車牌辨識系統

一、資料前處理

01. 處理正樣本影像

A. 匯入模塊函數

【01】
```
import os, shutil, glob, PIL
from time import sleep
from PIL import Image
```

B. 建立 emptydir 函數，功用為建立資料夾，若原本就存在則做刪除再建立

【02】
```
def emptydir(dirname):
    if os.path.isdir(dirname):
        shutil.rmtree(dirname)
        sleep(2)
    os.mkdir(dirname)
```

1. os.path.isdir(path): 回傳布林值，表示該位置(path)資料夾是否存在
2. shutil.rmtree(path): 依序刪除該資料夾中的檔案
3. sleep(): 延遲秒數
4. os.mkdir():創建資料夾

C. 建立 dirResize 函數，功用為輸建資料夾與影像位置，將影像改大小（300,225）儲存至 emptydir 所建立的資料夾中

【03】
```
def dirResize(src, dst):
    myfiles = glob.glob(src+'/*.jpg')
    emptydir(dst)
    for i, f in enumerate(myfiles):
        img = Image.open(f)
        imgnew = img.resize((300,225),PIL.Image.ANTIALIAS)
        imgnew.save(dst+'/'+'resize'+str('{:0>3d}').format(i+1)+'.bmp')
```

5. glob.glob(path): 返回該位置的文件名稱（只包括當前資料夾的文件，不包括子資料夾裡的文件）
6. Image.open(path): 讀取該位置的影像
7. img.resize((大小), PIL.Image.ANTIALIAS): 更改 img 的影像大小，並用插值方式縮放
8. img.save(path): 儲存 img 的影像至該位置

D. 使用 dirResize 函數來將 carPlate_sr 資料夾內的影像更改大小，並儲存至 carPlate 資料夾中

【04】
```
dirResize('carPlate_sr', 'carPlate')
```

02. 處理負樣本影像

A. 以同樣方式將負樣本影像作一樣的前處理，但是考慮正樣本車牌部分幾乎皆為黑白，為了提高訓練時的難易度，並將負樣本轉為灰階，並且將訓練負樣本的影像大小（500,376）比正樣本大

【05】

```
myfiles = glob.glob("carNegative_sr/*.jpg")
emptydir('carNegative')
for i, f in enumerate(myfiles):
    img = Image.open(f)
    imgnew = img.resize((500, 375), PIL.Image.ANTIALIAS)
    imgnew = imgnew.convert('L')
    imgnew.save('carNegative/'+'negGray'+str('{:0>3d}').format(i+1)+'.bmp')
```

9. img.convert('L'): 將該 img 的影像轉為灰階
10. {:0>3d} 表示 3d 只顯示三位整數。:0>表示原數字在左邊沒滿三位數，則用 0 補起來。:0< 則是補右邊

03. 建立負樣本的標註檔案

A. 新建負樣本的標註 txt 檔案，檔案內容包含影像位置與標註數量及位置，由於負樣本都沒有車牌，所以自行建立

【06】

```
fp = open('Haar-Training-master/Haar-Training-master/training/negative/bg.txt','w')
files = glob.glob('Haar-Training-master/Haar-Training-master/training/negative/*.bmp')
text = ""
for file in files:
    basename = os.path.basename(file)
    filename = 'negative/'+ basename
    text += filename + "\n"
print(text)
fp.write(text)
fp.close()
```

11. file_obj = open("path.txt", mode='w'): 開啟 path 的檔案，mode 常用的有:
    (1) r 為預設，只開啟檔案供讀取
    (2) w 開啟檔案供寫入，如果原先檔案有內容將被覆蓋
    (3) a 開啟檔案供寫入，如果原先檔案有內容，新寫入資料將附加在後面
    (4) x 開啟一個新的檔案供寫入，如果鎖開啟的檔案已經存在會產生錯誤

    file_obj.write("字串"): 將字串寫入檔案

    file_obj.close(): 不使用檔案時，將檔案正常關閉
12. os.path.basename(path): 用於去掉資料夾的的路徑，只返回文件名。（若要用於去文件名，但保留路徑，則使用 os.path.dirname(path) ）

04. 建立正樣本的標註檔案

A．由於 04 與 05 這部分需要使用到正樣本影像已經標註好的檔案，因此要先行去下載 Open Haar 分類器，自行標註。可先跳到下個單元（二、安裝與訓練車牌號碼 Haar 特徵分類器模型）

B. 安裝好 openCV 的 harr 後，開啟 Haar-Training-master\training\positive\objectmarker.exe 進行檔案標註

- 按下空白鍵就可將框選資料記錄下來
- 按下 Enter 鍵完成這張影像標記

C. 標注完成後，會在 Haar-Training-master\training\positive 的資料夾中，產生 info.txt 的標注檔案，它的資料結構為影像位置、標註數量及位置

05. 調整正樣本的影像與標註檔案

A. Haar 分類器會根據訓練時的寬高比來框選物件，由於舊式車牌為六碼而新式車牌為七碼，因此有機率會考慮舊式車牌來框選物件，導致新式車牌只截取到部分車牌號碼。因此需要調整寬高比，根據計算新式車牌的寬高比為 1:3.8。以下程式要把寬高比小於 3.8 的影像調整至 3.8:

【07】

```
fp = open('Haar-Training-master/Haar-Training-master/training/positive/info.txt', 'r')
lines = fp.readlines()
rettext = ''

for line in lines:
    data = line.split(' ')
    rettext += data[0] + ' ' + data[1] + ' '
    for i in range(int(n)):
        x = float(data[2+i*4])
        y = float(data[3+i*4])
        w = float(data[4+i*4])
        h = float(data[5+i*4])
        if (w/h) < 3.8:
            newW = h * 3.8
            x -= int((newW - w) / 2)
                if x<=0:
                    x=0
            w = int(newW)
        rettext = rettext+str(int(x))+' '+data[3+i*4]+' '+str(int(w))+' '+data[5+i*4]
fp.close()

fp = open('Haar-Training-master/Haar-Training-master/training/positive/info.txt', 'w')
fp.write(rettext)
fp.close()
```

13. str.split(str="分割符號", num=數值)：將 str 使用分割符號進行分割，分割 num+1 個。若沒有設置 num 數值，則字串採用全部分割

06. 影像增量

A. 由於正樣本影像太少（不均勻類別）會使得訓練的模型預測能力很差。增量的方法很多，本教材我們使用裁減的方法進行增量，我們針對四個角落各別移除邊緣長寬 10%來產生新影像，以左上角為例：移除上方 30 像素宇左方 22 像素，再將移除後的新影像放大至 300x225 像素（需要設立條件式防止裁減的位置將車牌分割掉）。除了影像進行增量外，標注檔案也要進行更新

【08】

```
path = 'Haar-Training-master/Haar-Training-master/training/positive/'
fp = open(path + 'info.txt', 'r')
lines = fp.readlines()
count = len(glob.glob("carPlate/*.bmp"))

rettext = ''
for line in lines:
    data = line.split(' ')
    img = Image.open(path + data[0])

    x = int(data[2])
    y = int(data[3])
    w = int(data[4])
    h = int(data[5])

    reduceW = 30 #減少的的寬度
    reduceH = int(reduceW*0.75) #減少的的高度
    multi = float(300/(300-reduceW)) #原圖與新圖比例
    neww = int(w*multi) #新圖的寬
    newh = int(h*multi) #新圖的高

    #移除左上角圖
    if (x-reduceW)>5 and (y-reduceH)>5: #左上角有空間才移除左上角
        count += 1 #編號加 1,此數值會做為檔名用
        newimg = img.crop((reduceW, reduceH, 300, 225)) #擷取圖形
        newimg = newimg.resize((300, 225), Image.ANTIALIAS) #放大圖形
        newimg.save(path + 'rawdata/bmpraw{:0>3d}.bmp'.format(count), 'bmp') #存檔
        newx = int((x-reduceW)*multi) #新圖 X 坐標
        newy = int((y-reduceH)*multi) #新圖 Y 坐標
        rettext = rettext+'rawdata/bmpraw{:0>3d}.bmp'.format(count)+' '+'1'+'
'+str(newx)+' '+str(newy)+' '+str(neww)+' '+str(newh)+'\n' #記錄新影像資料

    #移除右上角圖
    if (x+w)<(300-reduceW-5) and y>(reduceW+5):
        count += 1
        newimg = img.crop((0, reduceH, (300-reduceW), 225))
```

```
            newimg = newimg.resize((300, 225), Image.ANTIALIAS)
            newimg.save(path + 'rawdata/bmpraw{:0>3d}.bmp'.format(count), 'bmp')
            newx = int(x*multi)
            newy = int((y-reduceH)*multi)
            rettext = rettext+'rawdata/bmpraw{:0>3d}.bmp'.format(count)+' '+'1'+' '+str(new
x)+' '+str(newy)+' '+str(neww)+' '+str(newh)+'\n'

        #移除左下角圖
        if (x-reduceW)>5 and (y+h)<(225-reduceH-5):
            count += 1
            newimg = img.crop((reduceW, 0, 300, 225-reduceH))
            newimg = newimg.resize((300, 225), Image.ANTIALIAS)
            newimg.save(path + 'rawdata/bmpraw{:0>3d}.bmp'.format(count), 'bmp')
            newx = int((x-reduceW)*multi)
            newy = int(y*multi)
            rettext = rettext+'rawdata/bmpraw{:0>3d}.bmp'.format(count)+' '+'1'+' '+str(new
x)+' '+str(newy)+' '+str(neww)+' '+str(newh)+'\n'

        #移除右下角圖
        if (x+w)<(300-reduceW-5) and (y+h)<(225-reduceH-5):
            count += 1
            newimg = img.crop((0, 0, (300-reduceW), 225-reduceH))
            newimg = newimg.resize((300, 225), Image.ANTIALIAS)
            newimg.save(path + 'rawdata/bmpraw{:0>3d}.bmp'.format(count), 'bmp')
            newx = int(x*multi)
            newy = int(y*multi)
            rettext = rettext+'rawdata/bmpraw{:0>3d}.bmp'.format(count)+' '+'1'+' '+str(new
x)+' '+str(newy)+' '+str(neww)+' '+str(newh)+'\n'

    fp.close()

    fpmake = open(path + 'Info.txt', 'a')  #以新增資料方式開啟檔案
    fpmake.write(rettext)  #寫入檔案
    fpmake.close()
```

14. file_obj.readlines():讀取整個檔案所有行，儲存在一個列表(list)變數中，每行作為一個元素，但讀取大檔案會比較佔記憶體
15. img.crop((x1, y1, x2, y2)): 將 img 的影像進行裁減，將左上角與右下角裁減為(x1, y1)與(x2, y2)

二、安裝與訓練車牌號碼 Harr 特徵分類器模型

01. 安裝與調整套件

- OpenCV Haar 特徵分類器模型訓練 https://github.com/sauhaardac/haar-training
- OpenCV 官網 https://opencv.org/releases/ 中，下載 opencv-3.4.9-vc14_vc.exe
- 安裝 opencv-3.4.9-vc14_vc.exe，並打開其資料夾，作以下處理：

  A. 將 opencv\build\x64\vc15\bin 中的三個檔案，複製到 haar-training\tarining：

  a. opencv_createsamples.exe

  b. opencv_traincascade.exe

  c. opencv_world349.dll

  B. 刪除 haar-training/tarining 中的兩個檔案：

  a. createsamples.exe

  b. traincascade.exe

- 加入正/負樣本的影像，刪除 Haar-Training-master\training\negative 與 Haar-Training-master\training\positive\rawdata 原先檔案

- 注意 1：Harr 分類器的訓練樣本只提供 bmp 的影像副檔名檔案
- 注意 2：刪除與複製是因為 Github 中的 Harr 分類器僅供 opencv 3.4 版本使用，以下程式指令可查看其版本

【09】
```
import cv2
print(cv2.__version__)
```

02. 開始使用 Haar 訓練模型

A．更正打包的批次檔：

a. 正樣本影像必須以 opencv_createsamples.exe 檔打包為向量檔才能訓練。將正樣本影像打包向量檔的批次檔為 training\samples_creation.bat，請使用記事本打開 bat 檔，並按照下面文字修改其內容（註：文字內容是同一列）：

【10】
```
opencv_createsamples.exe -info positive/info.txt -vec vector/facevector.vec -num 497 -w 76 -h 20
```

16. 參數意義如下：
    - -info：正樣本標記檔路徑
    - -vec：產生的向量檔路徑
    - -num：正樣本影像數量
    - -w：偵測物件的寬度
    - -h：偵測物件的高度

b. 在使用 Haar 特徵分類器模型時，偵測物件的寬度與寬度設定非常重要，小於此設定值的區域將無法偵測，而且偵測時會使用此寬高比進行偵測。因為車牌號碼寬高比為 3.8，所以我們將設定最小高度為 20 像素，再將寬度設為 20x3.8=76 像素

B．執行打包的批次檔，開始打包向量檔：

更正好後，點兩下執行 samples_creation.bat 會在 training\vector 的資料夾中產生 facevector.vec 向量檔

C．更正 Haar 特徵分類器訓練的批次檔：

訓練 Haar 特徵分類器的批次檔為 training\haarTraining.bat，請使用記事本打開 bat 檔，並按照下面文字修改其內容（註：文字內容是同一列）：

【10】	opencv_traincascade.exe -data cascades -vec vector/facevector.vec -bg negative/bg.txt -numPos 497 -numNeg 293 -numStages 15 -w 76 -h 20 -minHitRate 0.9999 -precalcValBufSize 512 -precalcIdxBufSize 512 -mode ALL

17. 參數意義如下：

- -data: 指定儲存訓練結果的資料夾
- -vec: 指定正樣本向量檔路徑
- -bg: 指定負樣本資料檔路徑
- -numPos: 指定正樣本影像數量
- -numNeg: 指定負樣本影像數量
- -numStage: 訓練級數，級數越多，模型的偵測準確率越高，但訓練花費的時間越長。通常級數設為 15 到 25 之間
- -w、-h: 偵測物件的寬度與高度
- -minHitRate: 每一級需要達到的準確率
- -precalcValBufSize 及-precalcIdxBufSize: 使用的記憶體，單位為 「M」，訓練及使用的記憶體大小，記憶體越大，訓練所花費的時間越短。如果訓練過程中出現 「記憶體不足」訊息時，可適度減小此數值
- -mode: 訓練模式，使用哪些 Haar 特徵類型來訓練。「ALL」是使用所有特徵類型，「BASIC」是使用線性 Haar 特徵類型，「CORE」是使用線性及中心 Haar 特徵類型

D．執行 Haar 特徵分類器訓練的批次檔，開始訓練：

請先清空 training\cascades 資料夾中所有資料，然後再點兩下執行 training\haarTraining.bat。訓練時間相當長（大約兩小時左右），請耐心等候。訓練完成後，訓練結果會存於 training\cascades 資料夾中。其中 cascades.xml 就是訓練完成的模型檔

03. 使用 Haar 特徵分類器偵測車牌

A．讀取模型後進行分類器的辨識，並將車牌用矩形標註出來：

【11】

```
import cv2
import matplotlib.pyplot as plt
img_path = 'carPlate/resize001.bmp'

img = cv2.imread(img_path)
detector = cv2.CascadeClassifier("License_Plate_Haar_cascade.xml")
signs = detector.detectMultiScale(img, minSize = (76, 20), scaleFactor = 1.1, m
inNeighbors=10)
```

```
if len(signs) > 0:
    for (sx, sy, sw, sh) in signs:
        cv2.rectangle(img, (sx, sy), (sx+sw, sy+sh), (0, 0, 255), 2)
        print(signs)
else:
    print('沒有辨識到車牌!')

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

18. cv2.CascadeClassifier(path)：讀取 OpenCV 模型，其中 path 為模型路徑。若想方便性的尋找 haar 的模型，可以使用 cv2.data.haarcascades 的方法快速查找，但是還是不建議使用這方式，例：cv2.CascadeClassifier(cv2.data.haarcascades+" License_Plate_Haar_cascade.xml ")
19. output = detector.detectMultiScale(img, minSize , maxSize, scaleFactor , minNeighbors, flags) 參數意義如下：
    - minSize：此參數設定最小偵測區塊
    - maxSize：此參數設定最大偵測區塊
    - scaleFactor：偵測原理是系統會以不同區塊大小對影像掃描進行特徵比對，以參數設定區塊的改變倍數。常設為「1.1」
    - minNeighbors：此為控制誤檢率參數。系統以不同區塊大小進行特徵比對時，在不同區塊中可能會多次成功取得特徵，成功取得特徵數需達到此參數設定值才算偵測成功。預設為「3」
    - flags：此參數設定檢測模式，常見的值有：
        - cv2.CASECADE_SCALE_IMAGE：按比例正常檢測
        - cv2.CASECADE_DO_CANNY_PRUNING：利用 Canny 邊緣檢測器來排除一些邊緣很少或很多的影像區域
        - cv2.CASECADE_FIND_BIGGEST_OBJECT：只檢測最大的物體
        - cv2.CASECADE_DO_ROUGH_SEARCH：只做初略檢測
20. cv2.rectangle(img, (x1, y1), (x2, y2), color, line size)：將 img 的影像繪製矩形，其中（x1, y1）與（x2, y2）為矩形左上角與右下角，color 為(B,G,R)的顏色，line size 為矩形線的寬度

圖一、Haar 特徵分類器偵測方式

圖二、車牌檢測結果

04. 車牌號碼各別擷取

A. 將車牌位置的區域轉為二值化的影像：

【12】

```
img = cv2.imread(img_path)
crop_img = img[sy:sy+sh, sx:sx+sw]
gray_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
_, binary_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY_INV)

plt.imshow(binary_img, cmap='gray')
plt.show()
```

21. ret, binary_img = cv2.threshold(img, threshold, max_value, type)，詳細如下：

回傳值：

- ret：回傳找到的最佳門檻值，一般情況都忽略此參數
- binary_img: 已二值化的影像

輸入參數：

- img：輸入的灰階影像
- threshold：對像素值進行分類的門檻值
- max_value：當像素值超過了門檻值（或是小於門檻值，根據 type 來決定），所賦予的值
- type：二值化操作的類型，有下列五種類型：
  - cv2.THRESH_BINARY：將大於門檻值的灰階值設為最大灰階值，小於門檻值設為 0
  - cv2.THRESH_BINARY_INV：將大於門檻值的灰階值設為 0，其他值設為最大灰階值
  - cv2.THRESH_TRUNC：將大於門檻值的灰階值設為門檻值，小於門檻值的值保持不變
  - cv2.THRESH_TOZERO：將小於門檻值的灰階值設為 0，大於門檻值的值保持不變
  - cv2.THRESH_ TOZERO_INV：將大於門檻值的灰階值設為 0，小於門檻值的值保持不變

圖三、車牌二值化後的影像

B．將二值化的影像進行輪廓偵測：

【13】

```
img = cv2.imread(img_path)
crop_img = img[sy:sy+sh, sx:sx+sw]
contours, hierarchy = cv2.findContours(binary_img, cv2.RETR_EXTERNAL, cv2.C
HAIN_APPROX_SIMPLE)
for i in range(len(contours)):
    (x, y, w, h) = cv2.boundingRect(contours[i])
    print((x, y, w, h))
    cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 1)
plt.imshow(crop_img)
plt.show()
```

22. contours, hierarchy = cv2.findContours(img，偵測模式，輪廓算法)

回傳值：contours 為輪廓 及 hierarchy 為階層

常見的偵測模式：

- cv2.RETR_EXTERNAL：只偵測輪廓外緣（常用）
- cv2.RETR_LIST：偵測輪廓時不建立等級關係
- cv2.RETR_CCOMP：偵測輪廓時建立兩個等級關係
- cv2.RETR_TREE：偵測輪廓時建立樹狀等級關係

常見的輪廓算法：

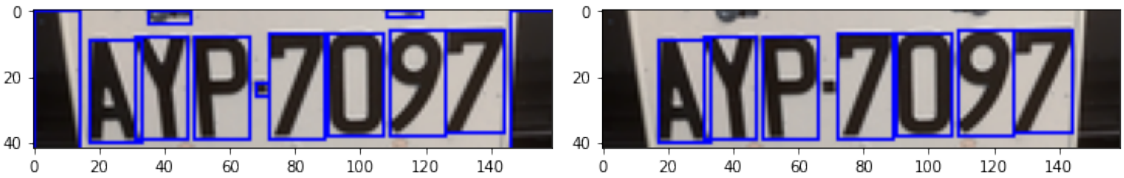
- cv2.CHAIN_APPROX_NONE：儲存所有輪廓點
- cv2. CHAIN_APPROX_SIMPLE：壓縮水平、垂直及對角線方向元素，只儲存該方向的終點，例如：舉行只儲存四個點，此算法速度較快（常用）

23. locations = cv2.boundingRect(輪廓資訊)：可將輪廓資訊以一個最小的矩形包圍起來，回傳該矩形的左上角位置與其寬高

C．設立條件式來移除非車牌號碼的輪廓區塊。若以像素的寬高來移除容易受到，鏡頭拍攝與車牌的距離而有落差，因此我們以比例關係來設立條件式：

【14】

```
img = cv2.imread(img_path)
crop_img = img[sy:sy+sh, sx:sx+sw]
contours, hierarchy = cv2.findContours(binary_img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
for i in range(len(contours)):
    (x, y, w, h) = cv2.boundingRect(contours[i])
    if sw*(3/20)>w>sw*(0.5/20) and sh*(19.5/20)>h>sh*(12/20):
        print((x, y, w, h))
        cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 1)
plt.imshow(crop_img)
plt.show()
```



圖四、最小矩形包圍後輪廓偵測的影像：（左圖）設條件式前 與（右圖）設條件式後

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

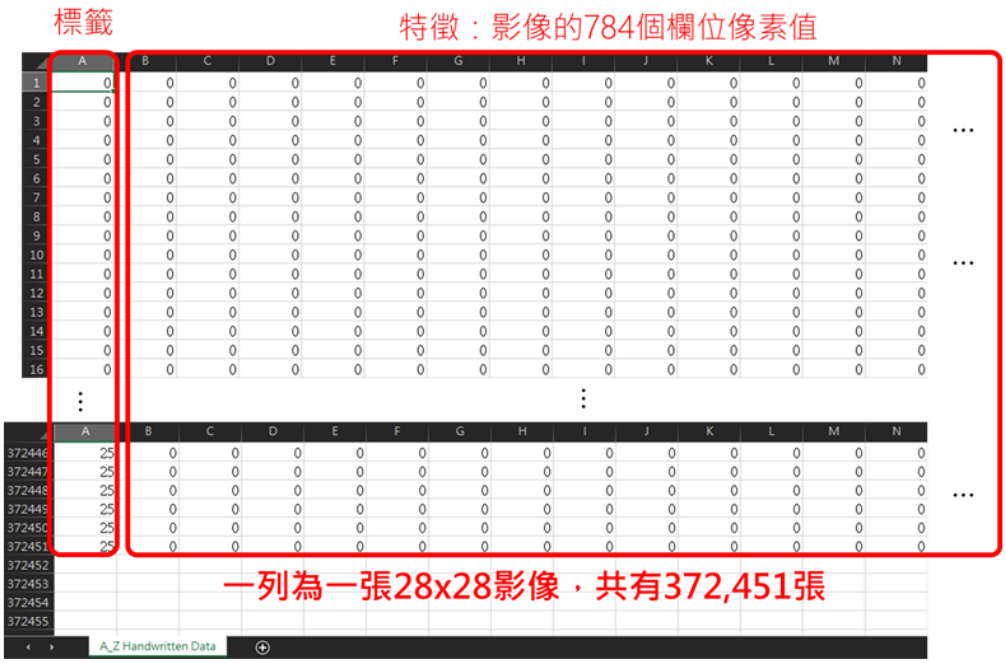
B. 讀取英文手寫數據集，將數據集分為訓練集與測試集，並建立標籤名稱

【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()
```

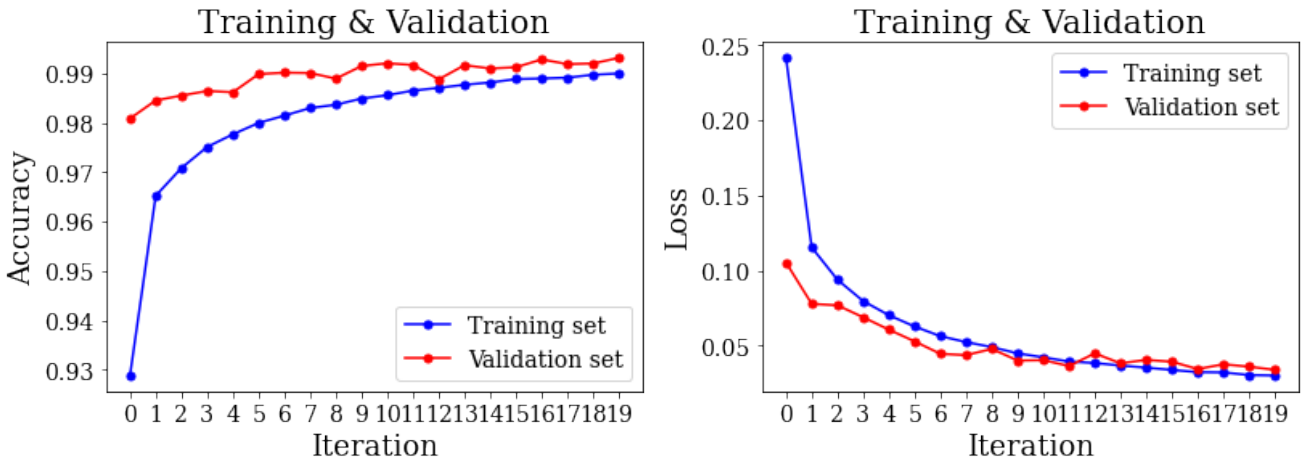
42. history.history 類似於字典的形式，內容儲存訓練與驗證集各別的準確度與損失，四種 key 分別是 acc、val_acc、loss 與 val_loss。若忘記 history 中的 key 可以打印 history.history.keys()來查看
43. 在 matplotlib 中更改字體可以使用下列方法：
    - 局部更改：

```
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)
```

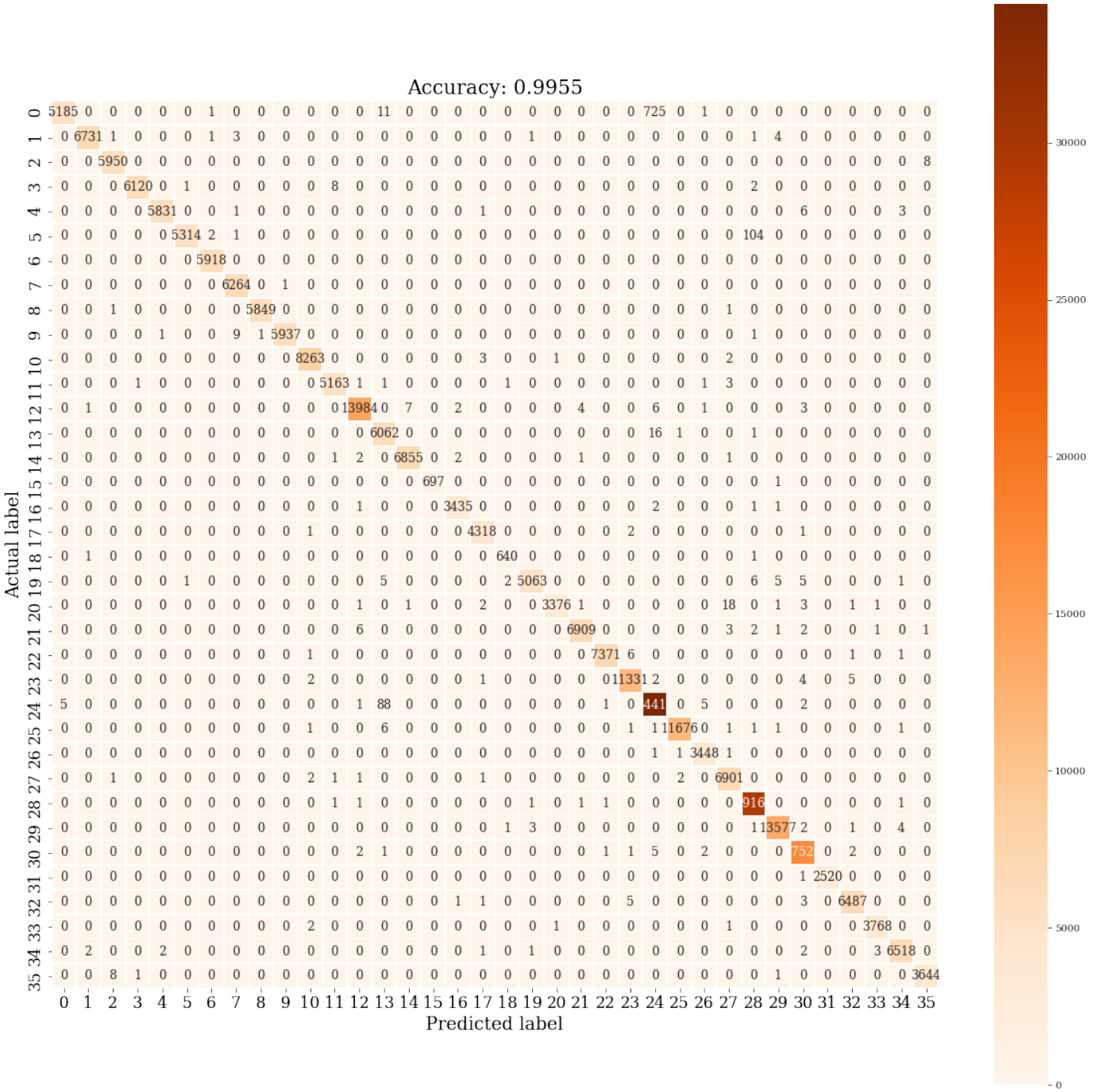
47. Scitik-learning 中的 confusion_matrix(y_true, y_pred)：回返混淆矩陣，y_true 為實際標籤，y_pred 為預測標籤
48. Seaborn 中的 heatmap(data, annot, fmt, linewidths, square , cmap, annot_kws, ...)：可繪製熱圖

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

四、車牌辨識應用

01. 影像測試車牌辨識系統

A. 匯入模塊、讀取影像 及 Haar 與 CNN 模型:

【27】

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.models import load_model

img_path = 'carPlate/resize010.bmp'
img = cv2.imread(img_path)

plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGRA2RGB))
plt.show()

detector = cv2.CascadeClassifier("License_Plate_Haar_cascade.xml")
cnn = load_model('cnn_model.h5')

mnist_label_name = [i for i in range(10)]
AZ_label_name = [chr(i+65) for i in range(26)]
label_name = np.hstack([mnist_label_name, AZ_label_name])
```

B. 進行 Haar 與 CNN 模型辨識:

【28】

```
signs = detector.detectMultiScale(img, minSize = (76, 20), scaleFactor = 1.1, minNeighb
ors=8)

if len(signs) > 0:
    for (sx, sy, sw, sh) in signs:
        crop_img = img[sy:sy+sh, sx:sx+sw]
        gray_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
        _, binary_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY_INV)

        save_predict_name = []
        save_contours = []
        contours, hierarchy = cv2.findContours(binary_img, cv2.RETR_EXTERNAL,
                                               cv2.CHAIN_APPROX_SIMPLE)
        for i in range(len(contours)):
            (x, y, w, h) = cv2.boundingRect(contours[i])
            save_contours.append((x, y, w, h))
        save_contours = sorted(save_contours, key=lambda x:x[0])
```

```
        for (x, y, w, h) in save_contours:
            if sw*(3/20)>w>sw*(0.5/20) and sh*(19.5/20)>h>sh*(12/20):
                number = binary_img[y:y+h, x:x+w]
                pad_number = cv2.copyMakeBorder(number, 13, 13, 10, 10,
                                                borderType=cv2.BORDER_CONSTANT)
                re_number = cv2.resize(pad_number, (28, 28),
                                       interpolation=cv2.INTER_CUBIC)
                intput_number = np.expand_dims(np.expand_dims(
                                re_number.astype('float32')/255,-1),0)
                predict = cnn.predict(intput_number)
                predict_name = label_name[np.argmax(predict)]
                save_predict_name.append(predict_name)
        print("".join(save_predict_name))
else:
    print('沒有辨識到車牌!')
```

51. sorted(list, key=函數)：可以將 list 列表進行小至大的排序，key 是根據函數來指定 index 排序，常搭配 lambda 來應用

    註：list.sort(reverse=True)：也相同於上述功能，reverse=True 是指排序由大到小，False 則相反，預設為 False。與上個函數不同是他會直接對 list 直接更改取代

52. cv2.copyMakeBorder(img, top, bottom, left, right, borderType, value)：增加影像邊界（填充）
    - img：預輸入的原影像
    - top, bottem, left, right：分別表示四個方向上增加邊界的長度
    - borderType：增加邊界的類型
        - BORDER_REFLICATE：直接用邊界的顏色填充
        - BORDER_REFLECT：倒映
        - BORDER_REFLECT_101：倒映（邊界隔離）
        - BORDER_CONSTANT：增加的數值為 value 的顏色
    - value：顏色，用於 BORDER_CONSTANT 模式，預測為黑色

02. 開啟攝影機

【29】

```
VIDEO_IN = cv2.VideoCapture(0)
while True:
    hasFrame, img = VIDEO_IN.read()
    cv2.imshow("Frame", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
VIDEO_IN.release()
cv2.destroyAllWindows()
```

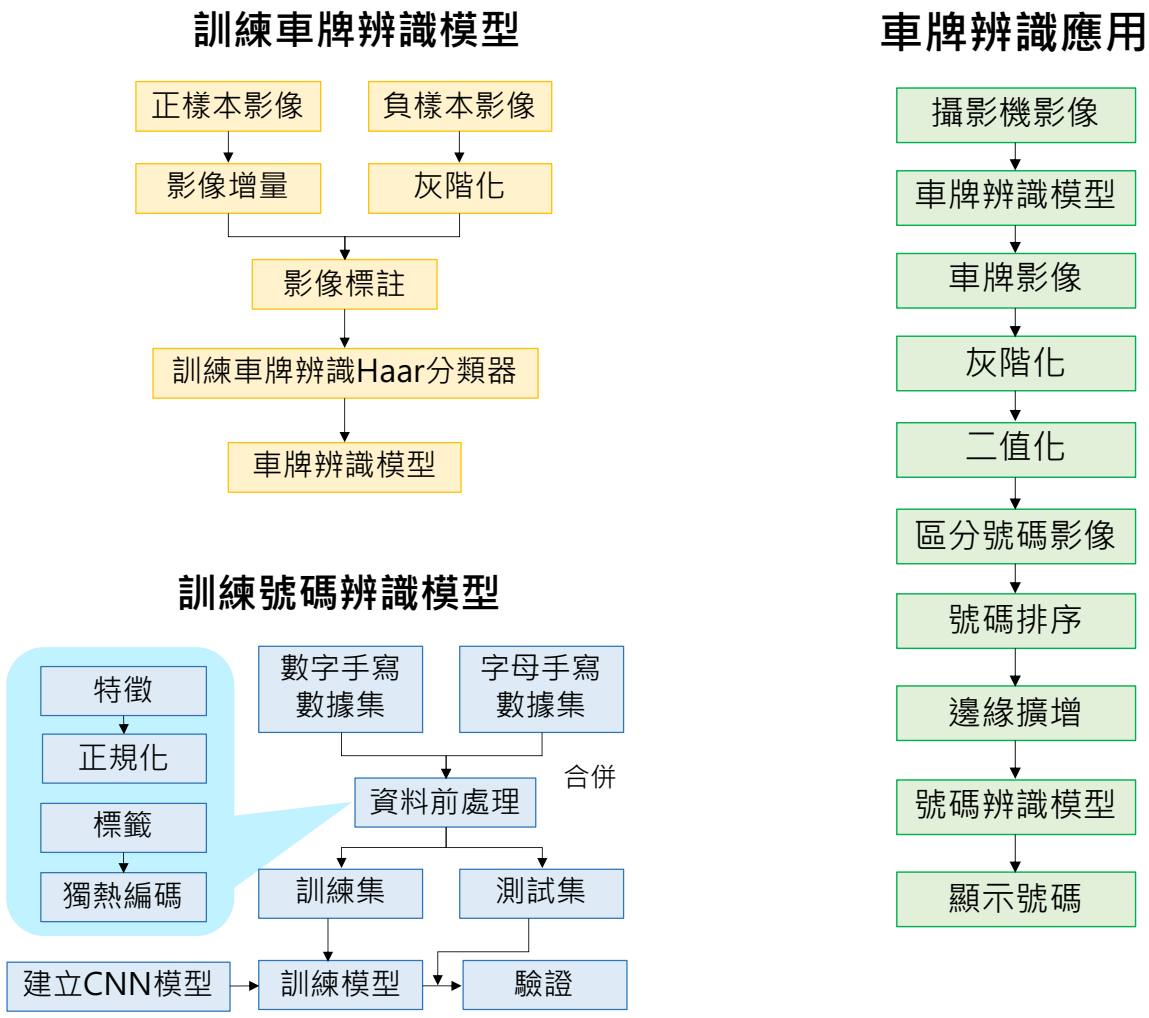
03. 攝影機測試車牌辨識系統

【30】

```
VIDEO_IN = cv2.VideoCapture(0)

while True:
    hasFrame, img = VIDEO_IN.read()
    signs = detector.detectMultiScale(img, minSize = (76, 20),
                                      scaleFactor = 1.1, minNeighbors=8)
    if len(signs) > 0:
        for (sx, sy, sw, sh) in signs:
            crop_img = img[sy:sy+sh, sx:sx+sw]
            gray_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
            _, binary_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY_INV)

            save_predict_name = []
            save_contours = []
            contours, hierarchy = cv2.findContours(binary_img, cv2.RETR_EXTERNAL,
                                                   cv2.CHAIN_APPROX_SIMPLE)
            for i in range(len(contours)):
                (x, y, w, h) = cv2.boundingRect(contours[i])
                save_contours.append((x, y, w, h))
            save_contours = sorted(save_contours, key=lambda x:x[0])

            for (x, y, w, h) in save_contours:
                if sw*(3/20)>w>sw*(0.5/20) and sh*(19.5/20)>h>sh*(12/20):
                    number = binary_img[y:y+h, x:x+w]
                    pad_number = cv2.copyMakeBorder(number, 13, 13, 10, 10,
                                                    borderType=cv2.BORDER_CONSTANT)
                    re_number = cv2.resize(pad_number, (28, 28),
                                           interpolation=cv2.INTER_CUBIC)
                    intput_number = np.expand_dims(np.expand_dims(
                                                   re_number.astype('float32')/255,-1),0)
                    predict = cnn.predict(intput_number)
                    predict_name = label_name[np.argmax(predict)]
                    save_predict_name.append(predict_name)

            if len(save_predict_name)>5:
                cv2.rectangle(img, (sx, sy), (sx+sw, sy+sh), (0, 0, 255), 2)
                cv2.putText(img, "".join(save_predict_name), (sx, int(sy-sh/4)),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
    else:
```

```
        cv2.putText(img, "No License_Plate", (580, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
    cv2.imshow("Frame", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
VIDEO_IN.release()
cv2.destroyAllWindows()
```

53. cv2.putText(img, text, org, fontFace, fontScale, color, boldface)：文字繪入影像上
    - img: 預繪入文字的影像
    - text: 預繪入的文字
    - org: 文字要放置的位置
    - fontFace: 字體的類型
    - fontScale: 字的顏色
    - boldface: 字的粗度



圖八、訓練車牌辨識與號碼模型及其應用的流程圖