勞動部產業新尖兵計畫

人工智慧金融應用與實務培訓班

課程模組： AI 金融科技課程 - AI 程式設計

4. 深度學習 1 - CNN

葉建華 (Yeh, Jian-hua)

tdi.jhyeh@tdi.edu.tw
au4290@gmail.com

講次內容

- 卷積神經網路 CNN 介紹
- CNN 案例：手寫辨識 MNIST
- CNN 案例： CIFAR-10 影像分類

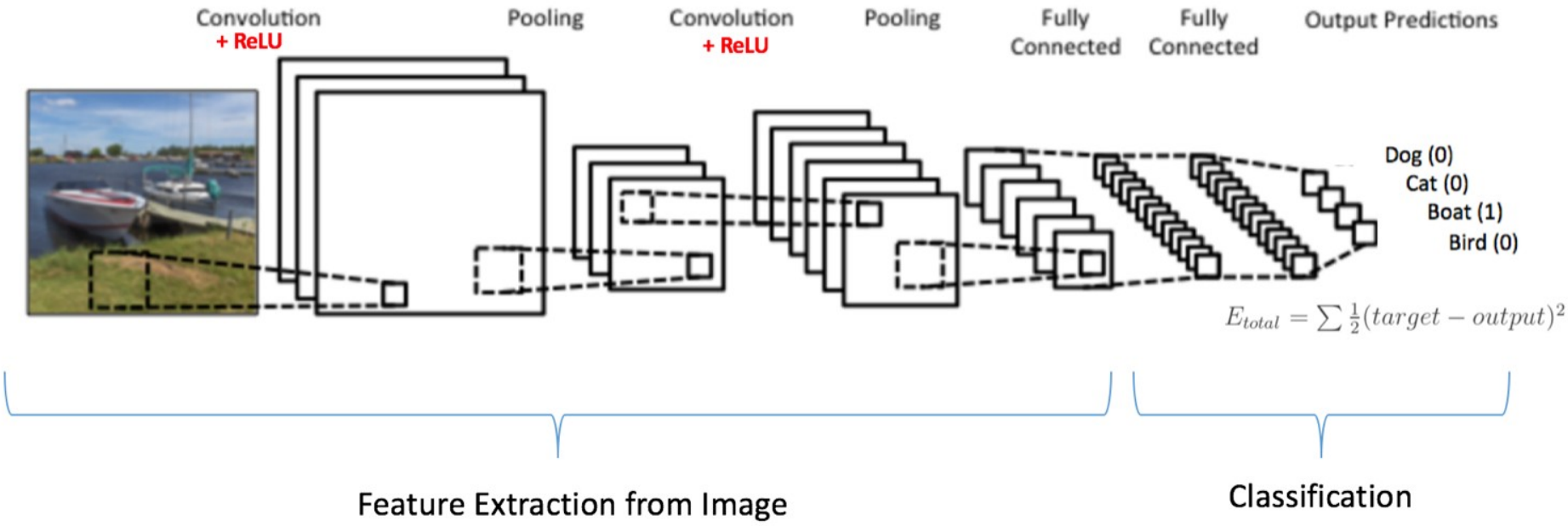
卷積神經網路 CNN 介紹

- 由 AI 三大家 Yann LeCun 所發明
  - 2018 Turing Award 獲獎人
  - 卷積神經網路發明人
  - 神經網路影像處理大師

卷積神經網路 CNN 介紹

- Convolutional Neural Networks，又稱為 ConvNets
- 是深度神經網絡，常用於分析視覺圖像
  - 透過使用卷積層 (Convolution Layer) 計算，由點的比對轉成局部的比對，逐步放大並堆疊綜合比對結果
  - 使用池化層 (Pooling Layer) 計算，來控制壓縮圖片資訊並保留其中重要的部份
- 圖片識辨別上超越人類精準度

卷積神經網路 CNN 介紹

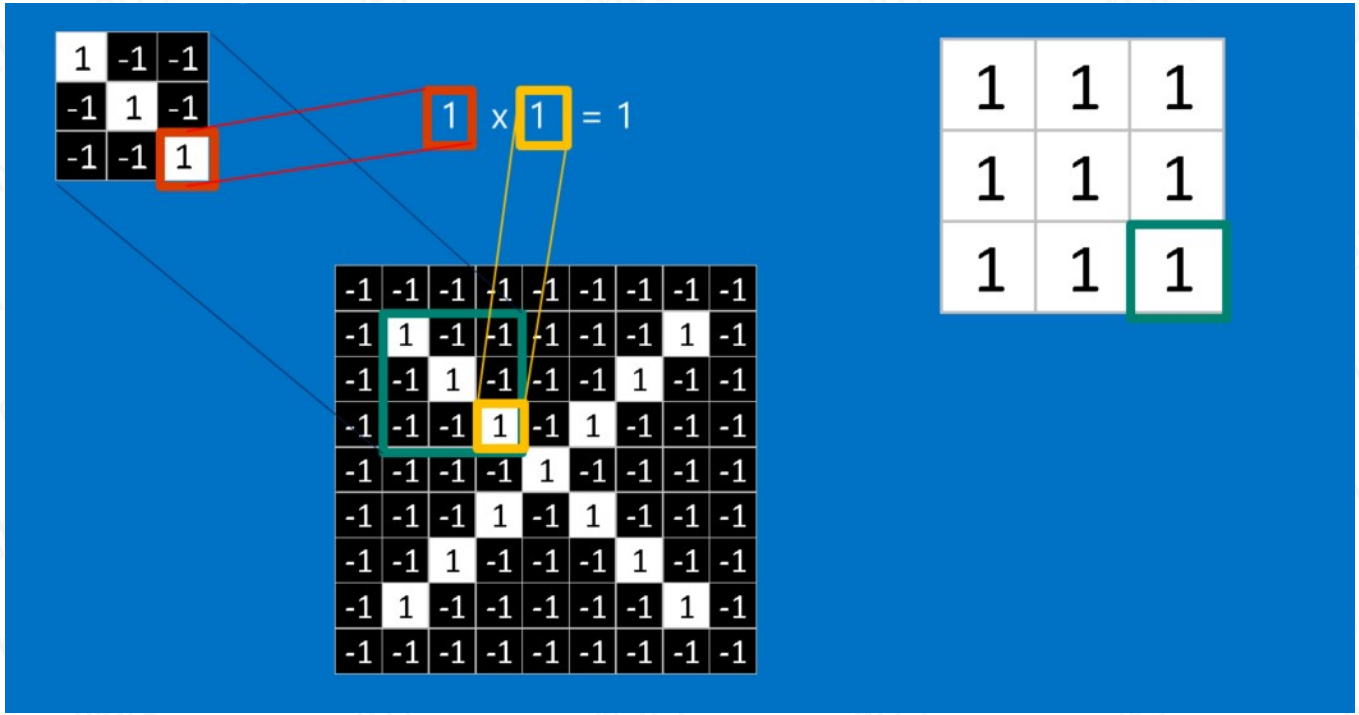


從 CNN 的模型開始，正式進入深度學習階段!

From https://e2eml.school/how_convolutional_neural_networks_work.html

卷積神經網路 CNN 介紹

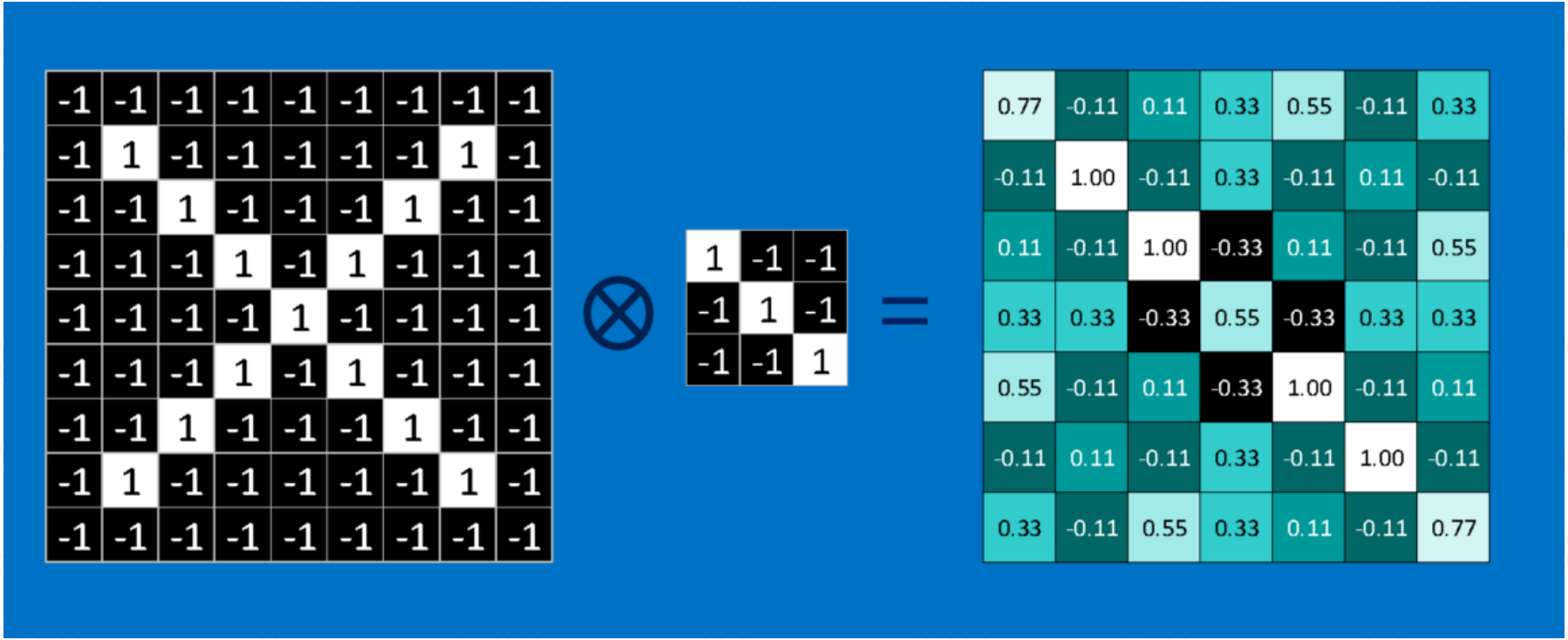
- 卷積計算：使用樣式過濾器 (filter)



From https://e2eml.school/how_convolutional_neural_networks_work.html

卷積神經網路 CNN 介紹

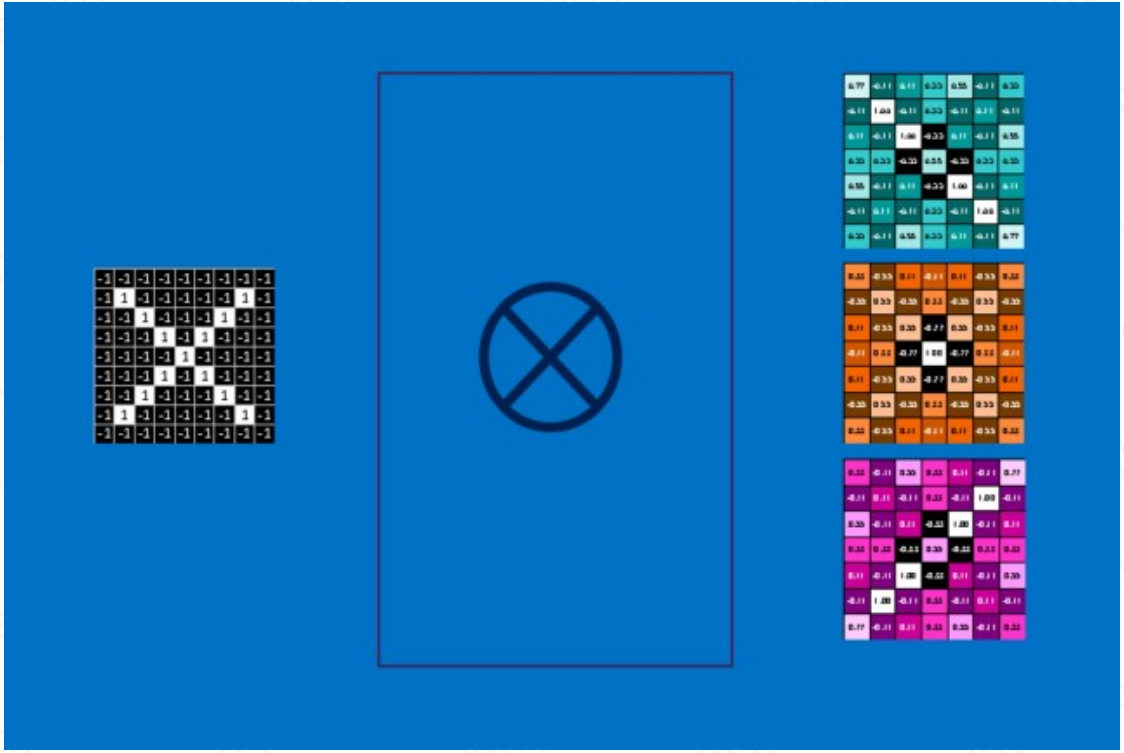
- filter 結果加總，以像素總數平均



From https://e2eml.school/how_convolutional_neural_networks_work.html

卷積神經網路 CNN 介紹

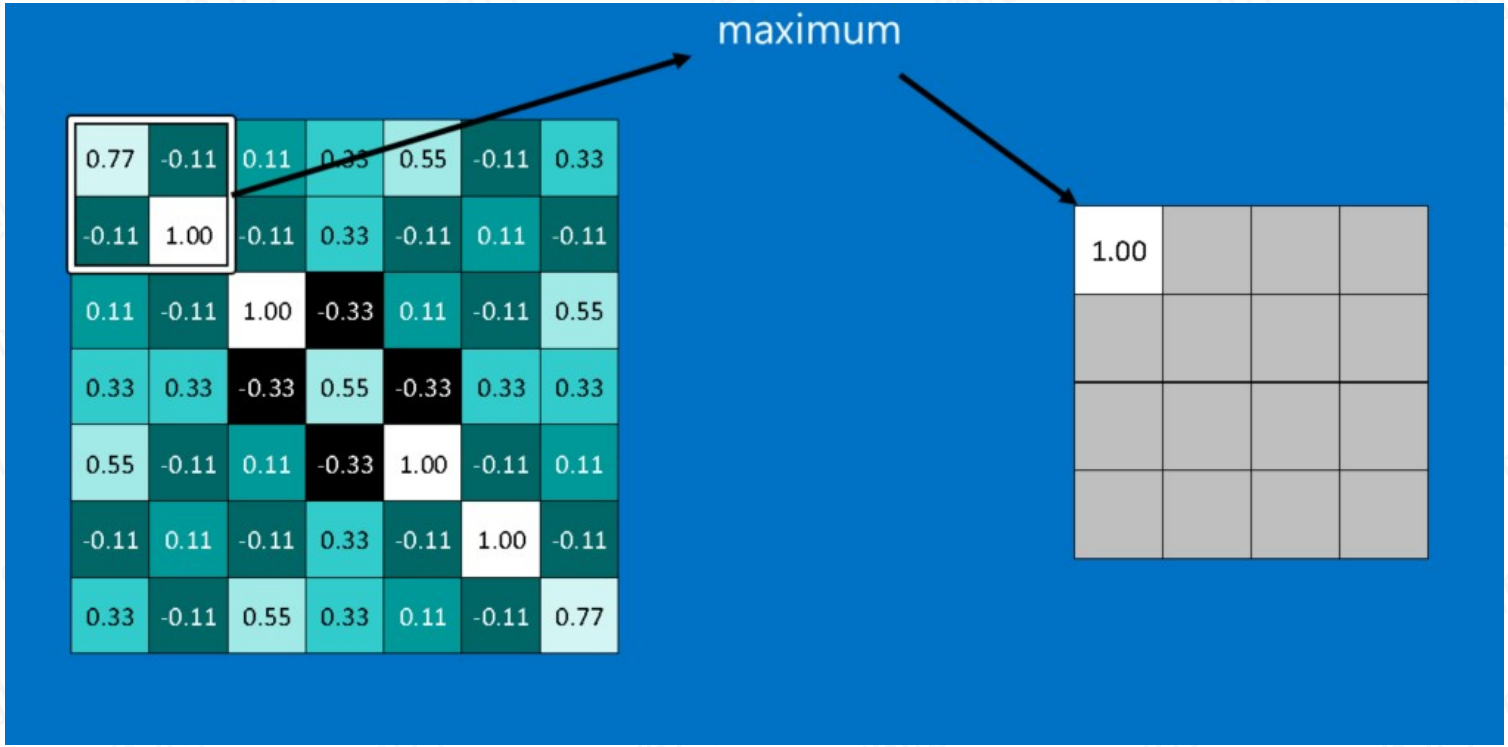
- 不同 filter 算出很多個「特徵圖」 (feature map)



From https://e2eml.school/how_convolutional_neural_networks_work.html

卷積神經網路 CNN 介紹

- 池化計算：選取窗框，並在此範圍中選最大值



CNN 是透過卷積和池化計算進行影像特徵的抽取來完成模型學習

使用 CNN 進行學習

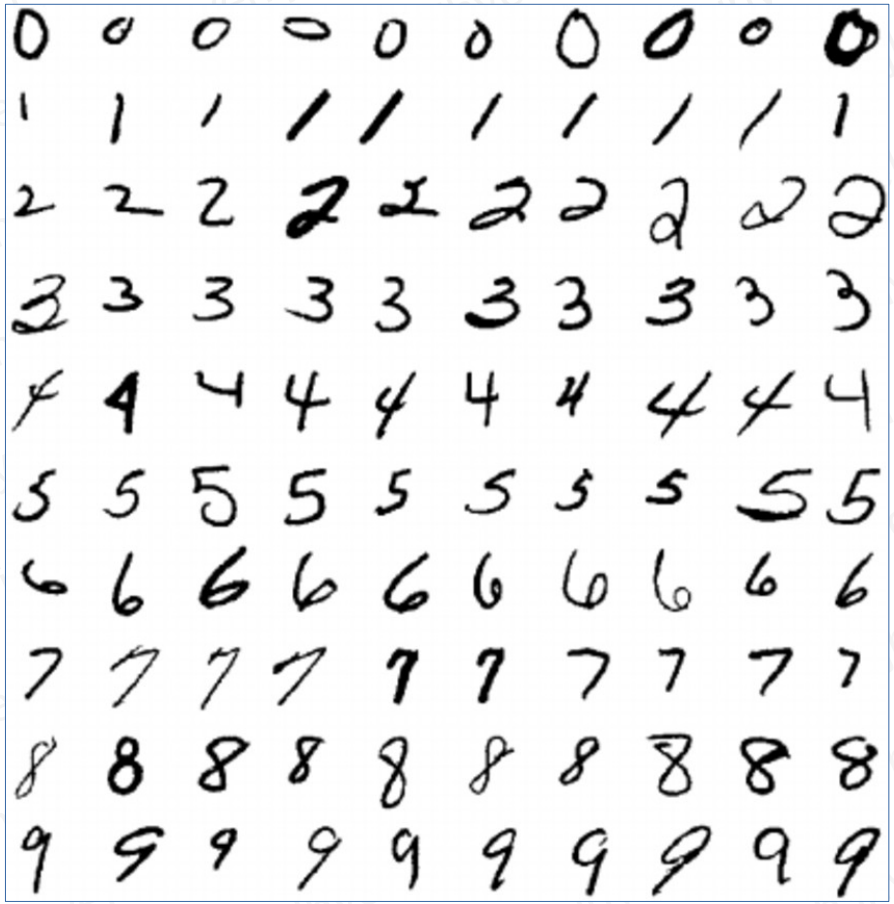
- 邏輯流程
  - 讀入資料檔
  - 分成 train 和 test 兩部分
  - ~~資料前處理~~ ---------- 真的嗎？!
  - 根據資料維度，建立 Keras CNN 模型
  - 訓練 train 資料集
  - 用 test 資料集做評估

講次內容

- 卷積神經網路 CNN 介紹
- CNN 案例：手寫辨識 MNIST
- CNN 案例： CIFAR-10 影像分類

CNN 案例：手寫辨識 MNIST

- 又是 MNIST！
  - 可是它確實是影像資料啊!



MNIST 資料集

- 由 AI 三大家 Yann LeCun 所建立
  - 2018 Turing Award 獲獎人
  - 卷積神經網路發明人
  - 神經網路影像處理大師
- 60000 筆訓練資料、 10000 筆測試資料
- 手寫數字的影像 (28x28 點灰階影像) 配合標籤

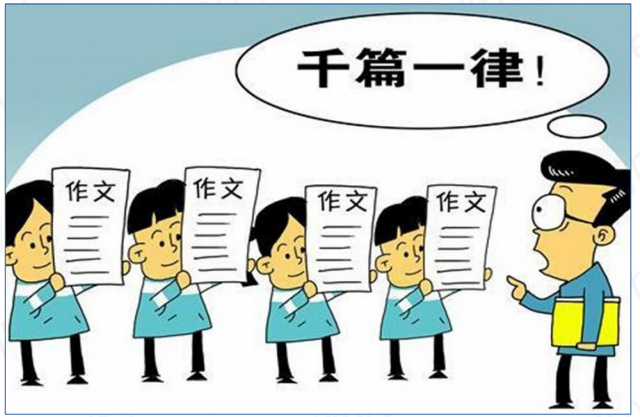
MNIST 使用 CNN 進行學習

- 邏輯流程
  - 載入 MNIST 資料集並分成

    train 和 test 兩部分 (Keras)
  - 資料前處理 (咦？！)
  - 根據資料維度，建立 Keras CNN 模型 (Keras)
  - 訓練 train 資料集 (Keras)
  - 用 test 資料集做評估 (Keras)

下載 MNIST 資料集

- 使用 Keras API 下載
  - from tensorflow.keras.datasets import mnist
  - (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
  - Train 和 Test 都幫你分好了！！



下載 MNIST 資料集

- 使
  -
  -
  -

```
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
print('type of train images: ', type(train_images))
print('type of train labels: ', type(train_labels))
print('shape of train images: ', train_images.shape)
print('shape of train labels: ', train_labels.shape)
print('shape of test images: ', test_images.shape)
print('shape of test labels: ', test_labels.shape)

print('first train image data:')
print(train_images[0])
print('first train image label:')
print(train_labels[0])

for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(train_images[i], 'gray')
plt.show()

print(train_labels[0:10])
```

下載 MNIST 資料集

```
type of train images:  <class 'numpy.ndarray'>
type of train labels:  <class 'numpy.ndarray'>
shape of train images:  (60000, 28, 28)
shape of train labels:  (60000,)
shape of test images:  (10000, 28, 28)
shape of test labels:  (10000,)
first train image data:
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
```

```
t_labels) = mnist.load_data()
es))
ls))
hape)
hape)
pe)
pe)
```

```
plt.show()

print(train_labels[0:10])
```

下載 MNIST 資料集

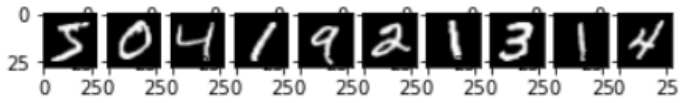
```
type of train images:  <class 'numpy.ndarray'>
type of train labels:  <class 'numpy.ndarray'>
shape of train images:  (60000, 28, 28)
shape of train labels:  (60000,)
shape of test images:  (10000, 28, 28)
shape of test labels:  (10000,)
first train image data:
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 2
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 2
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 2
    0   0   0   0   0   0   0   0   0   0]
```

```
plt.show()

print(train_labels[0:10])
```

```
t_labels) = mnist.load_data()
es))
ls))
hape)
hape)
```

```
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
first train image label:
5
```



請注意是灰階影像!

```
[5 0 4 1 9 2 1 3 1 4]
```

MNIST 使用 CNN 進行學習

- 資料前處理 (還是有啊！！)
  - 訓練資料維度： 60000x28x28
  - 代表 60000 張圖，每張 28x28，二維資料
  - 準備使用 Convolution Layer，不需要拉平
    - 但是需要為 Convolution Layer 調整維度成 28x28x1 （顏色深度問題！）
    - 還是要正規化到 (0,1) 之間
  - 標籤本來是 0 到 9，我們也將它 categorical 化

MNIST 使用 CNN 進行學習

-

```python
from tensorflow.keras.utils import to_categorical

# reshape for Conv2D
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# data normalization to 0~1
train_data = train_images.astype('float32')/255.0
test_data = test_images.astype('float32')/255.0
print('shape of train images: ', train_data.shape)
print('shape of test images: ', test_data.shape)

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
print('shape of train labels: ', train_labels.shape)
print('shape of test labels: ', test_labels.shape)

print('first train image label:')
print(train_labels[0])
```

料

平

28x28x1 （顏色深度問題！）

ical 化

MNIST 使用 CNN 進行學習

-

```
from tensorflow.keras.utils import to_categorical

# reshape for Conv2D
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# data normalization to 0~1
train_data = train_images.astype('float32')/255.0
test_data = test_images.astype('float32')/255.0
print('shape of train images: ', train_data.shape)
print('shape of test images: ', test_data.shape)

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
print('shape of train labels: ', train_labels
print('shape of test labels: ', test_labels.s

print('first train image label:')
print(train_labels[0])
```

料

平

28x28x1 （顏色深度問題！）

```
shape of train images:  (60000, 28, 28)
shape of test images:  (10000, 28, 28)
shape of train labels:  (60000, 10)
shape of test labels:  (10000, 10)
first train image label:
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

MNIST 使用 CNN 進行學習

- 根據資料維度，建立 Keras CNN 模型 (Keras)
  - 留意 Conv2D(), MaxPool2D(), Flatten()

MNIST 使用 CNN 進行學習

```
from tensorflow.keras.layers import Activation, Conv2D, MaxPool2D, Dense, Dropout, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(28,28,1)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

print(model.summary())
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['acc'])
```

MNIST 使用 CNN 進行學習

```
from tensorflow.keras.lay
from tensorflow.keras.mo
from tensorflow.keras.op

model = Sequential()

model.add(Conv2D(32, (3,
model.add(Conv2D(32, (3,
model.add(MaxPool2D(pool
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,
model.add(Conv2D(64, (3,
model.add(MaxPool2D(pool
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, act
model.add(Dropout(0.5))
model.add(Dense(10, acti

print(model.summary())
model.compile(loss='cate
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1606144
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,676,266
Trainable params: 1,676,266
Non-trainable params: 0
_________________________________________________________________
None
```

```
, Dropout, Flatten


ape=(28,28,1)))
```

有沒有注意到

參數量爆增!

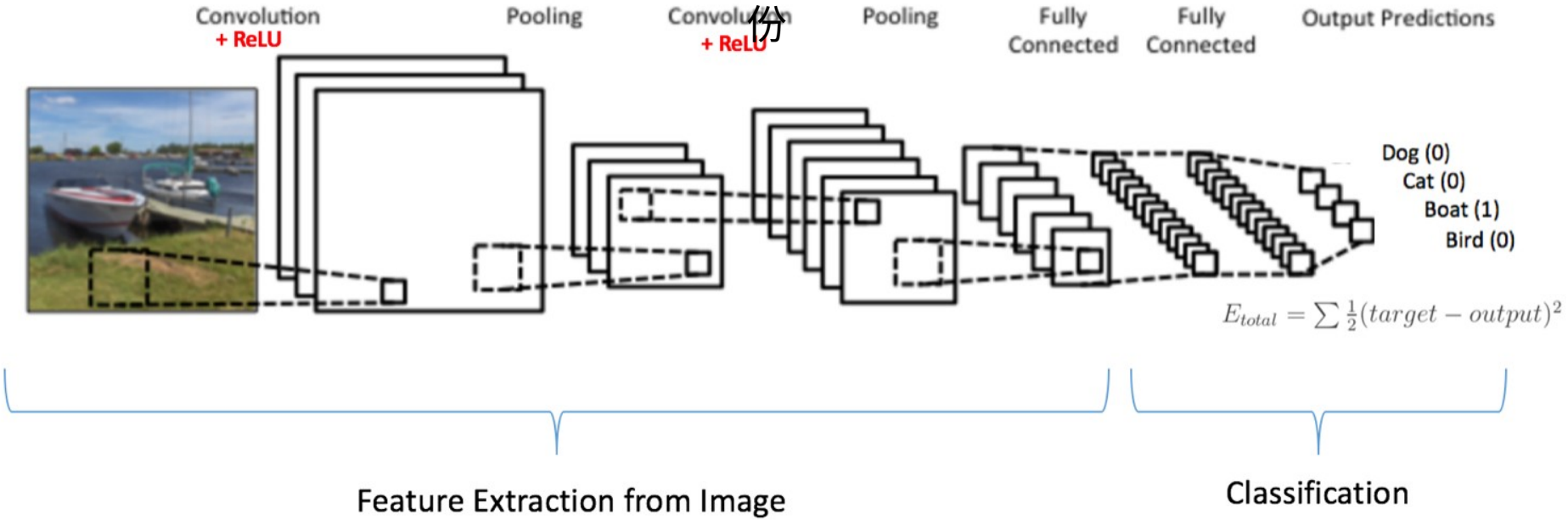
```
metrics=['acc'])
```

談談 Flatten 吧!

- 因為我們已經談過 Convolution 和 Pooling 了
- 到 Flatten() 之前，資料都還是 2D 的
- 進入 Dense，相當於進入 MLP，所以要「拉平」
  - Flatten() 是要幫你把當時的資料拉平!

談談 Flatten 吧!

拉平才能接到 Fully Connected，就是 MLP 部份



MNIST 使用 CNN 進行學習

- 訓練 train 資料集 (Keras)
  - model.fit() 又出現了!

```
import matplotlib.pyplot as plt

train_history = model.fit(train_data, train_labels, batch_size=500, epochs=100, validation_split=0.2)
print(train_history.history)

plt.plot(train_history.history['acc'], label='acc')
plt.plot(train_history.history['val_acc'], label='val_acc')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(loc='best')
plt.show()
```

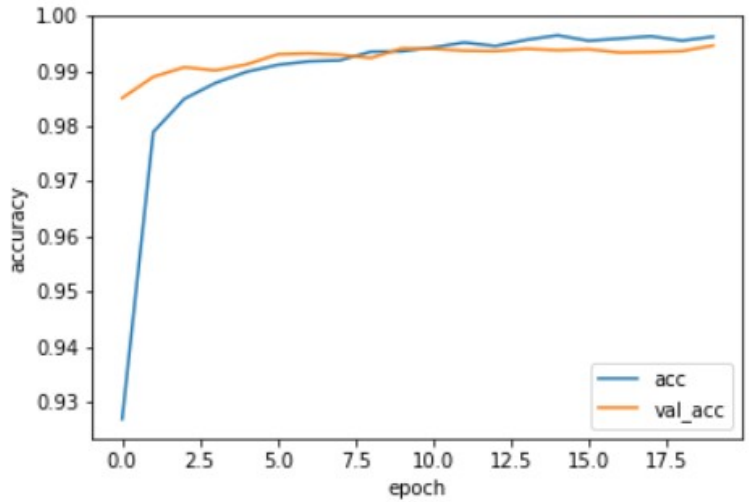
MNIST 使用 CNN 進行學習

```
Epoch 18/20
48000/48000 [==============================] - 2s 46us/sample - loss: 0.0121 - acc: 0.9963 - val_loss: 0.0287 - va
l_acc: 0.9934
Epoch 19/20
48000/48000 [==============================] - 2s 46us/sample - loss: 0.0130 - acc: 0.9955 - val_loss: 0.0289 - va
l_acc: 0.9936
Epoch 20/20
48000/48000 [==============================] - 2s 47us/sample - loss: 0.0113 - acc: 0.9962 - val_loss: 0.0264 - va
l_acc: 0.9946
```

```
imp

tra                                                                    2)
pri

plt
plt
plt
plt
plt
plt
```



MNIST 使用 CNN 進行學習

- 用 test 資料集做評估 (Keras)
  - 使用 model.evaluate() 和 model.predict()

```
import numpy as np

test_loss, test_acc = model.evaluate(test_data, test_labels)
print('loss: {:.3f}'.format(test_loss))
print('accuracy: {:.3f}'.format(test_acc))

# 只取第一筆測試資料來做預測，使用predict()函數
test_predictions = model.predict(test_data[0:1])
print([round(i,4) for i in test_predictions[0].tolist()])
print('real answer: ', test_labels[0])

test_predictions = np.argmax(test_predictions, axis=1)
print(test_predictions[0])
```

MNIST 使用 CNN 進行學習

- 用 test 資料集做評估 (Keras)
  - 使用 model.evaluate() 和 model.predict()

```
import numpy as np

test_loss, test_acc = model.evaluate(test_data, test_labels)
print('loss: {:.3f}'.format(test_loss))
print('accuracy: {:.3f}'.format(test_acc))
```

```
10000/10000 [==============================] - 0s 30us/sample - loss: 0.0194 - acc: 0.9955
loss: 0.019
accuracy: 0.996
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0]
real answer:  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
7
```

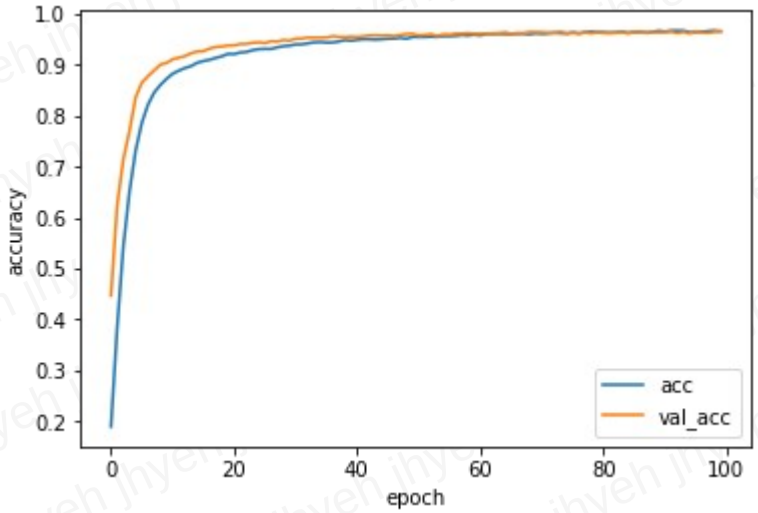
所以這次的成果就是 99.6% 正確性!

MNIST 使用 CNN 進行學習

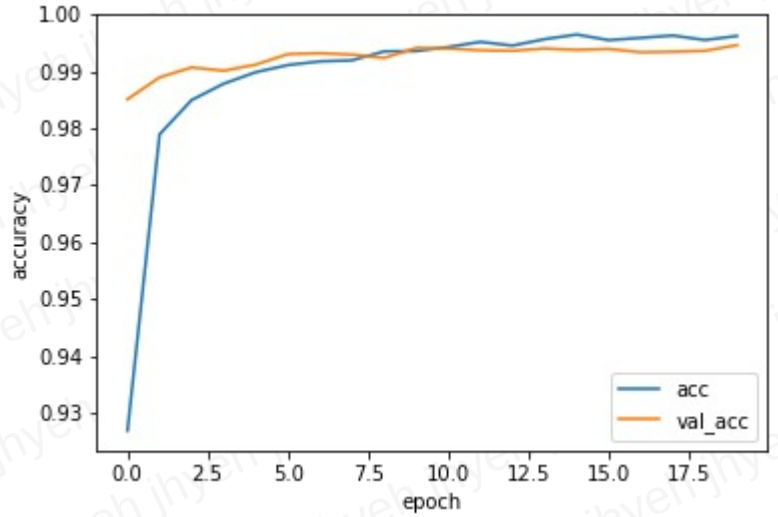
- 回顧檢討一下
  - 這次特徵處理和 MLP 的特徵處理有什麼不同?
  - 又換回了 SGD，有比較好嗎?
  - model.fit() 的參數 ...

比較一下 MLP 和 CNN

- MLP
  - Training: 96.5%
  - Test: 96.5%



- CNN
  - Training: 99.5%
  - Test: 99.6%



講次內容

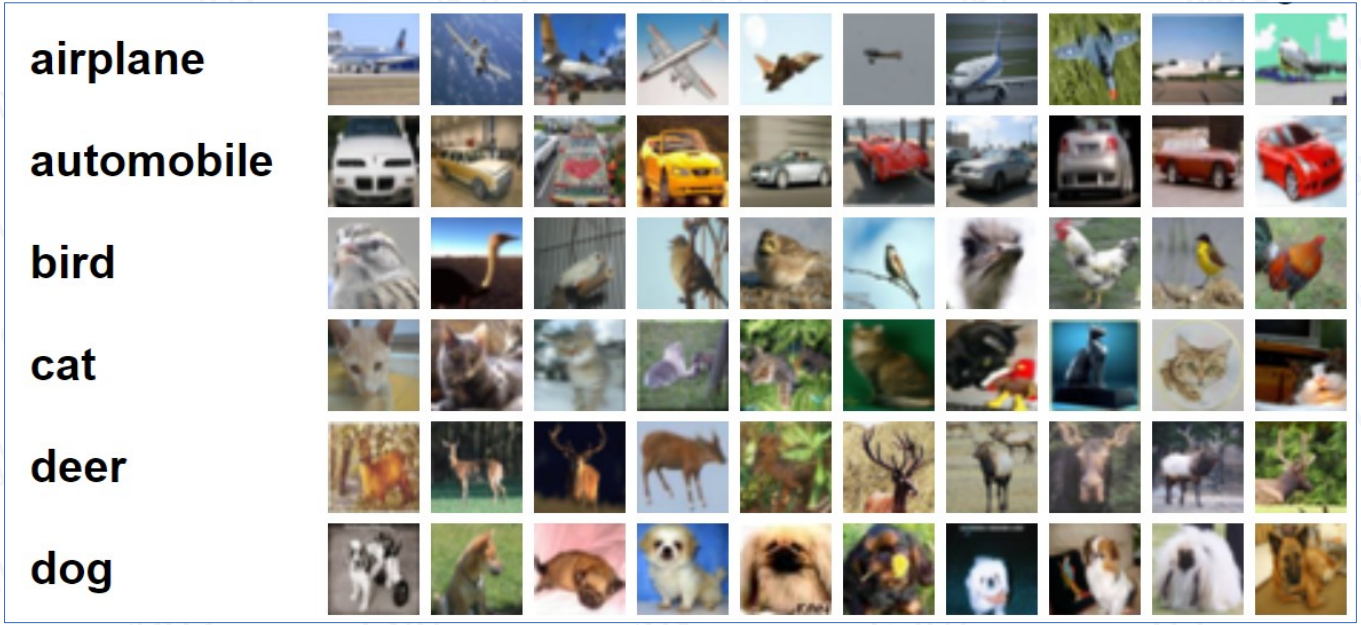
- 卷積神經網路 CNN 介紹
- CNN 案例：手寫辨識 MNIST
- CNN 案例： CIFAR-10 影像分類

CNN 案例： CIFAR-10 影像分類

- CIFAR-10？ 有特別嗎?

要注意資料維度囉!

  - 和 MNIST 灰階不同，是全彩影像



CIFAR-10 資料集

- 由 Hinton 的學生 Krizhevsky、 Sutskever 建立
  - 用來識別物體的資料集
- 10 個類別， 50000 筆訓練資料、 10000 筆測試資料
  - 另有 CIFAR-100 資料集， 100 個類別
- 各類物體的影像 (32x32 點全彩影像) 配合標籤
- 和 MNIST 比起來?
  - 類別發散、干擾資訊多、圖中識別的物體大小差異大

    難多

    了 ...

CIFAR-10 使用 CNN 進行學習

- 邏輯流程
  - 載入 CIFAR-10 資料集並分成

    train 和 test 兩部分 (Keras)
  - 資料前處理 (咦？！)
  - 根據資料維度，建立 Keras CNN 模型 (Keras)
  - 訓練 train 資料集 (Keras)
  - 用 test 資料集做評估 (Keras)

下載 CIFAR-10 資料集

- 使用 Keras API 下載
  - from tensorflow.keras.datasets import cifar10
  - (train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
  - Train 和 Test 都幫你分好了！！

下載 CIFAR-10 資料集

- ```
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
print('type of train images: ', type(train_images))
print('type of train labels: ', type(train_labels))
print('shape of train images: ', train_images.shape)
print('shape of train labels: ', train_labels.shape)
print('shape of test images: ', test_images.shape)
print('shape of test labels: ', test_labels.shape)

for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(train_images[i])
plt.show()

print(train_labels[0:10])
```
```

下載 CIFAR-10 資料集

-
```
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, te
print('type of train images: ', type(train_ima
print('type of train labels: ', type(train_lab
print('shape of train images: ', train_images.
print('shape of train labels: ', train_labels.
print('shape of test images: ', test_images.sh
print('shape of test labels: ', test_labels.sh

for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(train_images[i])
plt.show()

print(train_labels[0:10])
```

```
type of train images:  <class 'numpy.ndarray'>
type of train labels:  <class 'numpy.ndarray'>
shape of train images:  (50000, 32, 32, 3)
shape of train labels:  (50000, 1)
shape of test images:  (10000, 32, 32, 3)
shape of test labels:  (10000, 1)
```

```
[[6]
 [9]
 [9]
 [4]
 [1]
 [1]
 [2]
 [7]
 [8]
 [3]]
```

CIFAR-10 使用 CNN 進行學習

- 資料前處理　(?!)
  - 訓練資料維度： 50000x32x32x3
  - 代表 50000 張圖，每張 32x32， 三維資料（顏色深度 3 bytes， 24 位元彩色）
  - 準備使用 Convolution Layer，不需要拉平
    - 但是需要正規化到 (0,1) 之間
  - 標籤本來是 0 到 9，我們也將它 categorical 化

CIFAR-10 使用 CNN 進行學習

-

```
from tensorflow.keras.utils import to_categorical

print('before conversion, first pixel RGB value of first train image')
print(train_images[0][0][0])
train_images2 = train_images.astype('float32')/255.0
test_images2 = test_images.astype('float32')/255.0
print('after conversion, first pixel RGB value of first train image')
print(train_images2[0][0][0])

print('before to_categorical: ', train_labels[0])
train_labels2 = to_categorical(train_labels, 10)
test_labels2 = to_categorical(test_labels, 10)
print('after to_categorical: ', train_labels2[0])
```

  - 但是需要正

```
before conversion, first pixel RGB value of first train image
[59 62 63]
after conversion, first pixel RGB value of first train image
[0.23137255 0.24313726 0.24705882]
before to_categorical:  [6]
after to_categorical:  [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```

- 標籤本來是 0

CIFAR-10 使用 CNN 進行學習

- 根據資料維度，建立 Keras CNN 模型 (Keras)
  - 留意 Conv2D(), MaxPool2D(), Flatten()

CIFAR-10 使用 CNN 進行學習

```
from tensorflow.keras.layers import Activation, Dense, Dropout, Conv2D, Flatten, MaxPool2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32,32,3)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

print(model.summary())
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['acc'])
```

CIFAR-10 使用 CNN 進行學習

-

```
from tensorflow.ker
from tensorflow.ker
from tensorflow.ker

model = Sequential(

model.add(Conv2D(32
model.add(Conv2D(32
model.add(MaxPool2D
model.add(Dropout(0

model.add(Conv2D(64
model.add(Conv2D(64
model.add(MaxPool2D
model.add(Dropout(0

model.add(Flatten()
model.add(Dense(512
model.add(Dropout(0
model.add(Dense(10,

print(model.summary
model.compile(loss=
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 512)               2097664
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 2,168,362
Trainable params: 2,168,362
Non-trainable params: 0
_________________________________________________________________
None
```

```
 Flatten, MaxPool2D



hape=(32,32,3)))

, metrics=['acc'])
```

有沒有注意到

參數量爆增更多!

CIFAR-10 使用 CNN 進行學習

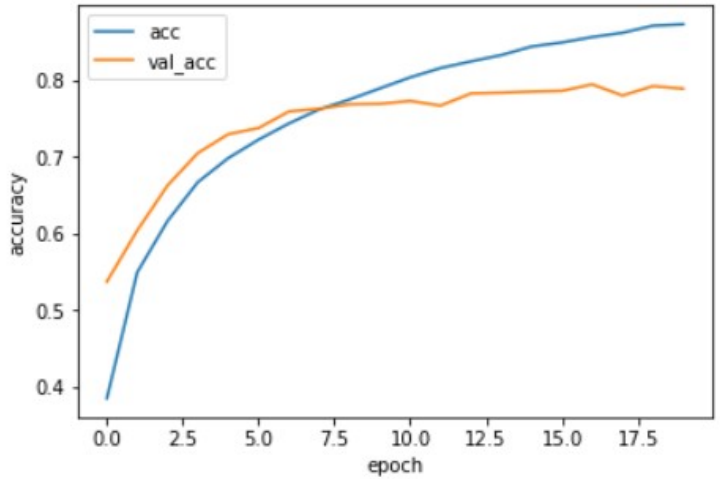
- 訓練 train 資料集 (Keras)
  - model.fit() 又出現了!

```
train_history = model.fit(train_images2, train_labels2, batch_size=128, epochs=20, validation_split=0.2)

plt.plot(train_history.history['acc'], label='acc')
plt.plot(train_history.history['val_acc'], label='val_acc')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(loc='best')
plt.show()
```

CIFAR-10 使用 CNN 進行學習

```
Epoch 18/20
40000/40000 [==============================] - 3s 63us/sample - loss: 0.3860 - acc: 0.8620 - val_loss: 0.6938 - va
l_acc: 0.7801
Epoch 19/20
40000/40000 [==============================] - 3s 66us/sample - loss: 0.3613 - acc: 0.8712 - val_loss: 0.6556 - va
l_acc: 0.7924
Epoch 20/20
40000/40000 [==============================] - 3s 63us/sample - loss: 0.3540 - acc: 0.8730 - val_loss: 0.6383 - va
l_acc: 0.7891
```



看得出來它比 MNIST 更難了嗎?

CIFAR-10 使用 CNN 進行學習

- 用 test 資料集做評估　(Keras)
  - 使用 model.evaluate() 和 model.predict()

CIFAR-10 使用 CNN 進行學習

- 用
  -

```
import numpy as np

test_loss, test_acc = model.evaluate(test_images2, test_labels2)
print('loss: {:.3f}'.format(test_loss))
print('acc: {:.3f}'.format(test_acc))

print('first 10 test labels ', np.argmax(test_labels2[:10], axis=1))
test_predictions = model.predict(test_images2[0:10])
print('first 10 test predictions ', np.argmax(test_predictions, axis=1))

labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',
          'frog', 'horse', 'ship', 'truck']
test_ans = np.argmax(test_labels2[0:10], axis=1)
print(type(test_ans))
print(test_ans.shape)
print('first 10 test labels', [labels[n] for n in test_ans])
pred = np.argmax(test_predictions, axis=1)
print(type(pred))
print(pred.shape)
print('first 10 prediction labels', [labels[n] for n in pred])
```

CIFAR-10 使用 CNN 進行學習

- 用

```
import numpy as np

test_loss, test_acc = model.evaluate(test_images2, test_labels2)
print('loss: {:.3f}'.format(test_loss))
```

```
10000/10000 [==============================] - 0s 48us/sample - loss: 0.6620 - acc: 0.7829
loss: 0.662
acc: 0.783
first 10 test labels  [3 8 8 0 6 6 1 6 3 1]
first 10 test predictions  [3 8 8 0 6 6 0 6 3 1]
<class 'numpy.ndarray'>
(10,)
first 10 test labels ['cat', 'ship', 'ship', 'airplane', 'frog', 'frog', 'automobile', 'frog', 'cat', 'automobile
']
<class 'numpy.ndarray'>
(10,)
first 10 prediction labels ['cat', 'ship', 'ship', 'airplane', 'frog', 'frog', 'airplane', 'frog', 'cat', 'automob
ile']
```

所以這次的成果就是 78.3% 正確性!

```
print('first 10 test labels', [labels[n] for n in test_ans])
pred = np.argmax(test_predictions, axis=1)
print(type(pred))
print(pred.shape)
print('first 10 prediction labels', [labels[n] for n in pred])
```

CIFAR-10 使用 CNN 進行學習

- 回顧檢討一下
  - 這次特徵處理和 MNIST 資料集有什麼不同?
  - 又換了 Adam，有比較好嗎?
  - model.fit() 的參數…
  - 為什麼訓練和測試正確性差那麼多?！

這個講次中，你應該學到了 ...

- 認識卷積神經網路 CNN 的運作方式
- 使用 CNN 處理灰階資料集 MNIST
- 使用 CNN 處理全彩資料集 CIFAR-10