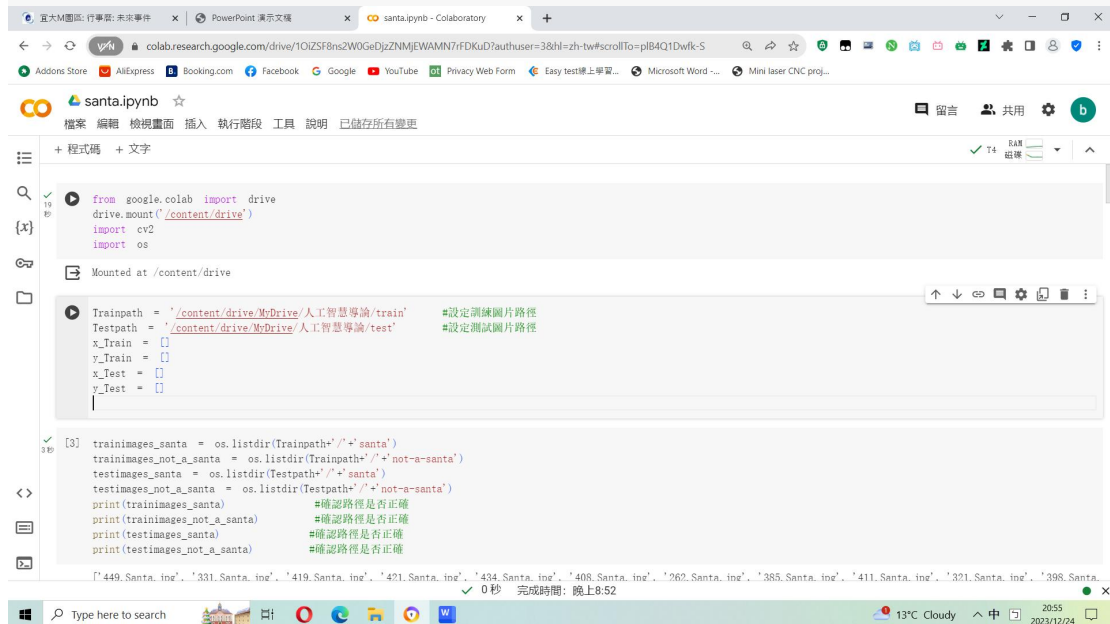


# B1043043 周定毅 資工三

首先從 colab 載入資料,然後設定訓練及測試圖片路徑,隨後確認路徑是否正確.



```
from google.colab import drive
drive.mount('/content/drive')
import cv2
import os

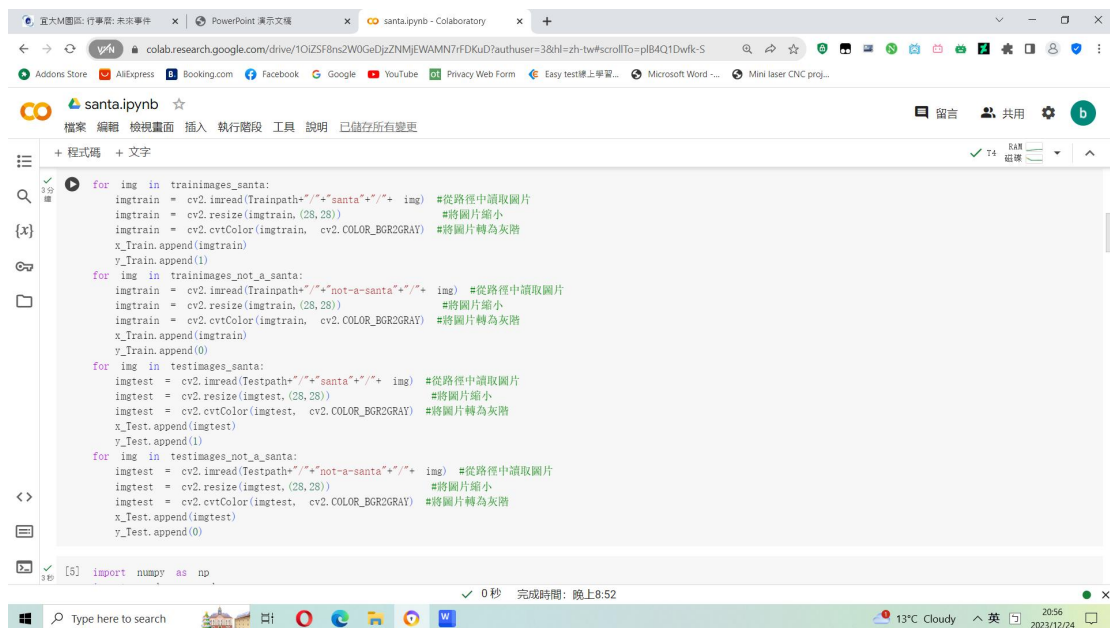
Mounted at /content/drive

Trainpath = '/content/drive/MyDrive/人工智能導論/train' #設定訓練圖片路徑
Testpath = '/content/drive/MyDrive/人工智能導論/test' #設定測試圖片路徑
x_train = []
y_train = []
x_test = []
y_test = []

[3] trainimages_santa = os.listdir(Trainpath+"/"+santa')
trainimages_not_a_santa = os.listdir(Trainpath+"/"+not-a-santa')
testimages_santa = os.listdir(Testpath+"/"+santa')
testimages_not_a_santa = os.listdir(Testpath+"/"+not-a-santa')
print(trainimages_santa) #確認路徑是否正確
print(trainimages_not_a_santa) #確認路徑是否正確
print(testimages_santa) #確認路徑是否正確
print(testimages_not_a_santa) #確認路徑是否正確

['449.Santa.jpg', '331.Santa.jpg', '419.Santa.jpg', '421.Santa.jpg', '434.Santa.jpg', '408.Santa.jpg', '262.Santa.jpg', '385.Santa.jpg', '411.Santa.jpg', '321.Santa.jpg', '398.Santa.jpg']
0 秒 完成時間: 晚上8:52
```

過後便開始讀取圖片,並將圖片縮小及轉為灰階,並將 label 為 santa 的圖片設定為 1,而 label 為 not-a-santa 的圖片設定為 0.



```
for img in trainimages_santa:
    imgtrain = cv2.imread(Trainpath+"/"+santa+"/"+img) #從路徑中讀取圖片
    imgtrain = cv2.resize(imgtrain, (28, 28)) #將圖片縮小
    imgtrain = cv2.cvtColor(imgtrain, cv2.COLOR_BGR2GRAY) #將圖片轉為灰階
    x_train.append(imgtrain)
    y_train.append(1)

for img in trainimages_not_a_santa:
    imgtrain = cv2.imread(Trainpath+"/"+not-a-santa+"/"+img) #從路徑中讀取圖片
    imgtrain = cv2.resize(imgtrain, (28, 28)) #將圖片縮小
    imgtrain = cv2.cvtColor(imgtrain, cv2.COLOR_BGR2GRAY) #將圖片轉為灰階
    x_train.append(imgtrain)
    y_train.append(0)

for img in testimages_santa:
    imgtest = cv2.imread(Testpath+"/"+santa+"/"+img) #從路徑中讀取圖片
    imgtest = cv2.resize(imgtest, (28, 28)) #將圖片縮小
    imgtest = cv2.cvtColor(imgtest, cv2.COLOR_BGR2GRAY) #將圖片轉為灰階
    x_test.append(imgtest)
    y_test.append(1)

for img in testimages_not_a_santa:
    imgtest = cv2.imread(Testpath+"/"+not-a-santa+"/"+img) #從路徑中讀取圖片
    imgtest = cv2.resize(imgtest, (28, 28)) #將圖片縮小
    imgtest = cv2.cvtColor(imgtest, cv2.COLOR_BGR2GRAY) #將圖片轉為灰階
    x_test.append(imgtest)
    y_test.append(0)

[5] import numpy as np
0 秒 完成時間: 晚上8:52
```

過後把已經分類完成的陣列轉成 np 陣列,並將影像的特徵值轉換為 4 維矩陣以符合 CNN 要求格式,同時將數字影像特徵值標準化,並把 label 轉換成 Onehot encoding

```
[5] import numpy as np
import pandas as pd
from keras.utils import to_categorical
x_Test=np.array(x_Test) #轉換成np陣列
x_Train=np.array(x_Train) #轉換成np陣列

[6] x_Train4D=x_Train.reshape(x_Train.shape[0],28,28,1).astype('float32') #將影像的特徵值轉換為4維矩陣以符合CNN要求格式
x_Test4D=x_Test.reshape(x_Test.shape[0],28,28,1).astype('float32') #將影像的特徵值轉換為4維矩陣以符合CNN要求格式

[7] x_Train4D_normalize = x_Train4D / 255 #數字影像特徵值標準化
x_Test4D_normalize = x_Test4D / 255 #數字影像特徵值標準化

[8] y_TrainOneHot = to_categorical(y_Train) #label轉換成Onehot encoding
y_TestOneHot = to_categorical(y_Test) #label轉換成Onehot encoding
```

定義一個模型,並設定其參數.

```
[9] from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D

[10] model = Sequential() #定義模型，先實例化一個優化器對象

[11] model.add(Conv2D(filters=16,kernel_size=(5,5),padding='same',input_shape=(28,28,1),activation='relu')) #新增一個卷積層並設定參數

[12] model.add(MaxPooling2D(pool_size=(2,2))) #新增一個池化層，設定池化窗口為2X2

[13] model.add(Conv2D(filters=36,kernel_size=(5,5),padding='same',activation='relu')) #再新增一個卷積層跟池化層
model.add(MaxPooling2D(pool_size=(2,2)))

[14] model.add(Dropout(0.25)) #關閉隱藏層

[15] model.add(Flatten()) #建立平坦層，進行攤平
model.add(Dense(128, activation='relu')) #建立隱藏層
model.add(Dropout(0.5)) #Dropout
model.add(Dense(2, activation='softmax')) #建立輸出層，維度為2(是否為santa)

[16] model.summary() #總結模型

Model: "sequential"
```

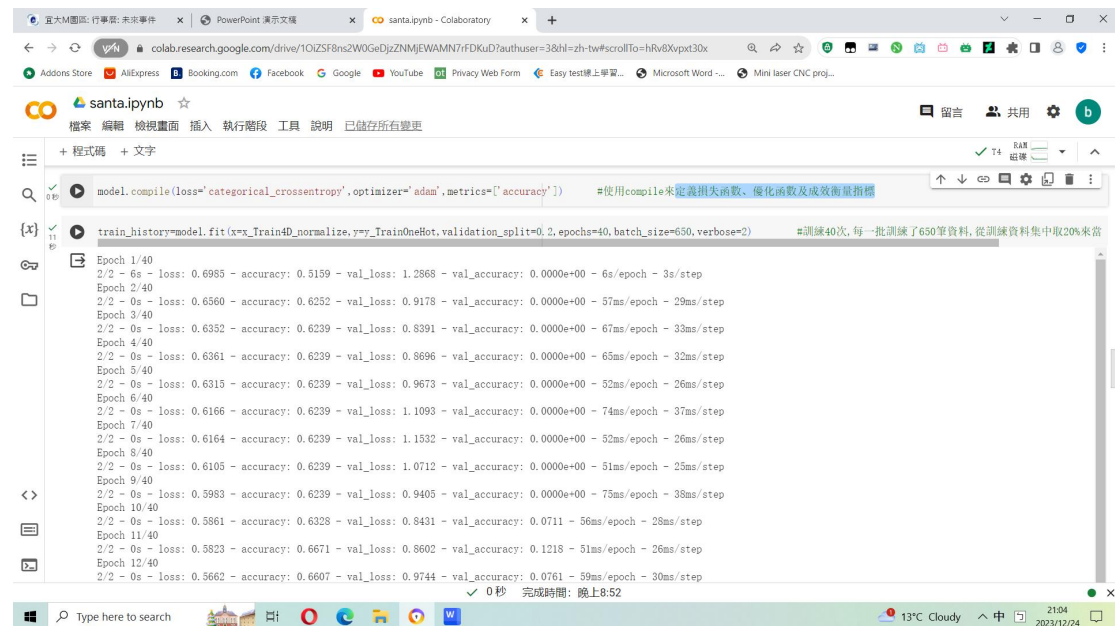
模型總結如下:

```
model.summary() #總結模型

Model: "sequential"
Layer (type) Output Shape Param #
-----
conv2d (Conv2D) (None, 28, 28, 16) 416
max_pooling2d (MaxPooling2D) (None, 14, 14, 16) 0
conv2d_1 (Conv2D) (None, 14, 14, 36) 14436
max_pooling2d_1 (MaxPooling2D) (None, 7, 7, 36) 0
dropout (Dropout) (None, 7, 7, 36) 0
flatten (Flatten) (None, 1764) 0
dense (Dense) (None, 128) 225920
dropout_1 (Dropout) (None, 128) 0
dense_1 (Dense) (None, 2) 258
-----
Total params: 241030 (941.52 KB)
Trainable params: 241030 (941.52 KB)
Non-trainable params: 0 (0.00 KB)
```

定義損失函數、優化函數及成效衡量指標,並開始訓練.

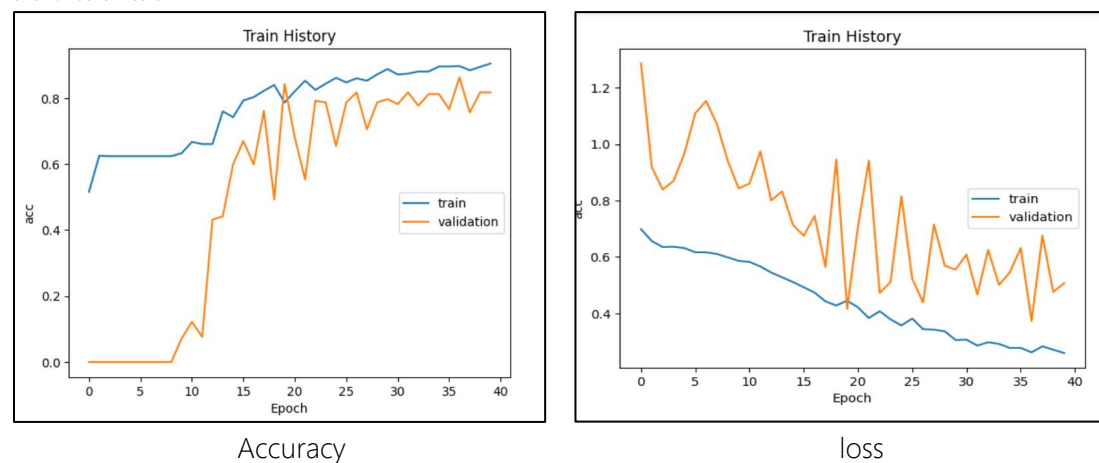
訓練 40 次,每一批訓練了 650 筆資料,從訓練資料集中取 20%來當作驗證資料集



```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])  
train_history=model.fit(x=x_Train4D_normalize,y=y_TrainOneHot,validation_split=0.2,epochs=40,batch_size=650,verbose=2)
```

Epoch 1/40  
2/2 - 6s - loss: 0.6985 - accuracy: 0.5159 - val\_loss: 1.2868 - val\_accuracy: 0.0000e+00 - 6s/epoch - 3s/step  
Epoch 2/40  
2/2 - 0s - loss: 0.6560 - accuracy: 0.6252 - val\_loss: 0.9178 - val\_accuracy: 0.0000e+00 - 57ms/epoch - 29ms/step  
Epoch 3/40  
2/2 - 0s - loss: 0.6332 - accuracy: 0.6239 - val\_loss: 0.8391 - val\_accuracy: 0.0000e+00 - 67ms/epoch - 33ms/step  
Epoch 4/40  
2/2 - 0s - loss: 0.6361 - accuracy: 0.6239 - val\_loss: 0.8696 - val\_accuracy: 0.0000e+00 - 65ms/epoch - 32ms/step  
Epoch 5/40  
2/2 - 0s - loss: 0.6315 - accuracy: 0.6239 - val\_loss: 0.9673 - val\_accuracy: 0.0000e+00 - 52ms/epoch - 26ms/step  
Epoch 6/40  
2/2 - 0s - loss: 0.6166 - accuracy: 0.6239 - val\_loss: 1.1093 - val\_accuracy: 0.0000e+00 - 74ms/epoch - 37ms/step  
Epoch 7/40  
2/2 - 0s - loss: 0.6164 - accuracy: 0.6239 - val\_loss: 1.1532 - val\_accuracy: 0.0000e+00 - 52ms/epoch - 26ms/step  
Epoch 8/40  
2/2 - 0s - loss: 0.6105 - accuracy: 0.6239 - val\_loss: 1.0712 - val\_accuracy: 0.0000e+00 - 51ms/epoch - 25ms/step  
Epoch 9/40  
2/2 - 0s - loss: 0.5983 - accuracy: 0.6239 - val\_loss: 0.9405 - val\_accuracy: 0.0000e+00 - 75ms/epoch - 38ms/step  
Epoch 10/40  
2/2 - 0s - loss: 0.5861 - accuracy: 0.6328 - val\_loss: 0.8431 - val\_accuracy: 0.0711 - 56ms/epoch - 28ms/step  
Epoch 11/40  
2/2 - 0s - loss: 0.5823 - accuracy: 0.6671 - val\_loss: 0.8602 - val\_accuracy: 0.1218 - 51ms/epoch - 26ms/step  
Epoch 12/40  
2/2 - 0s - loss: 0.5662 - accuracy: 0.6607 - val\_loss: 0.9744 - val\_accuracy: 0.0761 - 59ms/epoch - 30ms/step

訓練結果如下:



模型準確率為 83.74%,loss 值為 0.36


✓ 0秒 [20] loss, accuracy = model.evaluate(x\_Test4D\_normalize , y\_TestOneHot) #評估模型準確率  
print( "\nLoss: %.2f, Accuracy: %.2f%%" % (loss, accuracy\* 100 ))

8/8 [=====] - 0s 15ms/step - loss: 0.3599 - accuracy: 0.8374

Loss: 0.36, Accuracy: 83.74%

✓ 0秒 [21] prediction=np.argmax(model.predict(x\_Test4D\_normalize), axis=1) #顯示混淆矩陣  
pd.crosstab(y\_Test,prediction,rownames=['label'],colnames=['predict'])

8/8 [=====] - 0s 2ms/step

predict 0 1 

label 

0	96	27
---	----	----

1	13	110
---	----	-----