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

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): #判斷是否是文件夾，不是文件夾才打開
```

```
                #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) #印出結果
    #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): #判斷是否是文件夾，不是文件夾才打開
            #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) #印出結果
    #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',      # 設置輸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,  # 設置應用在每一個輸入的預處理函数
        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。

```
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): #判斷是否是文件夾，不是文件夾才打開
            #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) #印出結果
    #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): #判斷是否是文件夾，不是文件夾才打開
            #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) #印出結果
    #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)
#Y_train = keras.utils.to_categorical(Y_train, num_classes)
#Y_test = keras.utils.to_categorical(Y_test, num_classes)
```

```
#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
```

(續下頁)

(續上頁)

```
# 将数据增加
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,  # 隨機圖像旋转角度範圍 (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',    # 設置輸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,  # 設置應用在每一個輸入的預處理函数
        data_format=None,  # 圖像數據格式
        validation_split=0.0) # 保留用于驗證的圖像的比例 (控制在 0 和 1 之间)
datagen.fit(X_train)
```

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。

```
#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")
```

(續下頁)

(續上頁)

```
#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
```

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。

```
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])
```

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。

```
#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")
```

```
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
```

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

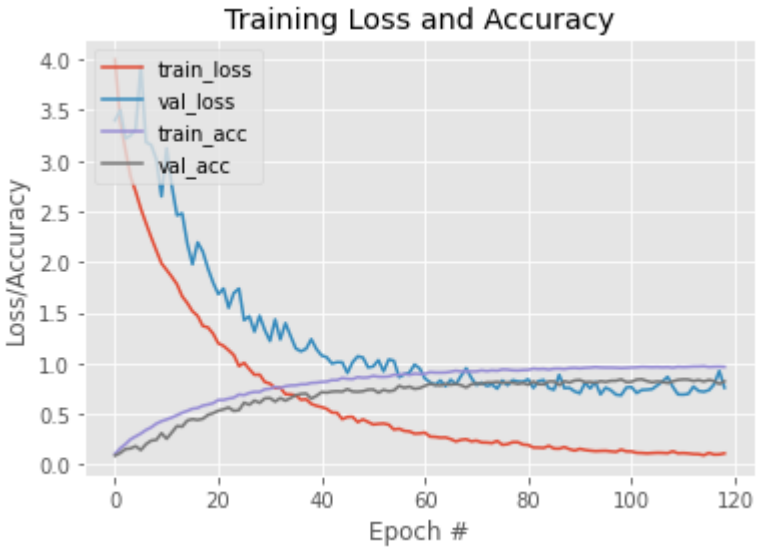
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_76 (Conv2D)           (None, 32, 32, 96)        2688
_________________________________________________________________
dropout_28 (Dropout)         (None, 32, 32, 96)        0
_________________________________________________________________
conv2d_77 (Conv2D)           (None, 32, 32, 96)        83040
_________________________________________________________________
conv2d_78 (Conv2D)           (None, 16, 16, 96)        83040
_________________________________________________________________
dropout_29 (Dropout)         (None, 16, 16, 96)        0
_________________________________________________________________
conv2d_79 (Conv2D)           (None, 16, 16, 192)       166080
_________________________________________________________________
conv2d_80 (Conv2D)           (None, 16, 16, 192)       331968
_________________________________________________________________
conv2d_81 (Conv2D)           (None, 8, 8, 192)         331968
_________________________________________________________________
dropout_30 (Dropout)         (None, 8, 8, 192)         0
_________________________________________________________________
conv2d_82 (Conv2D)           (None, 8, 8, 192)         331968
_________________________________________________________________
activation_22 (Activation)   (None, 8, 8, 192)         0
_________________________________________________________________
conv2d_83 (Conv2D)           (None, 8, 8, 192)         37056
_________________________________________________________________
activation_23 (Activation)   (None, 8, 8, 192)         0
_________________________________________________________________
conv2d_84 (Conv2D)           (None, 8, 8, 10)          1930
_________________________________________________________________
global_average_pooling2d_8 ( (None, 10)                0
_________________________________________________________________
activation_24 (Activation)   (None, 10)                0
=================================================================
Total params: 1,369,738
Trainable params: 1,369,738
Non-trainable params: 0


10000/10000 [==============================] - 2s 205us/step
Test Acc : 0.8974
Test Loss : 0.35012616671468133
```

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

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 160, 160, 32)	896
activation_1 (Activation)	(None, 160, 160, 32)	0
batch_normalization_1 (Batch	(None, 160, 160, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 53, 53, 32)	0
dropout_1 (Dropout)	(None, 53, 53, 32)	0
conv2d_2 (Conv2D)	(None, 53, 53, 64)	18496
activation_2 (Activation)	(None, 53, 53, 64)	0
batch_normalization_2 (Batch	(None, 53, 53, 64)	256
conv2d_3 (Conv2D)	(None, 53, 53, 64)	36928
activation_3 (Activation)	(None, 53, 53, 64)	0
batch_normalization_3 (Batch	(None, 53, 53, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 26, 26, 64)	0
dropout_2 (Dropout)	(None, 26, 26, 64)	0
conv2d_4 (Conv2D)	(None, 26, 26, 128)	73856
activation_4 (Activation)	(None, 26, 26, 128)	0
batch_normalization_4 (Batch	(None, 26, 26, 128)	512
conv2d_5 (Conv2D)	(None, 26, 26, 128)	147584
activation_5 (Activation)	(None, 26, 26, 128)	0
batch_normalization_5 (Batch	(None, 26, 26, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 13, 13, 128)	0
dropout_3 (Dropout)	(None, 13, 13, 128)	0
conv2d_6 (Conv2D)	(None, 13, 13, 256)	295168

```
_________________________________________________________________
activation_6 (Activation)    (None, 13, 13, 256)       0
_________________________________________________________________
batch_normalization_6 (Batch (None, 13, 13, 256)       1024
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 13, 13, 256)       590080
_________________________________________________________________
activation_7 (Activation)    (None, 13, 13, 256)       0
_________________________________________________________________
batch_normalization_7 (Batch (None, 13, 13, 256)       1024
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 6, 6, 512)         1180160
_________________________________________________________________
activation_8 (Activation)    (None, 6, 6, 512)         0
_________________________________________________________________
batch_normalization_8 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
activation_9 (Activation)    (None, 6, 6, 512)         0
_________________________________________________________________
batch_normalization_9 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_5 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              4719616
_________________________________________________________________
activation_10 (Activation)   (None, 1024)              0
_________________________________________________________________
batch_normalization_10 (Batc (None, 1024)              4096
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 20)                20500
=================================================================
Total params: 9,454,996
Trainable params: 9,449,044
Non-trainable params: 5,952
```




```
1127/1127 [==============================] - 1s 888us/step
loss: 0.6834349823254553, acc: 0.8402839396628217
```

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/