智慧化企業整合

汽車辨識(便是)王

第 4 組

110034547 邱韵婷

110034548 張瑋芩

110034568 陳彥碩

目錄

圖目錄

表目錄

一、 背景介紹

1.1 背景介紹

據相關資料顯示 70%的父母沒有足夠的時間陪伴孩子的成長,本組觀察到許多孩童對於路上汽車的喜愛,常看到小孩指著車子說是哪個品牌，而繁忙的父母，只能讓小孩透過書本自我學習，但小孩都喜歡生動有趣的學習方式。

因此本組發想「辨識王」平台，主要是應用 kaggle 上汽車品牌的圖片資料訓練以預測出該汽車品牌，再將此模型結合網站平台，只需父母幫小孩拍下欲認識的汽車品牌 1ogo(街上看到的汽車或是網路上開放的照片皆可)並上傳至辨識王平台，此平台即可準確預測出該圖片的品牌，不僅可顯示於網站中，還會提供發音及更多品牌的 1ogo 語音車款供孩童學習，不僅培養父母與孩童更多的互動，也讓小孩的成長階段更加豐富有趣。

1.2 問題定義 5W1H

進行本研究前，先以 5W1H 針對我們的問題定義進行分析，以更了解問題的本質。

Table 1 5W1H 分析

Who	想瞭解汽車的孩童。
What	孩童父母可藉由本研究開發的網頁來供小孩學習。
Why	孩童父母往往因為工作等因素，導致無法陪伴小孩一同成長與學習。
When	父母想與小孩互動，增進感情或是想與小孩一同學習時。
Where	家裡或是任何能與小孩互動的地點。
How	架設網站、深度學習。

二、 模型訓練與績效

2.1 資料蒐集

本研究的資料來源來自 Kaggle 網站上的 Car Brand Logos，資料包含了 8 種著名的汽車品牌。

Table 2 Logom 原始資料量

Car	Train	Test
Hyundai	302	50
Lexus	301	50
Mazda	317	50
Mercedes	342	50
Opel	301	50
Skoda	314	50
Toyota	306	50
Volkswagen	330	50

由 Table 2 可以發現，原始資料集的汽車品牌包含了 Hyundai、Lexus、Mazda、Mercedes、Opel、Skoda、Toyota 以及 Volkswagen 8 種汽車品牌。但因為特斯拉(Tesla)在近期是個非常熱門的議題，因此本研究便自行蒐集特斯拉的 Logo，總共蒐集了 300 張的特斯拉 Logo 圖片。

2.2 資料前處理

2.2.1 資料清理與整理

由於原本的資料集包含了一些跟 Logo 無關或是 Logo 圖片太小的資料，如 Figure 1 所示：

Figure 1 不符合的資料

並且將資料集依照 6-2-2 的比例，也就是 6 成是訓練集，2 成是驗證集，剩下的 2 成為測試集資料，整理結果如 Table 3 所示。

Table 3 整理後資料量

Car	Train	Validation	Test
Hyundai	163	55	55
Lexus	147	49	49
Mazda	190	64	64
Mercedes	186	61	62
Opel	156	51	52
Skoda	172	57	58
Tesla	164	54	54
Toyota	187	62	63
Volkswagen	200	67	67

2.2.2 資料擴增

因為拆分後的資料數太小，以至於之後可能會出現準確率太低等問題，因此便利用了資

料擴增的技術，將訓練集和驗證集的資料變為 4 倍。但單純的將同一張圖片從一張變成四張是不行的，因為這樣模型在訓練時會重複學到同一張圖片，測試出的準確率是不可信的，因此便利用了 keras 中的 ImageDataGenerator 來生成。

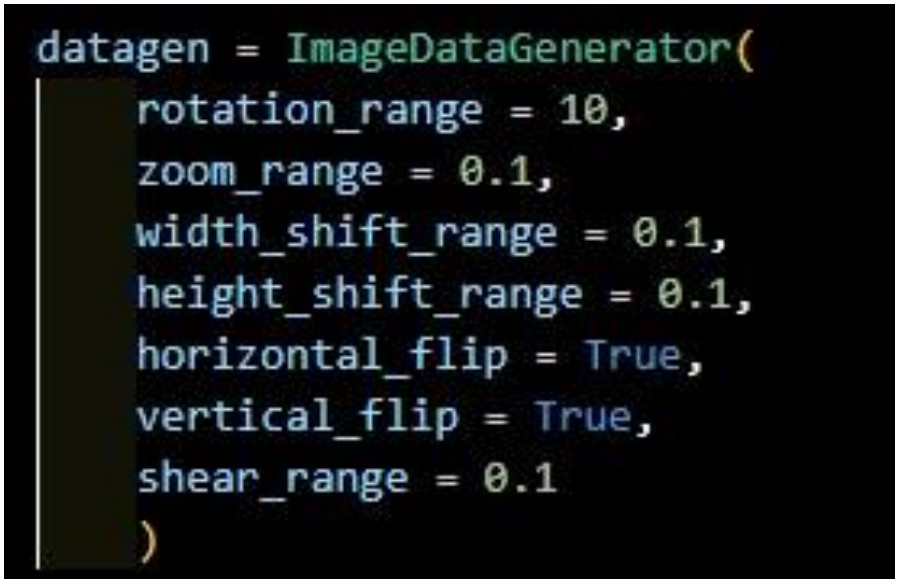

Figure 2 資料擴增程式碼

ImageDataGenerator 可以用來改變圖片的一些特性，像是可以調整翻轉的角度、縮放比例、水平翻轉、垂直翻轉、銳利度等等，雖然經過這些改變以我們人眼來說還是一樣的圖片，但是對於電腦來說，只要有些微的變化就是兩張不同的圖片。Table 4 為資料擴增後的資料筆數。

Table 4 資料擴增後資料量

Car	Train	Validation	Test
Hyundai	652	220	55
Lexus	588	196	49
Mazda	760	252	65
Mercedes	744	252	60
Opel	624	204	52
Skoda	688	228	58

Tesla	656	216	54
Toyota	748	248	63
Volkswagen	800	268	67

2.2.3 資料標準化

由於數據之間的差異性會影響到後續模型訓練的準確度，因此需統一將資料進行標準化的處理。

```
train_datagen = ImageDataGenerator(rescale = 1./255)

validation_datagen = ImageDataGenerator(rescale = 1./255)

test_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = train_datagen.flow_from_directory(train_data_dir,
                                                    target_size = (img_height, img_width),
                                                    batch_size = batch_size,
                                                    class_mode = 'categorical')

validation_generator = validation_datagen.flow_from_directory(validation_data_dir,
                                                              target_size = (img_height, img_width),
                                                              batch_size = batch_size,
                                                              class_mode = 'categorical')
```

Figure 3 資料標準化程式碼

2.3 模型架構

本研究選定四種在各資料預測競賽中表現優良的 CNN 模型架構，分別是 VGG16、VGG19、ResNet 與 DenseNet，藉由上述四種預訓練模型（Pre-Trained Model）並搭載本組所建之 DNN 模型，以找出能預測出本資料集最佳結果的模型，並透過實驗設計的方式進行參數最佳化，以訓練出績效最好的模型。

Table 5 本組 DNN 模型層級及說明

層級	說明
平坦層（Flatten）	銜接 CNN 層(Pre-Trained Model)與全連接層。
全連接層（FC）	主要在做最後的特徵提取，並且利用最後一層 FC 當作分類器
隱藏層 - Dropout 層	目的為防止過度擬合

輸出層	預測出的 9 種類別，Dense(9 , activation = 'softmax')

```
model.add(Flatten())
model.add(Dense(1024, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(512, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(512, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(number_of_class, activation = 'softmax'))
model.summary()
```

Figure 4 DNN 模型

VGG 是英國牛津大學 Visual Geometry Group 的縮寫，主要貢獻是使用更多的隱藏層，以大量的圖片訓練，提高準確率至 90%。其結構簡潔，由 5 層卷積層、3 層全連接層、輸出層（激活函數 softmax）構成，層與層之間使用Max-pooling（最大化池）分開，所有隱藏層的激活函數都採用 ReLU 函數。

VGG 應用多個較小卷積核（3x3）的卷積層代替一個較大卷積核的卷積層，一方面可以減少參數，另一方面相當於進行了更多的非線性映射，可以增加網絡的擬合及表達能力，由於卷積核專注於擴大通道數、池化專注於縮小寬和高，使模型層數更深且特徵圖更寬，也控制計算量的增加規模，其缺點為參數量龐大，計算資源需求高，且較難調整參數。

A. VGG16：

16 層包含 13 個卷積層及 3 個全連接層，其架構為卷積後接池化層，一來可減少參數的數量，加快計算速度，而減少參數可以防止過擬合的情況，且每個卷積都有一個激活函數，故可擬合更複雜的數據。

VGG16 架構圖如下，可觀察到其架構為每一塊包含若干卷積層和一個池化層，且同一塊內卷積層的通道（channel）數是相同的，隨著層數的增加通道數翻倍，圖片高和寬減半。

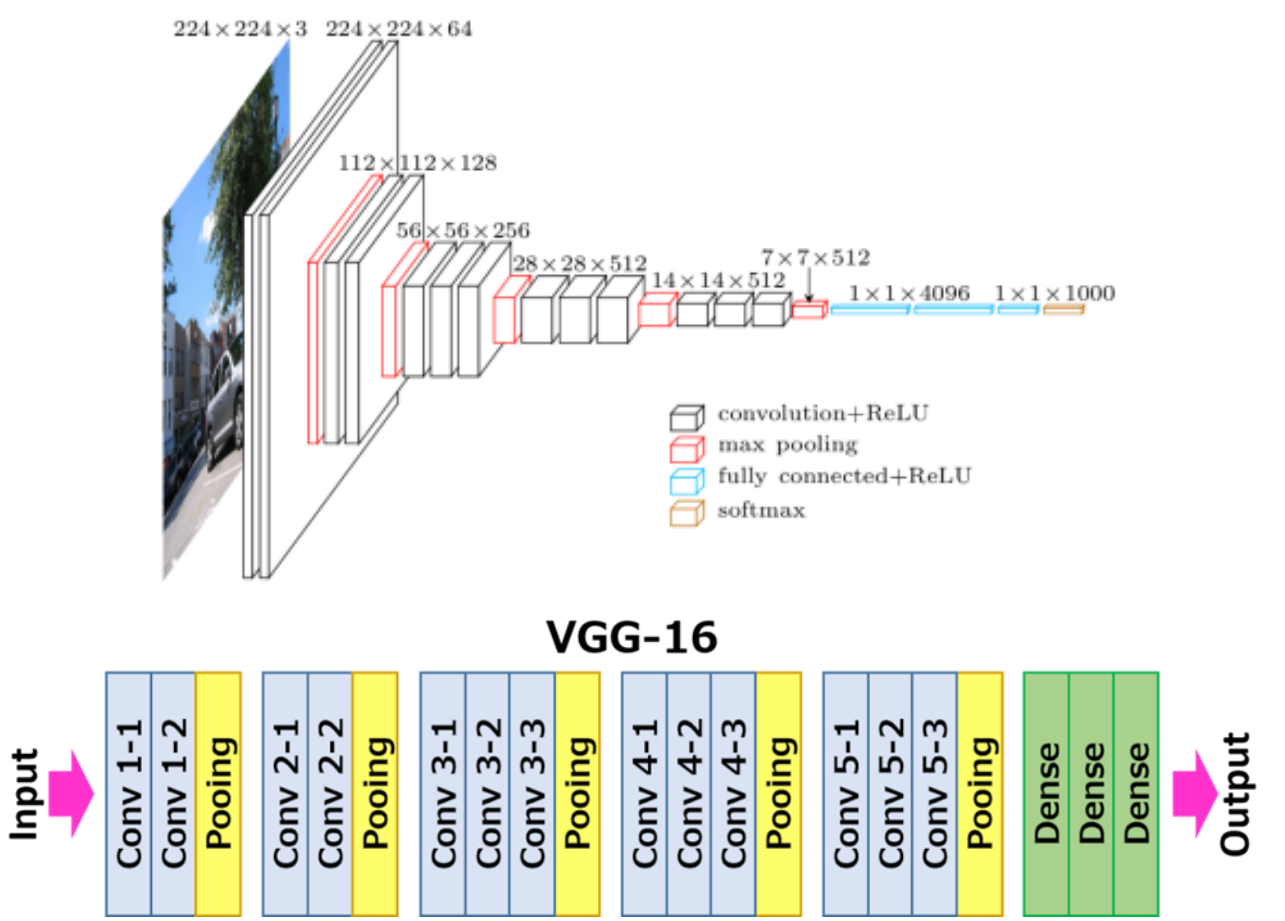

Figure 5 VGG 16 模型架構

資料來源：VGG16 -用于分类和检测的卷积网络

B. VGG19

19 層包含 16 個卷積層及 3 個全連接層，VGG19 架構與 VGG16 大同小異，差別在於網絡深度不一樣，也就是第三、四、五塊多增加一個卷積層，其架構圖如下。

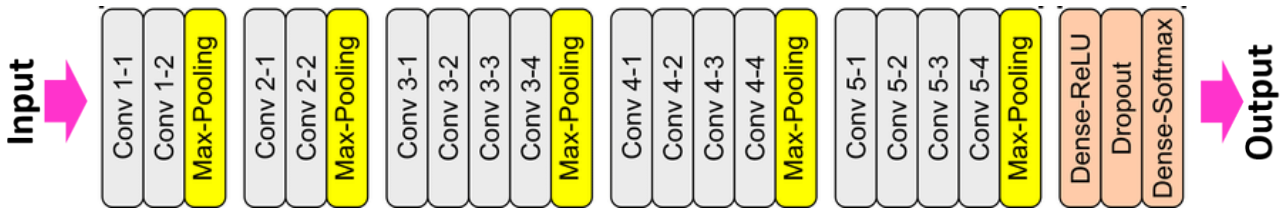

Figure 6 VGG19 模型架構

C. ResNet

具有兩種映射，分別為 Identity mapping 及 Residual mapping，前者指方程式中的 x，後

者則指 y-x，因此可用 F(x)表示殘差。殘差網路使用 Shortcut connection 的方式，目的是為了降低參數的數目。其建立在 VGG19 的基礎上進行修改，透過短路機制加入殘殘單元，變化在於 ResNet 使用 Stride=2 的卷積做下採樣，並用 Global Average Pooling（GAP）層取代全連接層。

另一項設計原則是當 feature map 降為原本的一半時，feature map 的數量增加一倍，其保持網絡層的複雜度，ResNet 相比普通網絡每兩層間增加短路機制，由此形成殘差學習。ResNet 結構非常容易修改和擴展，通過調整其 channel 數量以及堆疊的 block 數量，即可很容易地調整網絡的寬度和深度，以得到不同表達能力的網絡，只要訓練數據足夠，逐步加深網絡，就可以獲得更好的性能表現，下圖為 ResNet 及 VGG19 的架構圖比較。

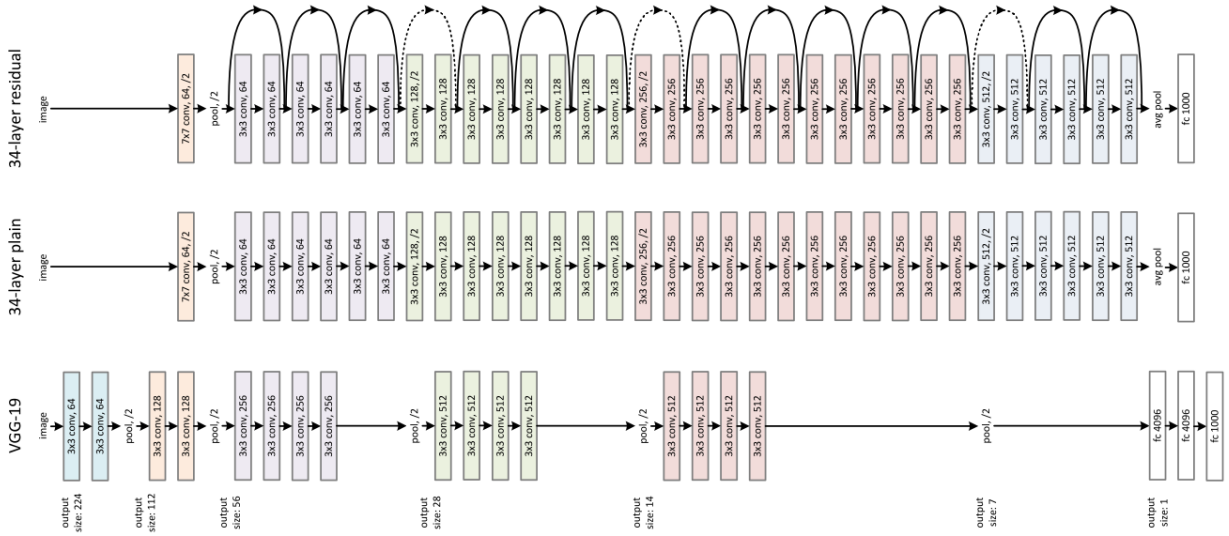

Figure 7 VGG19 及 ResNet 模型架構

資料來源：(Jia-Yau Shiau)Residual Leaning: 認識 ResNet 與他的冠名後繼者 ResNeXt、ResNeSt

D. DenseNet

其基本架構與 ResNet 類似，差別在於所有層之間都是密集連接的（Dense connection），另一大特色是更專注在特徵重用（feature reuse），這些優點使 DenseNet 在參數及成本更少的情況下，仍實踐比 ResNet 更優的性能。

DenseNet 互相連接所有的層，具體來說就是每個層都會接受其前面所有層作為其額外的輸入，ResNet 是每層與前面的某層短路連接在一起，但 DenseNet 則是每個層都會與前面所有層在 channel 維度上連接在一起，並作為下一層的輸入，可實現特徵重用，提升效率。除了上述優點外，其還可減緩梯度消失（vanishing-gradient）的問題及加強特徵傳播。

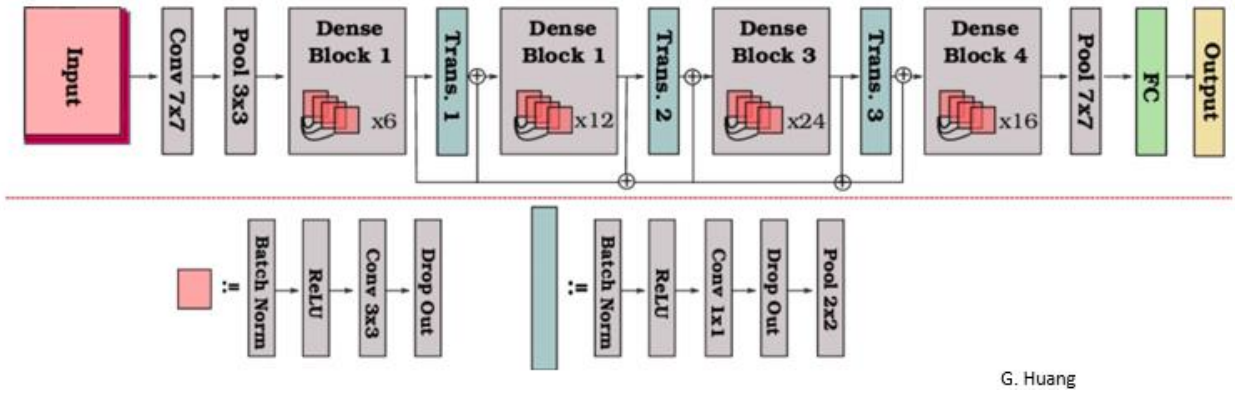

Figure 8 DenseNet 模型架構

2.4 超參數調整與模型績效比較

初步訓練只使用本組搭建的 CNN 模型，但最後預測結果準確度不超過 0.5，因此透過上述四種 Pre-Train 模型再搭配本組建之 CNN 模型，準確度大幅提升，所設定參數如下表。

Table 6 超參數調整項目

Model	Optimizer	Learning rate
VGG16	Adam	0.01~0.001
VGG19	SGD	0.001~0.0001
ResNet152V2	Adagrad	
DenseNet		

將上表中 Model、Optimizer（優化器）及 Learning rate（學習率）排列組合，完成本組實驗設計，以找到準確率最高的模型，當中學習率的部分使用 Keras 的回調函数 ReduceLROnPlateau，其於訓練過程中會依據所設定監控值以調整學習率，所輸入的值為上表中的數，但結果呈現上會以訓練當下所調整的最終學習率作為紀錄。

以下介紹所運用的優化器：

A. Adam：

結合 AdaGrad 和 RMSProp 兩種優化演算法的優點，對梯度的一階矩估計（First Moment

Estimation，即梯度的均值）和二階矩估計（Second Moment Estimation，即梯度的未中心化的方差）進行綜合考慮，其優點主要在於其有做偏置校正，使每次迭代（Epoch）學習率都有個確定範圍，讓參數的更新較為平穩，引數的更新不受梯度的伸縮變換影響，具有高校的運算速度且對記憶體需求少。

B. SGD：

梯度下降法 (Gradient descent ,GD)是一次用全部訓練集的數據計算其損失函 數的梯度，即更新一次參數，由於每次都抽取相同的樣本集耗時且冗餘，而 隨機梯度下降法(Stochastic gradient descent, SGD)則是每次隨機抽取一個樣本或小批次(mini-batch)樣本即計算其梯度的平均後更新參數，由於隨機梯度下 降演算法每次只隨機選擇一個樣本來更新模型參數，因此每次的學習是非常快速的，並且可以進行線上更新。也會處於一個高變異的狀態，更新時 loss 比較震盪，可能會使得其跳出區域性最優點到達一個更好的區域性最優。

C. Adagrad

學習速度與參數會互相對應，舉例來說，頻繁常見的參數，那麼其會進行小部分的更新，反之，則會進行很大的更新，因此很適合處理稀疏資料集，且減少 Learning rate 的手動調整，但缺點就是分母會不斷積累，這樣學習率就會收縮並最終會變得非常小。

Table 7 其他參數設定

參數	數值
Epochs	30
Batch size	16
Loss	Categorical_crossentropy

```
vgg = VGG16(include_top = False, weights = 'imagenet', input_shape = (img_height, img_width, 3))
vgg_layer_list = vgg.layers

model = Sequential()

for layer in vgg_layer_list:
    model.add(layer)
for layer in model.layers:
    layer.trainable = False

model.add(Flatten())
model.add(Dense(1024, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(512, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(512, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(rate = 0.1))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(number_of_class, activation = 'softmax'))
model.summary()

#選擇一優化器
opt = Adam(learning_rate = 0.01, beta_1 = 0.9, beta_2 = 0.999, epsilon = 10E-8)
#? opt = SGD(learning_rate = 0.01, momentum = 0, nesterov = False)
#? opt = Adagrad(learning_rate = 0.01, epsilon = None, decay = 0)

model.compile(optimizer = opt, loss = "categorical_crossentropy", metrics = ["accuracy"])
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.8, patience = 3, min_lr = 0.001, verbose = 1)
history = model.fit_generator(train_generator, epochs = 16, validation_data = validation_generator, callbacks = [reduce_lr])
```

Figure 9 訓練模型程式碼（以 Pre-Train Model 為 VGG16 為例）

Table 8 本組實驗設計的紀錄結果

DOE	Model	Optimizer	Learning rate(Final)	Train Acc	Test Acc
1	VGG16	Adam	0.01~0.001(**0.001**)	0.5048	0.5516
2	VGG16	Adam	0.001~0.0001(**0.001**)	0.6898	0.8298
2.5	VGG16	Adam	0.0001~0.00001(**0.000025**)	0.8144	0.8718
3	VGG16	SGD	0.01~0.001(**0.01**)	0.6449	0.8049
4	VGG16	SGD	0.001~0.0001(**0.0005**)	0.6667	0.8203
5	VGG16	Adagrad	0.01~0.001(**0.005**)	0.7896	0.8432
6	VGG16	Adagrad	0.001~0.0001(**0.001**)	0.7377	0.8508
7	VGG19	Adam	0.01~0.001(**0.01**)	0.6553	0.7954
8	VGG19	Adam	0.001~0.0001(**0.0005**)	0.6578	0.8203
9	VGG19	SGD	0.01~0.001(**0.01**)	0.6421	0.7935
10	VGG19	SGD	0.001~0.0001(**0.001**)	0.6217	0.8221
11	VGG19	Adagrad	0.01~0.001(**0.005**)	0.6642	0.826
12	VGG19	Adagrad	0.001~0.0001(**0.001**)	0.701	0.8375
13	ResNet152V2	Adam	0.01~0.001(**0.0025**)	0.8456	0.8846
14	ResNet152V2	Adam	0.001~0.0001(**0.0001**)	0.8756	0.8946
15	ResNet152V2	SGD	0.01~0.001(**0.0025**)	0.9649	0.9522
16	ResNet152V2	SGD	0.001~0.0001(**0.000125**)	0.946	0.9598
17	ResNet152V2	Adagrad	0.01~0.001(**0.005**)	0.9465	0.9503
18	ResNet152V2	Adagrad	0.001~0.0001(**0.001**)	0.9436	0.9345
19	DenseNet	Adam	0.01~0.001(**0.001**)	0.8154	0.836
20	DenseNet	Adam	0.001~0.0001(**0.0005**)	0.8364	0.8492
21	DenseNet	SGD	0.01~0.001(**0.01**)	0.9501	0.9513
22	DenseNet	SGD	0.001~0.0001(**0.00025**)	0.9423	0.9536
23	DenseNet	Adagrad	0.01~0.001(**0.005**)	0.9416	0.9541
24	DenseNet	Adagrad	0.001~0.0001(**0.001**)	0.9412	0.9434

上表中可觀察到綠色網底的儲存格為各模型中訓練資料集準確度最高的，而粉紅色網底的儲存格為測試資料集準確度最高的。VGG16 訓練及測試資料集（DOE2.5）準確度最高的組合：優化器 Adam、最低學習率設置為 0.0001~0.00001；VGG19 訓練及測試資料集（DOE12）

準確度最高的組合：優化器 Adagrad、最低學習率設置為 0.001~0.0001；ResNet 訓練資料集（DOE15）準確度最高：優化器 SGD、最低學習率設置為 0.01~0.001，測試資料集（DOE16）準確度最高：優化器 SGD、最低學習率設置為 0.001~0.0001；DenseNet 訓練資料集（DOE21）準確度最高：優化器 SGD、最低學習率設置為 0.01~0.001，測試資料集（DOE23）準確度最高：優化器 Adagrad、最低學習率設置為 0.01~0.001。

分別將各模型準確度高的訓練過程陳列至下圖，左邊皆為訓練資料及驗證資料的 Loss、右邊皆為訓練資料及驗證資料的準確度，可觀察到雖然訓練資料集 Loss 會不斷下降，但驗證資料集隨著 Epoch 增加漸漸地收斂，而準確度的部分可以觀察到訓練資料集跟驗證資料集的準確率都持續上升，沒有發生只有訓練資料準確度持續上升但驗證資料準確度沒上升的情況，也證實本組的模型沒有過擬合。

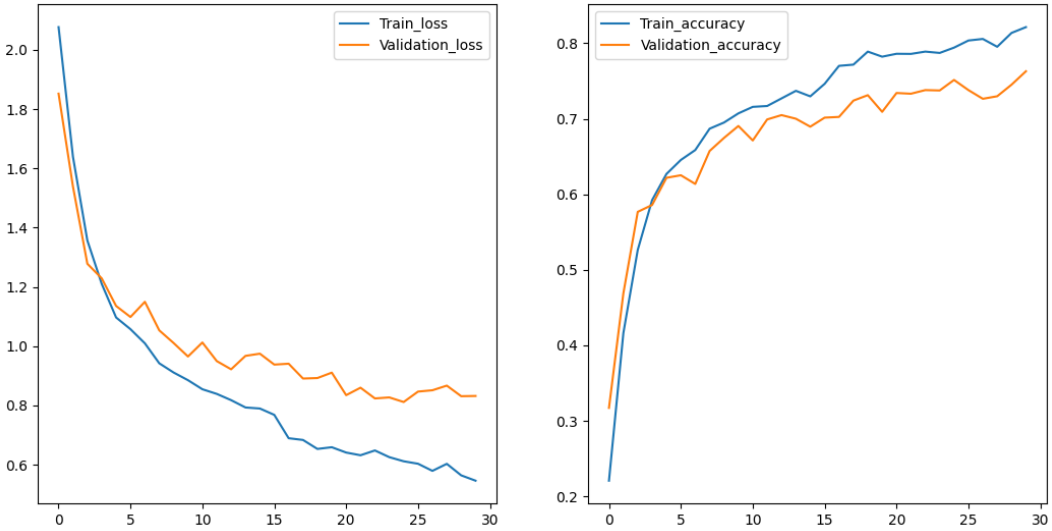

Figure 10 DOE 2.5

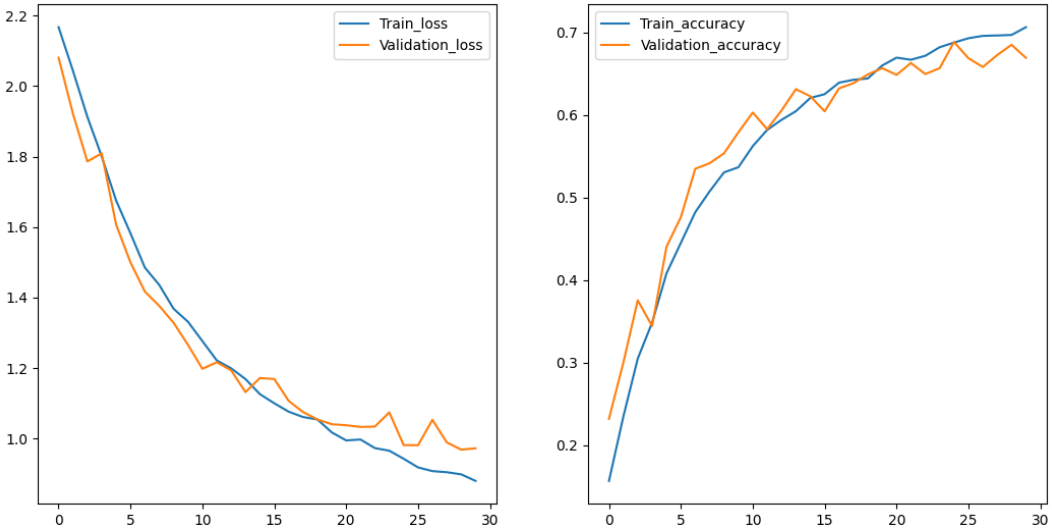

Figure 11 DOE 12

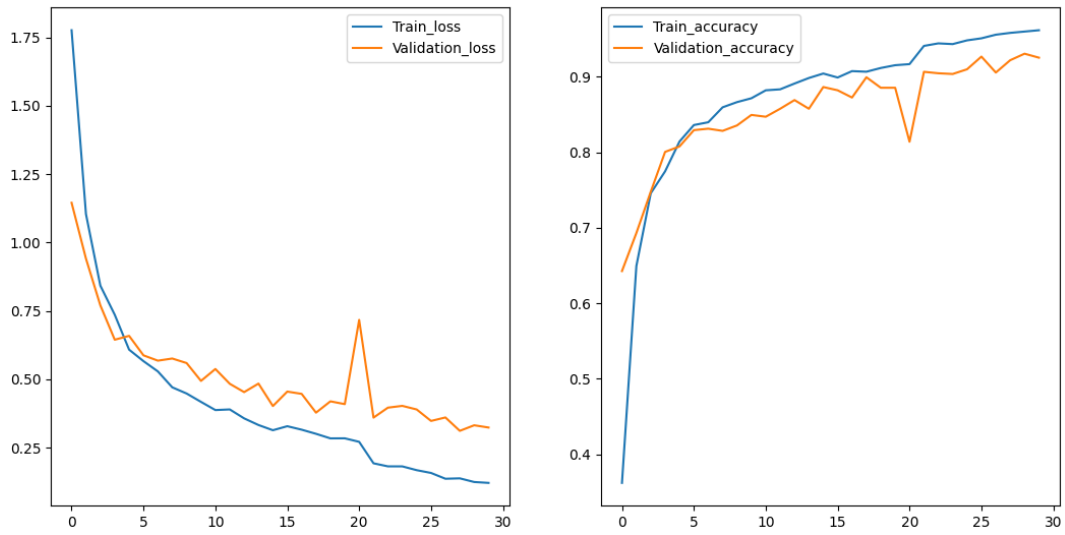

Figure 12 DOE 15

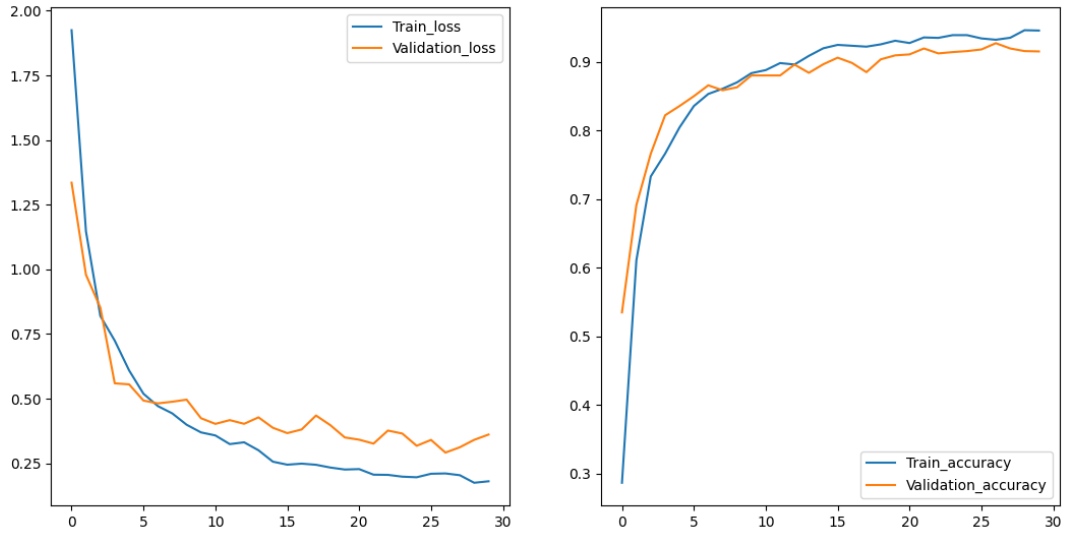

Figure 13 DOE 15

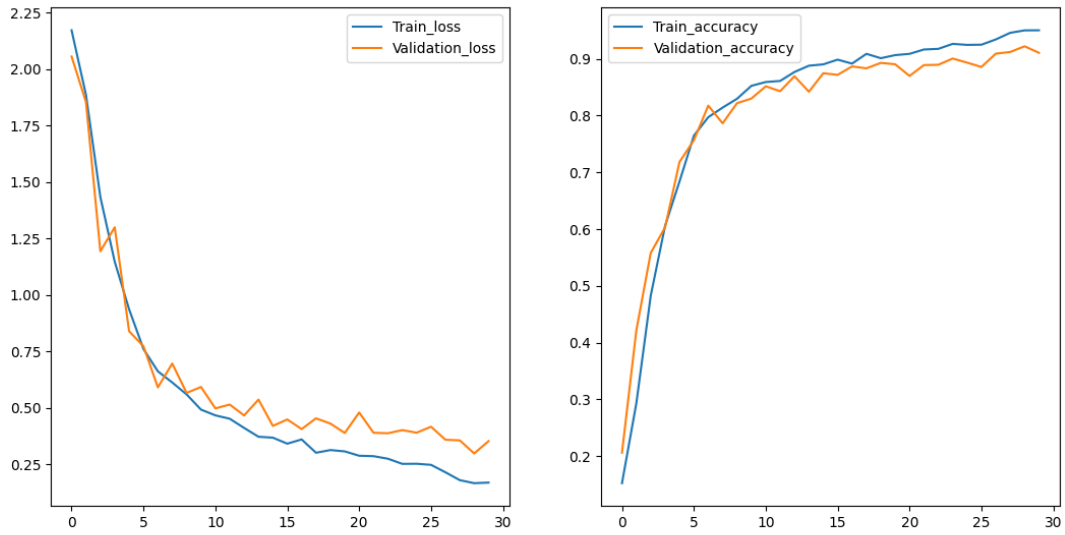

Figure 14 DOE 21

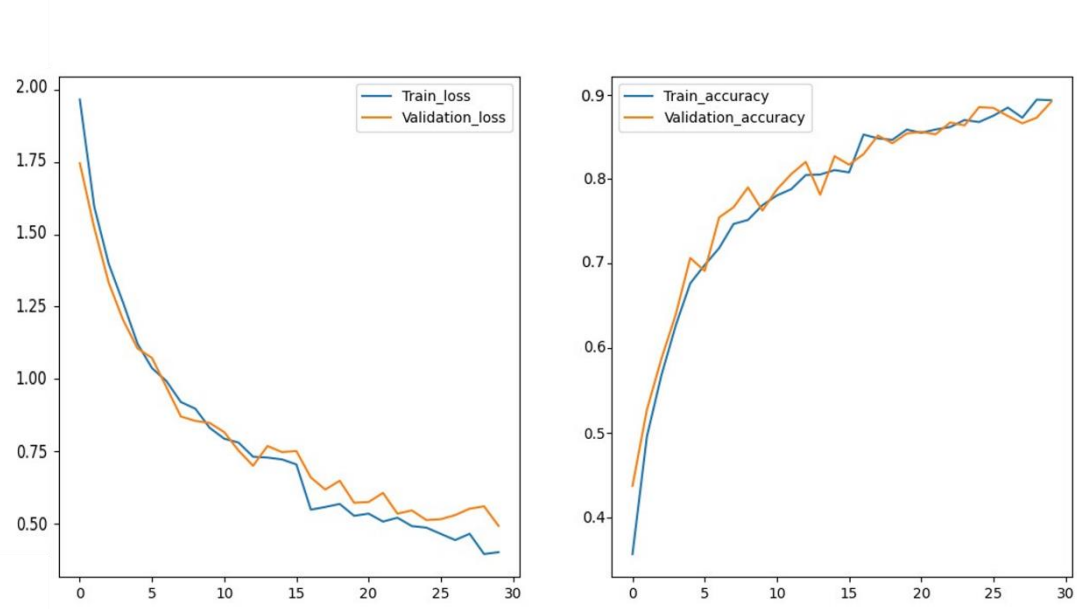

Figure 15 DOE 23

本組最終選擇實驗設計中的第 16 個其 Pre-Train 模型為 ResNet152、優化器為 SGD、最終學習率為 0.000125，將本模型的訓練結果放入本組辨識王網站中，以測試所上傳照片可準確辨識，沒有選擇訓練準確率最高的模型是因為其只代表其訓練的結果，真正餵入沒有訓練過的圖片時準確率也要高，才可代表本組所訓練的模型是真正有用的，也就是模型是具有泛化能力的。

三、 Web

本次 project 結合網頁與演算法。能夠匯入圖片至網頁中，透過 CNN 辨識出汽車品牌與 Logo，並在網頁中顯示還能夠教小朋友正確的汽車品牌發音。

3.1 Server 建構

利用 XAMPP 在本機架設伺服器，能夠讓網站在呼叫演算法 python 時比較有效率。

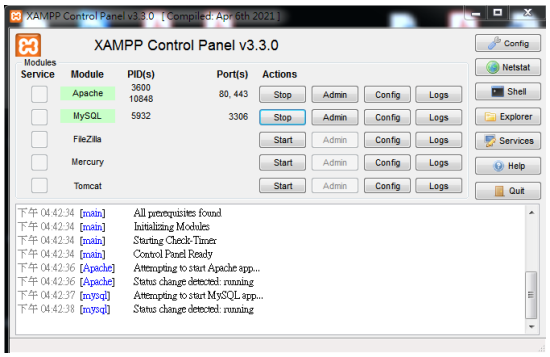

Figure 16 XAMPP 控制介面

3.2 網站執行 AI

透過 move_uploaded_file 將照片移至指定的檔案路徑，並透過 command 指令將網站與 Python 演算法程式結合，接收圖像辨識結果。

```php
# 檢查檔案是否上傳成功
if ($_FILES['img']['error'] === UPLOAD_ERR_OK){
  //echo '檔案名稱: ' . $_FILES['img']['name'] . '<br/>';
  //echo '檔案類型: ' . $_FILES['img']['type'] . '<br/>';
  //echo '檔案大小: ' . ($_FILES['img']['size'] / 1024) . ' KB<br/>';
  //echo '暫存名稱: ' . $_FILES['img']['tmp_name'] . '<br/>';

  # 檢查檔案是否已經存在
  if (file_exists('upload/' . $_FILES['img']['name'])){
    //echo '檔案已存在。<br/>';
  } else {
    $file = $_FILES['img']['tmp_name'];

    #xampp
    $dest = 'Test_PIC/' . $_FILES['img']['name'];

    # 將檔案移至指定位置
    move_uploaded_file($file, $dest);
  }
} else {
  echo '錯誤代碼：' . $_FILES['img']['error'] . '<br/>';
}
```

Figure 17 php 語法(上傳照片)

```php
$command2=escapeshellcmd('python test.py '.$imagename);
$output=shell_exec($command2);
```

Figure 18 php 語法(連接 python 演算法)

3.3 網頁介面

網站運行的影片與照片，顯示出我們的演算法能夠完善的適用在現實生活中，具有可行性。

影片.mp4

Figure 19 網頁影片

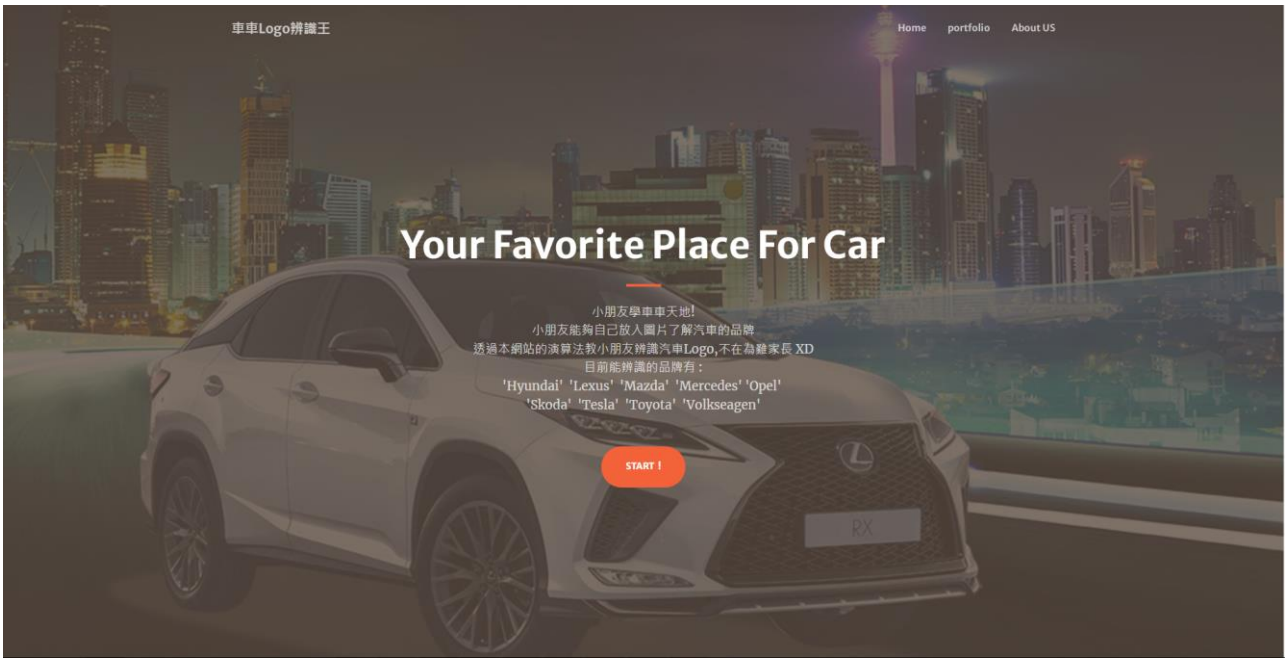

Figure 20 Home Page

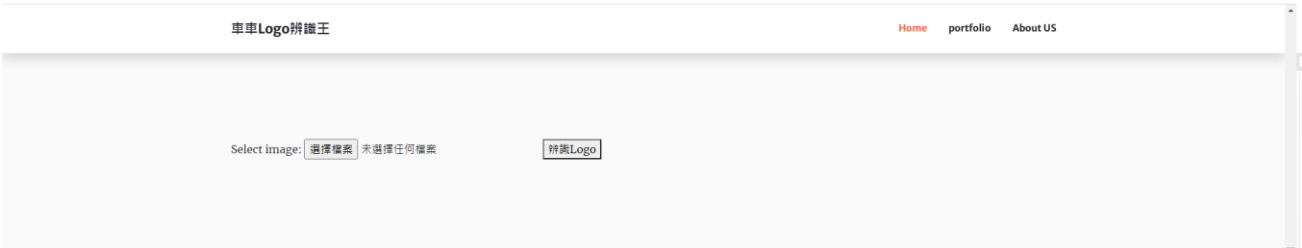

Figure 21 主要功能區

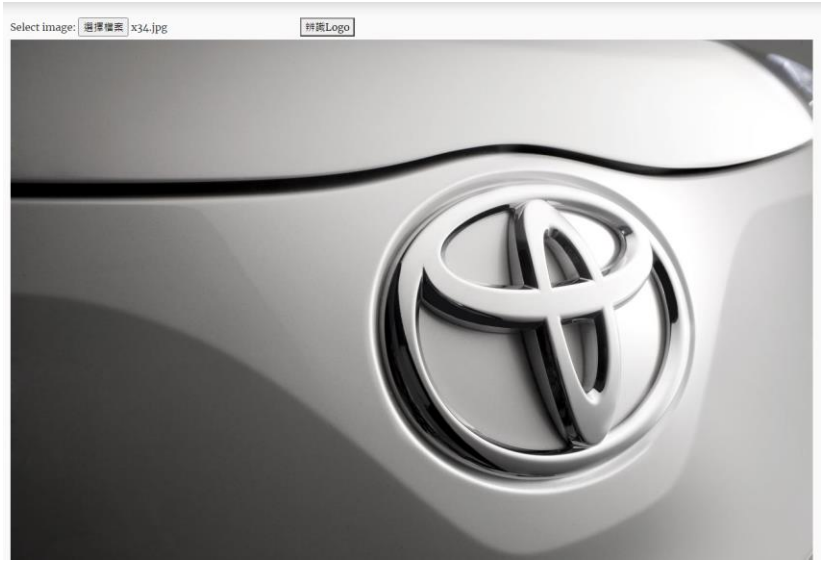

Figure 22 匯入圖片

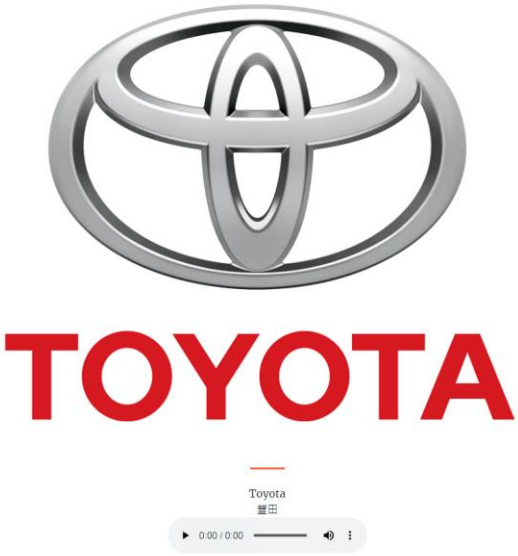

Figure 23 執行後介面

Figure 24 汽車展示介面

四、 結論及未來展望

4.1 整體改善

從模型結果可以得出，嘗試上述不同模型組合後，本研究在 ResNet 的模型表現最佳，訓練準確率能達 0.946，測試的準確率能達 0.9598。

而在實際測試中真的將網路或路邊隨手拍的照片上傳辨識王網站也可準確辨識。

4.2 未來展望

目前需要 Logo 較大的圖示較可成功辨識，若在輪胎上的 Logo 較少，模型較難找到圖片中 Logo 所在位置，因而會有誤判的情況發生，未來若有更多汽車 Logo 資料數據可供訓練，本網站的效益應該會大幅提升，不僅促使父母有更多與孩子互動的機會，也讓孩子可以正確學對發音，讓其彷彿真的有老師在教學一樣生動有趣。

並且在未來應用方面，網頁部分可以加入會員制，追蹤小朋友的喜好，進而推薦適合的汽車款式給小朋友。