三、數字與英文字母手寫辨識

01. 安裝套件

A. 安裝完成後，成功匯入模塊函數

【15】

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from keras.datasets import mnist
from keras.utils import to_categorical
from collections import Counter
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten,
Dropout
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
```

02. 兩個資料集讀取及合併與資料前處理

A. 讀取數字手寫數據集，並建立標籤名稱:

【16】

```
(x_train_mnist,y_train_minst),(x_test_mnist,y_test_minst) = mnist.load_data()

mnist_label_name = [i for i in range(10)]
print("標籤名稱：",mnist_label_name)
#print("訓練集數字手寫的類別數量:%s" %Counter(y_train_minst))
print("訓練集數字手寫的維度:",x_train_mnist.shape)
#print("測試集數字手寫的類別數量:%s" %Counter(y_test_minst))
print("測試集數字手寫的維度:",x_test_mnist.shape)
```

24. (x_train, y_train),(x_test, y_test) = mnist.load_data()
    讀取 keras 的數字手寫，回傳兩組數據集分別為訓練集與驗證集，其中各別包含特徵與標籤
25. Counter(number): 可以回傳 number 中每個元素的數量

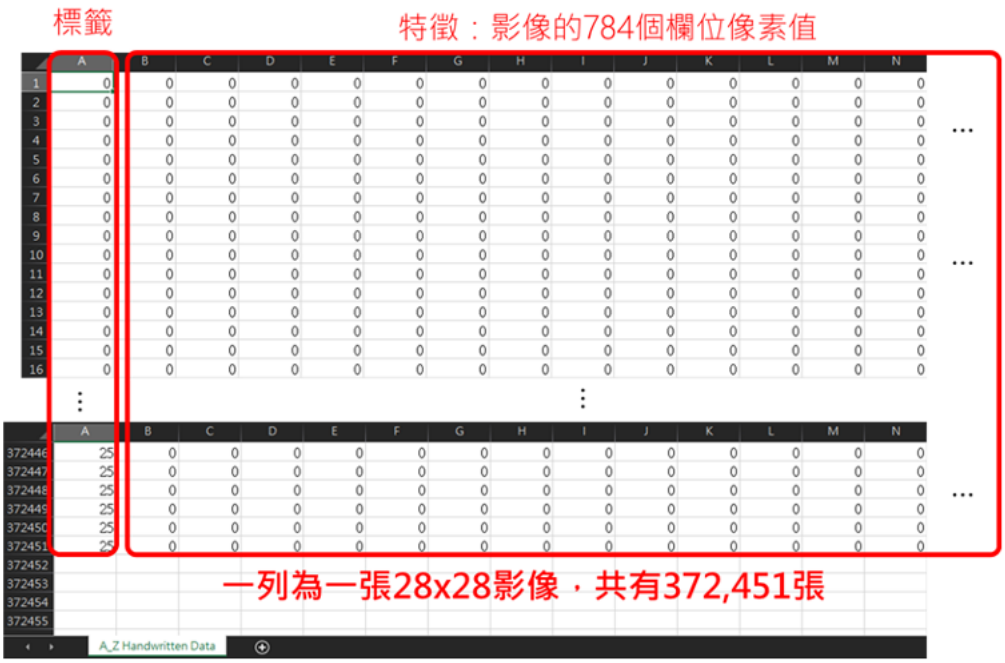
B. 讀取英文手寫數據集，將數據集分為訓練集與測試集，並建立標籤名稱

【17】

```
AZdata = pd.read_csv('A_Z Handwritten Data.csv', header = None)
print("英文手寫維度：",AZdata.shape)
AZ_label_name = [chr(i+65) for i in range(26)]
print("標籤名稱：",AZ_label_name)

AZ_label = np.array(AZdata)[:,0]
```

```
AZ_feature = np.array(AZdata)[:,1:785]
AZ_feature = AZ_feature.reshape(len(AZdata),28,28)
#print("英文手寫的類別數量:%s" %Counter(AZ_label))
print("英文手寫的維度:",AZ_feature.shape)

x_train_AZ, x_test_AZ, y_train_AZ, y_test_AZ = train_test_split(AZ_feature,
 AZ_label, random_state = 0, test_size = 0.4)

#print("訓練集英文手寫的類別數量:%s" %Counter(y_train_AZ))
print("訓練集英文手寫的維度:",x_train_AZ.shape)
#print("測試集英文手寫的類別數量:%s" %Counter(y_test_AZ))
print("測試集英文手寫的維度:",x_test_AZ.shape)
```

26. pandas.read_csv(path)：讀取 path 中的 csv 檔案，詳細參數如下列網址：
    https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html
27. numpy.array(data)：將 data 格式轉為陣列
28. numpy 中的 data.reshape(size)：將 data 改變陣列形狀為 size 大小並回傳，其中 data 必須為陣列。它與 numpy.reshape(data, size)類似，但不會回傳該改變後大小的陣列，而是直接取代原 data 陣列
29. x_train, x_test, y_train, y_test = train_test_split(x, y, random_state , train_size, test_size)：
    將 x 與 y 分為訓練與兩部分，分別為 x_train, x_test 與 y_train, y_test，通常 x 為特徵，y 為標籤，且兩個格式必需為列表（lists），numpy 中的陣列（arrays），或是 pandas 中的 dataframes 格式
    - random_state：預設為浮動數字，若固定其數值則每次對同樣資料拆分結果是相同的
    - train_size 與 test_size：預設 train_size 為 0.75，test_size 為 0.25，數值可設為 0 至 1 的浮點數，通常只設其中一個 size，因為另一半的 size 就已被確定了



圖五、英文字母手寫的 csv 格式

C．將兩個資料集合併，包含訓練與測試集中特徵與標籤以及標籤名稱：

【18】

```
x_train = np.vstack([x_train_mnist, x_train_AZ])
x_test = np.vstack([x_test_mnist, x_test_AZ])

y_train = np.hstack([y_train_minst, y_train_AZ+10])
y_test = np.hstack([y_test_minst, y_test_AZ+10])

label_name = np.hstack([mnist_label_name, AZ_label_name])
print(label_name)

no_one_hot_y_train = y_train
no_one_hot_y_test = y_test
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

x_train = np.expand_dims(x_train.astype('float32')/255,-1)
x_test = np.expand_dims(x_test.astype('float32')/255,-1)
print(x_train.shape)
```

30. numpy.vstack([A,B,C,...]): 沿著豎直方向將 A ,B,C 等矩陣堆疊起來
31. numpy.hstack([A,B,C,...]): 沿著水平方向將 A ,B,C 等矩陣堆疊起來
32. keras 中的 to_categorical(lable)：可將 label 作獨熱編碼（One-hot Encoding）
33. numpy.expand_dims(陣列, axis=num): 可將陣列在 num 增加一維度

03. 卷積神經網路訓練

A. 使用 Keras 中的序列模型來建立卷積神經網路，並添加兩層卷積層與池化層，後接全連接層：

【19】

```
cnn = Sequential()
cnn.add(Conv2D(32,(3,3),activation='relu', padding='same', input_shape=(28,28,1)))
cnn.add(Dropout(0.25))
cnn.add(MaxPooling2D((2,2)))
cnn.add(Conv2D(64,(3,3),activation='relu', padding='same'))
cnn.add(Dropout(0.25))
cnn.add(MaxPooling2D((2,2)))
cnn.add(Flatten())
cnn.add(Dropout(0.25))
cnn.add(Dense(1024, activation='relu'))
cnn.add(Dropout(0.25))
cnn.add(Dense(36, activation='softmax'))
cnn.summary()
```

34. model = Sequential(模型架構)：為順序性模型，可以直接在模型架構添加各網路層，或是 model.add（）方式來添加，如下兩個例子：
    - model = Sequential([Dense(32,input_shape=(784,)),
      Activation('relu'),
      Dense(10),
      Activation('softmax'),])
    - model = Sequential()
      model.add(Dense(32, input_dim=784))
      model.add(Activation('relu'))
      model.add(Dense(10))
      model.add(Activation('softmax'))

35. model.summary()：輸出神經網路架構，包含各層的輸出維度與其權重（參數）數量
36. Conv2D(filters, kernel_size, activation=激勵函數, padding=填充, ...)：2D 卷積層
    - filters：卷積濾波器數量
    - kernel：卷積核大小
    - activation 激勵函數：常見的有「softmax」、「elu」、「tanh」、「sigmoid」與「relu」等
    - padding 填充方式：可選擇「valid」，「causal」 或 「same」，詳細如下
        - valid：表示不填充，為預設值
        - causal：表示因果膨脹卷積
        - same：表示零填充輸入以使輸出具有與原始輸入相同的長度

    詳細參數可以參照網站：https://keras.io/api/layers/convolution_layers/convolution2d/

37. Dropout(prob)：捨棄層，在訓練中每次更新時，將輸入單元的按機率隨機設置為 0
38. Flatten()：將輸入展平
39. Dense(units, activation=激勵函數, ...)：全連接層，units 為該層的神經元數量

B. 模型進行編譯，定義要使用的損失函數與優化器：

【20】
```
cnn.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

40. model.compile(loss= 損失函數, optimizer= 優化器, metrics=[評價函數])
    - 損失函數常見有：
        - mean_squared_error：用於回歸任務
        - categorical_crossentropy：用於分類任務，輸出層為 softmax
        - binary_crossentropy：用於二元分類任務，輸出層為 sgmoid
    - 優化器常見有「SGD」、「RMSprop」、「Adagrad」、「Adam」與「AdamW」等
    - 評價函數與損失函數相似，只不過評估函數的結果不會用於訓練過程中

C. 模型進行訓練：

【21】
```
history = cnn.fit(x=x_train, y=y_train, batch_size=128, epochs=20, validation_split= 0.1)
```

41. history = model.fit (x, y, batch_size, epochs, verbose, validation_split, validation_data,...):

回傳值：將每迭代次數訓練存在 history 的變數內

輸入參數：

- x: 訓練集的特徵，必須維度與輸入層所定義的維度相同
- y: 訓練集的標籤，必須與輸出層所定義的維度相同（需要注意是否用 one-hot encoding）
- batch_size: 每次更新梯度的樣本數量，預設為 32
- epochs: 訓練模型迭代次數，預設為 1
- verbose: 訓練紀錄顯示模式，0 = 安靜模式, 1 = 進度條, 2 = 每輪一行，預測為 1
- validation_split: 從訓練集分割的驗證集數量比例，必須為 0 至 1 的浮點數，預設為 0.0
- validation_data: 指定給予的驗證集，預設為 None

其他詳細參數如下列網址：https://keras.io/api/models/model_training_apis/

04. 模型驗證、預測與可視化

A. 繪製學習曲線：

【22】
```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.rcParams["font.family"] = "serif"
plt.title("Training & Validation", fontsize=20)
plt.xlabel("Iteration", fontsize=18)
plt.ylabel("Accuracy", fontsize=18)
plt.plot(np.arange(len(acc)), acc,color='b', label="Training set", marker='o', markers
ize=5)
plt.plot(np.arange(len(val_acc)), val_acc,color='r', label="Validation set", marker='o
', markersize=5)
plt.xticks(np.linspace(0,19,20,endpoint=True),fontsize=14)
plt.yticks(fontsize=14)
plt.legend(loc='lower right',fontsize=14)
plt.show()

plt.title("Training & Validation", fontsize=20)
plt.xlabel("Iteration", fontsize=18)
plt.ylabel("Loss", fontsize=18)
```

```
plt.plot(np.arange(len(loss)), loss,color='b', label="Training set", marker='o', markersize=5)
plt.plot(np.arange(len(val_loss)), val_loss,color='r', label="Validation set", marker='o', markersize=5)
plt.xticks(np.linspace(0,19,20,endpoint=True),fontsize=14)
plt.yticks(fontsize=14)
plt.legend(loc='upper right',fontsize=14)
plt.show()
```

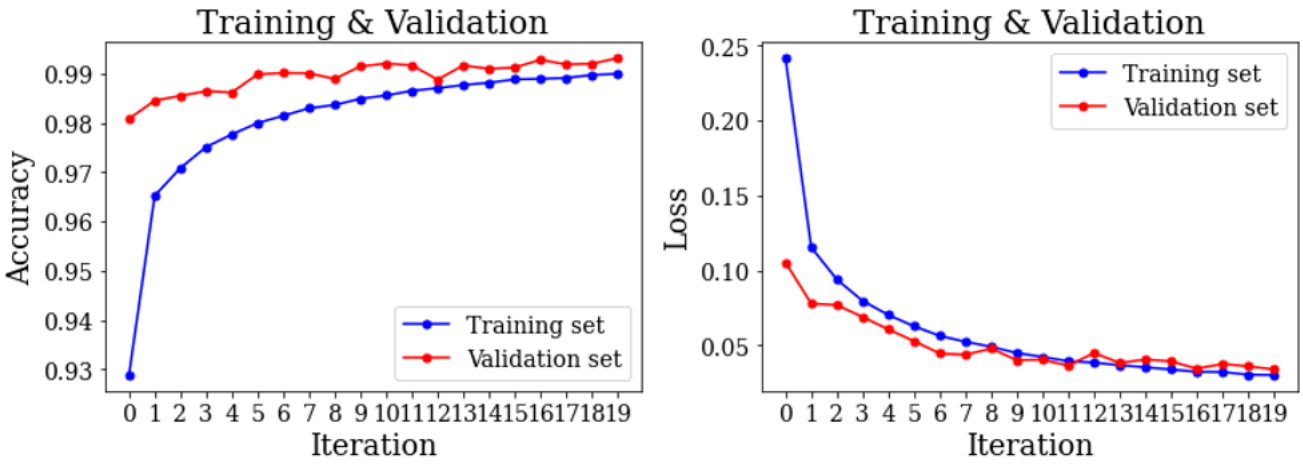
42. history.history 類似於字典的形式，內容儲存訓練與驗證集各別的準確度與損失，四種 key 分別是 acc、val_acc、loss 與 val_loss。若忘記 history 中的 key 可以打印 history.history.keys()來查看
43. 在 matplotlib 中更改字體可以使用下列方法：
    - 局部更改：

```
csfont = {'fontname':'Comic Sans MS'}
hfont = {'fontname':'Helvetica'}
plt.title('title',**csfont)
plt.xlabel('xlabel', **hfont)
plt.show()
```

    - 全域更改：

```
plt.rcParams["font.family"] = "serif"
```

    詳細參數如下網址：https://matplotlib.org/2.0.2/users/customizing.html
44. np.linspace(start, stop, num=5, endpoint=True,…)：生成等差級數的陣列
    - start：開始數值
    - stop：結束數值
    - num：start 到 stop 之間的數值個數
    - endpoint：True(包含 stop)；False(不包含 stop)，預設為 True



圖六、訓練與驗證集的準確度與損失學習曲線

B. 訓練集的評估與預測：

【23】
```
train_loss, train_acc = cnn.evaluate(x_train, y_train)
print("訓練集的準確度為：%0.4f" %(train_acc))
print("訓練集的損失值為：%0.4f" %(train_loss))
```

45. loss, acc = model.evaluate(x, y)：用於評估已經過訓練的模型。返回模型的損失值與準確度

【24】
```
predict = cnn.predict(x_train)
predictions = [np.argmax(one_hot)for one_hot in predict]
```

46. predict = model.predict(x)：用於實際預測。它爲輸入樣本的輸出預測機率

C. 訓練集的混淆矩陣：

【25】
```
cm = confusion_matrix(no_one_hot_y_train, predictions)
plt.figure(figsize=(20,20))
sns.heatmap(cm, annot=True, fmt=".0f", linewidths=1.0,
            square = True, cmap = 'Oranges',annot_kws={"size": 12})
plt.ylabel('Actual label', size = 18)
plt.xlabel('Predicted label', size = 18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.title('Accuracy: %0.4f' %(train_acc), size = 20)
```

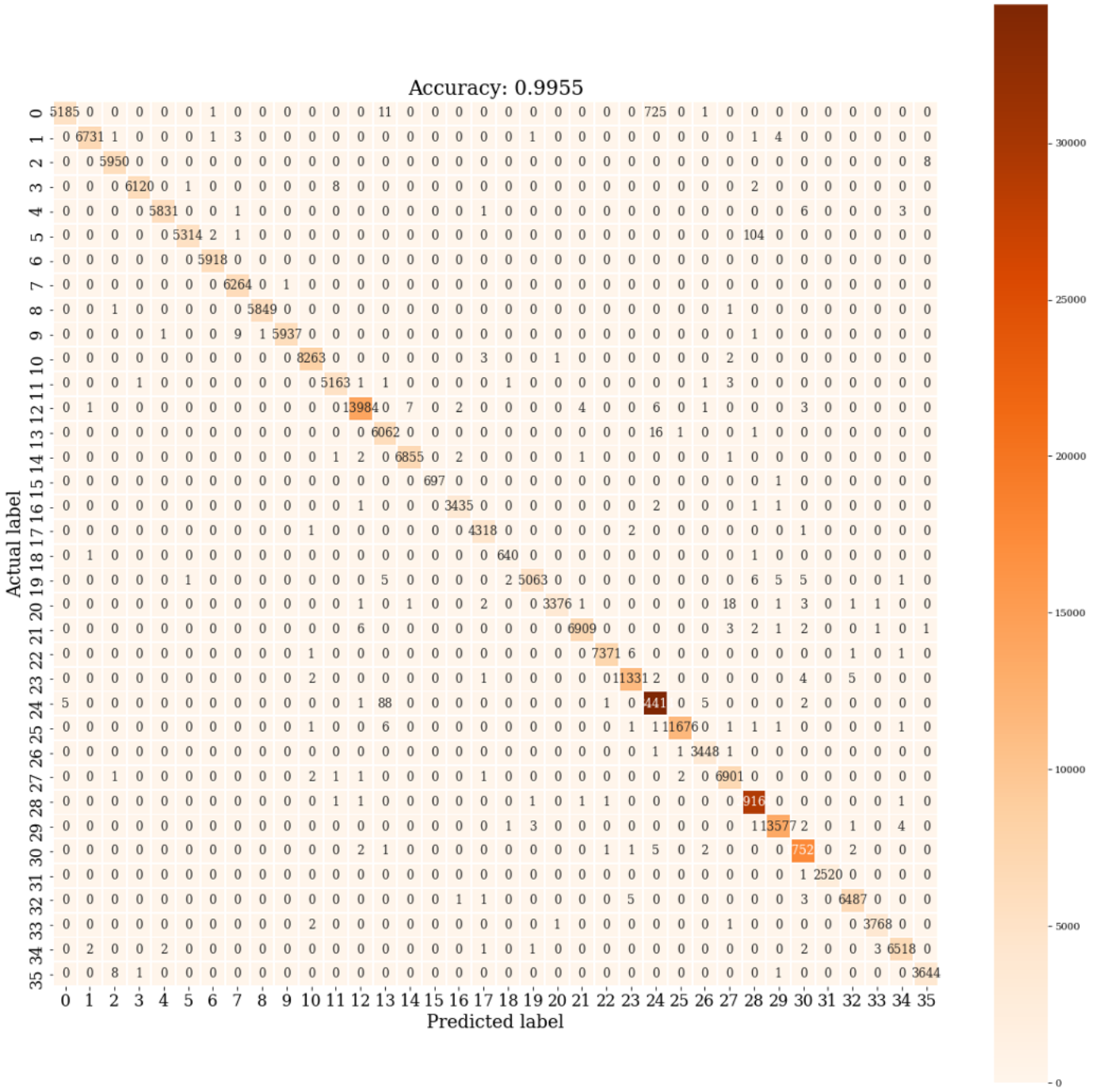
47. Scitik-learning 中的 confusion_matrix(y_true, y_pred)：回返混淆矩陣，y_true 為實際標籤，y_pred 為預測標籤
48. Seaborn 中的 heatmap(data, annot, fmt, linewidths, square , cmap, annot_kws, ...)：可繪製熱圖

常見的參數值為：

- data：繪出熱圖的輸入資料
- annot：若為 True，則在每個單元格中呈現數值，預設為 None
- fmt：顯示的數字格式
- linewidths：將劃分每個單元格的線寬度
- square：若為 True，則將兩軸縱橫長設置為相同，即每個單元格為正方形
- cmap：從數據值在色彩空間的映射。如果未提供，則預設為 center

詳細參數如下網址：https://seaborn.pydata.org/generated/seaborn.heatmap.html

測試集的評估、預測與混淆矩陣，相同上述方法



圖七、訓練集的混淆矩陣

04. 模型儲存與讀取

【26】

```
cnn.save('cnn_model.h5')
from tensorflow.keras.models import load_model
cnn = load_model('cnn_model.h5')
```

49. model.save('path.h5')：將 model 模型儲存到 path 位置，副檔名必須為.h5
50. model = load_model('path.h5')：讀取 path 位置的模型