

投稿類別：資訊類

篇名：

深度學習－胎紋辨識

作者：

吳宜庭。國立新豐高中。高三 6 班。
張育芳。國立新豐高中。高三 6 班。
陳羿雯。國立新豐高中。高三 6 班。

指導老師：
林雪白老師
黃柏翔老師

壹、前言

一、研究動機

為了避免胎紋過淺，造成輪胎與路面摩擦力和抓地力降低，除了影響煞車距離使油耗量增加之外，而且容易爆胎以及排水性下降，導致產生水漂現象，進而造成不必要的意外，因此交通部早在 2016 年 7 月 1 日著手修法，規定胎紋不得少於零點八公釐，藉此避免胎紋不足所釀的禍，違規者將會吊扣牌照至改善為止；雖然有報章媒體的報導，去教導一般民眾，該如何判斷是否更換輪胎，但在日常生活中往往容易被忽略，都是發現車況有問題，進車廠才會更換，本論文希望能夠找到一個快速又方便的檢測，提升自身家人與其他用路人的安全。

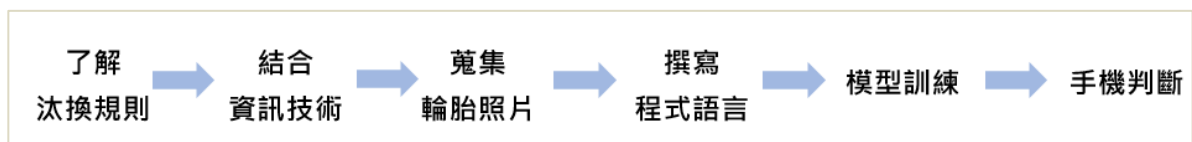
二、研究目的

現今手機功能多元化，而且幾乎人人皆有，希望透過手機將輪胎狀況拍下，可將深度學習已訓練好的模型作為判斷，快速得知是否該更換，讓駕駛者行車安全更有保障，本論文將研究目的分為四個：

- (一) 深度了解各種輪胎狀況，可供模型學習。
- (二) 收集各種輪胎的圖片，做為訓練檔案。
- (三) 深度學習模型程式撰寫。
- (四) 手機辨識程式撰寫。

三、研究方法

了解輪胎汰換規則後，討論該如何結合資訊技術，選擇近期火紅的深度學習實作，蒐集輪胎的圖片可供訓練，在網路上和書本中學習相關程式語言，論深度學習可使用的語言有 Pytorch、Keras 或 Tensorflow 等...，在當中選擇最適合易理解的語言製作，最後利用 Keras 該程式語言實作出一個可判斷輪胎優劣的模型，透過電腦內多次的運算學習達到一定的精準度，並且製作手機 APP 去判斷，讓車主可快速了解輪胎狀況，如圖一所示。



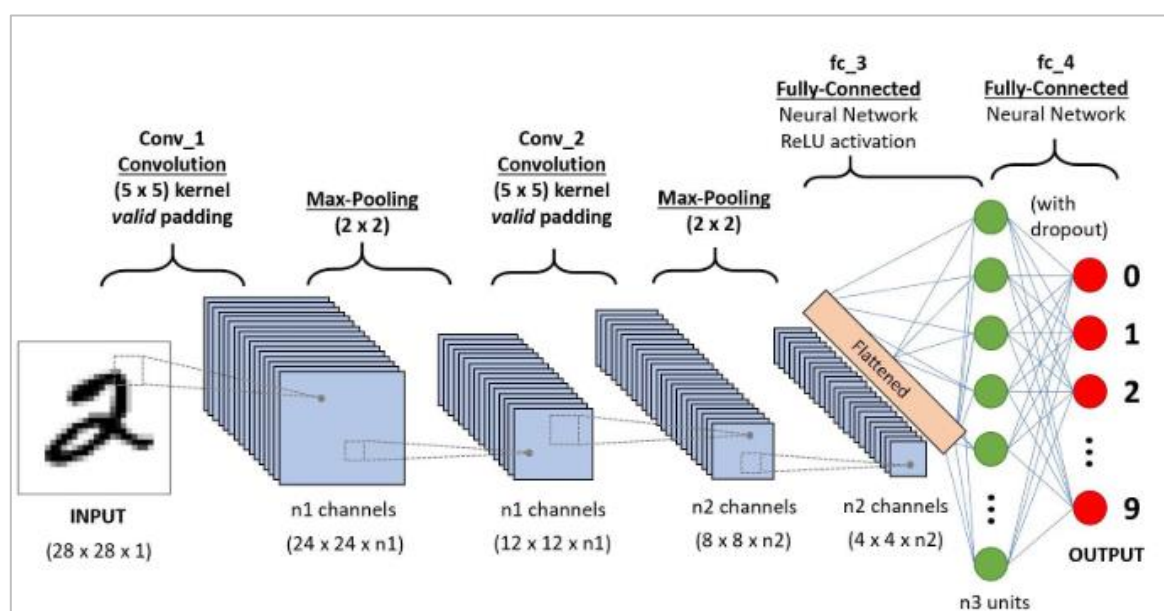
圖一、研究過程圖(資料來源：研究者繪製)

貳、正文

一、相關研究探討

(一) 深度學習

「自從 2006 年學者 **Geoffrey Hinton** 證明，多層次深度神經網路比傳統的類神經辨識效果好，深度神經網路改名為深度學習」(陳信銘、鄭維恆、黃子峻、郭萱聖、簡大為，2018)，因此深度學習架構上基本上就是人工類神經網路，也是機器學習的一種；在深度學習影像辨識應用中，卷積神經網路(Convolutional Neural Network, CNN)是基本演算法之一，主要分為三個部分，分別是卷積層(Convolutional Layer)、池化層(Pooling Layer)以及全連接層(Fully Connected Layer)，其運算首先會給予一個標準檔案，並且以亂數去執行，經由卷積運算和池化運算，可把特徵擷取下來，最後再由全連接層進行分類，如圖二所示。



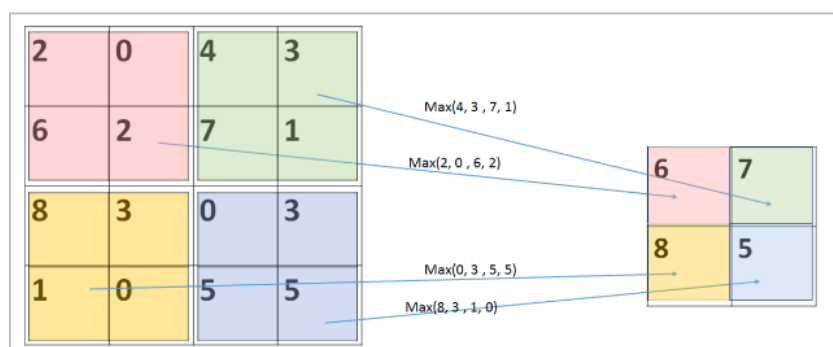
圖二、卷積神經網路架構圖 (資料來源：Sumit Saha, 2018)

1、卷積層(Convolutional Layer)

首先會比較兩張圖片裡的各個局部，這些局部被稱為特徵 (feature)，且捕捉兩張圖片的共同特徵去進行比對；當 CNN 在判別圖片時，在不知道上述特徵在哪裡的情況下，CNN 會比對圖片中的每個地方，且為了計算整張圖片裡有多少相符的特徵，創造了一套篩選機制，這套機制背後的數學原理被稱為卷積，計算特徵和圖片局部的相符程度，只要將兩者各個像素上的值相乘、再將總和除以像素的數量。舉例：如果兩個像素都是白色 (值為 1)，乘積就是 $1 * 1 = 1$ ；如果都是黑色 (值為 -1)，乘積就是 $(-1) * (-1) = 1$ 。也就是說像素相符的乘積為 1，像素相異的乘積為 -1。如果兩張圖的每一個像素都相符，將這些乘積加總、再除以像素數量就會得到 1；反之，如果兩者的像素完全相異，就會得到 -1。

2、池化層(Pooling Layer)

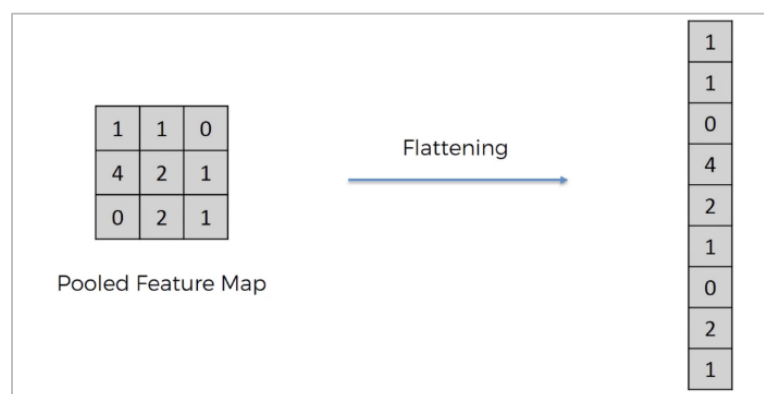
池化是一種將圖片資料量減少並保留重要資訊的方法，把原本的資料做一個最大化或是平均化的降維計算。它有三種不同型式 **Mean pooling**、**Max pooling**、**Sum pooling**，以 **Max pooling** 為例，先將 **4X4** 的矩陣分別以四個特徵去抓取最大值，再將最大值做為新的矩陣代表，此時矩陣將從 **4X4** 簡化為 **2X2**，如圖三所示，**Max pooling** 擁有去雜訊的功能，且當圖片平移的話也不會影響電腦的判斷。



圖三、池化層運算概念圖 (資料來源：Chtseng, 2017)

3、全連接層(Fully Connected Layer)

全連接將池化運算完矩陣的數值轉為一維陣列，此過程稱之為平坦化(Flatten)，最後將完成分類並且輸出該判斷狀況，如圖四所示。

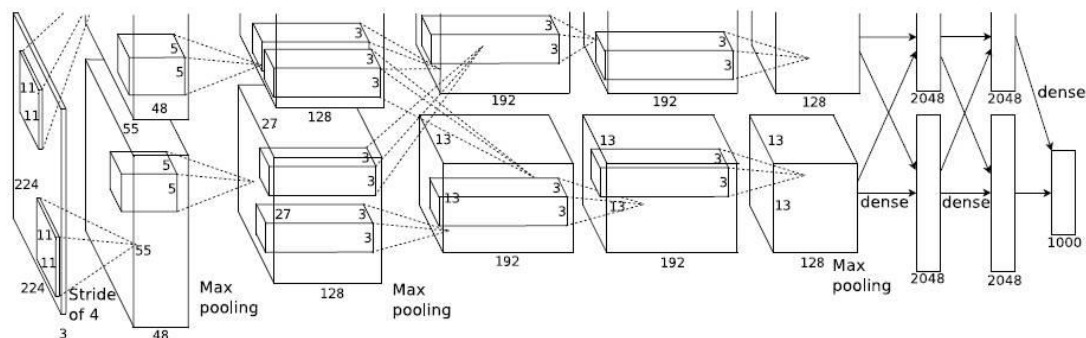


圖四、全連接層運算概念圖(資料來源：Yeh James, 2017)

(二) 神經網路模型－AlexNet

不論是哪種學習架構方式，都要給予機器模型，並且讓機器不斷的去運算，此部分是深度學習中最重要的一環，然而現今有很多種的模型，舉例來說有 **Alexnet**、**ResNet**、**VGGNet** 以及 **GoogleNet** 等...，而在一篇論文中有提及「**Alexnet** 其卷積層相較於其他如 **VGG16**、**GoogleNet** 架構而言來得簡便，運算量較小更能夠減少即時物件辨識時所需的時間」(李子暄, 2019)，因本論文圖片複雜度低，

不需要太多層數去運算，導致訓練時間拉長，因此選擇 AlexNet 運算，內有五層卷積層(Convolutional Layer)和三層全連接層(Fully Connected Layer)，運算完後就會輸出結果。

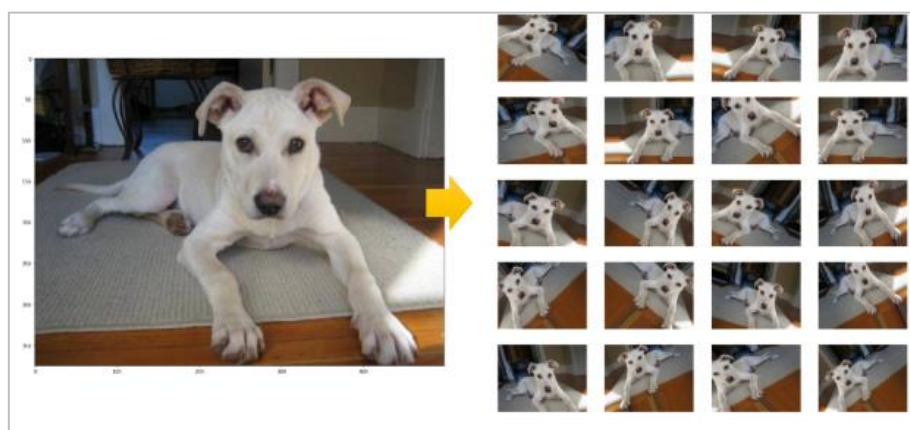


圖五、Alexnet 架構圖

(資料來源：Alex Krizhevsky、Ilya Sutskever、Geoffrey E. Hinton,2012)

(三) 影像資料擴增(Image Data Augmentation)

影像資料擴增(Image Data Augmentation)是為了彌補資料量的不足，因此將要給予模型學習的圖片，透過影像處理的方式，可以將圖片旋轉、剪裁或縮放等，去增加圖片的多元性以及提升資料量，如圖六所示。



圖六、資料擴增範例圖(資料來源：chtseng，2017)

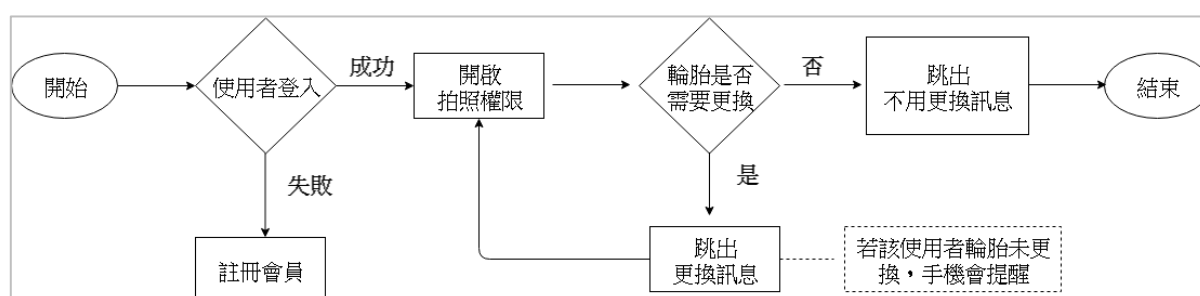
(四) OpenCV

OpenCV(全名 Open Source Computer Vision Library)為跨平台的視覺庫，可支援多種程式語言 C++、C#和 Python 等...，針對電腦視覺相關的演算法的處理，可運用在人機互動、物體識別、影像分割和機器視覺等領域，在機器學習當中，所使用的圖片集，可利用 OpenCV 將圖片做前處理，在一篇論文當中有提及使用「圖像標準化目的，為了加快速度，包括灰度轉換、標準化圖像大小和直方圖均衡處理，都必須將圖片尺寸統一。」(刘云鹏、李 瑾、潘 闻，2014)，因圖檔的增加導致精準度提升，因此本論文將會去尋找 OpenCV 相關影像處理的方式，選

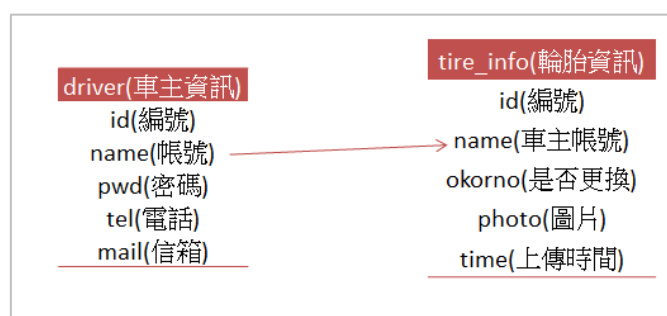
擇較合適的寫入圖片集。

二、系統設計

本論文實作會以手機 APP 操作，起初先把模型訓練好以及放置伺服器內，使用 APP 前車主需要登入可取得自身的資訊，並且再利用手機鏡頭去拍攝輪胎狀況，將會直接上傳到伺服器判斷，判斷結果會存放於資料庫，若判斷結果為需要更換輪胎，系統將會定期提醒車主該更換，更換後需要再拍攝，才會停止更換提醒，如圖七所示；本論文資料庫所使用兩個表(table)，如圖八所示，讓車主對輪胎車況有一個紀錄，希望藉由此可提升駕駛安全。



圖七、系統流程圖 (資料來源：研究者繪製)



圖八、資料庫表內容圖 (資料來源：研究者繪製)

三、系統實作

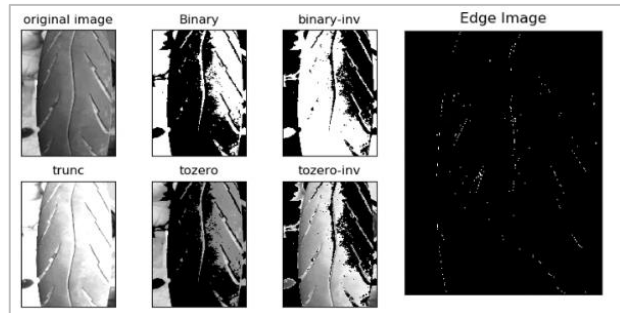
(一) 訓練模型

判斷輪胎是否更換的方式有幾種，除了量測胎紋不得少於零點八公釐之外，在輪胎上每隔一段就有標記塊，如圖九所示，將拍攝的輪胎圖片先行學習，因圖片的量不多，利用 Python 程式撰寫在 OpenCV 內尋找適合的影像處理演算法，希望可凸顯出輪胎的紋路，分別有邊緣檢測(Edge Detection)和二值化(Binary)等，如圖十、十一所示，選擇方式直接人眼觀看何種方式，可明顯區分出標記塊和輪胎的紋路，探討後決定影像資料擴增，會以灰階存放於圖片集內；利用 Keras 撰寫，程式讀取會載入執行需要的函式庫時，而照片載入進去的尺寸為 256x256，

如圖十二所示，照片會轉成電腦讀取的數據，並且至 Alexnet 模型去運算學習，最後把模型儲存下來，便於之後判斷不需要再重新跑學習，如圖十三、十四所示，利用視覺化可看到模型學習曲線趨近於零，而精準度約在 84 或 85%，如圖十五所示。



圖九、實際輪胎判斷圖
(資料來源：研究者拍攝)



圖十、OpenCV 影像處理
(資料來源：研究者製作)

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('./image/75761.jpg',0)
ret, thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)

titles = ['original image','Binary','binary-inv','trunc','tozero','tozero-inv']
images = [img,thresh1,thresh2,thresh3,thresh4,thresh5]

for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))

plt.show()
```

圖十一、OpenCV 程式碼撰寫
(資料來源：研究者製作)

```
import cv, shutil, random, glob
import cv2
import numpy as np
import pandas as pd
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D, BatchNormalization
import matplotlib.pyplot as plt
from datetime import datetime

resize = 256
def load_data():
    imgs = os.listdir("./train/")
    num = len(imgs)
    train_data = np.empty((400, resize, resize, 3), dtype='uint8')
    train_label = np.empty((400, ), dtype='uint8')
    test_data = np.empty((400, resize, resize, 3), dtype='uint8')
    test_label = np.empty((400, ), dtype='uint8')
    for i in range(400):
        img = cv2.imread("./train/%s" % imgs[i], cv2.IMREAD_COLOR)
        train_data[i] = cv2.resize(img, (resize, resize))
        train_label[i] = 1
    for i in range(400,800):
        img = cv2.imread("./train/%s" % imgs[i], cv2.IMREAD_COLOR)
        test_data[i] = cv2.resize(img, (resize, resize))
        test_label[i] = 0
```

圖十二、載入函示庫和照片程式轉寫
(資料來源：研究者製作)

```
for i in range(400):
    img = cv2.imread("./train/%s" % imgs[i], cv2.IMREAD_COLOR)
    test_data[i] = cv2.resize(img, (resize, resize))
    test_label[i] = 0
return train_data, train_label, test_data, test_label

train_data, train_label, test_data, test_label = load_data()
train_data, test_data = train_data.astype('float32'), test_data.astype('float32')
train_data, test_data = train_data/255, test_data/255

img = train_data[0]
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.axis('off')
plt.show()

train_label = keras.utils.to_categorical(train_label, 2)
test_label = keras.utils.to_categorical(test_label, 2)

model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', input_shape=(resize,resize,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
# output layer
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
model.summary()

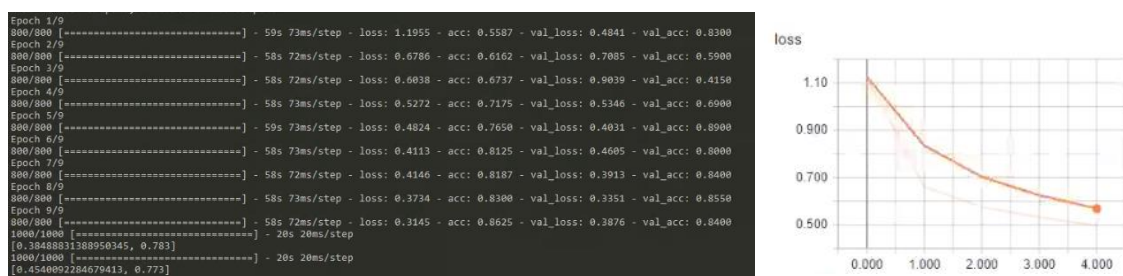
logdir = os.path.join("./", datetime.now().strftime("%Y%m%d%H%M%S"))
filepath = os.path.join(logdir, "best_model.h5")
callbacks = keras.callbacks.ModelCheckpoint(filepath, save_freq='epoch')
train_data, train_label, test_data, test_label = load_data()
model.fit(train_data, train_label, batch_size=64, epochs=50, validation_split=0.2, shuffle=True, callbacks=[callbacks])
scores = model.evaluate(train_data, train_label, verbose=1)
print(scores)
model.save("model.h5")
```

圖十三、Alexnet 模型程式碼撰寫
(資料來源：研究者製作)

```
model.add(Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
# output layer
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
model.summary()

logdir = os.path.join("./", datetime.now().strftime("%Y%m%d%H%M%S"))
filepath = os.path.join(logdir, "best_model.h5")
callbacks = keras.callbacks.ModelCheckpoint(filepath, save_freq='epoch')
train_data, train_label, test_data, test_label = load_data()
model.fit(train_data, train_label, batch_size=64, epochs=50, validation_split=0.2, shuffle=True, callbacks=[callbacks])
scores = model.evaluate(train_data, train_label, verbose=1)
print(scores)
model.save("model.h5")
```

圖十四、Alexnet 儲存模型程式碼撰寫
(資料來源：研究者製作)



圖十五、模型學習和模型視覺化

(二) 輪胎判斷

手機利用 **WebView** 內嵌網頁的方式使用，起始頁面為登入畫面，若未有會員帳號需註冊才可啟用該功能，如圖十六所示，伺服器內若收到輪胎畫面，將執行判斷程式，如圖十七所示，並且會將其資料紀錄至資料庫內，如圖十八所示，最後讀取此資料庫，是否更換輪胎在資料庫內，用數值零和一去判斷，零為需要更換，而一為不需更換，若該車主一直停留零的資料，會定期提醒該換輪胎，除非換好輪胎後再次判斷為一，才會結束該提醒。

圖十六、會員登入註冊畫面 (資料來源：研究者製作)

1	import os	id	name	passwd	tel	email	
2	os.environ["KERAS_BACKEND"] = "plaidml.keras.backend"	1	demo1	Z8tF6uc=	0988888888	demo1@demo1.com	
3	#import plaidml.keras		2	demo2	Z8tF6uQ=	02-22222135	demo2@gmail.com
4	#plaidml.keras.install_backend()		3	user01	MJwZ	0911115479	user321@kk.com
5			4	user02	MZ0atuT2qk4=	0971477224	
6	from keras.models import Sequential						
7	from keras.layers import Conv2D, MaxPooling2D						
8	from keras.layers import Activation, Dropout, Flatten, Dense						
9	from keras import backend as K						
10	from keras.models import load_model						
11	from keras.preprocessing.image import img_to_array, load_img						
12	import cv2						
13	import numpy as np						
14	import mysql.connector						
15							
16	model = load_model('alexnet.h5')						
17							
18	model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])	id	name	okorno	photo	time	
19	img = load_img('./right.1.jpg', target_size=(256, 256))	1	demo1	0	./photo/tire20190910150402.jpg	2019-09-10 15:04:02	
20	x = img_to_array(img)	2	demo1	1	./photo/tire20190911123002.jpg	2019-09-10 12:30:14	
21	arrayresized = cv2.resize(x, (256, 256))	3	demo1	1	./photo/tire201909112000.jpg	2019-09-10 20:03:16	
22	inputarray = arrayresized[np.newaxis, ...]						
23	prediction = model.predict_classes(inputarray)						
24	print(prediction)						

圖十七、模型判斷程式碼撰寫
(資料來源：研究者製作)

圖十八、資料庫資料
(資料來源：研究者製作)

參、結論

透過這次「深度學習」的研究，了解人工智慧能運用之處，也發現人工智慧並非想像中如此萬能，以本論文為例，在實驗的過程中，模型學習過後，精準度在 84% 左右，在判斷上還是有一定的誤差，並且會影響提醒功能的判斷，所以需要將模型的精準度再次提升，而在測試過程中，發現該功能在使用者操作上略顯不順，因此可再探討是否有更合適的方法，去提醒車主更換輪胎，而這次的學習到也只是深度學習當中的一小環節，希望日後在技術的探討與發展，還有更多角度和方向可以去思考，可使本論文功能更加的完善且可至相關 APP 平台上供人使用。

肆、引註資料

公路總局嘉義區監理所(2016)。認識輪胎與胎紋。2019 年 6 月 30 日，取自 <https://reurl.cc/zyAWVe>。

陳信銘、鄭維恆、黃子峻、郭萱聖、簡大為(2018)。結合深度學習演算法應用於道路車輛影像識別。電工通訊季刊，4。30-39。

Sumit Saha(2018)。A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way。2019 年 7 月 15 日，取自 <https://reurl.cc/24WQK9>。

Chtseng(2017)。初探卷積神經網路。2019 年 7 月 15 日，取自 <https://reurl.cc/72RVny>。

Yeh James (2017)。[資料分析&機器學習] 第 5.1 講：卷積神經網絡介紹(Convolutional Neural Network)。2019 年 7 月 15 日，取自 <https://reurl.cc/XXLQge>。

李子暄(2019)。設計與實現 AlexNet 架構應用於即時物件辨識。國立高雄科技大學：碩士論文。

Alex Krizhevsky、Ilya Sutskever、Geoffrey E. Hinton(2012)。ImageNet classification with deep convolutional neural networks。University of Toronto。

Chtseng(2017)。Data Augmentation 資料增強。2019 年 7 月 15 日，取自 <https://reurl.cc/Rdz19e>。

刘云鹏、李瑾、潘闻(2014)。基于 OpenCV 的智能相册系统。计算机系统应用，9(23)。61-64。