國立雲林科技大學

資訊管理所

機器學習專案報告

學生/學號：

張靖 M10823001

陳逢麒 M10823028

平靖翔 M10823024

古力維 M10823040

中華民國109年5月

摘要

本研究利用CIFAR10的資料集與VGG Face2資料集，分別進行圖片分類與人臉辨別，藉由卷曲類神經網路()演算法進行訓練，本研究使用了3種不同的卷曲類神經架構，當中包含Vgg16預訓練模型與simple Resnet模型。在影像辨識的熱潮下，許多關於影像辨識的架構與論文相繼出現，像是GoogleNet、VGG19、ResNet等，為此本研究選取普遍績效較好的架構當作訓練模型，建立出10種類行的辨識模型與20個不同人的人臉辨識，績效分別為0.8974以及0.90。

關鍵字：機器學習、CNN、Optimizer、VGG19

目錄

[摘要 i](#_Toc38803426)

[一、緒論 1](#_Toc38803427)

[1.1動機 1](#_Toc38803428)

[1.2目的 1](#_Toc38803429)

[二、研究方法 2](#_Toc38803430)

[2.1程式架構-每周工作時數 2](#_Toc38803431)

[2.2程式架構-年收入預測 6](#_Toc38803432)

[三、實驗 11](#_Toc38803433)

[3.1資料集 11](#_Toc38803434)

[3.2前置處理 13](#_Toc38803435)

[3.3實驗設計 14](#_Toc38803436)

[3.4實驗結果 19](#_Toc38803437)

[四、結論 23](#_Toc38803438)

[五、參考文獻 24](#_Toc38803439)

一、緒論

1.1動機

卷曲類神經網路(Convolutional neural network, CNN)的出現讓影像辨識的技術達到了新的高峰，同時許多組織或是學術界也相繼投入大量的資源及時間發展影像辨識，像是史丹佛大學每年都會舉辦ImageNet的視覺辨識的競賽，許多科技龍頭像是，Google、微軟等科技龍頭都相繼參與，加快了影像辨識的成熟度。Alex Krizhevsky、Vinod Nair與Geoffrey Hinton三人蒐集了8百萬張32x32的圖片，而其中CIFAR-10提供了60000張32x32，10種類別的圖片資料集以攻深度學習研究；而牛津大學的視覺研究社也提供了VGGFace的公開人臉資料集，為此本研究將以這兩種常用資料集做為訓練資料，來實作影像辨識。

1.2目的

本研究透過Keras來去建構整個影像辨識模型的架構與預測分類，並藉由現今論文所提供泛用的卷曲神經架構當作架構基底，並以此為出發點延伸新的架構或圖片前處理，加強模型的預測、分類的績效。

二、研究方法

2.1程式架構-影像辨識

【CNN by CIFAR10】

import numpy as np

import pandas as pd

import sklearn

from sklearn.metrics import accuracy\_score

from sklearn import preprocessing

from keras import layers, optimizers, models

from sklearn.preprocessing import LabelEncoder

import keras

from keras.layers import Dense, Conv2D, BatchNormalization, Activation, Dropout

from keras.layers import AveragePooling2D, Input, Flatten , GlobalAveragePooling2D ,MaxPooling2D

from keras.optimizers import Adam

from keras.callbacks import ModelCheckpoint, LearningRateScheduler

from keras.callbacks import ReduceLROnPlateau

from keras.preprocessing.image import ImageDataGenerator

from keras.regularizers import l2

from keras.models import Model

from keras.models import Sequential

import os

import tensorflow as tf

from keras.utils import plot\_model

import os

import cv2

categories = os.listdir("test") #取得CIFAR10下的所有名稱

X\_test = []

Y\_test = []

for categories\_name in categories:

path = "test/"+categories\_name #CIFAR10內的文件

files= os.listdir(path) #取得CIFAR10內所有資料夾下的資料名稱

for file in files: #走遍所有文件

if not os.path.isdir(file): #判断是否是文件夾，不是文件夹才打開

#print(path+"/"+file)

f = open(path+"/"+file,encoding='utf-8'); #打開文件

name = cv2.imread(path+"/"+file)

Y\_test.append(categories\_name)

X\_test.append(name) #每個文件的資料存到list中

#print(X\_test) #印出结果

#print(Y\_test)

import os

import cv2

categories = os.listdir("train") #取得CIFAR10下的所有名稱

X\_train = []

Y\_train = []

for categories\_name in categories:

path = "train/"+categories\_name #CIFAR10內的文件

files= os.listdir(path) #取得CIFAR10內所有資料夾下的資料名稱

for file in files: #走遍所有文件

if not os.path.isdir(file): #判断是否是文件夾，不是文件夹才打開

#print(path+"/"+file)

f = open(path+"/"+file,encoding='utf-8'); #打開文件

name = cv2.imread(path+"/"+file)

Y\_train.append(categories\_name)

X\_train.append(name) #每個文件的資料存到list中

#print(X\_train) #印出结果

#print(Y\_train)

#將預測值轉換成數字

Y\_train = pd.get\_dummies(Y\_train)

Y\_test = pd.get\_dummies(Y\_test)

#將list轉換成陣列

X\_train=np.array(X\_train)

X\_test=np.array(X\_test)

Y\_train=np.array(Y\_train)

Y\_test=np.array(Y\_test)

#正規化

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255

X\_test /= 255

#控制顯卡內核

os.environ["CUDA\_VISIBLE\_DEVICES"] = "0"

config = tf.ConfigProto(allow\_soft\_placement = True)

gpu\_options = tf.GPUOptions(per\_process\_gpu\_memory\_fraction = 0.5)

config.gpu\_options.allow\_growth = True

sess0 = tf.InteractiveSession(config = config)

# 將數據增加

datagen = ImageDataGenerator(

featurewise\_center=False, #在整個數據及上將輸入均值置為false

samplewise\_center=False, #將每個樣本均值置為false

featurewise\_std\_normalization=False,

samplewise\_std\_normalization=False,

zca\_whitening=False, # ZCA 白化

zca\_epsilon=1e-06, # ZCA 白化的 epsilon 值

rotation\_range=0, # 隨機圖像旋轉角度範圍 (deg 0 to 180)

width\_shift\_range=0.1, # 隨機水平平移圖像

height\_shift\_range=0.1, # 隨機垂直平移圖像

shear\_range=0., # 設置隨機裁剪範圍

zoom\_range=0., # 設置隨機缩放範圍

channel\_shift\_range=0., # 設置隨機通道切換範圍

fill\_mode='nearest', # 設置输入邊界之外的點的數據填充模式

cval=0., # 在 fill\_mode = "constant" 時使用的值

horizontal\_flip=True, # 隨機翻轉圖像

vertical\_flip=False, # 隨機翻轉圖像

rescale=None, # 設置重缩放因子 (應用在其他任何變換之前)

preprocessing\_function=None, # 設置應用在每一個输入的預處理函数

data\_format=None, # 圖像數據格式

validation\_split=0.0) # 保留用於驗證的圖像的比例 (控制在 0 和 1 之間)

datagen.fit(X\_train)

from keras.callbacks import EarlyStopping

Eearly\_stopping = EarlyStopping(monitor='val\_acc',mode='max', patience=20,restore\_best\_weights=True)

model = Sequential()

#當val\_acc最大時且取20次內最大值

model.add(Conv2D(96, (3, 3), activation='relu', padding = 'same', input\_shape=(X\_train.shape[1:]))) #input 96節點 kernel 3\*3

model.add(Dropout(0.2))

model.add(Conv2D(96, (3, 3), activation='relu', padding = 'same')) #input 96節點 kernel 3\*3

model.add(Conv2D(96, (3, 3), activation='relu', padding = 'same', strides = 2)) #input 96節點 kernel 3\*3

model.add(Dropout(0.5))

model.add(Conv2D(192, (3, 3), activation='relu', padding = 'same')) #input 192節點 kernel 3\*3

model.add(Conv2D(192, (3, 3), activation='relu', padding = 'same')) #input 192節點 kernel 3\*3

model.add(Conv2D(192, (3, 3), activation='relu', padding = 'same', strides = 2)) #input 192節點 kernel 3\*3

model.add(Dropout(0.5))

#input 192節點 kernel 3\*3 #same=Feature map寬高不變

model.add(Conv2D(192, (3, 3), activation='relu', padding = 'same'))

#input 192節點 kernel 3\*3 #valid=Feature map寬高會依據kernel size縮小一點

model.add(Conv2D(192, (1, 1), activation='relu',padding='valid'))

model.add(Conv2D(10, (1, 1), padding='valid')) #input 10節點 kernel 1\*1

model.add(GlobalAveragePooling2D()) #將特徵明顯化

model.add(Activation('softmax'))

model.summary()

model.compile(loss='categorical\_crossentropy', optimizer=Adam(lr=1.0e-4), metrics = ['accuracy'])

model\_details = model.fit(datagen.flow(X\_train, Y\_train, batch\_size = 32),epochs=2000,

validation\_data=(X\_test, Y\_test),callbacks=[Eearly\_stopping])

2.2程式架構-人臉辨識

【CNN by VGGFace2 】

from keras.models import Sequential

from keras.layers.normalization import BatchNormalization

from keras.layers.convolutional import Conv2D

from keras.layers.convolutional import MaxPooling2D

from keras.layers.core import Activation

from keras.layers.core import Flatten

from keras.layers.core import Dropout

from keras.layers.core import Dense

from keras import backend as K

from keras.preprocessing.image import ImageDataGenerator

from keras.optimizers import Adam

from keras.preprocessing.image import img\_to\_array

from sklearn.preprocessing import LabelBinarizer

import matplotlib.pyplot as plt

from imutils import paths

import numpy as np

import random

import cv2

import os

import tensorflow as tf

tf.logging.set\_verbosity(tf.logging.ERROR)

#function

#載入資料並將label轉換數字

def load\_data(imagePaths):

data = []

labels = []

for imagePath in imagePaths:

image = cv2.imread(imagePath)

image = cv2.resize(image, (IMAGE\_DIMS[1], IMAGE\_DIMS[0]))

image = img\_to\_array(image)

data.append(image)

label = imagePath.split(os.path.sep)[-2]

labels.append(label)

data = np.array(data, dtype="float") / 255.0

labels = np.array(labels)

lb = LabelBinarizer()

labels = lb.fit\_transform(labels)

return data, labels, lb

#資料路徑排序打亂

def randomPath(random\_seed, path\_x, path\_y):

random.seed(random\_seed)

random.shuffle(path\_x)

return path\_x, path\_y

#建立CNN

def build\_model(height, width, depth, classes):

model = Sequential()

inputShape = (height, width, depth)

chanDim = -1

if K.image\_data\_format() == "channels\_first":

inputShape = (depth, height, width)

chanDim = 1

model.add(Conv2D(32, (3, 3), padding="same",input\_shape=inputShape))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(3, 3)))

model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(64, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(128, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(256, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(256, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(512, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(512, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(1024))

model.add(Activation("relu"))

model.add(BatchNormalization())

model.add(Dropout(0.5))

model.add(Dense(classes, activation="softmax"))

return model

#set early stopping

from keras.callbacks import EarlyStopping

early\_stopping = EarlyStopping(monitor='val\_loss', min\_delta=0, patience=20, verbose=0,mode='min', baseline=None, restore\_best\_weights=True)

EPOCHS = 200 #epochs

INIT\_LR = 0.0001 #learning rate

BS = 32 #batch\_size

IMAGE\_DIMS = (160, 160, 3)

CLASSES = 20

#loading data

print("[INFO] Loading images...")

trainImagePaths = sorted(list(paths.list\_images("./VGGFace2/train")))

testImagePaths = sorted(list(paths.list\_images("./VGGFace2/test")))

trainImagePaths, testImagePath = randomPath(42,trainImagePaths,testImagePaths)

train\_data, train\_labels, train\_lb = load\_data(trainImagePaths)

test\_data, test\_labels, test\_lb = load\_data(testImagePaths)

(trainX, trainY, testX, testY) = (train\_data, train\_labels, test\_data, test\_labels)

#Data Augmentation

aug = ImageDataGenerator(rotation\_range=25, width\_shift\_range=0.1,

height\_shift\_range=0.1, shear\_range=0.2, zoom\_range=0.2,horizontal\_flip=True, fill\_mode="nearest")

#building model and training

print("[INFO] Compiling model...")

model = build\_model(width=IMAGE\_DIMS[0], height=IMAGE\_DIMS[1],depth=IMAGE\_DIMS[2], classes=len(train\_lb.classes\_))

opt = Adam(lr=INIT\_LR, decay=INIT\_LR / EPOCHS)

model.compile(loss="categorical\_crossentropy", optimizer=opt,metrics=["accuracy"])

model.summary()

print("[INFO] Training data...")

H = model.fit\_generator(aug.flow(trainX, trainY, batch\_size=BS),

validation\_data=(testX, testY),steps\_per\_epoch=len(trainX) / BS,epochs=EPOCHS, verbose=1,callbacks=[early\_stopping])

# save the model

print("[INFO] Save model...")

model.save("VGG\_new3.h5")

三、實驗

3.1資料集

表 1

CIFAR10 資料集

|  |  |  |  |
| --- | --- | --- | --- |
| 數據及特徵 | 影像 | 實例數 | 60000 |
| 屬性特徵 | 矩陣 | 屬性數量 | 10 |
| 相關任務 | 分類 | 缺少缺失值 | 否 |

表2

CIFAR10屬性資料

|  |  |  |
| --- | --- | --- |
| 屬性名稱 | 屬性類別 | 屬性說明 |
| airplane | Matrix | 32\*32\*3 |
| automobile | Matrix | 32\*32\*3 |
| bird | Matrix | 32\*32\*3 |
| cat | Matrix | 32\*32\*3 |
| deer | Matrix | 32\*32\*3 |
| dog | Matrix | 32\*32\*3 |
| frog | Matrix | 32\*32\*3 |
| horse | Matrix | 32\*32\*3 |
| ship | Matrix | 32\*32\*3 |
| truck | Matrix | 32\*32\*3 |

表3

VGGFace2 資料集

|  |  |  |  |
| --- | --- | --- | --- |
| 數據及特徵 | 影像 | 實例數 | 5593 |
| 屬性特徵 | 矩陣 | 屬性數量 | 20 |
| 相關任務 | 分類 | 缺少缺失值 | 否 |

表2

VGGFace2 屬性資料

|  |  |  |
| --- | --- | --- |
| 屬性名稱 | 屬性類別 | 屬性說明 |
| n000002 | Matrix | 160\*160\*3 |
| n000003 | Matrix | 160\*160\*3 |
| n000004 | Matrix | 160\*160\*3 |
| n000005 | Matrix | 160\*160\*3 |
| n000006 | Matrix | 160\*160\*3 |
| n000007 | Matrix | 160\*160\*3 |
| n000008 | Matrix | 160\*160\*3 |
| n000010 | Matrix | 160\*160\*3 |
| n000011 | Matrix | 160\*160\*3 |
| n000012 | Matrix | 160\*160\*3 |
| n000013 | Matrix | 160\*160\*3 |
| n000014 | Matrix | 160\*160\*3 |
| n000015 | Matrix | 160\*160\*3 |
| n000016 | Matrix | 160\*160\*3 |
| n000017 | Matrix | 160\*160\*3 |
| n000018 | Matrix | 160\*160\*3 |
| n000019 | Matrix | 160\*160\*3 |
| n000020 | Matrix | 160\*160\*3 |
| n000021 | Matrix | 160\*160\*3 |
| n000022 | Matrix | 160\*160\*3 |

3.2前置處理

1. CIFAR-10 dataset

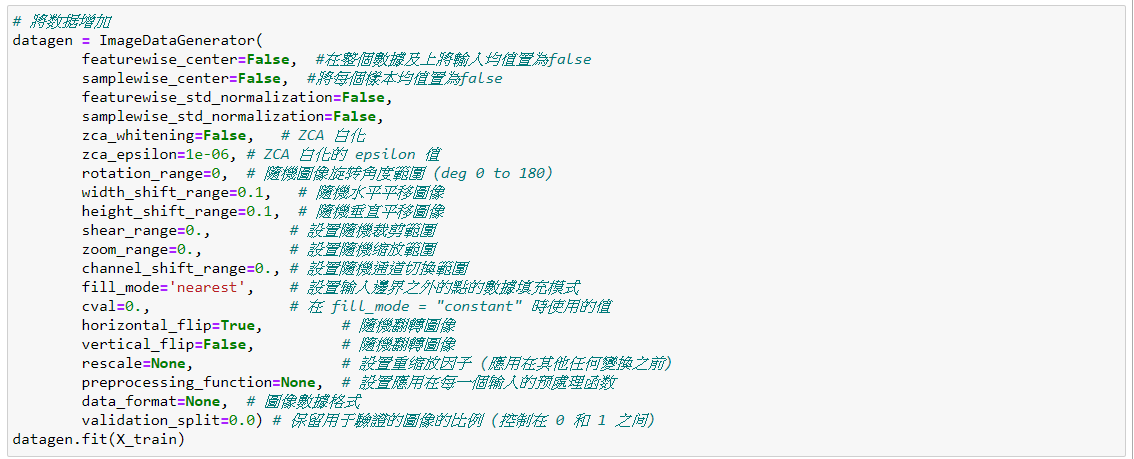
原始資料分為訓練和測試內容分為10個類別，將兩個資料讀取後，將Y\_train、Y\_test兩者資料中文字類別轉換成數字(One hot encoding)，轉換後將list資料轉換成陣列，對X\_train、X\_test做正規化，最後對dataset作增加，如：翻轉、位移等…，以利後續訓練資料夠多而做出較佳的績效。







(續下頁)



(續上頁)

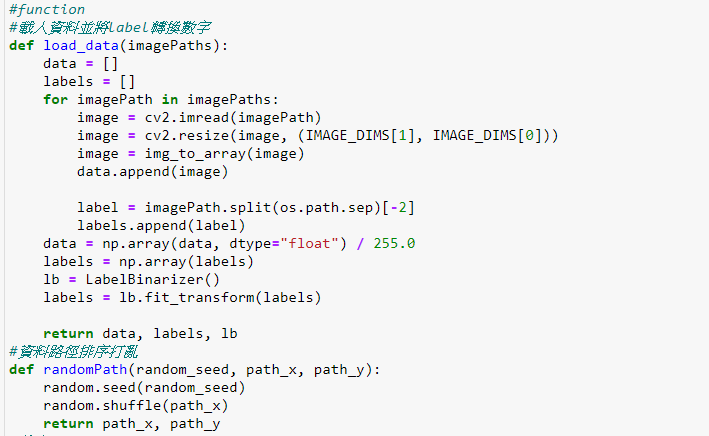
圖 1 CIFAR-10 dataset資料前處理

2. VGGFace2 dataset

原始資料分為訓練和測試內容分為20個類別，將兩個資料讀取後，將trainY、testY兩者資料中文字類別轉換成數字(One hot encoding)，轉換後將list資料轉換成陣列，對trainX、testX做正規化並將資料路徑排序打亂，最後對dataset做增加，如：翻轉、位移等…，以利後續訓練資料夠多而做出較佳的績效。



(續下頁)



(續上頁)

圖 2VGGFace2 dataset前處理

3.3實驗設計

1. CIFAR-10 dataset

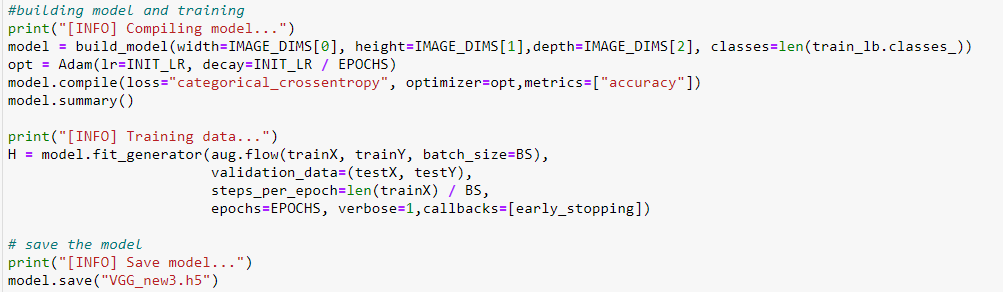
本研究是利用Anaconda3 Jupyter Notebook(Python version = 3.7)環境進行開發，使用的套件有sklearn的keras、pandas、numpy。對資料進行前處理後，使用卷積神經網路進行類別預測，而在類別預測中，神經層總共9層，節點數分別為96和192最後一層輸出層為10，padding設定same和valid，使用GlobalAveragePooling2D將特徵明顯化，使用Dropout參數設定0.2和0.5、激活函數使用relu，優化器使用adam；學習率設定為0.0001，loss使用categorical\_crossentropy，進而獲得績效以及準確度。



圖 3 CIFAR-10 dataset model訓練程式碼

2. VGGFace2 dataset

本研究是利用Anaconda3 Jupyter Notebook(Python version = 3.7)環境進行開發，使用的套件有sklearn的keras、imutils、random、cv2等。對資料進行前處理後，使用卷積神經網路進行類別預測，而在類別預測中，神經層總共10層，節點數分別為32、64、128、256、512、1024最後一層輸出層為label數量，每一層都使用BatchNormalization，padding設定same，使用MaxPooling2D將特徵明顯化，使用Dropout參數設定0.25和0.5、激活函數使用relu，優化器使用adam；學習率設定為0.0001，loss使用categorical\_crossentropy，進而獲得績效以及準確度。



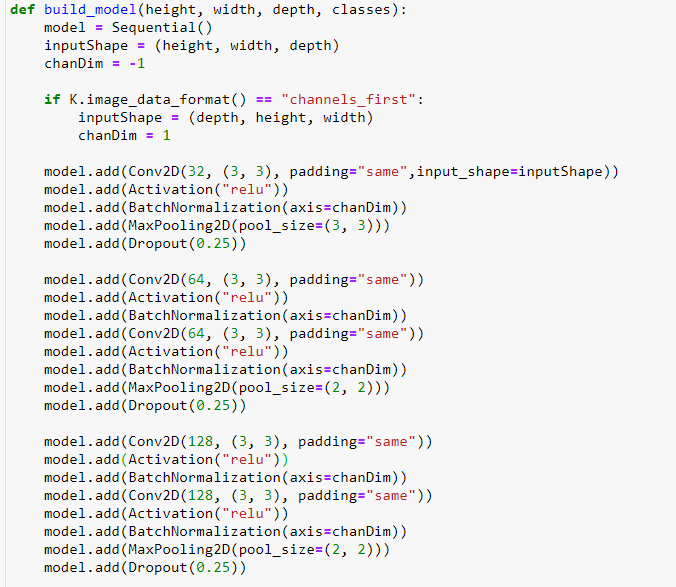
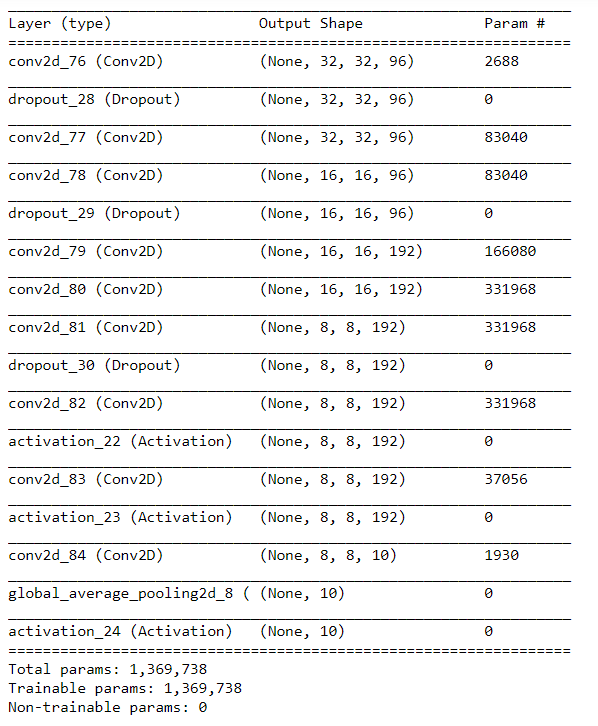


圖 4 VGGFace2 model訓練程式碼

3.4實驗結果

本研究小組利用卷積神經網路進行預測對資料集進行分析的結果如下：

1. CIFAR-10 dataset預測績效(類別預測)Accuracy約等於0.8974



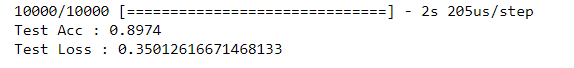
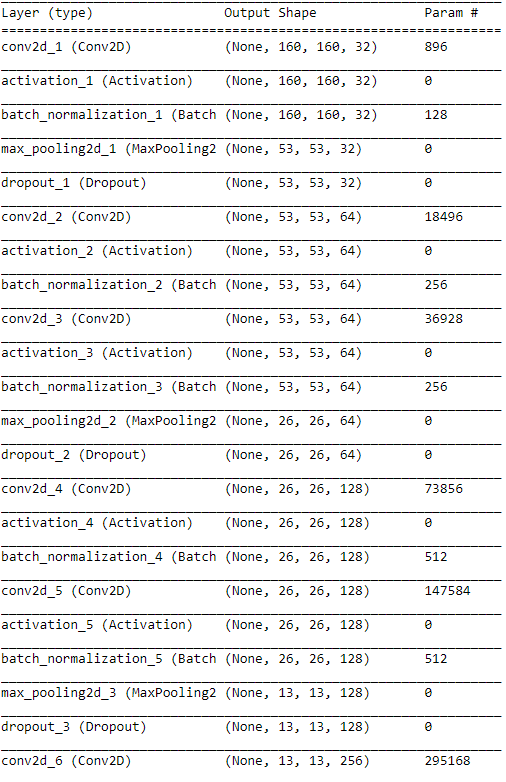
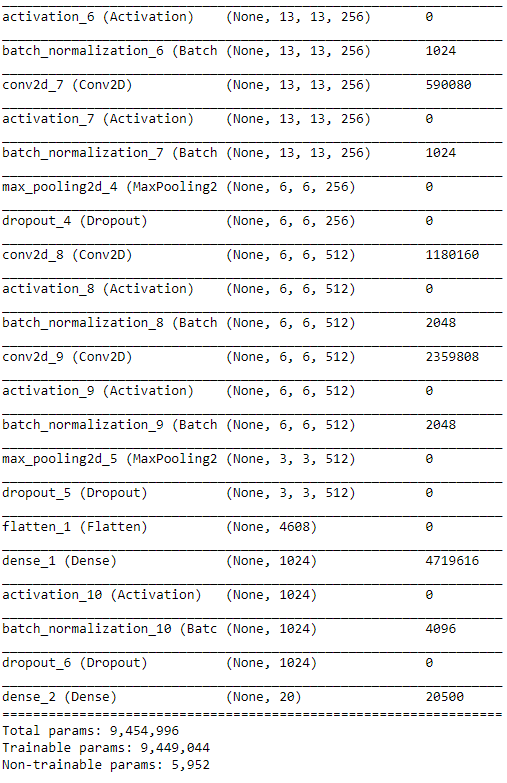


圖 5 CIFAR-10 dataset預測績效

2. VGGFace2 dataset預測績效(類別預測) Accuracy約等於0.84





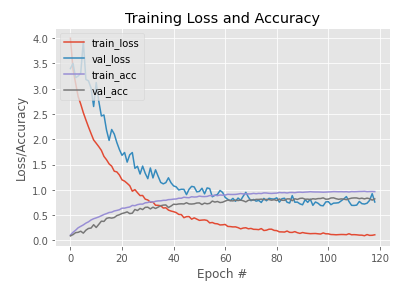




圖 6 VGGFace2 dataset預測績效

四、結論

本研究是以CIFAR-10 dataset 以及VGGFace2 dataset為主的資料，並使用卷積神經網路去進行分類並評估兩者的績效及準確度，最終我們得出的結果個別為CIFAR-10 dataset預測績效(類別預測) Accuracy約等於0.8974，VGGFace2 dataset預測績效(類別預測) Accuracy約等於0.8402。

五、參考文獻

CNN架構：

<https://keras-zh.readthedocs.io/examples/cifar10_resnet/>

<https://github.com/09rohanchopra/cifar10>

<https://www.pyimagesearch.com/2018/04/16/keras-and-convolutional-neural-networks-cnns/>