

2021 機器學習期末報告

復仇者聯盟人臉辨識

-使用 CNN 模型

網科三甲 A107223007 陳佳玟

網科三甲 A107223205 張恩慈

2021/6/17

目錄

組員分工表.....	3
題目構想及功能.....	3
資料夾存放位置.....	4
程式操作說明.....	4
CNN 架構 (James, 2017)	5
程式架構.....	7
套件說明.....	8
基本定義.....	11
1. 資料前處理.....	12
完整程式.....	15
2.建構模型	16
1.CNN(Conv2d)	16
Model Summary	20
完整程式.....	22
模型訓練結果.....	23
2.VGG16 (李馨伊, 2020)	29
完整程式.....	30
Model Summary	31

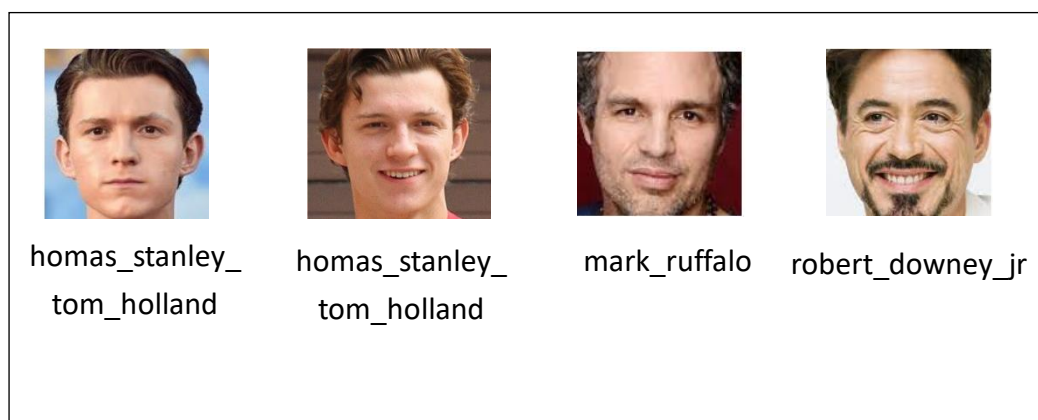
3. CNN+RNN(LSTM (嘗試理解 LSTM 和 CNN 的結合, 無日期)....	31
完整程式.....	33
訓練結果.....	34
3. 模型訓練及優化.....	35
4. 結果呈現.....	38
5. 調整超參數.....	40
預測程式.....	42
參考資料來源.....	46
主要程式來源.....	46
影片學習.....	47
心得.....	47

組員分工表

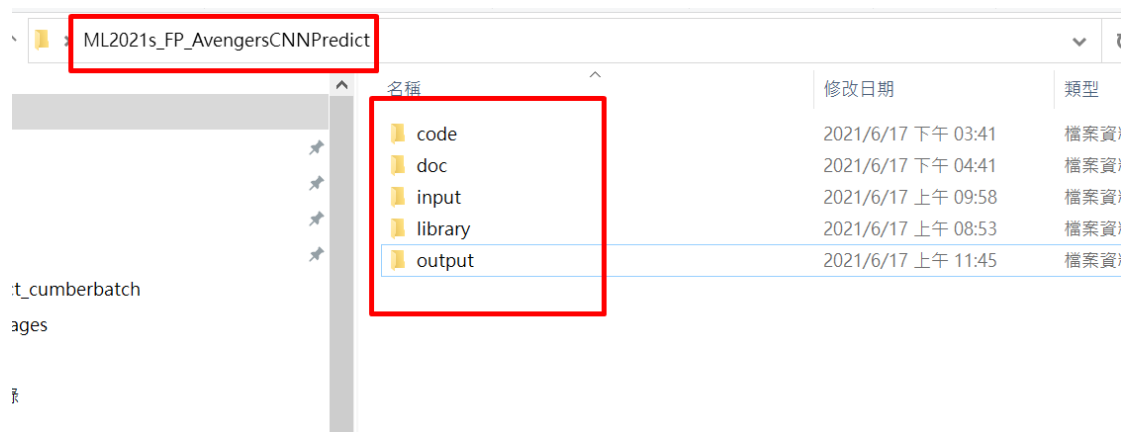
分配內容	陳佳妤	張恩慈
圖片蒐集	40%	60%
程式	100%	
參數調整、模型訓練	100%	
GUI 介面	100%	
書面報告	100%	
簡報製作	50%	50%
口頭報告	30%	70%

題目構想及功能

報告將透過 keras 實作卷積網路 CNN，進行「復仇者聯盟人臉辨識」。將整理好的資料放入 CNN 模型訓練，辨識出人物角色並輸出圖片正確對應之人物名稱，結果示意圖如下。



資料夾存放位置



程式碼存放在 code 資料夾底下；

相關文件存放在 doc 資料夾底下；

輸入圖片存放在 input 資料夾底下之 characters 資料夾，預測圖片請放在 test_images 資料夾內；

程式 module 環境存放在 library 底下，已壓縮 conda 環境；

輸出圖表及資料集存放在 library 資料夾底下。

程式操作說明

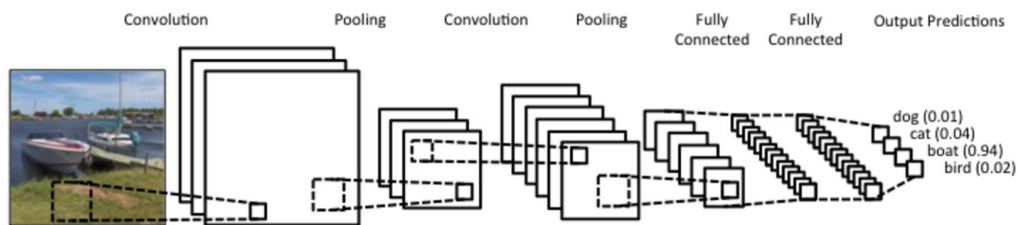
請打開 code 資料夾底下之 Avengers_CNN_Predict.py 檔，直接執行，預設模型為 CNN，若要使用其他模型，可執行已註解之程式碼。若要新增人物角色做訓練，請在../input/characters 資料夾新增以人物命名的資料夾，將圖片放置裡面，並在 Avengers_CNN_Predict.py 的 map_characters(人物類別名稱)新增新的角色名稱。主要程式目前有兩隻，Avengers_CNN_Predict.py 為主程

式，test_image.py 為預測程式，predict.py 為 GUI 介面。

CNN 架構 (James, 2017)

卷積神經網絡(Convolutional Neural Network)簡稱 CNN，

CNN 的概念圖如下：

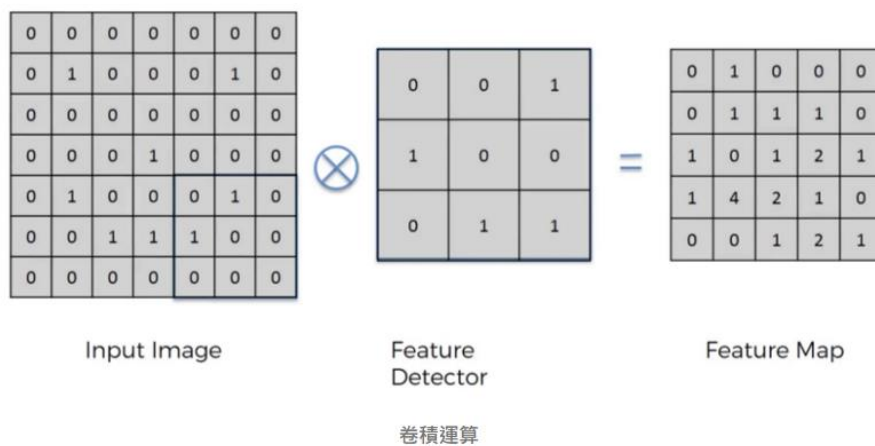


CNN 概念圖

1. Convolution Layer 卷積層

卷積運算就是將原始圖片的與特定的 Feature Detector(filter)做卷積運算

(符號 \otimes)，卷積運算就是將下圖兩個 3x3 的矩陣作相乘後再相加。



在其中使用激勵函數(activation function)，常見的激勵函數有 sigmoid,

tanh, Relu · 實用上最常使用 ReLU · ReLU 的優點為有效的克服梯度消失

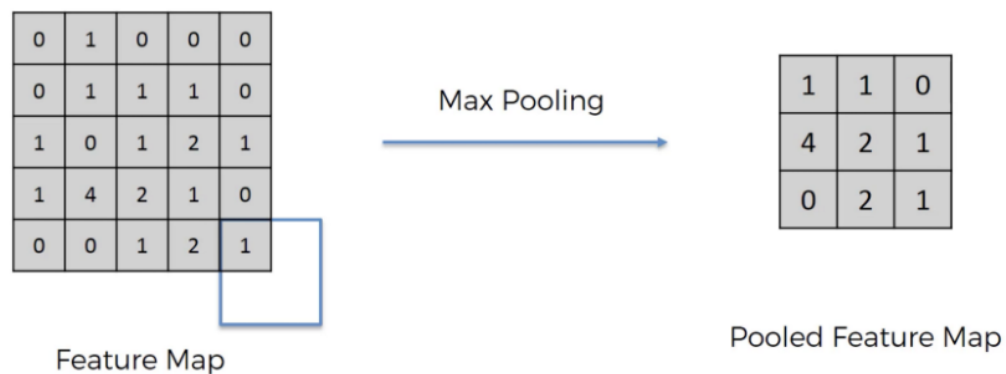
的問題、運算時間快，並可達到無窮多 sigmoid function 疊加的結果。

2. Pooling Layer 池化層

主要是採用 Max Pooling，計算過程中只挑出矩陣當中的最大值，好處是當

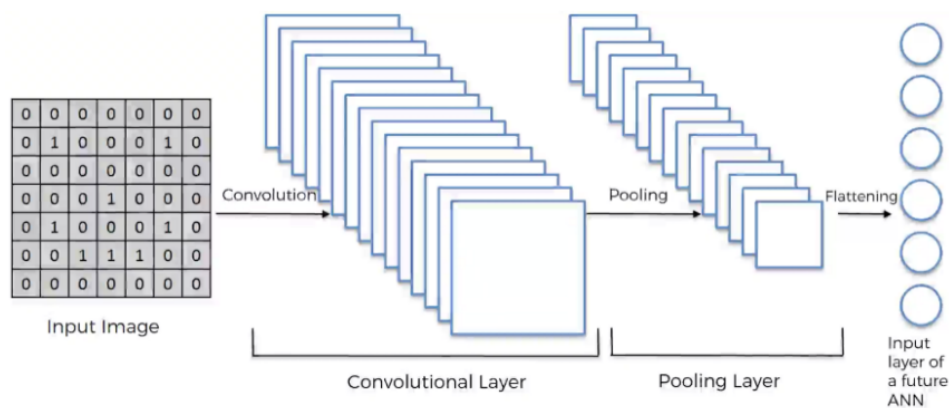
圖片整個平移幾個 Pixel 的話對判斷上完全不會造成影響，以及有很好的抗

雜訊功能。



3. Fully Connected Layer 全連接層->Dense Layer

將之前的結果平坦化之後接到最基本的神經網絡。



程式架構

1. 資料前處理
2. 建構模型
 - A. 模型訓練
 - B. 模型評估
 - C. 模型優化
 - D. 儲存模型
3. 結果展示

套件說明

定義資料夾路徑

```
import os
inputPath=os.path.join("../","input")
outputPath=os.path.join("../","output")

import glob #查找符合特定規則的文件路徑名
```

匯入一般套件

```
import numpy as np
import time
import matplotlib.pyplot as plt
```

匯入 OpenCV 套件

```
import cv2
```

*下載指令為

```
pip install opencv-python
pip install cv2
```

匯入 data process 套件

```
import h5py #儲存資料集

from random import shuffle #將序列的所有元素隨機排序
```

匯入機器學習套件

```
from collections import Counter
import math

import itertools #笛卡爾

import sklearn
from sklearn.model_selection import train_test_split

#劃分數據集
```

```
from sklearn.metrics import classification_report#分類報告
```

```
from sklearn.metrics import confusion_matrix  
from sklearn.metrics import plot_confusion_matrix  
import keras
```

#圖片生成器

```
from keras.preprocessing.image import ImageDataGenerator
```

#回調函數(call back) 返回學習速率；在每次訓練之後保存最佳模型

```
from keras.callbacks import LearningRateScheduler,  
ModelCheckpoint, TensorBoard, EarlyStopping
```

#回調函數(call back) 返回學習速率；在每次訓練之後保存最佳模型

匯入 CNN 模型套件

```
from keras.models import Sequential  
from keras.layers import Dense, Dropout, Activation,  
Flatten, BatchNormalization  
from keras.layers import Conv2D, MaxPooling2D  
from keras.optimizers import SGD, Adam, RMSprop #優化器套件  
from keras.utils import np_utils #可視化
```

匯入 LSTM(RNN)套件

#CNN+LSTM

```
from keras.layers import TimeDistributed  
from keras.layers import LSTM
```

```
#VGG16

from keras.applications.vgg16 import VGG16

from keras.applications.vgg16 import preprocess_input

from keras.applications.vgg16 import decode_predictions
```

使用到的套件版本








glob2	0.7
h5py	3.1.0
Keras	2.4.3
keras-nightly	2.5.0.dev2021032900
Keras-Preprocessing	1.1.2
matplotlib	3.3.2
numpy	1.19.2
numpydoc	1.1.0
opencv-python	4.5.2.54
scipy	1.4.1
tensorboard	2.5.0
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.0
tensorflow	2.5.0
tensorflow-estimator	2.5.0

基本定義

```
map_characters={0:'chris_evans',1:'chris_hemsworth',  
2:'jeremy_lee_renner',3:'mark_ruffalo',  
4:'robert_downey_jr',5:'scarlett_johansson',  
6:'thomas_stanley_tom_holland'} #人物類別名稱  
  
pic_size = 224 #設定圖片大小  
  
batch_size = 128 #batch_size 以 2 的次方做設定  
  
#這裡 batch_size=128 , 表示我們要把 128 張隨機選擇的 image 放到一個  
batch 裡面, 然後把所有的 image 分成一個個不同的 batch, Keras 會自動完  
成隨機選擇 image 的過程。  
  
epoch 不是固定的, 會依據每次訓練結果做調整。  
  
epochs = 150  
  
num_classes = len(map_characters)  
  
pictures_per_class = 300  
  
test_size = 0.3 #資料切割 7 3 分
```

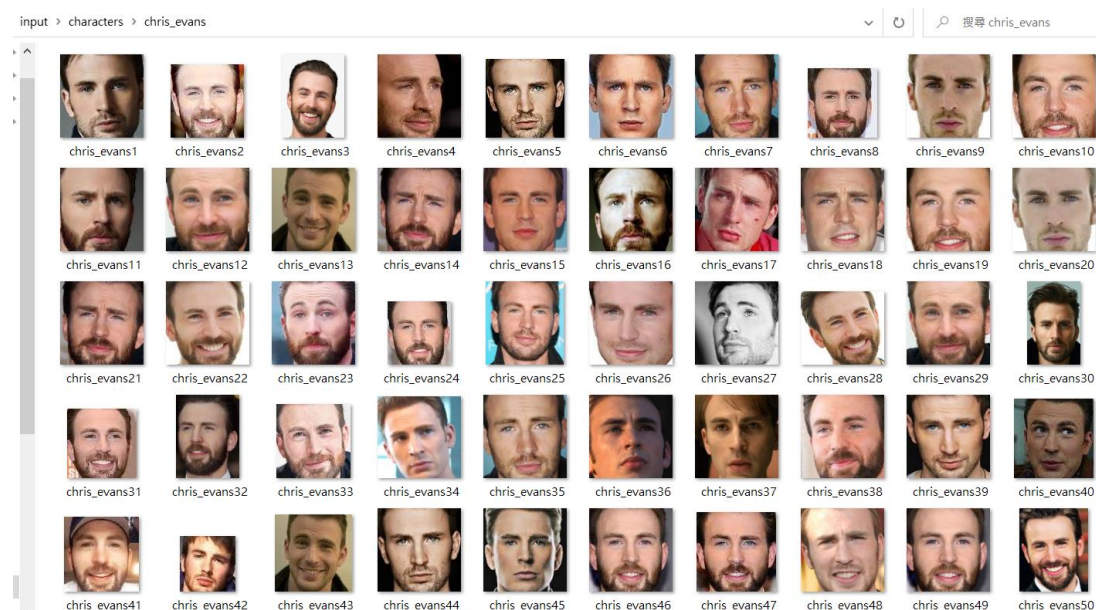
1. 資料前處理

報告總共蒐集復仇者聯盟 7 個腳色人物的圖片，圖片來源為 google，圖片格式為.jpg 檔。

照片	人物名稱	照片張數
	Chris Evans	300
	Chris Hemsworth	300
	Thomas Stanley Tom Holland	300
	Jeremy Lee Renner	300
	Mark Ruffalo	300
	Robert Downey Jr	300
	Scarlett Johansson	300

放入以名字命名的資料夾，圖片蒐集結果如下。

input > characters >				
名稱	修改日期	類型	大小	
chris_evans	2021/6/11 下午 01:12	檔案資料夾		
chris_hemsworth	2021/6/11 下午 01:12	檔案資料夾		
thomas_stanley_tom_holland	2021/6/11 下午 01:12	檔案資料夾		
jeremy_lee_renner	2021/6/11 下午 01:12	檔案資料夾		
mark_ruffalo	2021/6/11 下午 01:12	檔案資料夾		
robert_downey_jr	2021/6/11 下午 01:12	檔案資料夾		
scarlett_johansson	2021/6/11 下午 01:12	檔案資料夾		



從文件夾中選取樣本，每個人物選取的訓練集樣本佔比為 0.7，300 個樣本，測

試集佔比 0.3

```
pictures_per_class = 220
```

```
test_size = 0.3
```

如果選取的圖片數目小於該人物的總圖片數，則從中隨機選取，否則選取該人物

所有的圖片作為樣本數據集。

透過 OpenCV 來讀取圖片，因為 OpenCV 默認為 BGR，所以需要對圖片轉換

為熟悉的 RGB 圖像。

```
a = cv2.cvtColor(a, cv2.COLOR_BGR2RGB)
```

將每張圖片大小調整為 224*224

```
a = cv2.resize(a, (pic_size,pic_size))
```

將 picture 及 label 放入陣列回傳，

```
pics.append(a)
```

```
labels.append(k)
```

接著將讀取的 label 轉為 one-hot 編碼。

```
y = keras.utils.to_categorical(y, num_classes)
```

拆分資料集

```
X_train, X_test, y_train, y_test = train_test_split(X, y,  
test_size=test_size)
```

調整好參數後，將樣本儲存起來，下次訓練直接讀取。

```
if save:
```

```
    h5f = h5py.File(outputPath+'/dataset.h5', 'w')
```

```
    h5f.create_dataset('X_train', data=X_train)
```

```
    h5f.create_dataset('X_test', data=X_test)
```

```
    h5f.close( )
```



```
    h5f = h5py.File(outputPath+'/labels.h5', 'w')
```

```
    h5f.create_dataset('y_train', data=y_train)
```

```
    h5f.create_dataset('y_test', data=y_test)
```

```
    h5f.close( )
```

儲存之資料集存放在輸出路徑底下，儲存結果如下。

 dataset.h5	2021/6/15 上午 08:43	H5 檔案	307,673 KB
 labels.h5	2021/6/15 上午 08:43	H5 檔案	60 KB

將讀取的資料集轉換，並印出訓練集資料

```
X_train = X_train.astype('float32') / 255.
```

#為了把 pixel 值轉成 0-1

```
X_test = X_test.astype('float32') / 255.
```

```
print("Train", X_train.shape, y_train.shape)
```

```
print("Test", X_test.shape, y_test.shape)
```

完整程式

1. 讀取資料夾圖片及調整大小

```
def load_pictures(BGR):
    pics = []
    labels = []
    for k, char in map_characters.items():
        pictures = [k for k in glob.glob(inputPath+'characters/%s/*' % char)] #從每類人物的資料夾裡返回所有圖片名字 pictures=[****]
        #print(pictures)
        #從 pictures 中選樣本集，如果樣本數目 < pictures 數目，則返回樣本數目；如果大於，則返回 pictures 數目
        nb_pic = round(pictures_per_class/(1-test_size)) if round(pictures_per_class/(1-test_size)) < len(pictures) else len(pictures)
        # nb_pic = len(pictures)
        for pic in np.random.choice(pictures, nb_pic): #從每類 pictures 中隨機選 np_pic 張圖片作為樣本數據集
            #以 cv2.imread 讀進來的資料，會存成 NumPy 陣列
            #讀取圖片，默認彩色圖，a.shape(x,x,3)
            a = cv2.imread(pic)
            if BGR:
                a = cv2.cvtColor(a, cv2.COLOR_BGR2RGB) #色彩空間轉換 BGR 轉為 RGB
            a = cv2.resize(a, (pic_size, pic_size)) #按比例縮放為 pic_size * pic_size 大小，此時 a.shape(64,64,3)
            pics.append(a) #放入 []
            labels.append(k)
    return np.array(pics), np.array(labels)
```

2. 資料集讀寫(輸入、輸出)

```
def get_dataset(save=True, load=False, BGR=True):
    if load:
        h5f = h5py.File(outputPath+'dataset.h5', 'r')
        X_train = h5f['X_train'][:]
        X_test = h5f['X_test'][:]
        h5f.close()

        h5f = h5py.File(outputPath+'Labels.h5', 'r')
        y_train = h5f['y_train'][:]
        y_test = h5f['y_test'][:]
        h5f.close()

    else:
        X, y = load_pictures(BGR) #讀取並獲得圖片信息
        #print(X.shape, y.shape)
        y = keras.utils.to_categorical(y, num_classes) #轉換為 one-hot 編碼
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size) #拆分數據集
        if save:
            h5f = h5py.File(outputPath+'dataset.h5', 'w')
            h5f.create_dataset('X_train', data=X_train)
            h5f.create_dataset('X_test', data=X_test)
            h5f.close()

            h5f = h5py.File(outputPath+'Labels.h5', 'w')
            h5f.create_dataset('y_train', data=y_train)
            h5f.create_dataset('y_test', data=y_test)
            h5f.close()

        X_train = X_train.astype('float32') / 255. #為了把 pixel 值轉成 0-1
        X_test = X_test.astype('float32') / 255.
        print("Train", X_train.shape, y_train.shape)

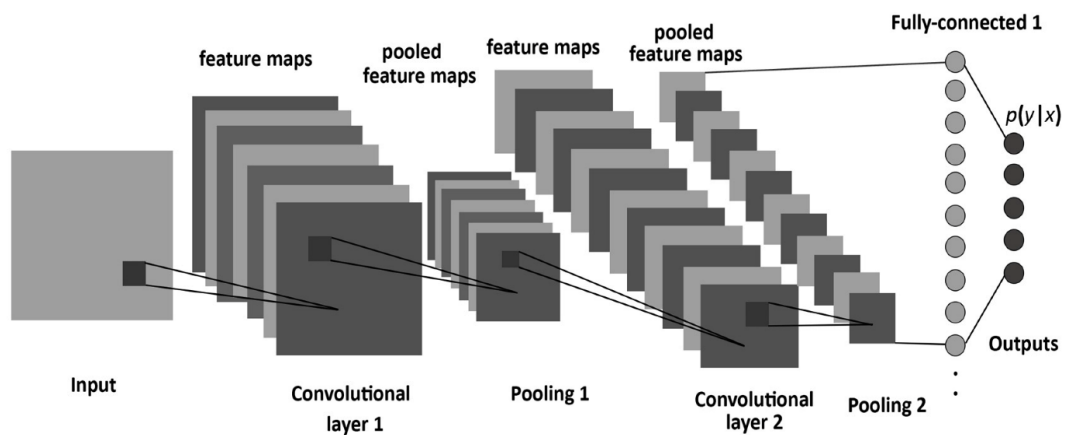
    print("Train", X_train.shape, y_train.shape)
    print("Test", X_test.shape, y_test.shape)
    # 把每類的訓練集和測試集數目印出來
    if not load:
        dist = {k: tuple(d[k] for d in [dict(Counter(np.where(y_train==1)[1])), dict(Counter(np.where(y_test==1)[1]))])
                for k in range(num_classes)}
        print('\n'.join(["%s : %d train pictures & %d test pictures" % (map_characters[k], v[0], v[1])
                        for k, v in sorted(dist.items(), key=lambda x: x[1][0], reverse=True)]))
    return X_train, X_test, y_train, y_test
```


2.建構模型

建構之模型為三種。

1.CNN(Conv2d)

模型架構圖如下: (CH.Tseng, 2017)



構建卷積神經網路，帶有 6 個 ReLU 激活函數的卷積層及 3 個池化層和 1 個全連接層(dense)。

卷積池化層(Pooling)還增加了 Dropout，可以有效防止過擬合，輸出層採用 softmax 函數來輸出各類概率，優化器 optimizer 選用隨機梯度下降 SGD，學習率 lr=0.01，學習率衰減 decay=1e-6，動量 momentum=0.9，動量是為了越過平坦區域或泥石流區域(左右震蕩)。

架構解釋:

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

- maxPooling2D 為最大池化器預設是 2*2

用一個 2x2 的矩陣來掃過輸入 (stride = 2)，然後在每個紅色區域裡都找那個區域裡最大值當作輸出(找出重要特徵)。

優點

1. 縮小 feature map 的尺寸，減少需要訓練的參數，避免 overfitting 的可能。
2. feature map 雖然縮小了，依舊可保持影像中的主要特徵。

- padding 是補零的方式

padding='same'，會用 zero-padding 的手法，讓輸入的圖不會受到 kernel map 的大小影響。每經過一層 conv 之後圖片的 size 就變小了，那如果要一樣的話，我們做的事情就是把輸入一開始補上 0，那再去跑一次 convolution 就會得到一樣大小的圖片了。

```
model.add(Conv2D(64, (3, 3), padding='same',  
                activation='relu'))
```

- Activation 選擇 relu 激活函數

原理:把輸出都通過一個函數，其實就是 $x < 0$ 的時候全部為 0，而 $x > 0$ 則為 x 。結果小於零就認定你辨識不出來，就把你關掉！

優點:計算速度快，省去複雜運算。

- Dropout:防止過度擬合(overfitting)

原理: 利用隨機關掉隱藏層節點與輸入神經元的連結，不更新權重(W)，

造成多個結果，再作比較去除極端值，即可達到避免過度擬合的現象。

```
model.add(Dropout(0.5))
```

配置為 0.5，表示每個神經元有 50% 的機率不參與下一層的傳遞。

- Flatten 扁平層:

把多維的輸入壓扁為一維輸出，常用在從卷積層到全連接 層的過渡，無

參數。因為要將矩陣拉直做運算。

- Batch Normalization 的使用時機

- 遇到收斂速度很慢，或梯度爆炸等無法訓練的狀況時可以嘗試

- 在一般使用情況下也可以加入，用來加快訓練速度，提高模型效能。

它的最大好處就是讓每一層的值在有效的範圍內傳遞下去。

參考: <https://ithelp.ithome.com.tw/articles/10204106>

- 全連接層又稱密集層 (Dense Layer)

其作用是用來進行分類。將卷積層與池化層輸出的特徵輸入到全連接層，

通過調整權重及偏差得到分類的結果

- Optimizer 優化器

要調整參數，來使 Loss 越小越好。

選擇:

- SGD: 最單純的梯度下降法。

- Momentum: 是「動量」，此優化器為模擬物理動量的概念，在同方向的維度上學習速度會變快，方向改變的時候學習速度會變慢。通常設定成 0.9。
- RMSprop: 自適應學習率的優化器，對於 RNN 效果很好。
- Learning Rate: 學習率

學習率直接影響我們的模型能夠以多快的速度收斂到局部最小值（也就是達到最好的精度）。一般來說，學習率越大，神經網絡學習速度越快。如果學習率太小，網絡很可能會陷入局部最優；但是如果太大，超過了極值，損失就會停止下降，在某一位置反覆震盪。

參考：<https://kknews.cc/code/936ylkj.html>

Model Summary

Layer (type)	Output Shape	Param #
conv2d_21 (Conv2D)	(None, 224, 224, 16)	448
conv2d_22 (Conv2D)	(None, 222, 222, 32)	4640
max_pooling2d_13 (MaxPooling)	(None, 111, 111, 32)	0
batch_normalization_5 (Batch Normalization)	(None, 111, 111, 32)	128
dropout_6 (Dropout)	(None, 111, 111, 32)	0
conv2d_23 (Conv2D)	(None, 111, 111, 64)	18496
max_pooling2d_14 (MaxPooling)	(None, 55, 55, 64)	0
batch_normalization_6 (Batch Normalization)	(None, 55, 55, 64)	256
dropout_7 (Dropout)	(None, 55, 55, 64)	0
conv2d_24 (Conv2D)	(None, 55, 55, 256)	147712
conv2d_25 (Conv2D)	(None, 53, 53, 256)	590080
max_pooling2d_15 (MaxPooling)	(None, 26, 26, 256)	0
batch_normalization_7 (Batch Normalization)	(None, 26, 26, 256)	1024
dropout_8 (Dropout)	(None, 26, 26, 256)	0
conv2d_26 (Conv2D)	(None, 26, 26, 512)	1180160
conv2d_27 (Conv2D)	(None, 24, 24, 512)	2359808
max_pooling2d_16 (MaxPooling)	(None, 12, 12, 512)	0
batch_normalization_8 (Batch Normalization)	(None, 12, 12, 512)	2048
dropout_9 (Dropout)	(None, 12, 12, 512)	0
flatten_5 (Flatten)	(None, 73728)	0
dense_7 (Dense)	(None, 1024)	75498496
activation_2 (Activation)	(None, 1024)	0
dropout_10 (Dropout)	(None, 1024)	0
dense_8 (Dense)	(None, 7)	7175
Total params: 79,810,471		
Trainable params: 79,808,743		
Non-trainable params: 1,728		

Conv2d_16 不算一層 param $3*3*3*32+32=896$

Conv2d_17 (222, 222, 32) param $3*3*32*32+32=9248$

剛開始輸入 image 大小是 224，經過 conv2D $3*3$ ，所以是 $224-3+1=222$

32 是 filter 數

Max_pooling2d_8 是 $2*2$ ，所以 $222/2=111$

Conv2d_18 (111, 111, 64) filter 數 64 $9248*2=18496$

Conv2d_19 (109, 109, 64) 因為 $111-3+1=109$

Maxpooling2d_9 (54, 54, 64) $109/2=54$

Conv2d_20 (54, 54, 256) filter=256

Conv2d_21 (52, 52, 256) $54-3+1=52$

Max_pooling2d_10 (26, 26, 256) $52/2=26$

Conv2d_22 (26, 26, 512) filter=512

Conv2d_23 (24, 24, 512) filter=512 $26-3+1=24$

Max_pooling2d_11 (12, 12, 512) $24/2=12$

$12*12$ 已經很小了，再小會看不出特徵值

Param=filter 長*filter 寬*輸入通道數*filter 個數+filter 個數

完整程式

```
def create_model_conv(input_shape):  
  
    model = Sequential()  
    model.add(Conv2D(16, (3, 3), padding='same', activation='relu', input_shape=input_shape))  
  
    #conv1  
    model.add(Conv2D(32, (3, 3), activation='relu'))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
    model.add(BatchNormalization())  
    model.add(Dropout(0.25))  
  
    #conv2  
    #Dropout->降低overfitting，配置為 0.5，表示每個神經元有 50% 的機率不參與下一層的傳遞。  
    #這種技術迫使神經網絡需要學習更為穩健的特徵，因此可有效降低 Overfitting。  
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))  
  
    # #conv3  
    # model.add(Conv2D(64, (3, 3), activation='relu'))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
    model.add(BatchNormalization())  
    model.add(Dropout(0.5))  
  
    #conv4  
    model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))  
    model.add(Conv2D(256, (3, 3), activation='relu'))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
    model.add(BatchNormalization())  
    model.add(Dropout(0.5))  
  
    #conv5  
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))  
    model.add(Conv2D(512, (3, 3), activation='relu'))  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
    model.add(BatchNormalization()) #加快收斂速度，提高效率  
    model.add(Dropout(0.5))  
  
    #扁平層，將值拉成直線，做運算  
    model.add(Flatten())  
    #Dense 表示加一個Fully connected的layer  
    model.add(Dense(1024))  
    model.add(Activation('relu'))  
    model.add(Dropout(0.5))  
  
    # model.add(Dense(256))  
    # model.add(Activation('relu'))  
  
    # model.add(Dropout(0.5))  
    ...  
  
    model.add(Dense(num_classes, activation='softmax'))  
    #opt = RMSprop(lr=0.0001, decay=1e-6)  
    opt = SGD(lr=0.02, decay=1e-6, momentum=0.9, nesterov=True)#nesterov=True使用動量  
    model.summary()  
    return model, opt
```

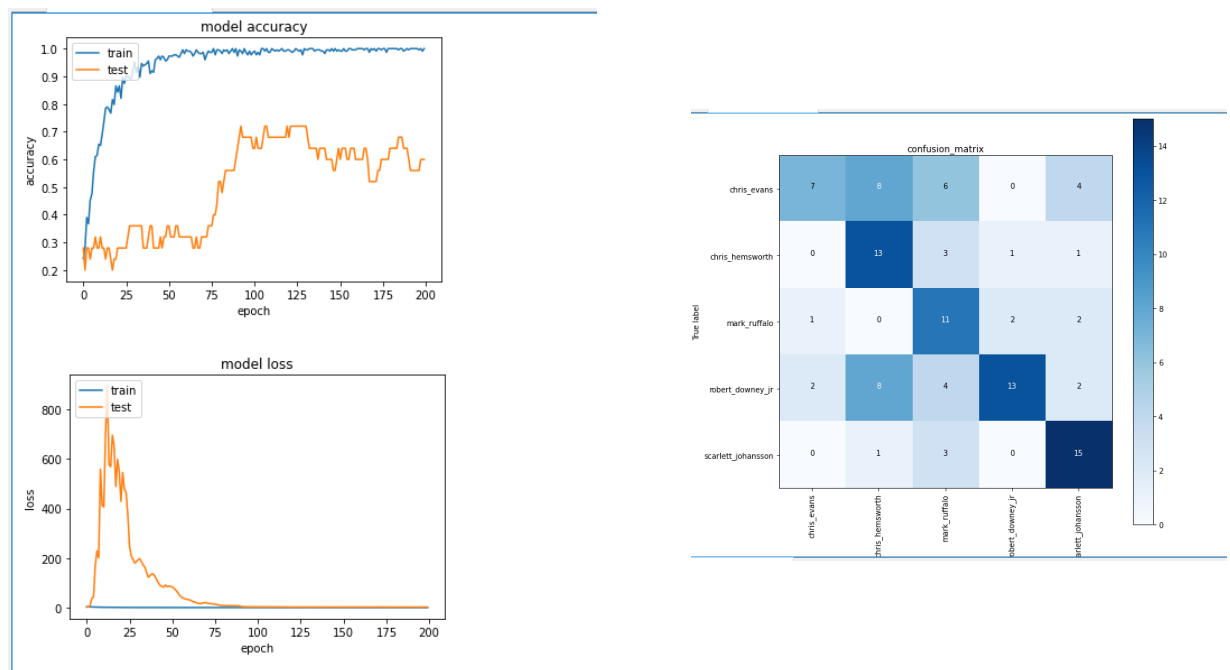
模型訓練結果

6/8 號，第一次訓練，當時人物只有 5 位，每位人物 100 張圖片。

照著原來的架構去訓練。

Epoch=200 learning rate=0.001

結果如下:

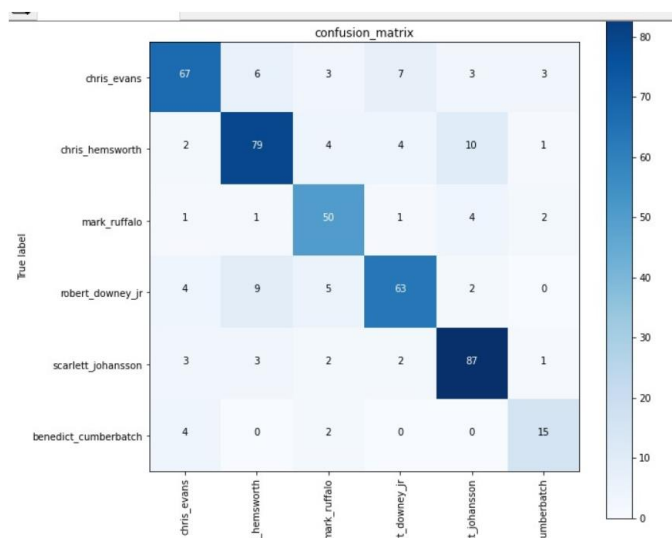


訓練結果發現資料集太少，新增一位人物角色，並將每個人物照片增加到 200

張。

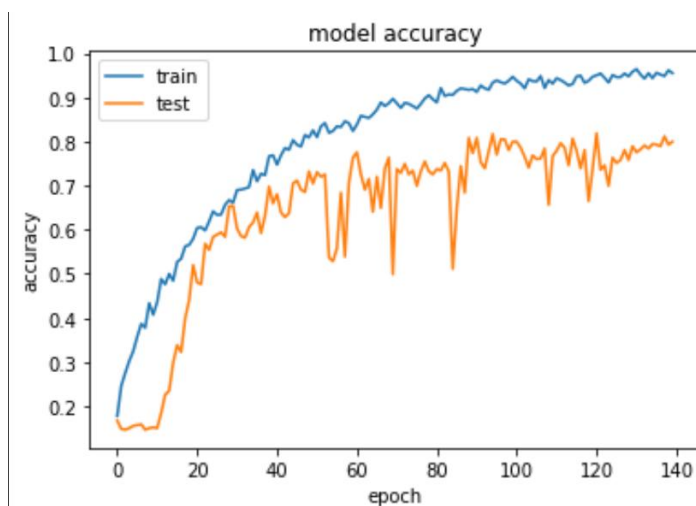
6/9 號第二次訓練，沒有調整變數。

訓練結果如下：

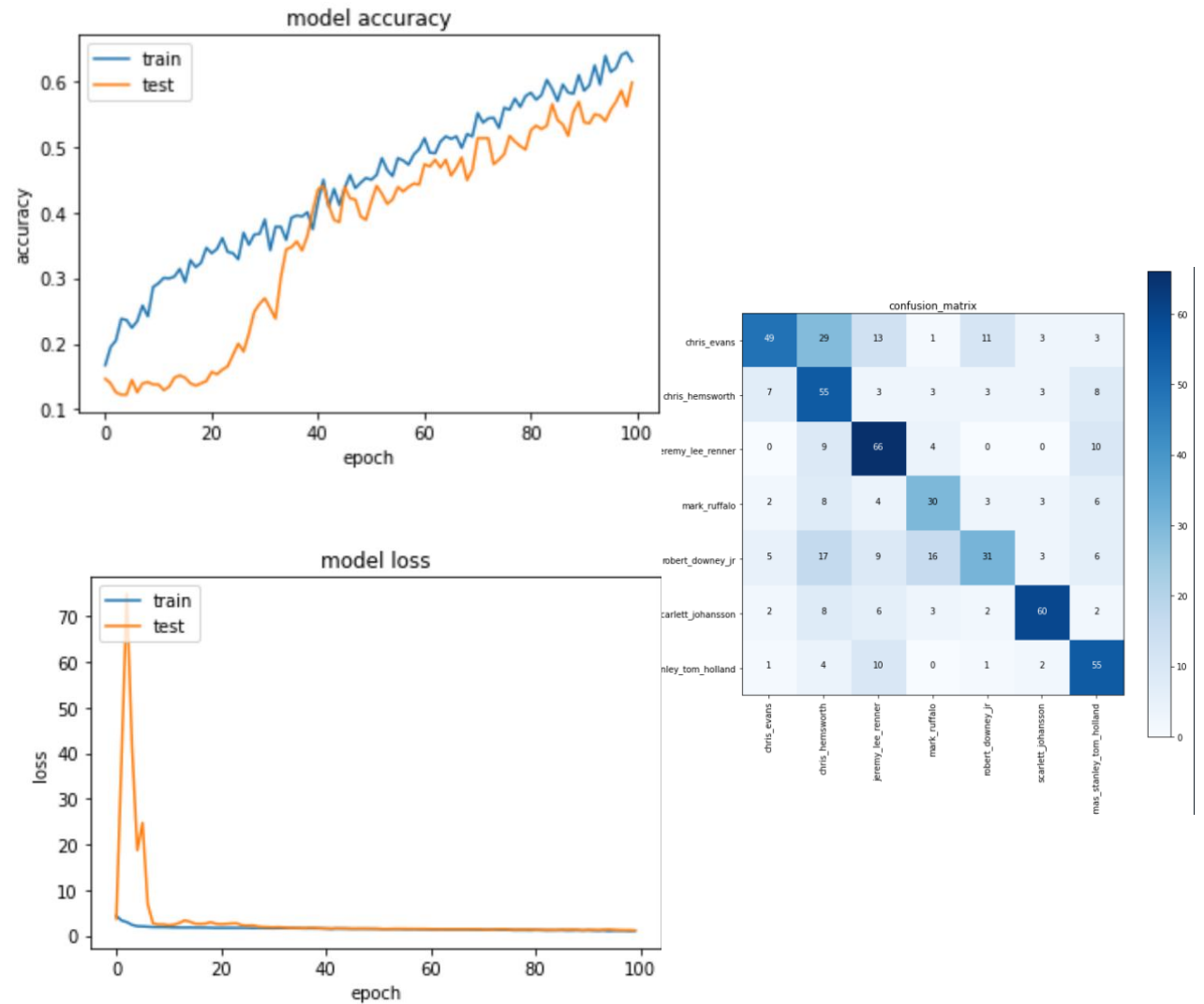


以 confusion_matrix 觀察訓練結果，明顯比上次進步很多。

6/10 號訓練，調整 epoch 為 140。



6/11 號訓練，將 batch_size 調大改成 256，epoch=100，發現 epoch 需要更大



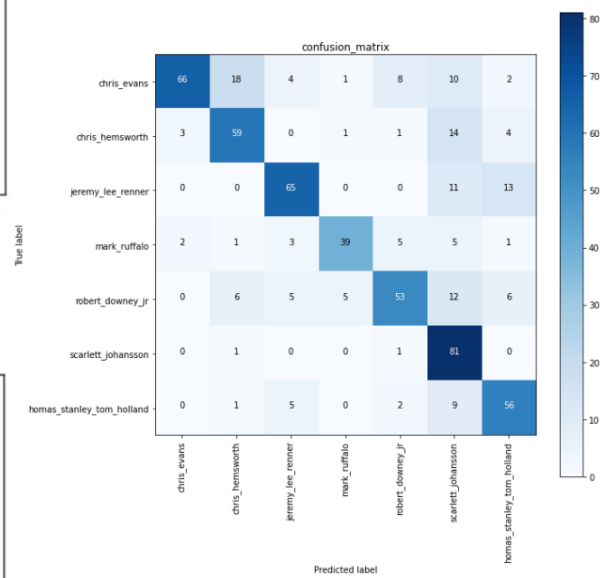
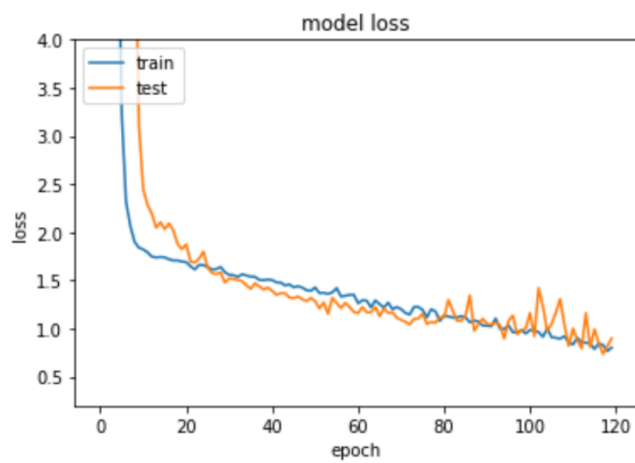
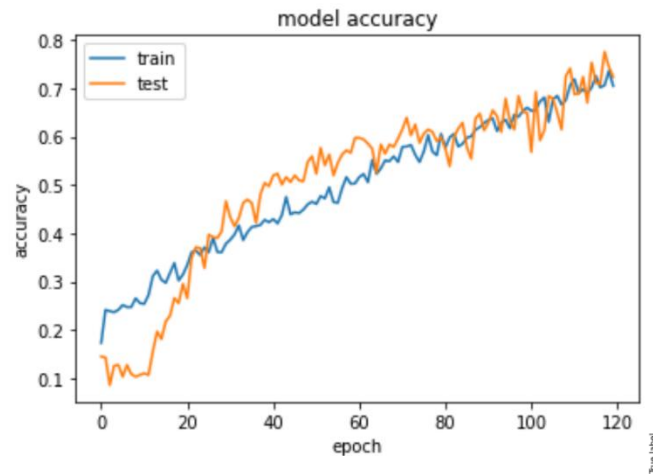
*要調整 Loss 的 y 軸，調整成比較好判讀

新增 `plt.ylim([0.1, 5])`

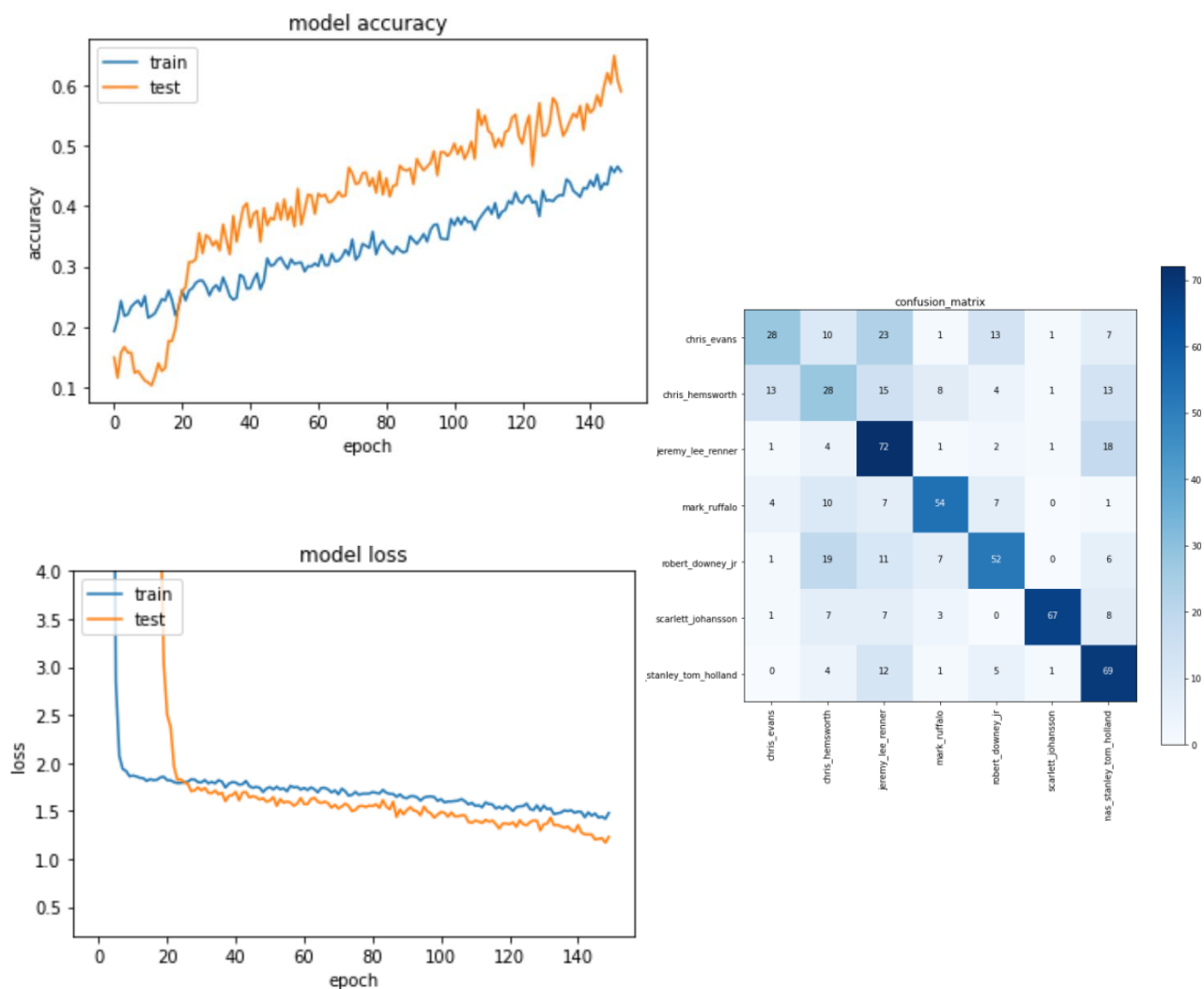
6/13 訓練，將 learning rate 調整為 0.02，發現 val_acc 學習率提高。

Batch_size 調回 128，epoch 為 120。(epoch 太少)

模型有進步。



6/14 號訓練，epoch 提高為 140，再訓練一次，覺得可以再增加 epoch。



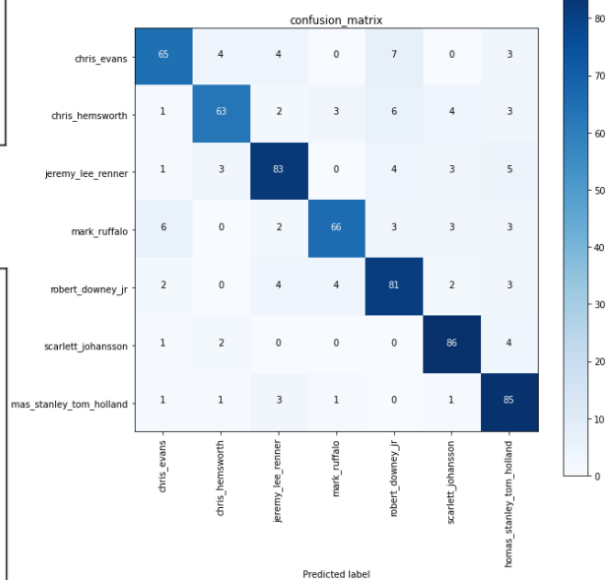
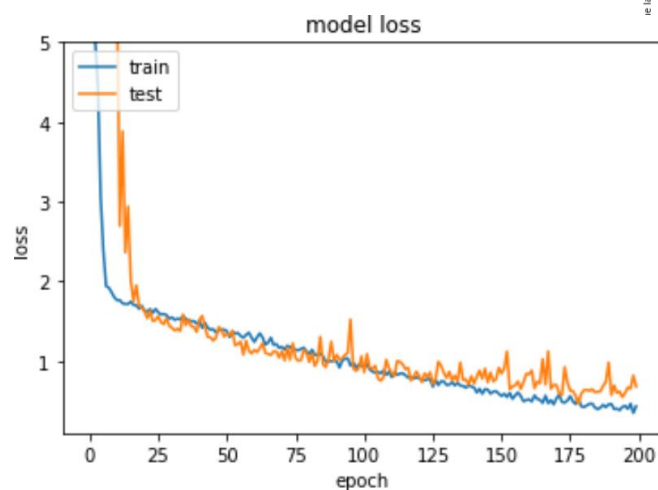
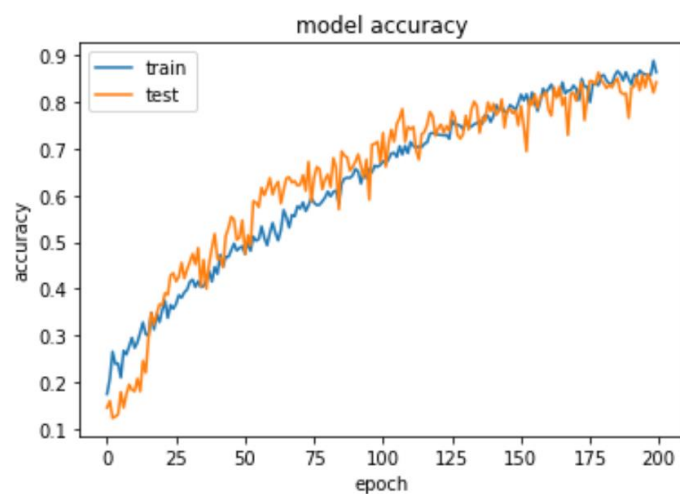
6/15 號訓練，將 epoch 設為 200，因為原本設的 Dropout 太多，

影響模型收斂，所以將 Dropout 減少，模型明顯變好。

結果:

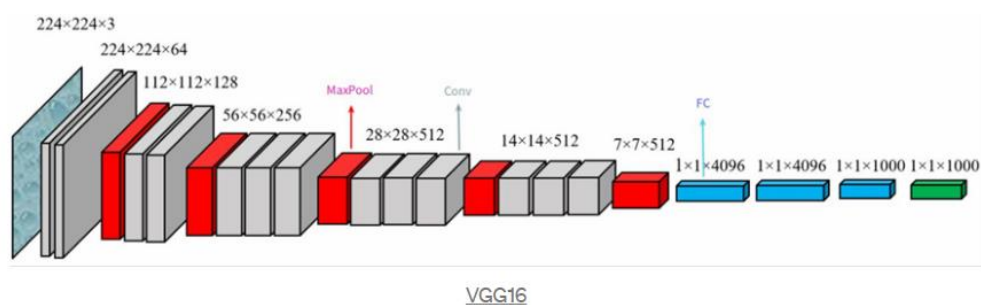
Test loss: 0.6831903457641602

Test accuracy: 0.8423566818237305



2.VGG16 (李馨伊, 2020)

VGG 特點是重複採用同一組基礎模組，並改用小卷積核替代中大型卷積核，這樣的作法可以達到相同的感受野，同時減少參數量。其架構由 n 個 VGG Block 與 3 個全連接層所組成。常見的 VGGNet 指的是 VGG16，其架構使用了五個卷積層與三個全連接層，其中前兩個卷積層內含 2 個基礎模組、後三個卷積層內含 3 個基礎模組，總共為 $2 \times 2 + 3 \times 3 + 3 = 16$ 層網路層數，架構圖如下。



程式參考

<https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c>

(Step by step VGG16 implementation in Keras for beginners)

完整程式

```
def create_model_vgg16(input_shape):
    conv_base=VGG16(weights='imagenet',input_shape=input_shape,include_top=False)
    model = Sequential()
    model.add(conv_base)
    '''
    opt = RMSprop(lr=0.0001, decay=1e-6)
    優化器用預設
    opt = SGD(lr=0.007, decay=1e-6, momentum=0.9, nesterov=True)#nesterov=True使用動量
    '''

    model.add(Flatten())
    model.add(Activation('relu'))
    model.add(Dense(num_classes, activation='softmax'))
    opt = Adam(lr=0.001)
    model.summary()
    return model, opt
```

Import 套件:

```
#VGG16
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
```

Model Summary

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1792
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
batch_normalization (Batch Normalization)	(None, 112, 112, 64)	256
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_7 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_8 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_10 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1024)	25691136
activation (Activation)	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1024)	1049600
activation_1 (Activation)	(None, 1024)	0
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 7)	7175
Total params: 41,462,855		
Trainable params: 41,462,727		
Non-trainable params: 128		

3. CNN+RNN(LSTM) (嘗試理解 LSTM 和 CNN 的結合, 無日期)

CNN+LSTM 也可以稱為 LRCN (long-term recurrent convolutional network), 因為該模型是要生成對圖像的文字描述, 所以關鍵就是我們需要一個已經 train 好的 CNN, 而該 CNN 是訓練來分類圖片現在我們可以將它用來提取描述所需要的說明。

文章中建議可以將該架構直接想成兩個子部分:

CNN: 提取特徵工具人

LSTM: 嘗試解釋按照時間序的特徵

而在 CNN+LSTM 的架構當中, 我們會希望 CNN 處理單筆資料, LSTM 處理時序性資料。所以我們要想辦法允許 CNN 一次只收一張圖片, 並且依照連續的時間段傳送給 LSTM。

如果想要達成上述的效果, 我們可以使用 TimeDistributed layer 將做好的 CNN 包裝起來, 把模型加進該 layer 就可以。因為該 layer 的用意就是將同時輸出的 data 包裝成

```
model.add(TimeDistributed(...))  
model.add(LSTM(...))  
model.add(Dense(...))
```

程式架構參考:

<https://hackmd.io/@subject/BJWLeCSNd>(嘗試理解 LSTM 和 CNN 的結合)

完整程式

```
def Conv_LSTM_model(input_shape):  
  
    model = Sequential()  
    print(input_shape)  
    model.add(TimeDistributed(Conv2D(1, (2,2), activation='relu',input_shape=input_shape)))  
  
    model.add(TimeDistributed(MaxPooling2D(pool_size=(2, 2))))  
  
    model.add(TimeDistributed(Conv2D(1, (2,2),activation='relu')))  
    model.add(TimeDistributed(MaxPooling2D(pool_size=(2, 2))))  
    model.add(TimeDistributed(BatchNormalization()))  
    model.add(TimeDistributed(Dropout(0.25)))  
    model.add(TimeDistributed(Flatten()))  
  
    model.add(LSTM(50))  
    model.add(Dense(num_classes,activation='sigmoid'))  
    model.add(Dropout(0.2))  
  
    opt = keras.optimizers.Adam(learning_rate=0.01)  
    model.summary()  
    return model,opt
```

主程式:

```
if __name__ == '__main__':  
    X_train, X_test, y_train, y_test = get_dataset(load=True)  
    # model, opt = create_model_conv(X_train.shape[1:])  
  
    # model, opt = create_model_vgg16(X_train.shape[1:])  
    # model,opt = load_model_from_checkpoint('../output/weights_6conv_20210615.hdf5', six_conv=True, input_shape=(pic_size,pic_size,3))  
  
    # model.compile(loss='categorical_crossentropy',  
    #               optimizer=opt,  
    #               metrics=['accuracy'])  
  
    model,opt=Conv_LSTM_model((1,224,224,3))  
  
    model.compile(loss='mean_squared_error',  
                  optimizer=opt,  
                  metrics=['accuracy'])  
    X_train=X_train.reshape(1465,1,224,224,3)  
    X_test=X_test.reshape(628,1,224,224,3)  
    model, history = training(model, X_train, X_test, y_train, y_test, data_augmentation=False)
```

參考資料:

<https://stackoverflow.com/questions/55433649/how-to-combine-lstm-and-cnn-models-in-keras>

訓練結果

```
Epoch 213/220
3/3 [=====] - 6s 2s/step - loss: 0.1262 - accuracy: 0.1711 - val_loss: 0.1243 - val_accuracy 0.1667
Epoch 214/220
3/3 [=====] - 6s 1s/step - loss: 0.1262 - accuracy: 0.1863 - val_loss: 0.1243 - val_accuracy 0.1667
Epoch 215/220
3/3 [=====] - 6s 2s/step - loss: 0.1276 - accuracy: 0.1901 - val_loss: 0.1245 - val_accuracy 0.1667
Epoch 216/220
3/3 [=====] - 6s 2s/step - loss: 0.1258 - accuracy: 0.1597 - val_loss: 0.1241 - val_accuracy 0.1667
Epoch 217/220
3/3 [=====] - 6s 2s/step - loss: 0.1280 - accuracy: 0.1711 - val_loss: 0.1236 - val_accuracy 0.1667
Epoch 218/220
3/3 [=====] - 6s 2s/step - loss: 0.1277 - accuracy: 0.1673 - val_loss: 0.1234 - val_accuracy 0.2000
```

*因為時間不夠的關係，我就沒有另外去調參數做訓練，一開始因為 LSTM 讀取的資料要加上 timesteps，原本的維度必須轉成 5 維。

將資料做轉換：

```
model,opt=Conv_LSTM_model((1,224,224,3))

model.compile(loss='mean_squared_error',
              optimizer=opt,
              metrics=['accuracy'])
X_train=X_train.reshape(1465,1,224,224,3)
X_test=X_test.reshape(628,1,224,224,3)
model, history = training(model, X_train, X_test, y_train, y_test, data_augmentation=False)
```

遇到的問題:資料集太少，讓訓練結果的 acc 無法提高。

之後會繼續嘗試。

3. 模型訓練及優化

回調函數(callbaks)

- ✧ ModelCheckpoint，實現的功能是將 val_acc 最高的模型保存下來，然後測試時直接載入使用。

程式碼:

```
##每當val_cc有提升就保存checkpoint
#save_best_only=True被監測數據的最佳模型就不會被覆蓋，mode='max'保存的是準確率最大值
filepath=outputPath+"/weights_6conv_%.hdf5" % time.strftime("%Y%m%d")
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

callbacks_list = [checkpoint]
history = model.fit_generator(datagen.flow(X_train, y_train, #傳入 Numpy 數據和標籤數組，生成批次的 增益的/標準化的數據。在生成的批次數據上無限地無限次循環。
                                batch_size=batch_size,
                                steps_per_epoch=X_train.shape[0] // batch_size,
                                epochs=epochs,
                                validation_data=(X_test, y_test),
                                verbose=1, #輸出進度條
                                callbacks=callbacks_list), #調用一些列回調函數
                             #callback會有一些回傳值，根據回傳值調整每次更新的頻率)
```

- ✧ Learning Rate schedule，自動調整學習率

```
#設置學習率衰減
def lr_schedule(epoch):
    initial_lr = 0.08 #初始學習率
    drop = 0.5 #衰減為原來的多少倍
    epochs_drop = 12.0 #每隔多久改變學習率
    lr = initial_lr * math.pow(drop, math.floor((1+epoch)/epochs_drop)) #math.pow(x,y)=x的y次方，math.floor向下取整
    #return lr if lr >= 0.0001 else 0.0001
    return lr
```

參考網址:

<https://androidkt.com/change-the-learning-rate-using-schedules-api-in-keras/>

- ✧ Early Stopping，針對迭代次數過多、訓練時間過長的問題，我們可以透過設置 Early Stopping 來解決，Early Stopping 其實就是透過觀察測試資料集 Loss 的變化來停止訓練，並透過調整 patience 決定容忍度。

程式碼:

```
#EarlyStopping  
#當val_loss不再下降，提前停止訓練  
#超過20次就停止訓練  
early_stopping = EarlyStopping(monitor='val_loss', patience=20, verbose=1, mode='min')  
#early_stopping = EarlyStopping(monitor='val_accuracy', patience=20, verbose=1, mode='max')
```

參考網址:

<https://medium.com/@CinnamonAITaiwan/cnn%E5%85%A5%E9%96%80-overfitting-d10acd15ec21>

✧ ImageDataGenerator:資料增強

影像資料增強是人工增加訓練集的一種方式，主要是通過修改資料集中的圖片達成。更多的訓練資料帶來的是更有效的深度學習模型，同時，資料增強技術會產生更多的圖片變體，這些變體會提高模型對新圖片的泛化能力。

可調整之參數舉例:

rotation_range: (0-180 度) 影像旋轉角度

width_shift_range:0-1 水平平移，相對總寬度的比例

height_shift_range:0-1 垂直平移，相對總高度的比例

參考網址:

<https://iter01.com/183646.html>

程式碼：

```
if data_augmentation:
    #資料擴增法:
    #將現有資料圖片，藉由水平旋轉、上下翻轉之類的，把輸入很少的照片，增加成很多照片。
    #data augmantation 對資料量少很適合
    datagen = ImageDataGenerator(
        featurewise_center=False, # 將輸入數據的均值設置為 0，依特徵進行
        samplewise_center=False, # 將每個樣本的均值設置為 0
        featurewise_std_normalization=False, # 將輸入除以數據標準差，依特徵進行
        samplewise_std_normalization=False, # 將每個輸入除以其標準差
        #zca_whitening=False, #應用 ZCA 白化
        rotation_range=12, # 隨機旋轉的度數範圍(degrees, 0 to 180)，旋轉角度
        width_shift_range=0.13, # 隨機水平移動的範圍，比例
        height_shift_range=0.13, # 隨機垂直移動的範圍，比例
        zoom_range=0.2,
        horizontal_flip=True, # 隨機水平翻轉，相當於鏡像
        vertical_flip=False) # 隨機垂直翻轉，相當於鏡像

    datagen.fit(X_train)
```

4. 結果呈現

分別畫出訓練集和測試集上的 accuracy 和 loss 變化曲線，注意，模型訓練完後返回的是一個 History 對象。其 History.history 屬性是連續 epoch 訓練損失和評估值，以及驗證集損失和評估值的記錄。

程式碼:

```
history = model.fit_generator(datagen.flow(X_train, y_train, #導入 Numpy 數據和標籤數組，生成批次的，增益的/標準化的數據。在生成的批次數據上無限制地無限次循環，
                                batch_size=batch_size),
                             steps_per_epoch=X_train.shape[0] // batch_size,
                             epochs=epochs,
                             validation_data=(X_test, y_test),
                             verbose=1, #輸出進度條
                             callbacks=checkpoint) #調用一些列回調函數
```

✧ 畫出 loss & val

```
#acc和loss可視化
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# plt.savefig('acc_%.png' % time.strftime("%Y%m%d"))

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim([0.1, 5])
plt.show()
# plt.savefig('loss_%.png' % time.strftime("%Y%m%d"))
```

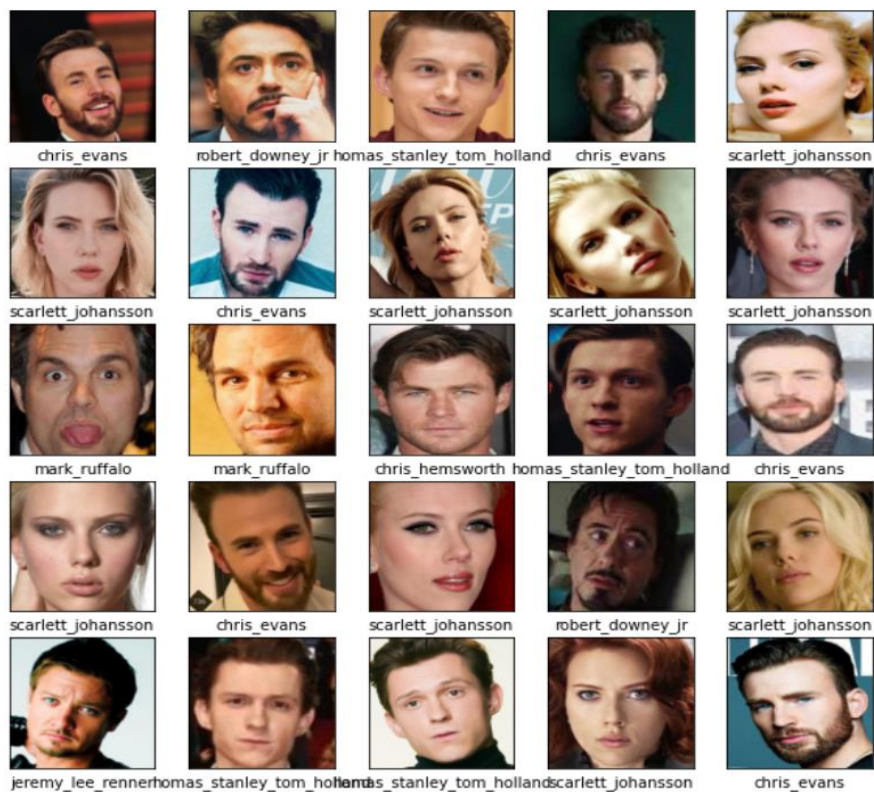
✧ 畫出 confusion matrix

```
#畫出混淆矩陣
plt.figure(figsize = (10,10))
cnf_matrix = sklearn.metrics.confusion_matrix(np.where(y_test > 0)[1], np.argmax(y_pred, axis=1))
classes = list(map_characters.values())
thresh = cnf_matrix.max() / 2. # 閾值
for i, j in itertools.product(range(cnf_matrix.shape[0]), range(cnf_matrix.shape[1])):
    plt.text(j, i, cnf_matrix[i, j], #在圖形中加註釋
             horizontalalignment="center", #水平對齊
             color="white" if cnf_matrix[i, j] > thresh else "black")
plt.imshow(cnf_matrix, interpolation='nearest', cmap=plt.cm.Blues) #cmap顏色圖譜，默認RGB(A)
plt.colorbar() #顯示顏色條
plt.title('confusion_matrix') #標題
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=90)
plt.yticks(tick_marks, classes)
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.savefig('confusion_matrix_%.png' % time.strftime("%Y%m%d"))
```

✧ 印出 25 張測試集結果

```
#y_test predict
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_test[i])
    a=np.where(y_test[i]==1)
    b=a[0][0]
    plt.xlabel(map_characters[b])
plt.show()
```

結果如下:



5. 調整超參數

上網查詢了幾種調整方法，以下是我照著這幾個原則調整模型：

1. 解決過擬合(Overfitting)

➤ 迭代次數過多

針對迭代次數過多、訓練時間過長的問題，我們可以透過設置 Early Stopping 來解決。

➤ Dropout

Dropout 為 Deep Learning 中常用的技巧，隨機關閉 NN 層中的一定比例神經元(讓其值為 0)，藉此降低模型對各個神經元的依賴性。但過高比例的 Dropout 會影響模型的收斂，尤其是在 Convolution 層，所以大家使用上也要特別注意。

➤ Batch Normalization

起到一定 Regularize(正則)的作用。

參考：

<https://medium.com/@CinnamonAITaiwan/cnn%E5%85%A5%E9%96%80-overfitting-d10acd15ec21>

2. 觀察 loss 及 val 圖

有 5 種情形

- train loss 不斷下降，test loss 不斷下降，說明 network 仍在學習;
(最好的)
- train loss 不斷下降，test loss 趨於不變，說明 network 過擬合;
(max pool 或者正則化)
- train loss 趨於不變，test loss 不斷下降，說明數據集 100%有問題;
(檢查 dataset)
- train loss 趨於不變，test loss 趨於不變，說明學習遇到瓶頸，需要減小學習率或批量數目 (減少學習率)
- train loss 不斷上升，test loss 不斷上升，說明 network 結構設計不當，訓練超参数設置不當，數據集經過清洗等問題 (最不好的情况)

參考網址:

<https://blog.csdn.net/qingfengxd1/article/details/107320580>

預測程式

使用的套件

```
from os import listdir
from os.path import isfile, join
import numpy as np
import cv2
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import glob
import keras
```

基本定義

```
modelPath = os.path.join("../", "model")
mypath= os.path.join("../", "input/test_images")
inputPath=os.path.join("../input")
outPath = os.path.join("../", "output")

class_names = {0: 'chris_evans', 1: 'chris_hemsworth', 2: 'homas_stanley_tom_holland',
                3: 'jeremy_lee_renner', 4: 'mark_ruffalo', 5: 'robert_downey_jr', 6: 'scarlett_johansson'}

test_images=[]
test_labels=[]
pic_size=224
```

讀取圖片並資料前處理

```
1
2 onlyfiles = [ f for f in listdir(mypath) if isfile(join(mypath,f)) ]
3 images = np.empty(len(onlyfiles), dtype=object)
4
5 for n in range(0, len(onlyfiles)):
6     images[n] = cv2.imread(join(mypath, onlyfiles[n]), cv2.IMREAD_COLOR)
7     images[n] = cv2.cvtColor(images[n], cv2.COLOR_BGR2RGB)
8     images[n] = cv2.resize(images[n], (224, 224))
9     test_images.append(images[n])
10 test_images = np.array(test_images)
11
12 plt.imshow(test_images[0])
13
14 test_images = test_images.reshape((1, 224, 224, 3))
15 test_images = test_images.astype('float32') / 255. #將色彩轉為0-1
16
17 test_labels = ['0']
18 test_labels = keras.utils.to_categorical(test_labels, 7)
19 test_labels=np.array(test_labels)
20
21
22 print(test_images.shape)
23 print(test_labels.shape)
```

利用已存好的模型去作預測

```
modelName = outPath+"/weights_6conv_20210617.hdf5"  
trainedModel = load_model(modelName)  
print("model summary", trainedModel.summary())
```

預測程式

```
testLoss, testAcc = trainedModel.evaluate(test_images, test_labels, verbose=1)  
print("testLoss", testLoss)  
print("testAcc", testAcc)  
  
##### Make prediction  
pc = trainedModel.predict_classes(test_images) #index of the label with max  
ps = trainedModel.predict(test_images) # the content of nodes in output layers  
print("Class of prediction: ", pc[0:7])  
print("Result of prediction: ", ps)  
print("Label of testing: ", test_labels)
```

印出圖片及結果

```
##### Plot output  
saveFileName = outPath + "/image_0.jpg"  
plt.figure(figsize=(27,10))  
plt.title("Avengers Image Test")  
  
ax = plt.subplot(1,1,1)  
ax.imshow(test_images[0])  
ax.set_title("label={}\n predi={}\n characters={}".format(str(test_labels[0]), str(pc[0]),  
                                                         str(class_names[pc[0]])), fontsize=18)  
ax.set_xticks([])  
ax.set_yticks([])  
  
plt.savefig("Result.jpg")  
plt.show()
```

顯示結果如下：（但因為模型 acc 不夠好，所以預測結果不準確）



GUI 介面

介面程式使用 python 內建的 tkinter。

Import 套件如下:

```
1
2
3 from os import listdir
4 from os.path import isfile, join
5 import numpy as np
6 import cv2
7 import os
8 from tensorflow.keras.models import load_model
9 from tensorflow.keras.utils import to_categorical
10 import matplotlib.pyplot as plt
11 import keras
12
13
14 from tkinter import *
15 from keras.preprocessing import image
16 # Loading Python Imaging Library
17 from PIL import ImageTk, Image
18 # To get the dialog box to open when required
19 from tkinter import filedialog
20
```

基本定義

```
modelPath = os.path.join("../", "model")
mypath= os.path.join("../", "input/test_images")
inputPath=os.path.join("../input")
outPath = os.path.join("../", "output")

class_names = {0:'chris_evans',1:'chris_hemsworth',2:'homas_stanley_tom_holland',
               3:'jeremy_lee_renner',4:'mark_ruffalo',5:'robert_downey_jr',
               6:'scarlett_johansson'}

modelName = outPath+"/weights_6conv_20210617.hdf5"
model = load_model(modelName)
print('Model Loaded Sucessfully')
```

這裡的 model 可以自由更換，可從 output 資料夾裡挑選，放入 modelName

使用者選取圖片後，會將圖片存成 test.jpg，再將圖片寫入

```
def open_img():
    # Select the Imagename from a folder
    x = openfilename()
    # opens the image
    img = Image.open(x)
    im1 = img.save("test.jpg")
    img = ImageTk.PhotoImage(img)
    # create a Label
    panel = Label(root, image = img)
    # set the image as img
    panel.image = img
    panel.place(bordermode=OUTSIDE, x=80, y=100)

def openfilename():
    # open file dialog box to select image
    # The dialogue box has a title "Open"
    filename = filedialog.askopenfilename(title = 'Select Image')
    return filename
```

預測程式碼與 test_image.py 相同。

執行結果如下：



程式架構參考：

<https://www.rs-online.com/designspark/python-tkinter-cn>

<https://yanwei-liu.medium.com/industrial-desktop-deep-learning-application-with-keras-and-tkinter-b2dcc273756a>

參考資料來源

- [資料分析&機器學習] 第 5.1 講：卷積神經網絡介紹(Convolutional Neural Network)
<https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC5-1%E8%AC%9B-%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E7%B5%A1%E4%BB%8B%E7%B4%B9-convolutional-neural-network-4f8249d65d4f>
- 卷積神經網絡 CNN 經典模型 — LeNet、AlexNet、VGG、NiN with Pytorch code
<https://medium.com/ching-i/%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E7%B5%A1-cnn-%E7%B6%93%E5%85%B8%E6%A8%A1%E5%9E%8B-lenet-alexnet-vgg-nin-with-pytorch-code-84462d6cf60c>
- Conv2D 架構解釋參考
<https://ithelp.ithome.com.tw/articles/10187424>
- LSTM+CNN
<https://ithelp.ithome.com.tw/articles/10223055>
- CNN 程式
<https://chtseng.wordpress.com/2017/09/23/%E5%AD%B8%E7%BF%92%E4%BD%BF%E7%94%A8keras%E5%BB%BA%E7%AB%8B%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E8%B7%AF/>

主要程式來源

- 辛普森 CNN 人臉辨識
<https://github.com/lpdsdx/Simpson-Recognition/blob/master/train.py>
- 辛普森 CNN 解釋
<https://www.getit01.com/p2018070437833815/>

影片學習

- ML Lecture 9-1: Tips for Training DNN
https://www.youtube.com/watch?v=xki6lj7z-30&list=PLJV_e13uVTsPy9oCRY30oBPNLCo89yu49&index=16&ab_channel=Hung-yiLeeHung-yiLee
- ML Lecture 10: Convolutional Neural Network
<https://www.youtube.com/watch?v=FrKWiRv254g>

心得

這學期的機器學習課對我幫助很大，了解很多模型的架構，也有很多的實作。但因為之前沒有修過相關課程，對很多專有名詞和變數不熟悉，在做期末報告的時候就要花很多時間理解 CNN 架構背後的理論；在網路上有很多資源，也有很多詳細的課程，對我幫助很大。期末報告讓我完整實作機器學習訓練的過程，花了很多時間理解，在每一次訓練調整參數中學經驗，我收穫很多，之後也想要在這個領域繼續努力。謝謝老師的指導。