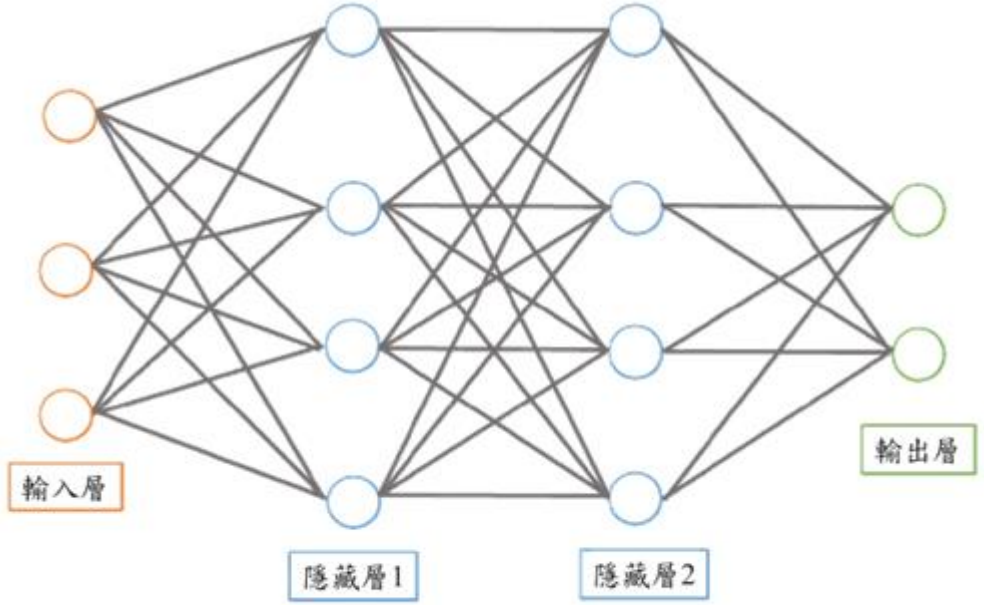
MNIST 手寫數字資料集中，第一筆資料及標籤



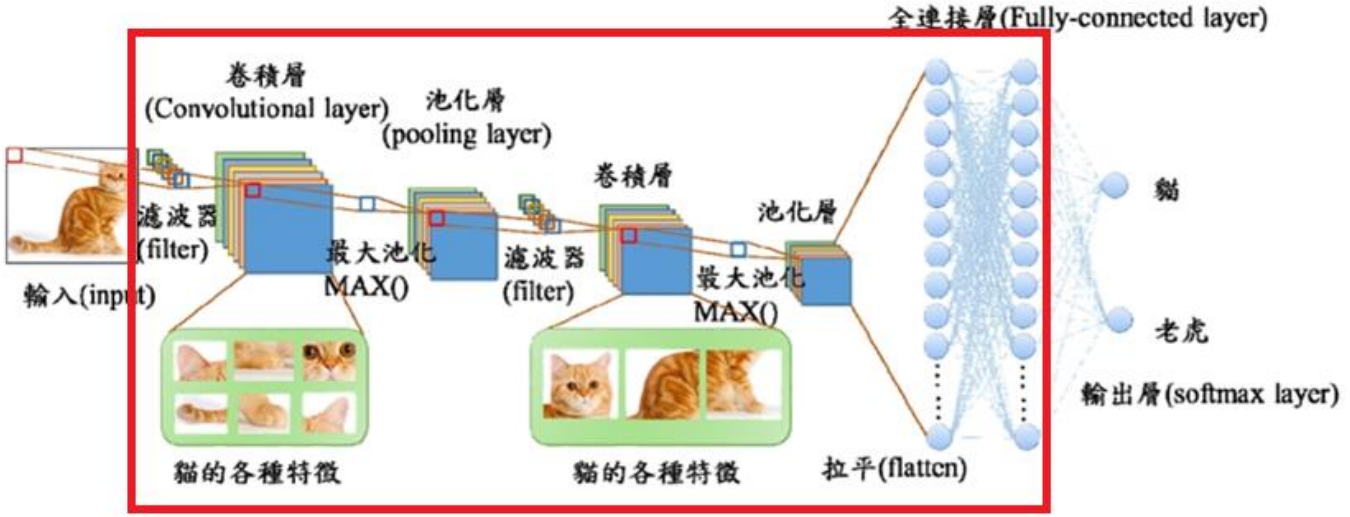
卷積神經網路架構圖

CNN架構

- 隱藏層（Hidden Layer）
  - 卷積層
  - 池化層
  - 全連接層



神經網路基本架構圖



卷積神經網路架構圖

CNN架構

- 隱藏層（Hidden Layer）
  - 卷積層
    - 濾波器(filter)
    - 特徵圖(feature map)
    - 激活函數(activation function)

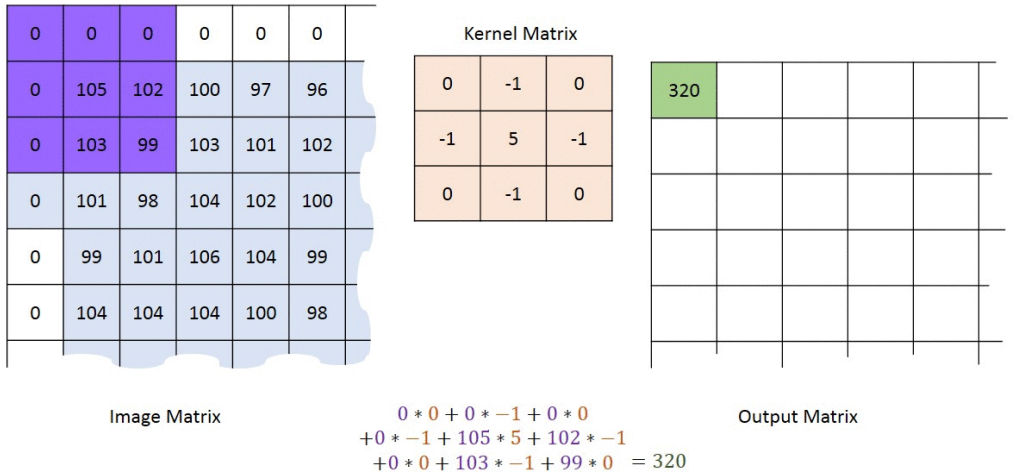
4.2.3:CNN架構

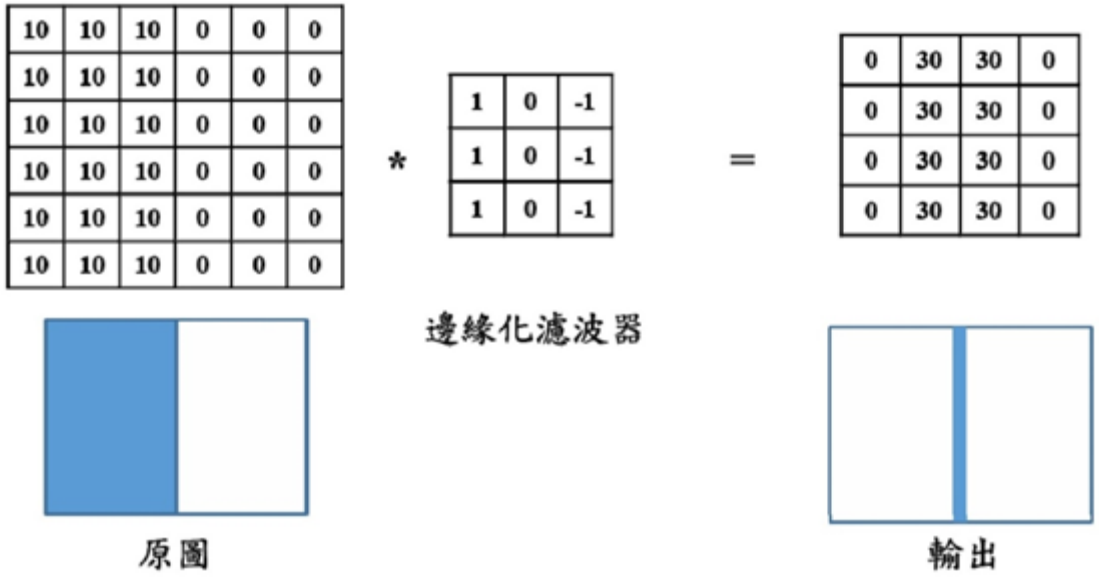
- 隱藏層（Hidden Layer）
  - 卷積層
    - 濾波器(filter)

透過濾波器處理圖像

CNN架構

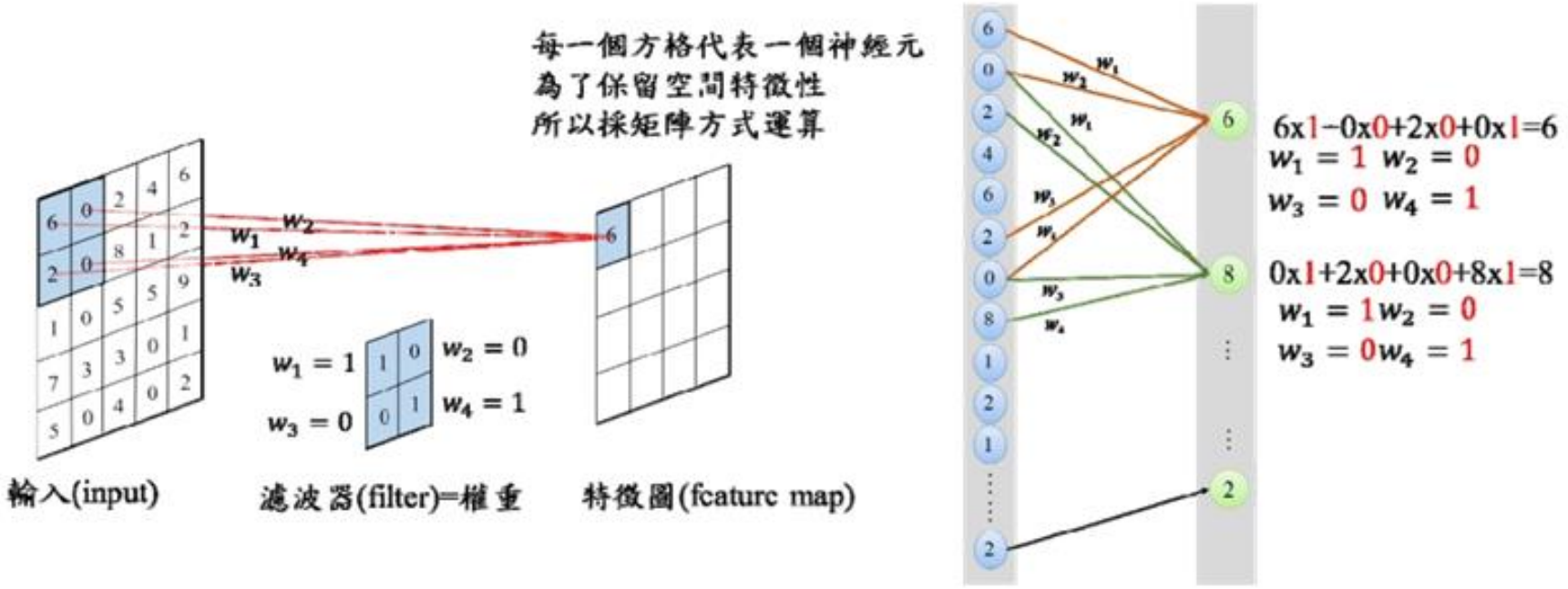
- 隱藏層（Hidden Layer）
  - 卷積層
    - 濾波器(filter)





CNN架構

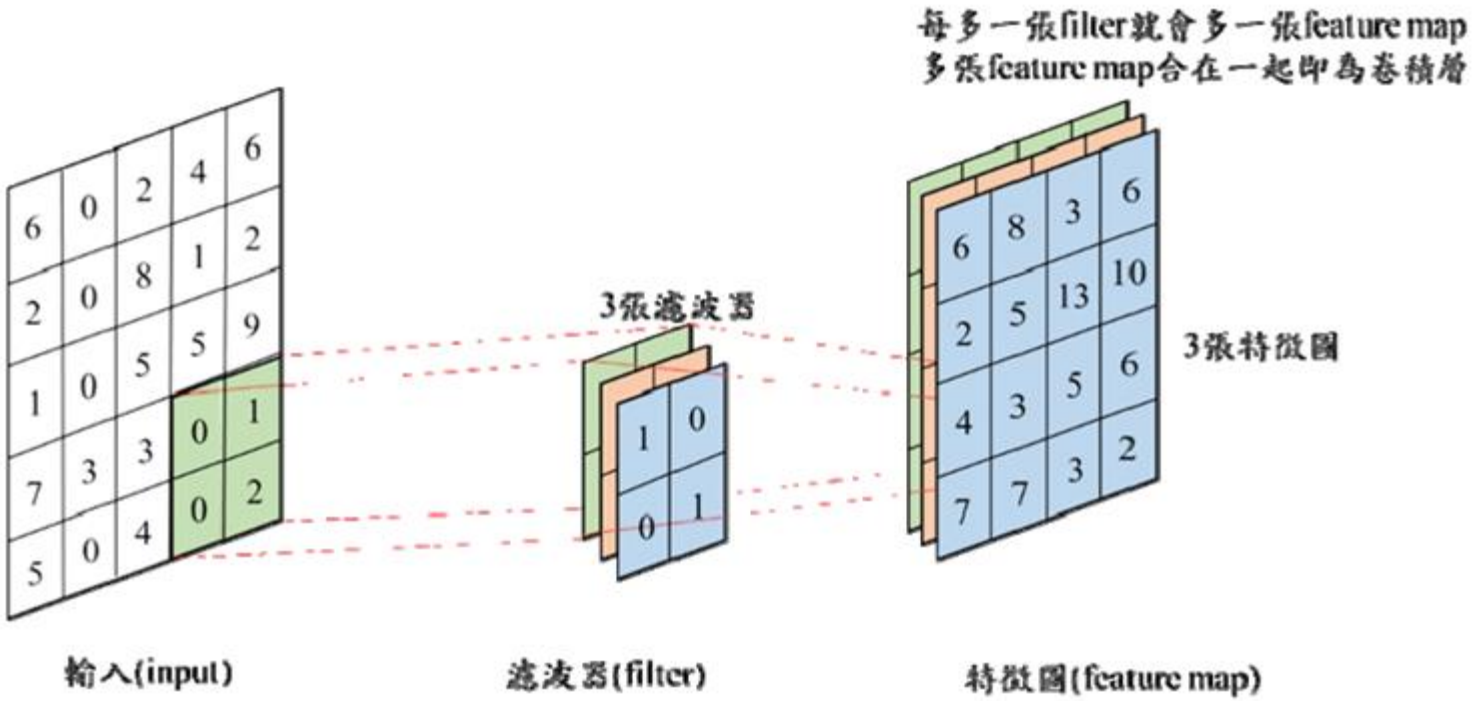
- 隱藏層（Hidden Layer）
  - 卷積層
    - 濾波器(filter)、特徵圖(feature map)



卷積神經網路中兩層神經元透過權重連接示意圖

CNN架構

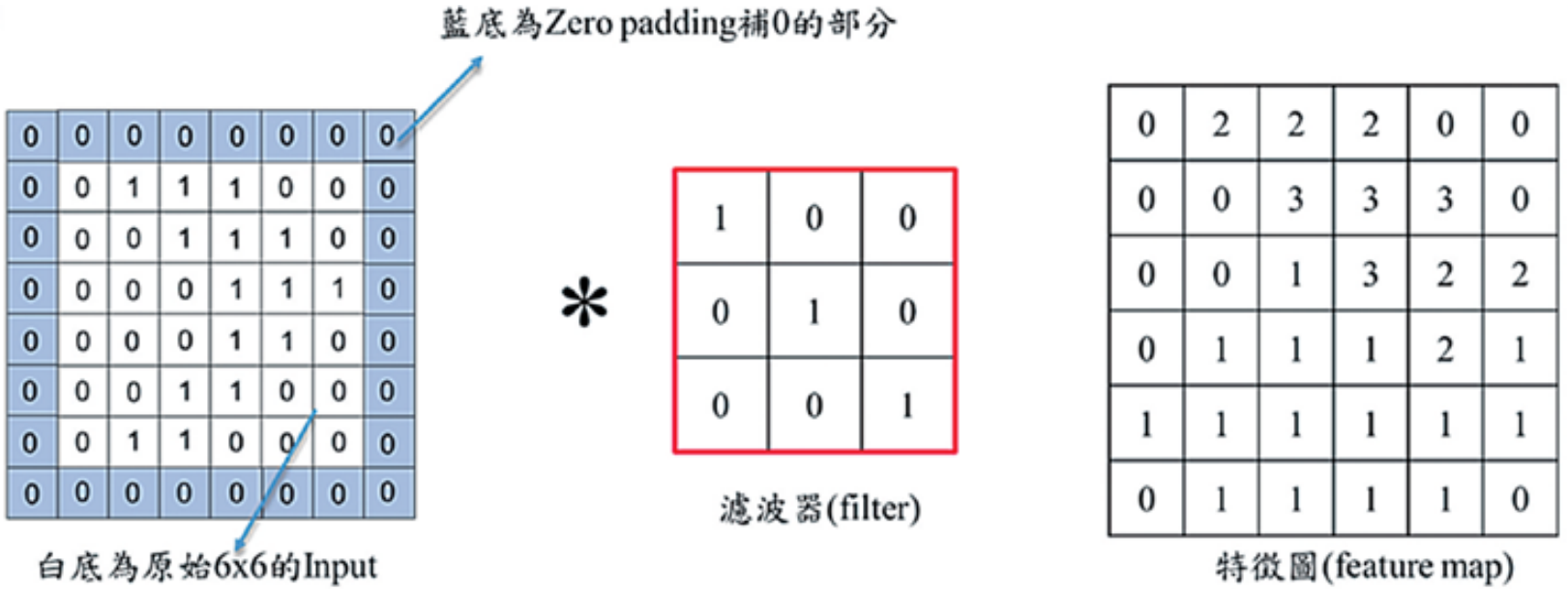
- 隱藏層（Hidden Layer）
  - 卷積層
    - 濾波器(filter)、特徵圖(feature map)



多張濾波器進行卷積的示意圖

CNN架構

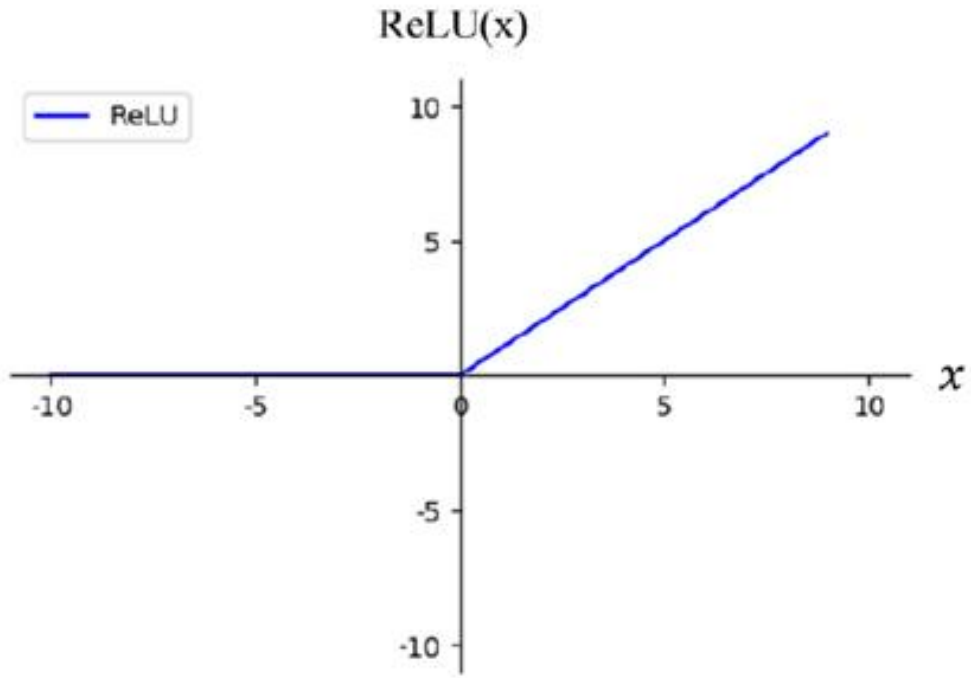
- 隱藏層（Hidden Layer）
  - 卷積層
    - 濾波器(filter)、特徵圖(feature map)



補零(zero padding)的運作方式

CNN架構

- 隱藏層（Hidden Layer）
  - 卷積層
    - 濾波器(filter)、特徵圖(feature map)、激活函數(AF)



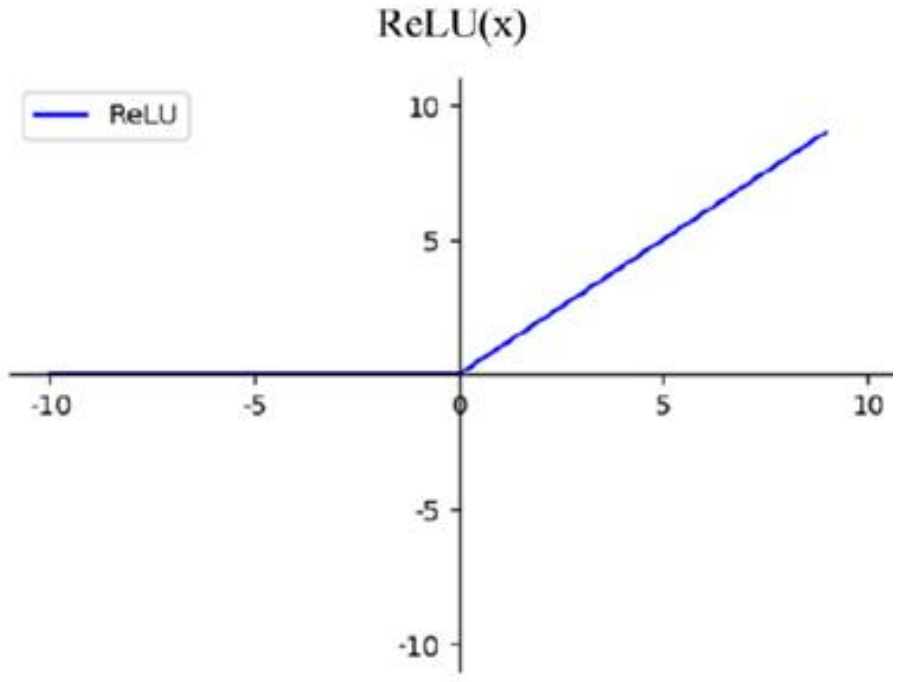
ReLU 函數圖

CNN架構

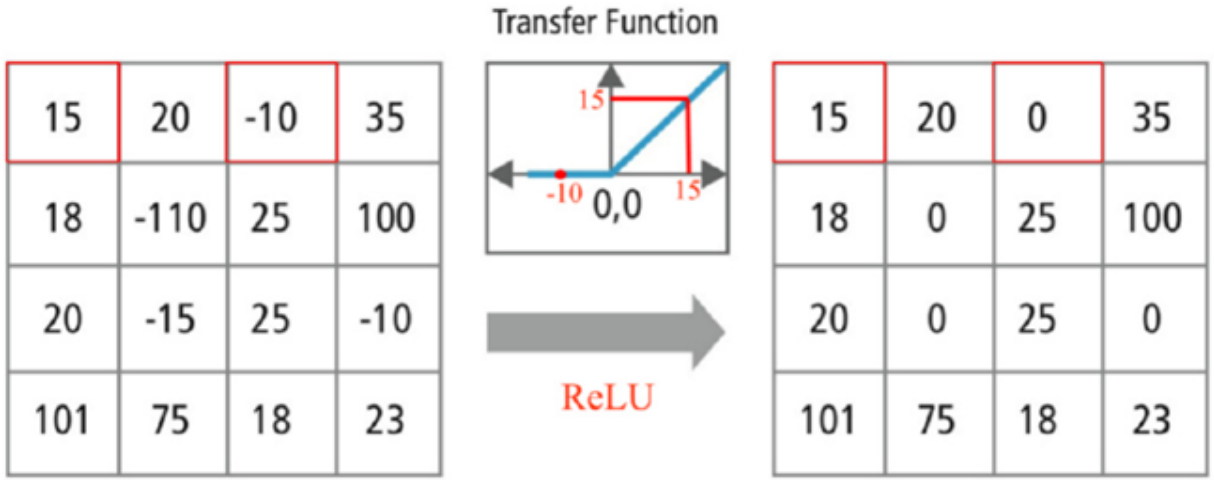
- 為什麼要用ReLU函數?
  - 克服梯度消失問題
  - 類神經網路的稀疏性
  - ReLU計算量很小

CNN架構

- 隱藏層（Hidden Layer）
  - 捲積層
    - 濾波器(filter)、特徵圖(feature map)、激活函數(AF)



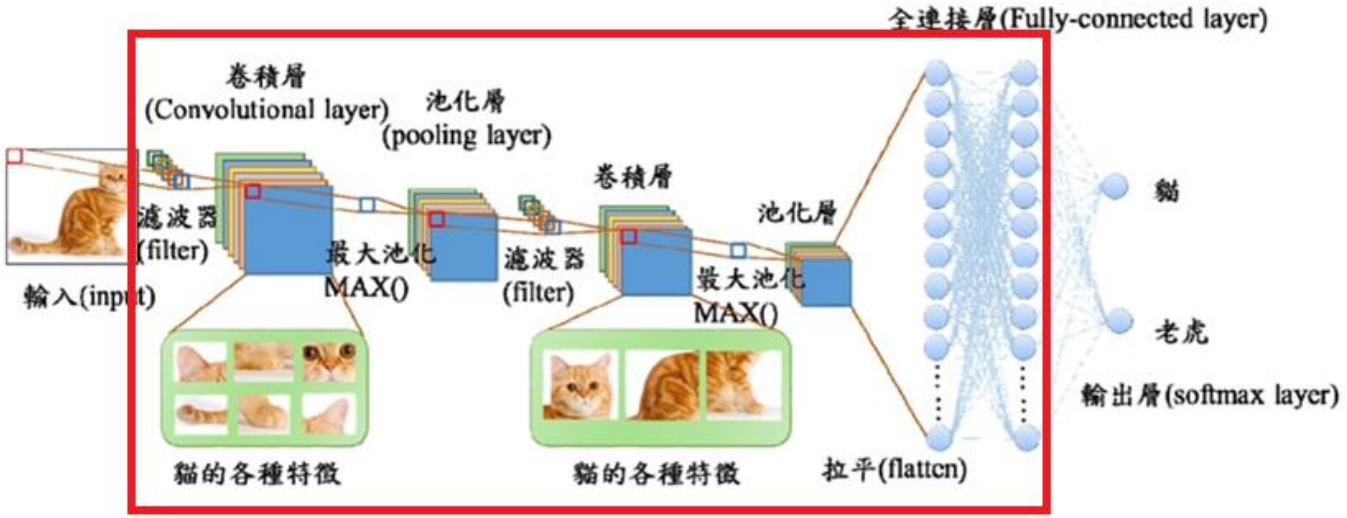
ReLU 函數圖



ReLU 函數的運作方式

CNN架構

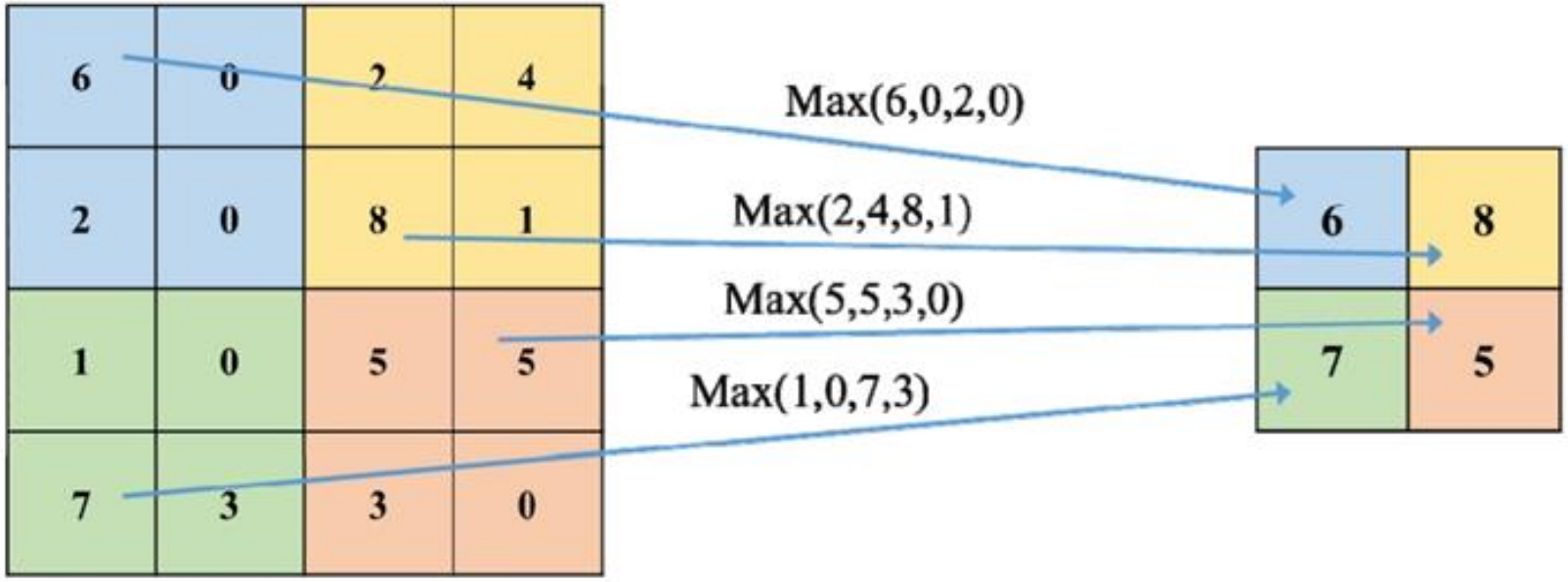
- 隱藏層（Hidden Layer）
  - 卷積層
  - 池化層
  - 全連接層



卷積神經網路架構圖

CNN架構

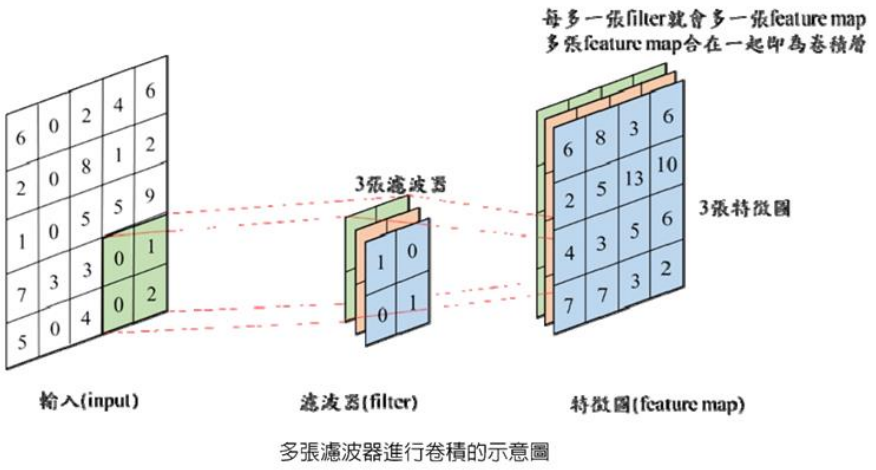
- 隱藏層（Hidden Layer）
  - 池化層(Max pooling)



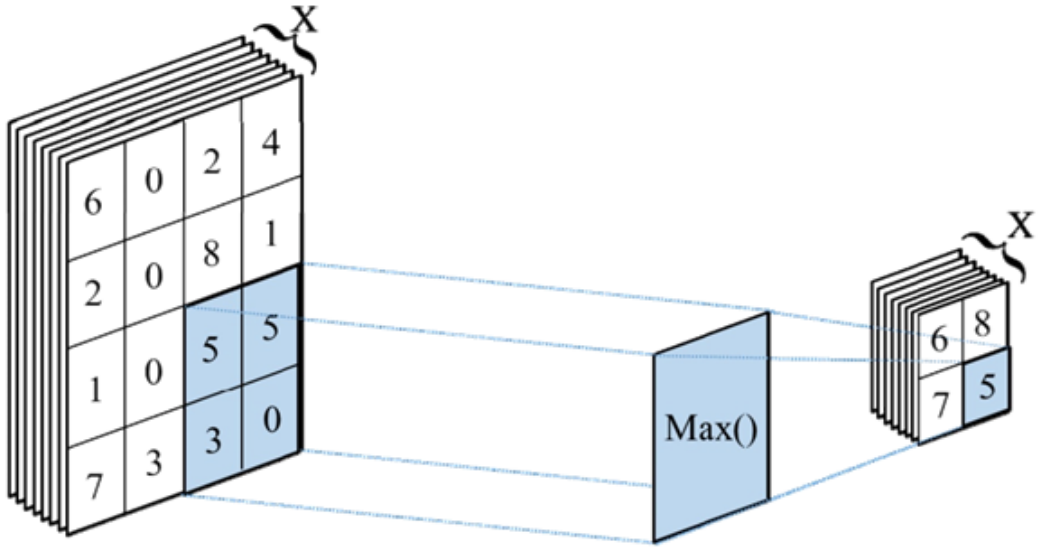
池化層透過池化函數進行池化的過程

CNN架構

- 隱藏層（Hidden Layer）
  - 池化層(Max pooling)



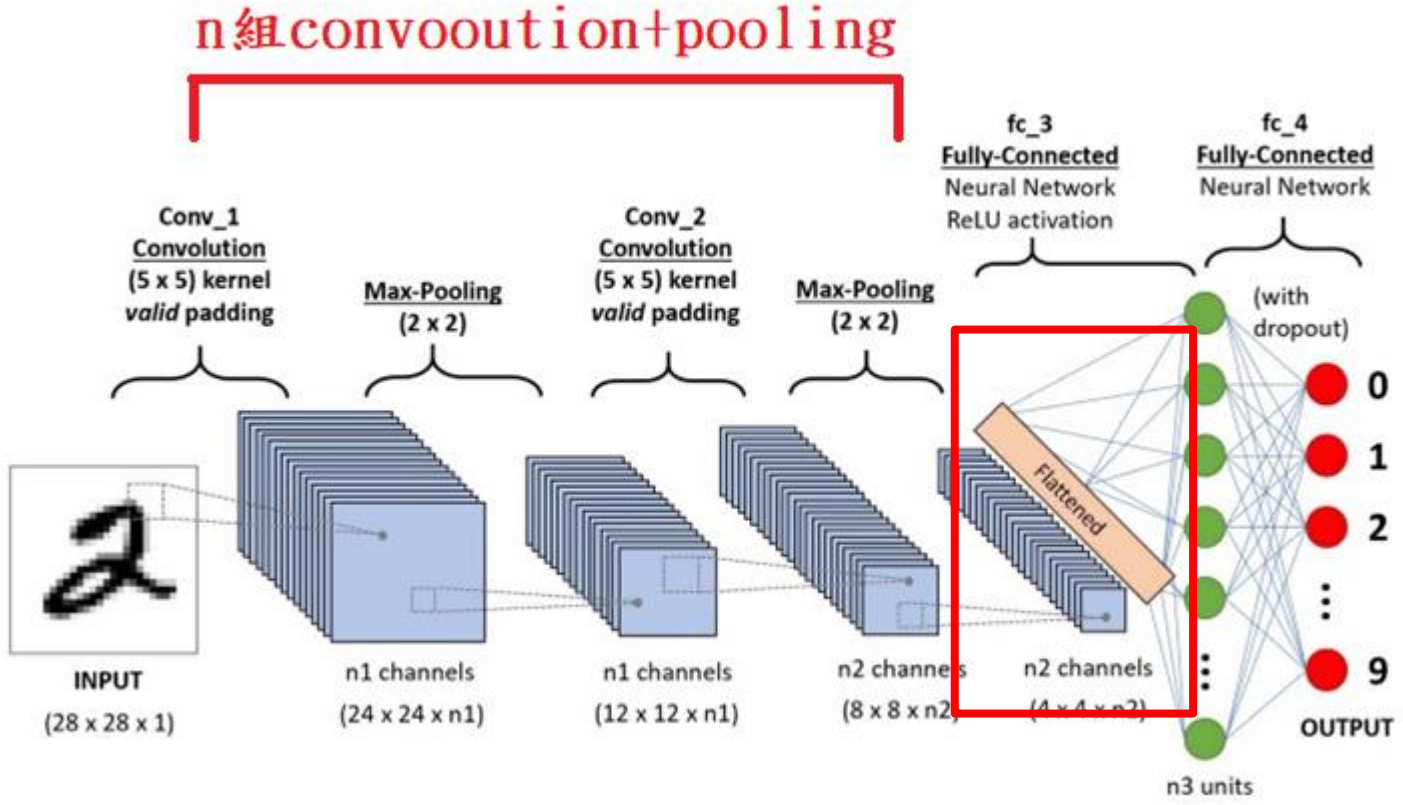
多張濾波器進行卷積的示意圖



透過池化函數輸出相同深度的池化層

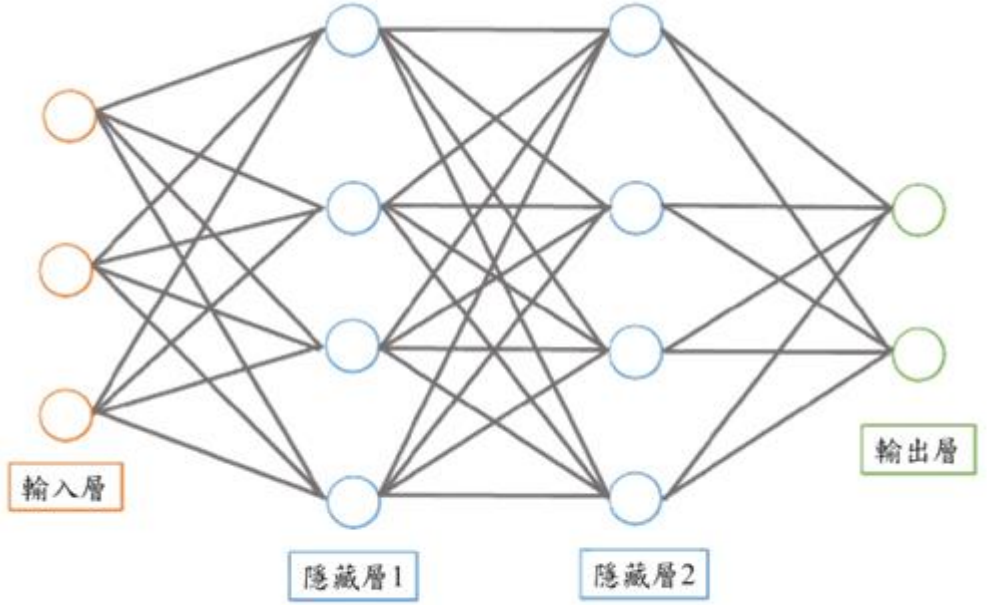
CNN架構

- 隱藏層（Hidden Layer）
  - 卷積層
  - 池化層

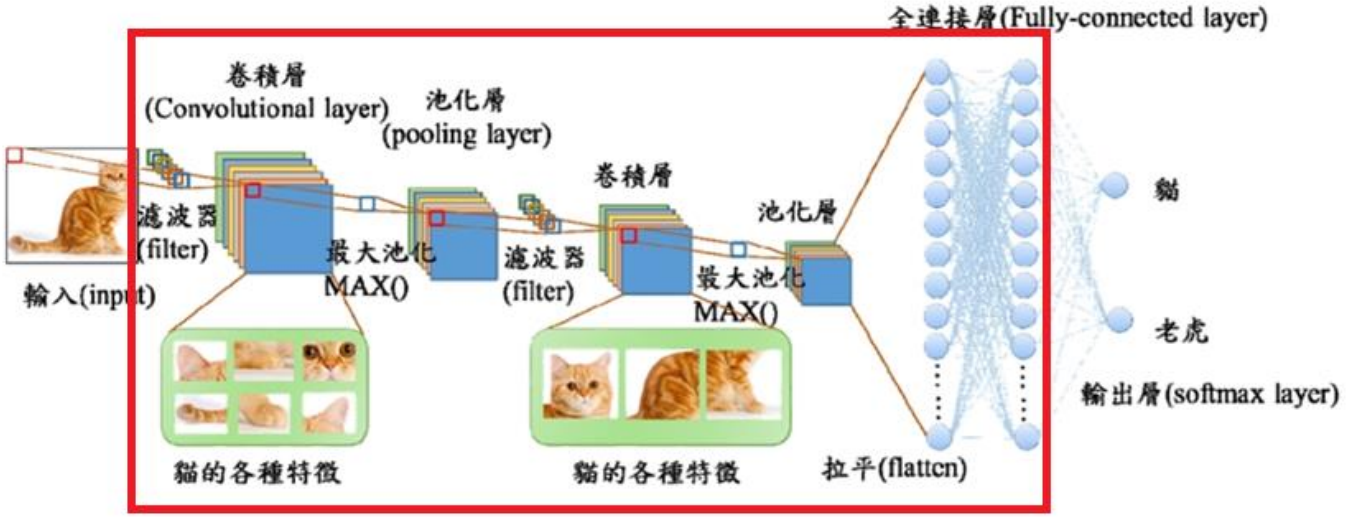


CNN架構

- 隱藏層（Hidden Layer）
  - 卷積層
  - 池化層
  - 全連接層



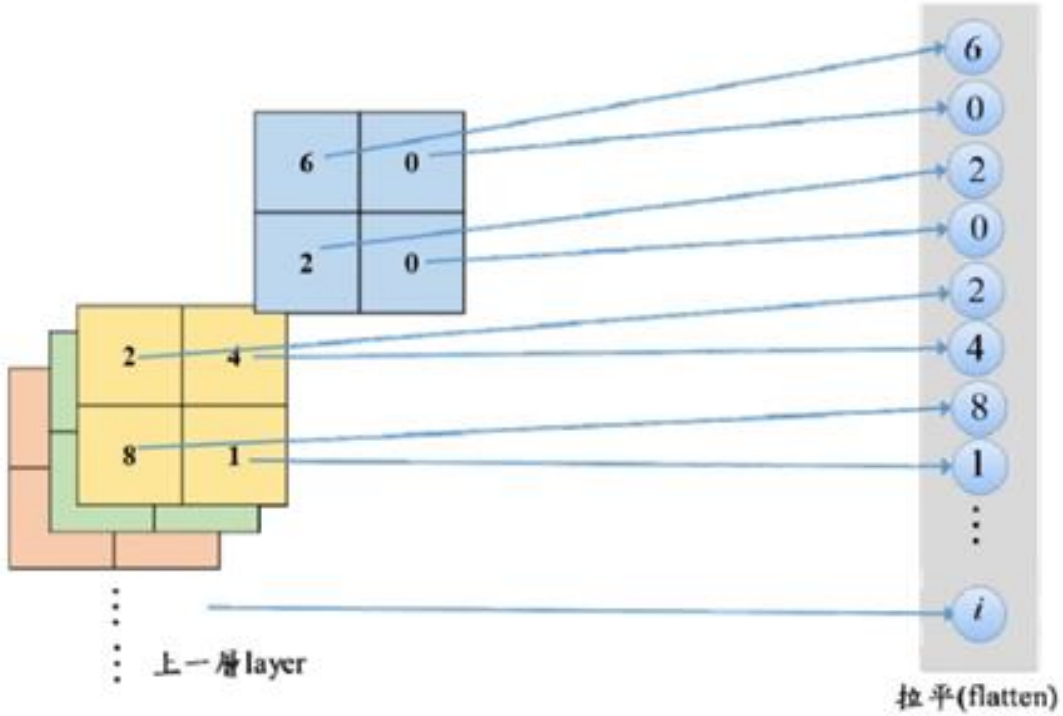
神經網路基本架構圖



卷積神經網路架構圖

CNN架構

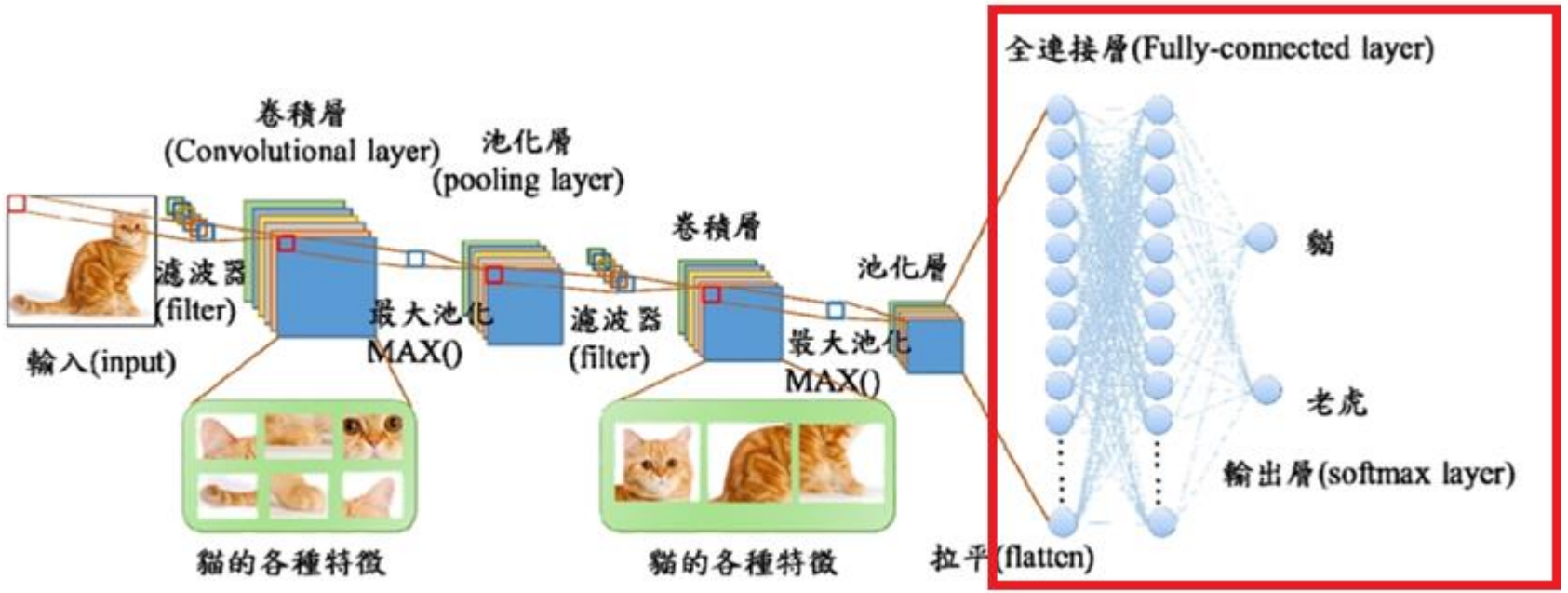
- 隱藏層（Hidden Layer）
  - 卷積層
  - 池化層
  - 全連接層



池化層的神經元連接全連接層前的動作

CNN架構

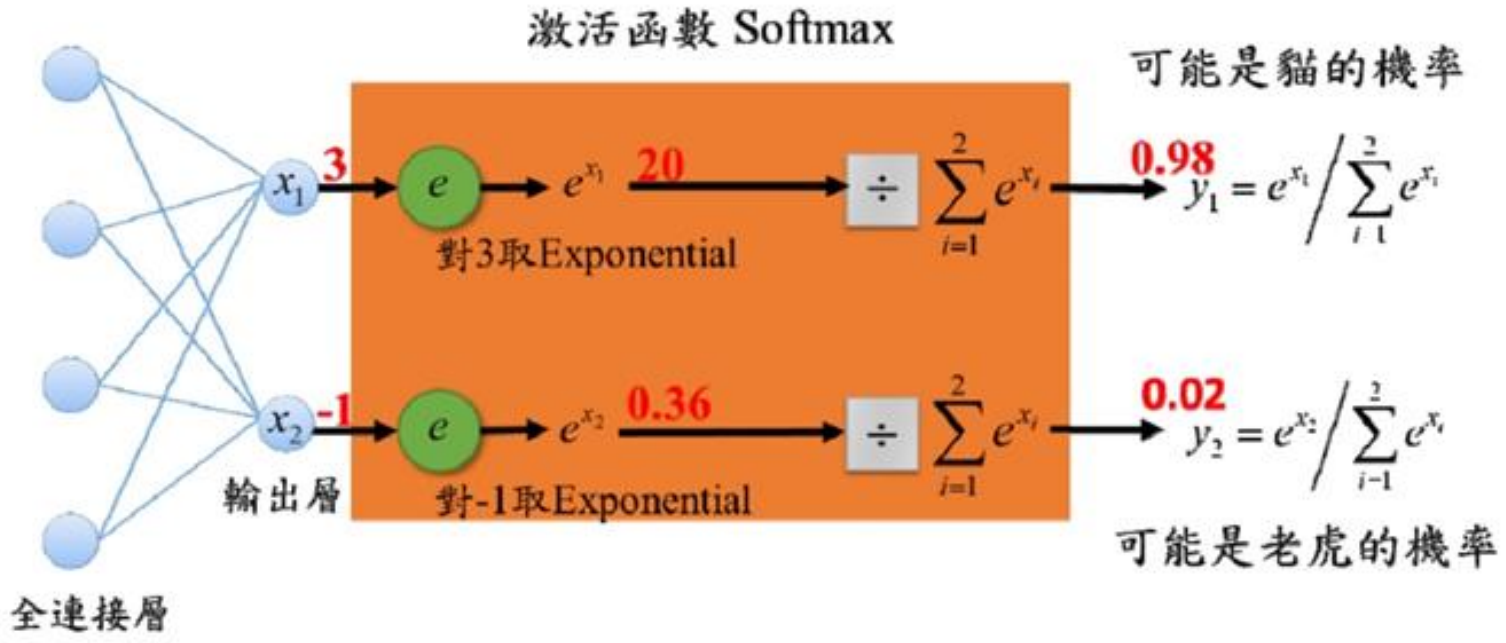
- 輸入層
- 隱藏層
  - 卷積層
  - 池化層
  - 全連接層
- 輸出層



卷積神經網路架構圖

CNN架構

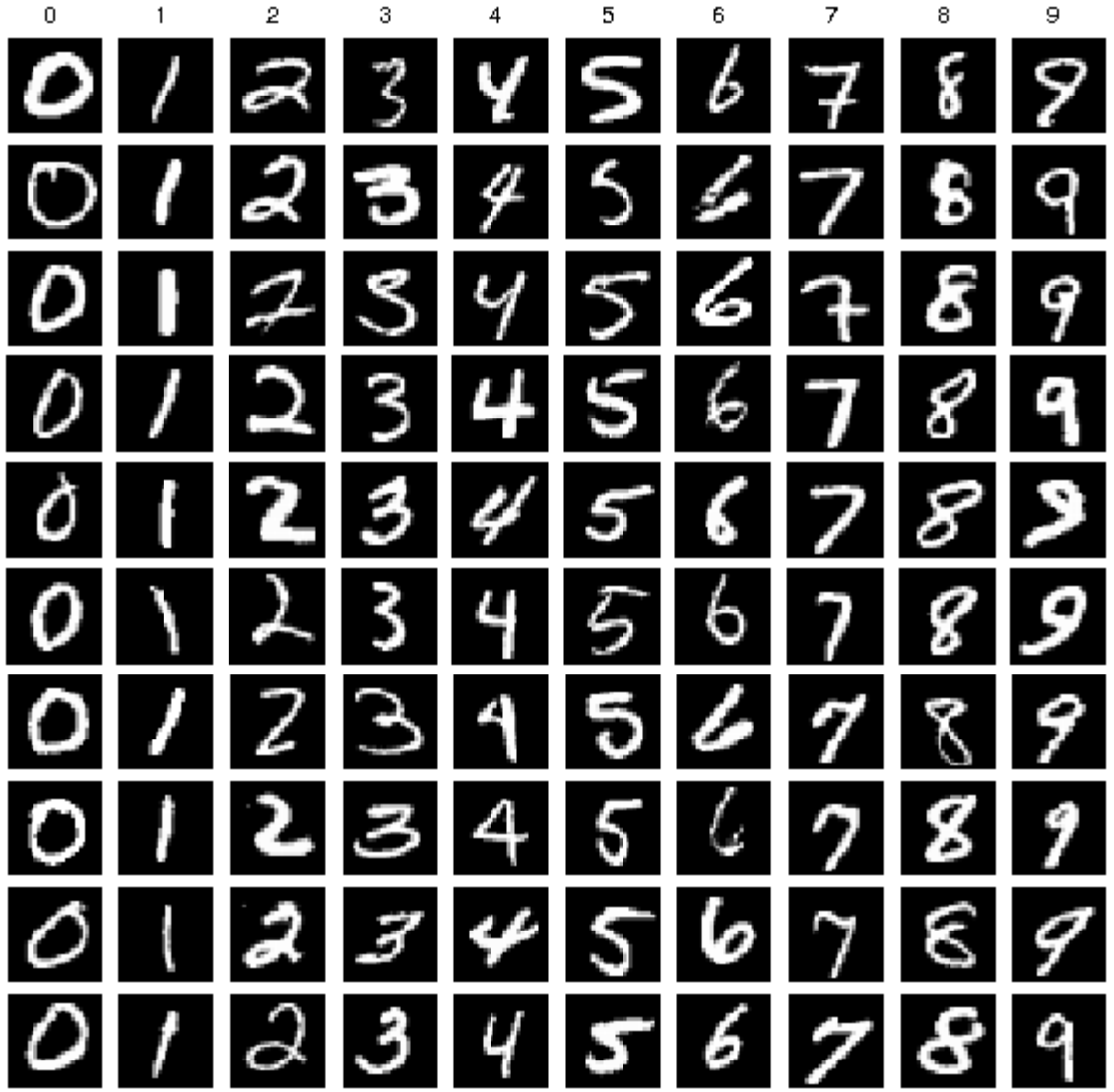
- 全連接層與輸出層



輸出層示意圖

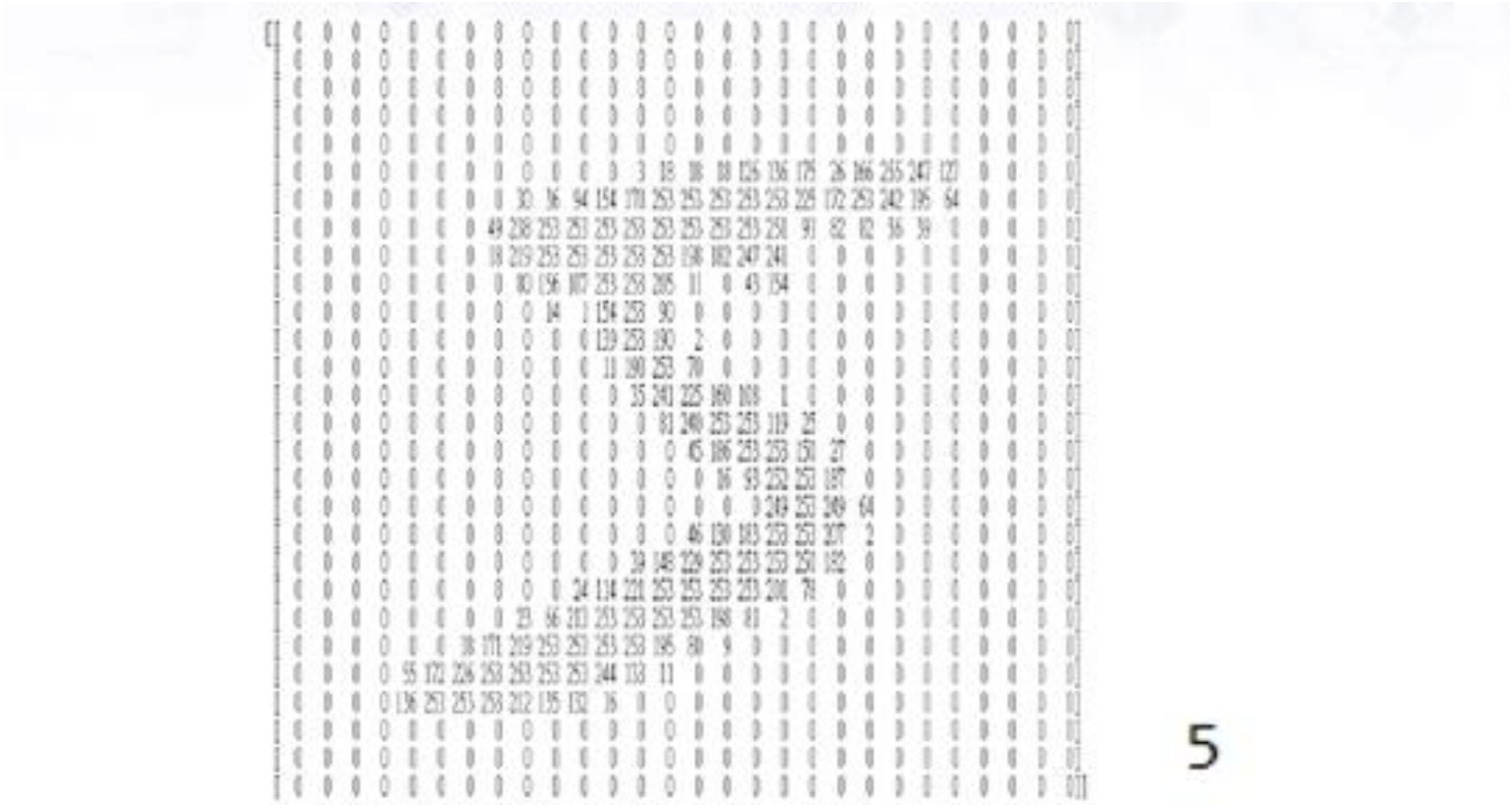
CNN架構

- Mnist手寫數字辨識



CNN架構

- Mnist手寫數字辨識

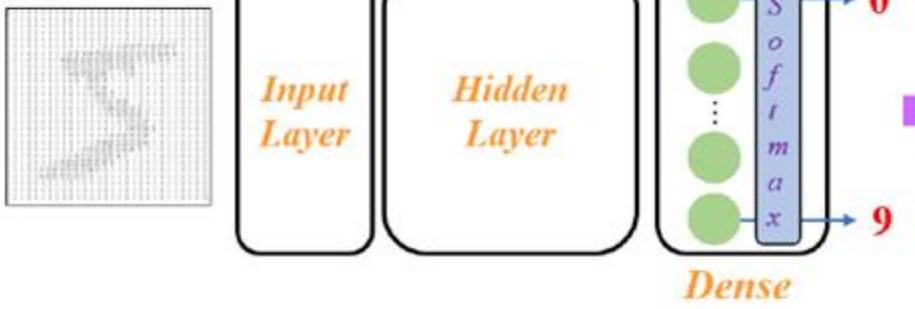


5

MNIST 手寫數字資料集中，第一筆資料及標籤

CNN架構

- 資料前處理
  1. 將特徵資料轉換成4D張量形狀(樣本數,28,28,1)
  2. 執行特徵標準化的正規化
  3. 將標籤資料進行 One-hot編碼



To_categorical

數字5

0
0
0
0
1
0
0
0
0
0

```
Y_train=to_categorical(Y_train)
Y_test=to_categorical(Y_test)
print("Y_train Shape:",Y_train.shape)
print(Y_train[0])
```

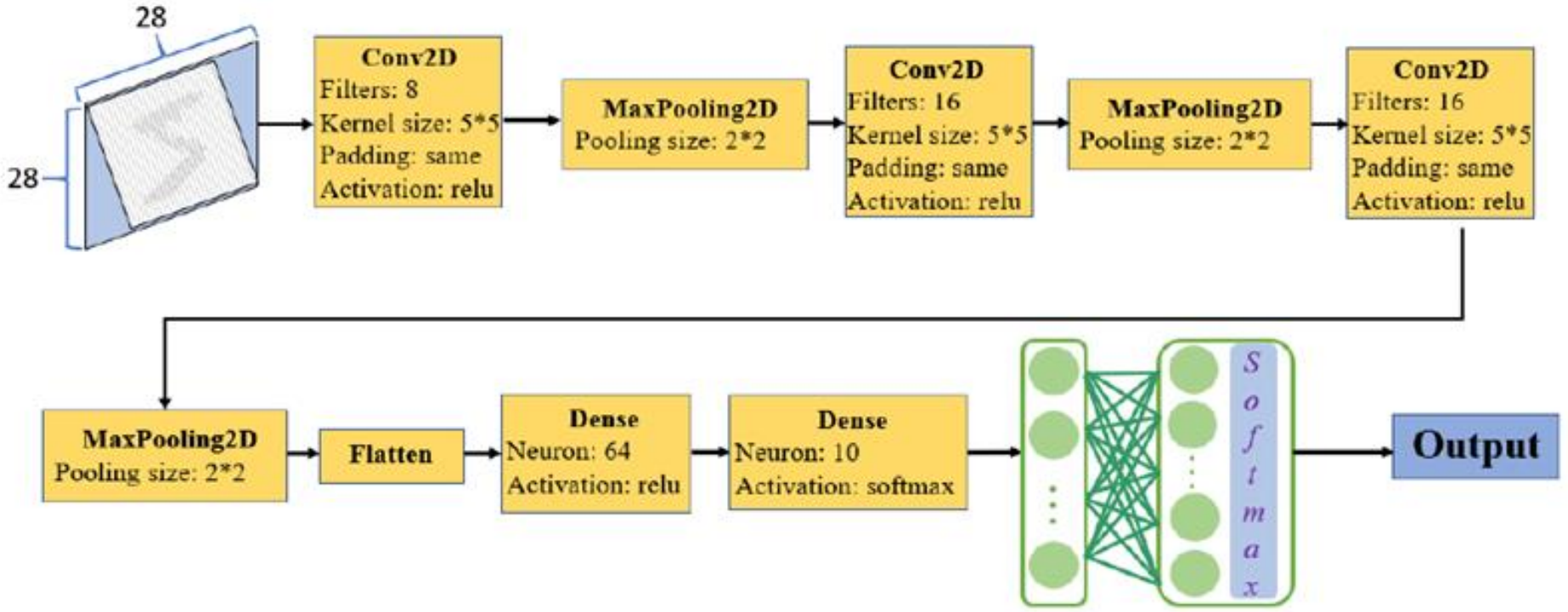
```
Y_train Shape: (60000, 10)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

One-hot code

以 One-hot encoding 將輸出的標籤編碼

CNN架構

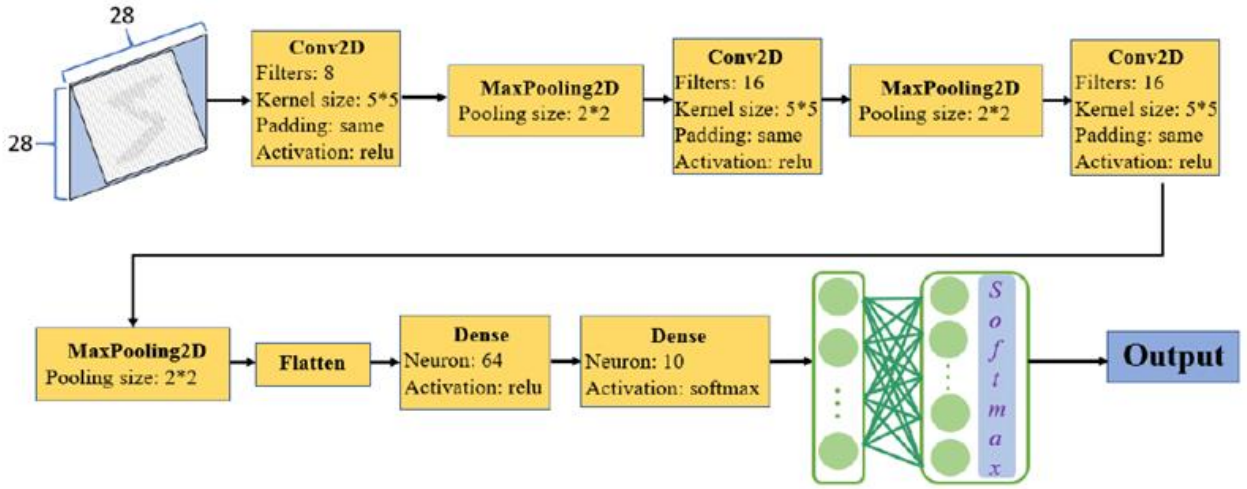
- 定義模型


28
28
Conv2D
Filters: 8
Kernel size: 5*5
Padding: same
Activation: relu
MaxPooling2D
Pooling size: 2*2
Conv2D
Filters: 16
Kernel size: 5*5
Padding: same
Activation: relu
MaxPooling2D
Pooling size: 2*2
Conv2D
Filters: 16
Kernel size: 5*5
Padding: same
Activation: relu
MaxPooling2D
Pooling size: 2*2
Flatten
Dense
Neuron: 64
Activation: relu
Dense
Neuron: 10
Activation: softmax
Softmax
Output

CNN架構

- 訓練模型

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 28, 28, 8)         208

 max_pooling2d_9 (MaxPooling  (None, 14, 14, 8)        0
 2D)

 conv2d_10 (Conv2D)          (None, 14, 14, 16)        3216

 max_pooling2d_10 (MaxPoolin  (None, 7, 7, 16)         0
 g2D)

 conv2d_11 (Conv2D)          (None, 7, 7, 32)          12832

 max_pooling2d_11 (MaxPoolin  (None, 3, 3, 32)         0
 g2D)

 flatten_3 (Flatten)         (None, 288)               0

 dense_6 (Dense)             (None, 64)                18496

 dense_7 (Dense)             (None, 10)                650

=================================================================
Total params: 35,402
Trainable params: 35,402
Non-trainable params: 0
_________________________________________________________________
```




```python
#定義模型
model=Sequential()
model.add(Conv2D(8,kernel_size=(5,5),padding="same",
input_shape=(28,28,1),activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(16,kernel_size=(5,5),padding="same",
activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(16,kernel_size=(5,5),padding="same",
activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(64,activation="relu"))
model.add(Dense(10,activation="softmax"))
model.summary()
```

CNN架構

- 訓練模型

```
_________________________________________________________________
Epoch 1/10
375/375 - 12s - loss: 1.1378 - accuracy: 0.8171 - val_loss: 0.1904 - val_accuracy: 0.9457 - 12s/epoch - 31ms/step
Epoch 2/10
375/375 - 10s - loss: 0.1419 - accuracy: 0.9576 - val_loss: 0.1133 - val_accuracy: 0.9668 - 10s/epoch - 28ms/step
Epoch 3/10
375/375 - 11s - loss: 0.0894 - accuracy: 0.9723 - val_loss: 0.0924 - val_accuracy: 0.9747 - 11s/epoch - 28ms/step
Epoch 4/10
375/375 - 11s - loss: 0.0656 - accuracy: 0.9796 - val_loss: 0.0753 - val_accuracy: 0.9778 - 11s/epoch - 28ms/step
Epoch 5/10
375/375 - 11s - loss: 0.0508 - accuracy: 0.9841 - val_loss: 0.0781 - val_accuracy: 0.9787 - 11s/epoch - 28ms/step
Epoch 6/10
375/375 - 11s - loss: 0.0427 - accuracy: 0.9864 - val_loss: 0.0907 - val_accuracy: 0.9765 - 11s/epoch - 28ms/step
Epoch 7/10
375/375 - 11s - loss: 0.0363 - accuracy: 0.9887 - val_loss: 0.0679 - val_accuracy: 0.9827 - 11s/epoch - 28ms/step
Epoch 8/10
375/375 - 11s - loss: 0.0317 - accuracy: 0.9894 - val_loss: 0.0731 - val_accuracy: 0.9841 - 11s/epoch - 29ms/step
Epoch 9/10
375/375 - 11s - loss: 0.0316 - accuracy: 0.9901 - val_loss: 0.0652 - val_accuracy: 0.9840 - 11s/epoch - 30ms/step
Epoch 10/10
375/375 - 11s - loss: 0.0301 - accuracy: 0.9904 - val_loss: 0.0796 - val_accuracy: 0.9789 - 11s/epoch - 28ms/step
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0402 - accuracy: 0.9879
訓練資料集的準確度=0.99
313/313 [==============================] - 1s 4ms/step - loss: 0.0659 - accuracy: 0.9800
測試資料集的準確度=0.98
```