汽車Logo
辨識（便是）王

Group 4：
110034547 邱韵婷
110034548 張瑋芩
110034568 陳彥碩

目錄

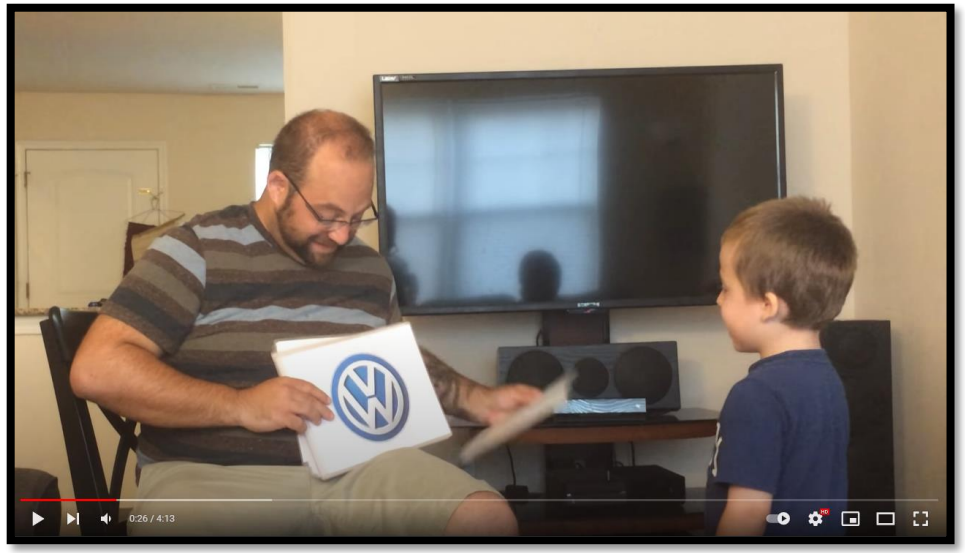
CONTENTS

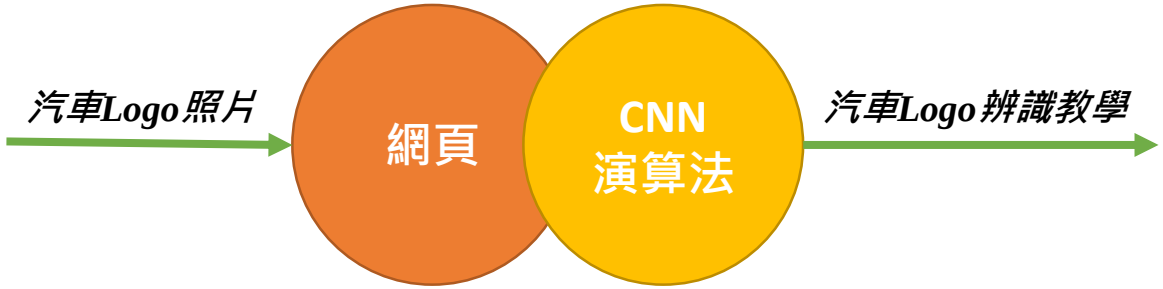
01

背 景 介 紹

1-1 背 景 介 紹

- 根據相關資料顯示70%的父母沒有足夠時間陪伴孩子成長
- 為了讓喜愛汽車的孩童能有更多元的學習方式
- 本組開發辨識王網站供孩童學習且能增加父母陪伴的時間





1-2 5W1H


便是王
網站
Who
喜愛汽車的孩童
What
供孩童學習
Why
父母工作繁忙
When
父母休假或假日
Where
任何地點
How
深度學習及網站

02

資料前處理

2-1 資 料 蒐 集

資料來源：Kaggle網站-Car Brand Logos

原始資料：8種汽車Logo

2-1 資料蒐集

Car
Hyundai
Lexus
Mazda
Mercedes
Opel
Skoda
Toyota
Volkswagen

+

上網蒐集Tesla Logo圖片

2-2 資 料 處 理

CAR	Train	Test
Hyundai	302	50
Lexus	301	50
Mazda	317	50
Mercedes	342	50
Opel	301	50
Skoda	314	50
Tesla	300	
Toyota	306	50
Volkswagen	330	50

資料清理
和分類

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
	6	2	2

2-3 資 料 擴 增

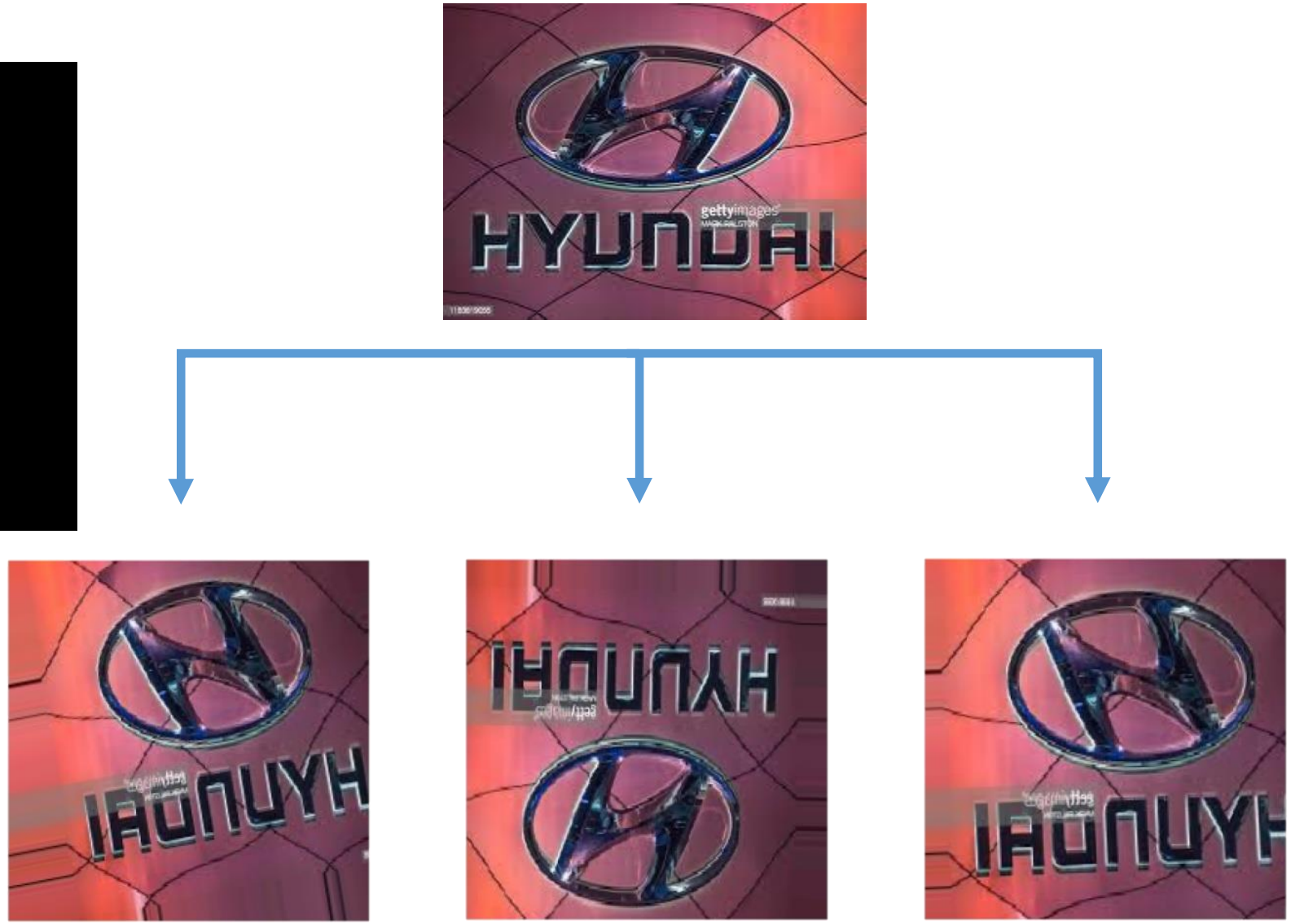
訓練集跟驗證集資料增加4倍

CAR	Train	Validation	Test
Hyundai	163	55	55
Lexus	147	49	49
Mazda	190	63	65
Mercedes	186	63	60
Opel	156	51	52
Skoda	172	57	58
Tesla	164	54	54
Toyota	187	62	63
Volkswagen	200	67	67

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

2-3 資料擴增

```
datagen = ImageDataGenerator(
    rotation_range = 10,
    zoom_range = 0.1,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    horizontal_flip = True,
    vertical_flip = True,
    shear_range = 0.1
    )
```



2-3 資 料 擴 增

```
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')

test_generator = test_datagen.flow_from_directory(test_data_dir,
                                                  target_size = (img_height, img_width),
                                                  batch_size = batch_size,
                                                  class_mode = "categorical")
```

03

模型訓練與績效

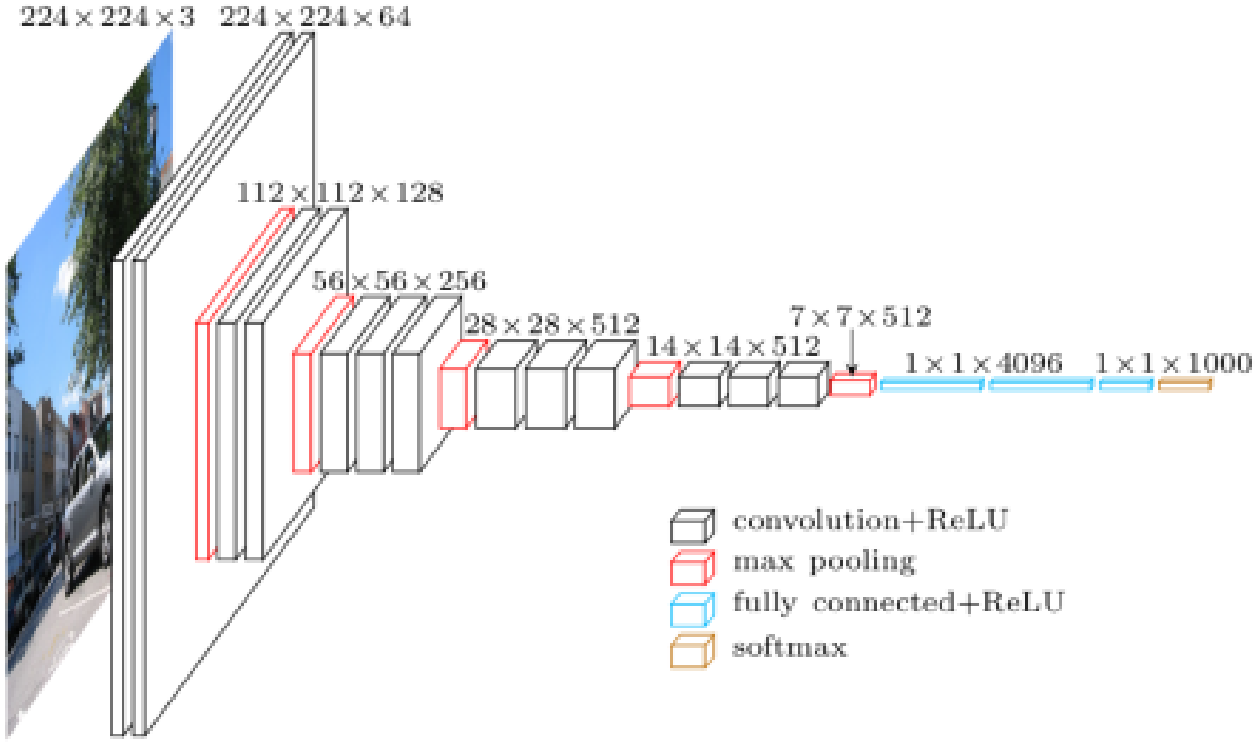
3-1 模 型 架 構

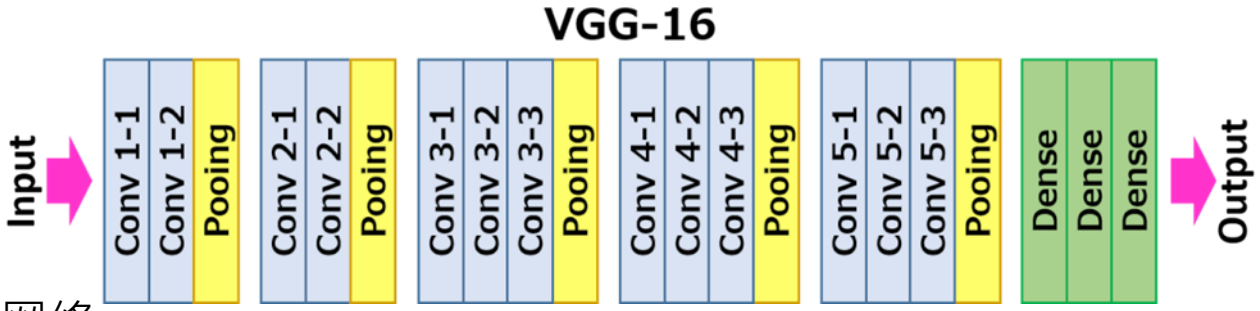
選定在各資料預測競賽中表現優良的四種CNN 模型架構作為主結構，分別是VGG16、VGG19、ResNet與DenseNet，再搭載本組所建之DNN模型。

Model	Size	Top-1 準確率	Top-5 準確率	參數數量	深度
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50V2	98 MB	0.772	0.938	25,613,800	-
DenseNet	33 MB	0.750	0.923	8,062,504	121

3-1 模 型 架 構

- VGG16

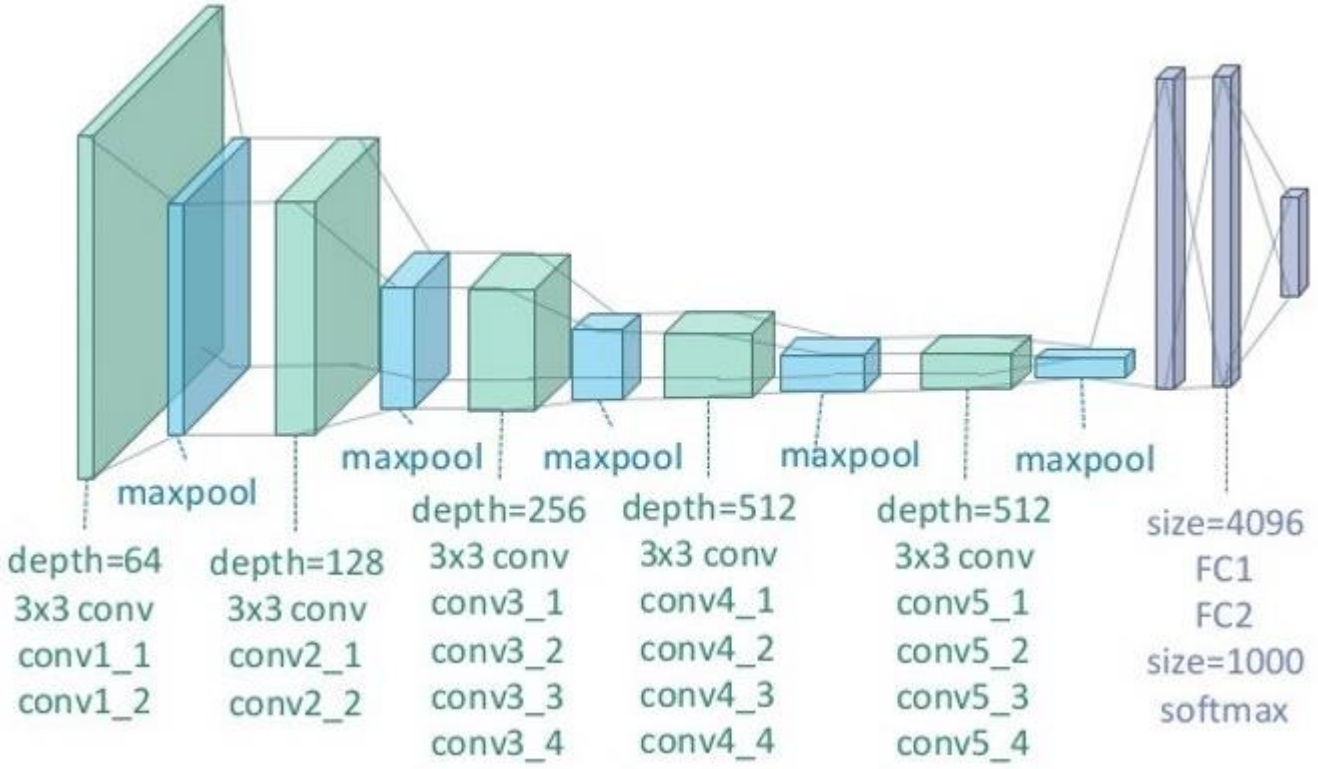




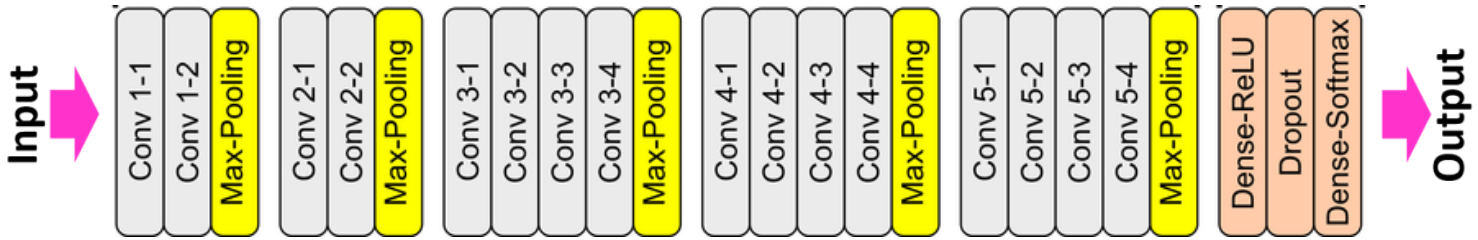
資料來源：VGG16 -用于分类和检测的卷积网络

3-1 模 型 架 構

- VGG19



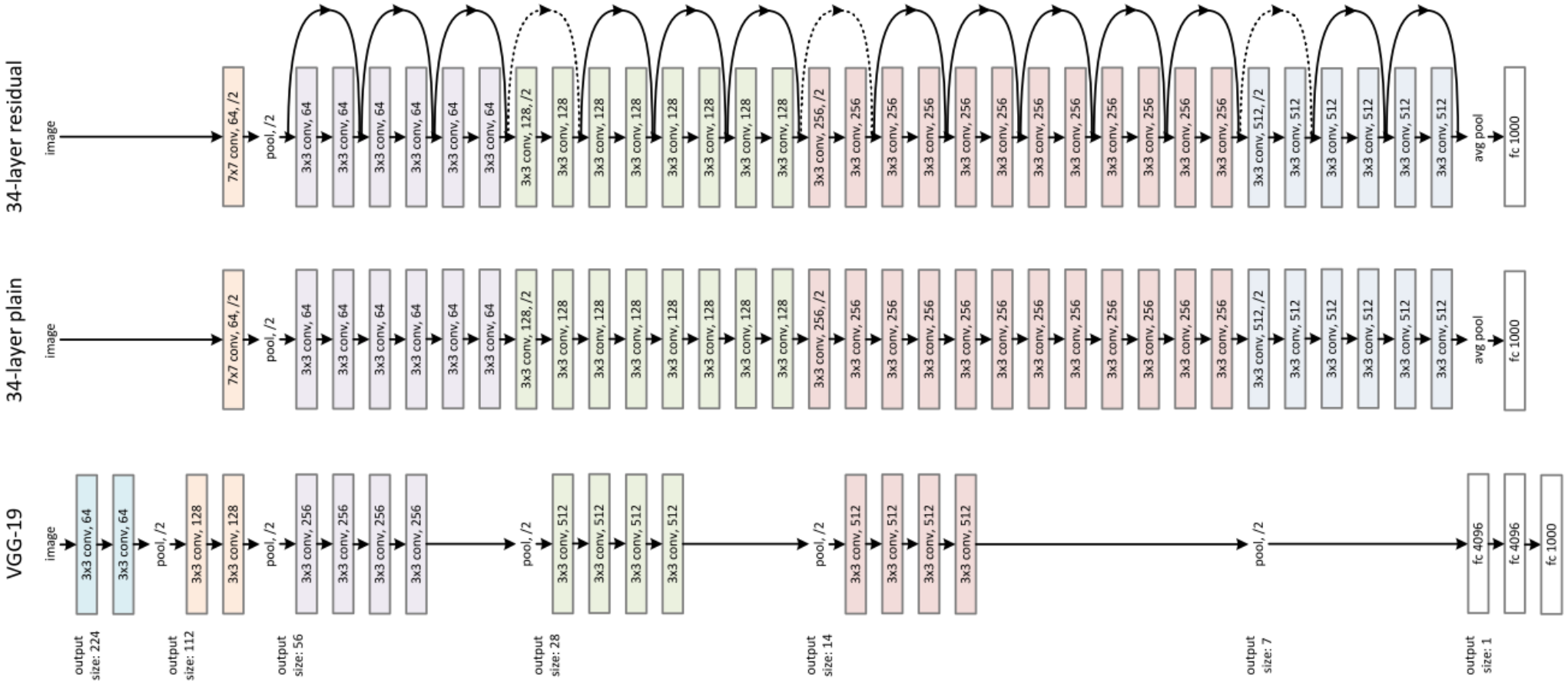
VGG - 19



資料來源：VGG16 -用于分类和检测的卷积网络

3-1 模 型 架 構

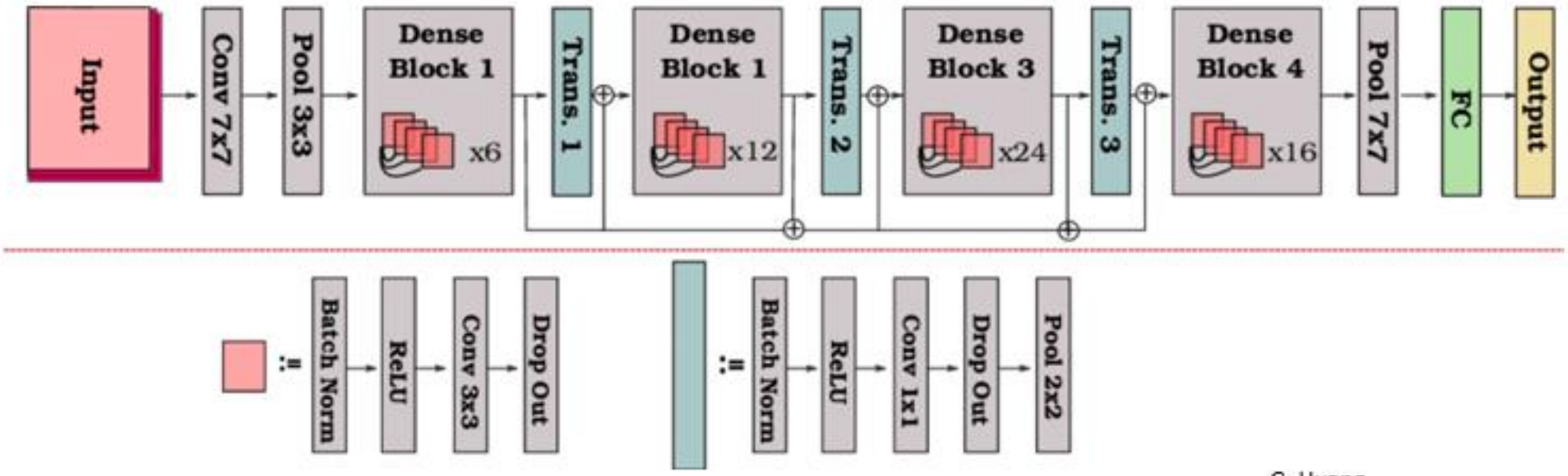
- ResNet



資料來源：Residual Leaning: 認識ResNet與他的冠名後繼者ResNeXt、ResNeSt

3-1 模 型 架 構

- DenseNet



G. Huang

3-1 模 型 架 構

- DNN模型

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')

```
_________________________________________________________________
flatten_1 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              18875392
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_6 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_7 (Dense)              (None, 9)                 297
=================================================================
Total params: 34,453,833
Trainable params: 19,739,145
Non-trainable params: 14,714,688
_________________________________________________________________
```

3-2 超參數調整

應用實驗設計得出所有結果，學習率的部分是使用Keras中的ReduceLROnPlateau，訓練過程中會以設定最小的值開始訓練，再根據訓練結果不斷調整。

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

參數	數值
Epochs	30
Batch size	16
Loss	Categorical_crossentropy

3-2 超參數調整

```
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])
```

3-2 超參數調整

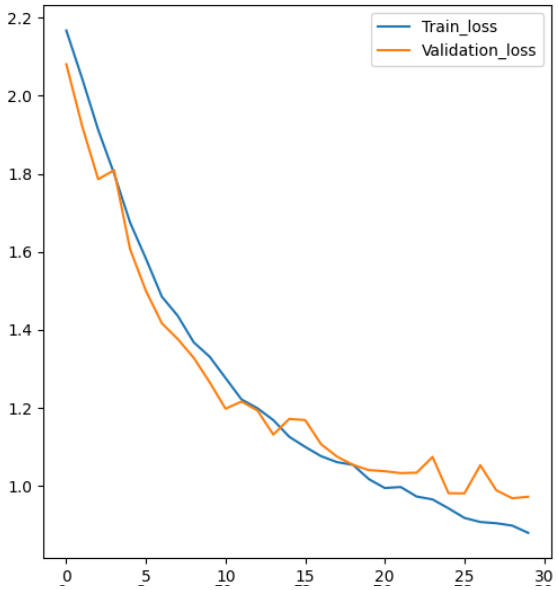
DOE	Model	Optimizer	Learning rate(Final)	Train Acc	Test Acc
1	VGG16	Adam	0.01~0.001(**0.001**)	0.5048	0.5516
2	VGG16	Adam	0.001~0.0001(**0.001**)	0.6898	0.8298
2.5	VGG16	Adam	0.0001~0.00001(**0.000025**)	0.8144	0.8718
3	VGG16	SGD	0.01~0.001(**0.01**)	0.6449	0.8049
4	VGG16	SGD	0.001~0.0001(**0.0005**)	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

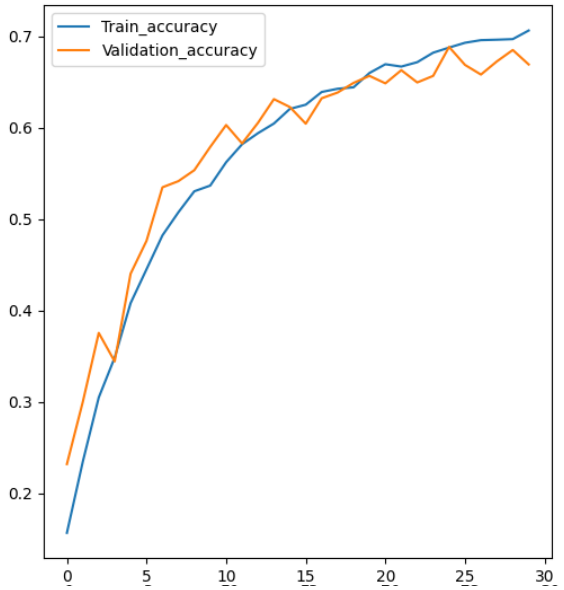
DOE	Model	Optimizer	Learning rate(Final)	Train Acc	Test Acc
13	ResNet152V2	Adam	0.01~0.001(0.0025)	0.8456	0.8846
14	ResNet152V2	Adam	0.001~0.0001(0.0001)	0.8756	0.8946
15	ResNet152V2	SGD	0.01~0.001(0.0025)	0.9649	0.9522
16	ResNet152V2	SGD	0.001~0.0001(0.000125)	0.946	0.9598
17	ResNet152V2	Adagrad	0.01~0.001(0.005)	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

3-3 模型績效比較

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

DOE 2.5

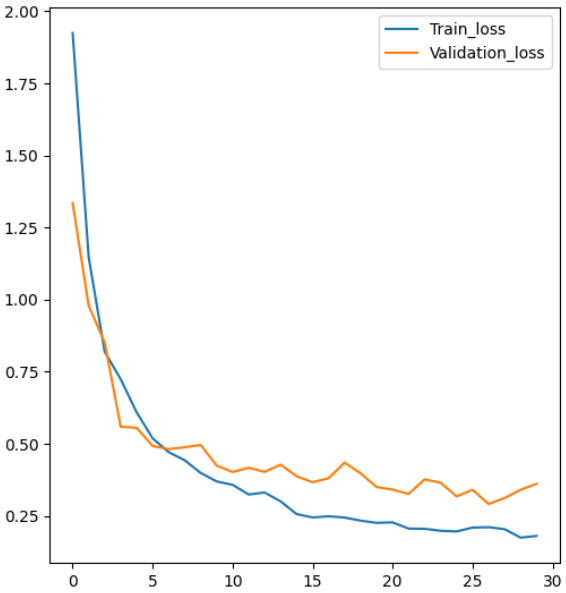


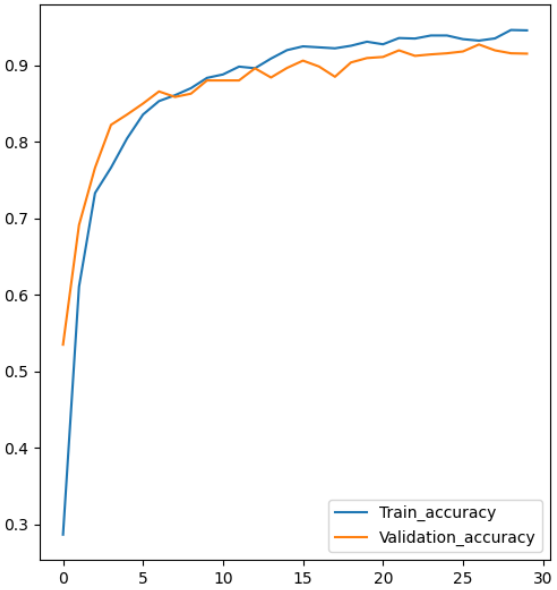


3-3 模型績效比較

DOE	Model	Optimizer	Learning rate(Final)	Train Acc	Test Acc
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

DOE 16

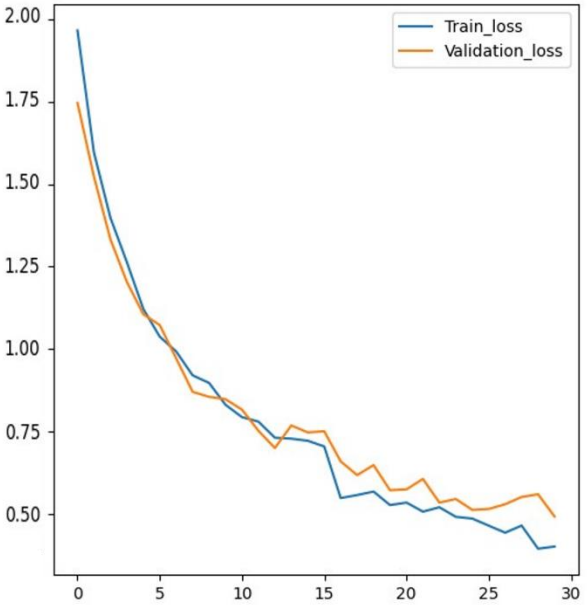


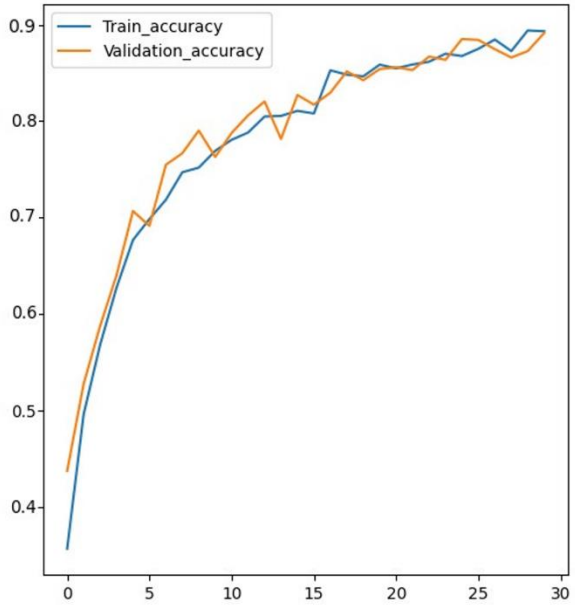


3-3 模型績效比較

DOE	Model	Optimizer	Learning rate(Final)	Train Acc	Test Acc
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

DOE 23





3-4 結果與討論

- 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，其測試準確率最高

Model	Training Accuracy/Loss	Validation Accuracy/Loss	Testing Accuracy/Loss
VGG16	0.8144/0.5438	0.783/0.8362	08718/0.4122
VGG19	0.701/0.5644	0.6943/1.052	0.8375/0.4521
ResNet	0.9649/0.1231	0.9021/0.3729	0.9598/0.2319
DenseNet	0.9501/0.271	0.8821/0.362	0.9541/0.2012

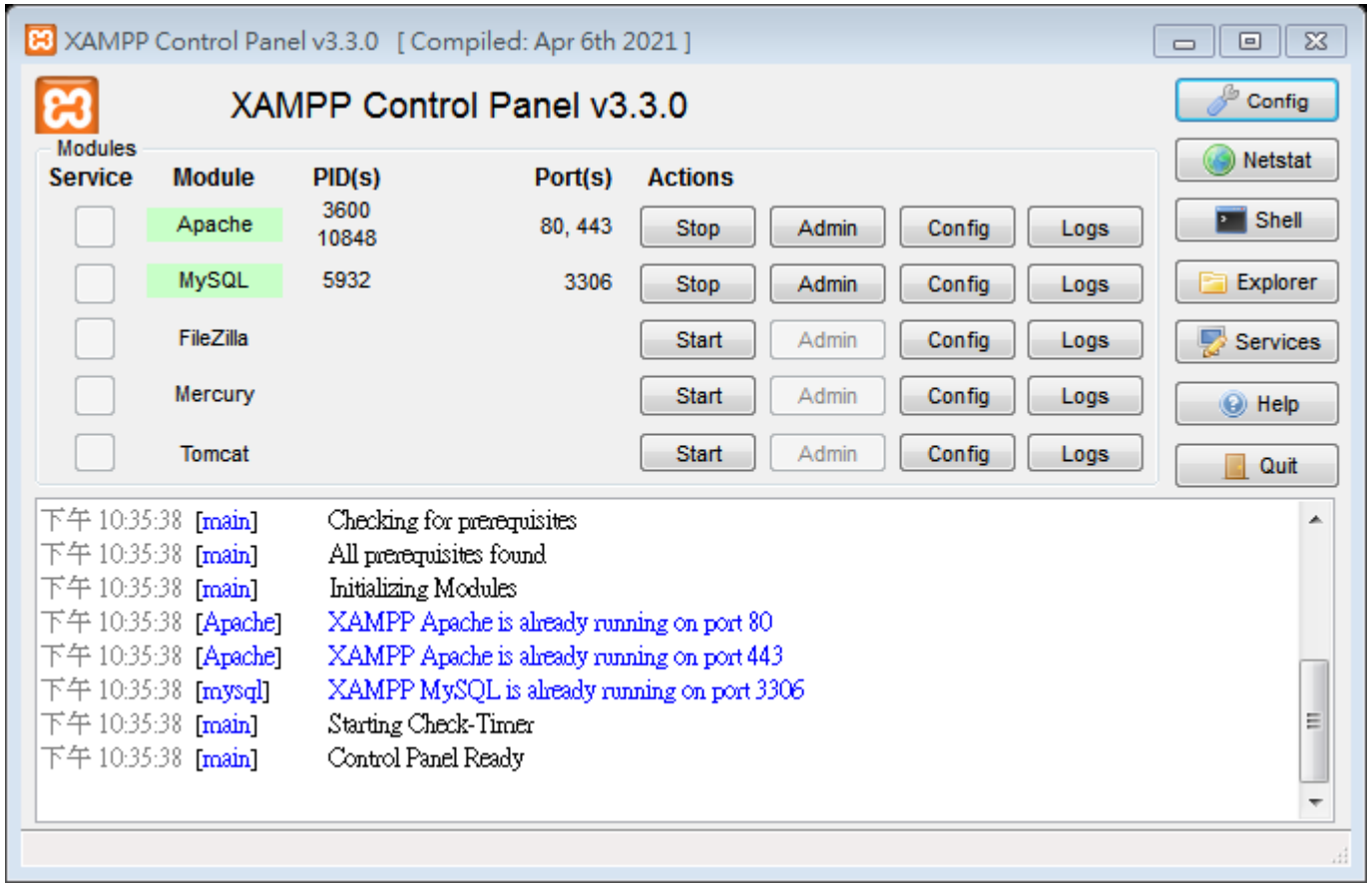
- 本組多個Pre-Train模型皆是較大的神經網路，訓練模型耗時許久

- 本模型的訓練結果餵入辨識王網站中，以測試所上傳照片可準確辨識

04

網 頁 系 統

4-1 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。

4-2 網站執行AI

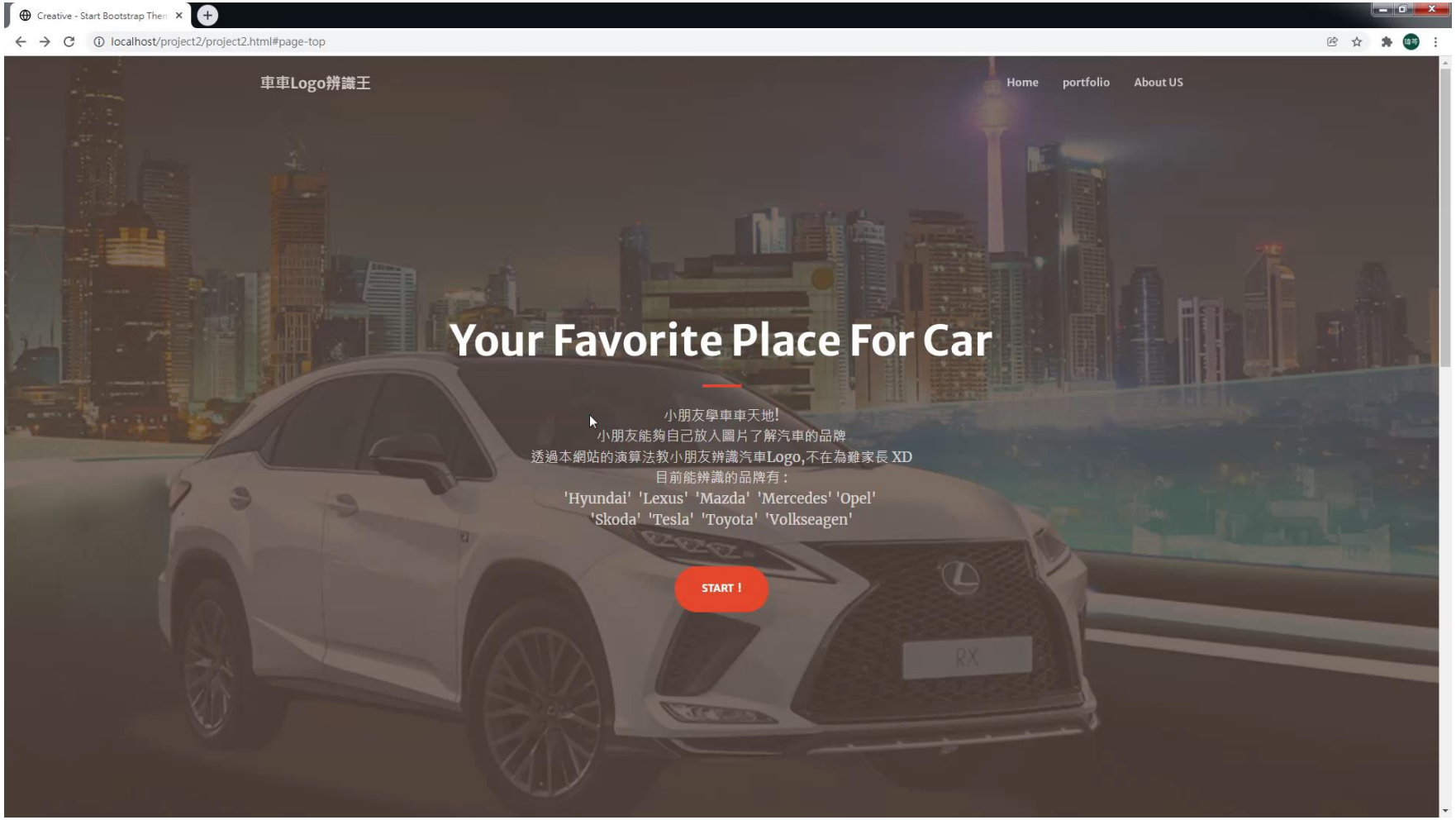
```php
# 檢查檔案是否上傳成功
if ($_FILES['img']['error'] === UPLOAD_ERR_OK){
  //echo '檔案名稱: ' . $_FILES['img']['name'] . '<br/>';
  //echo '檔案類型: ' . $_FILES['img']['type'] . '<br/>';
  //echo '檔案大小: ' . ($_FILES['img']['size'] / 1024) . ' KB<br/>';
  //echo '暫存名稱: ' . $_FILES['img']['tmp_name'] . '<br/>';

  # 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/>';
}
```

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

透過move_uploaded_file
將照片移至指定的檔案路徑

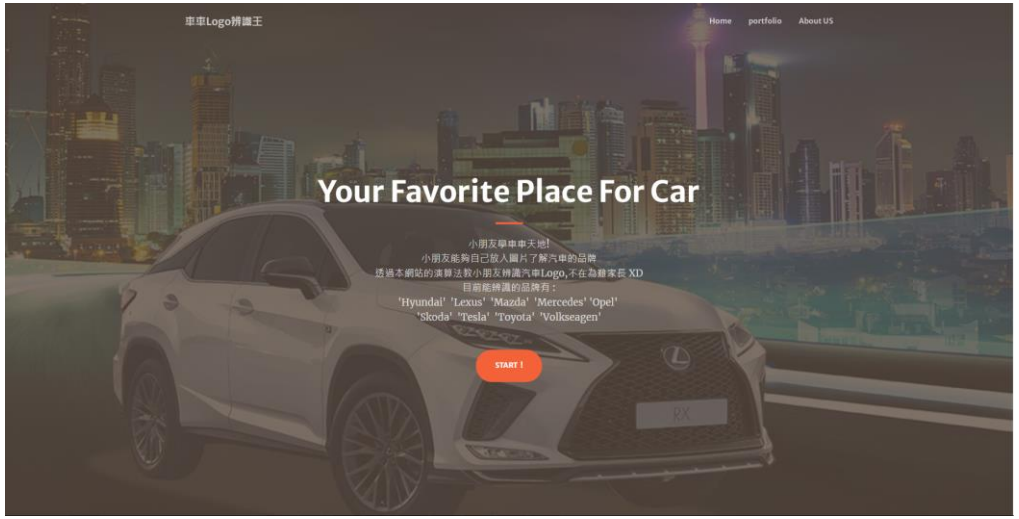
透過command指令將網站與Python演算法程式結合，並接收圖像辨識結果。

4-3 網頁介面



網站影片Demo

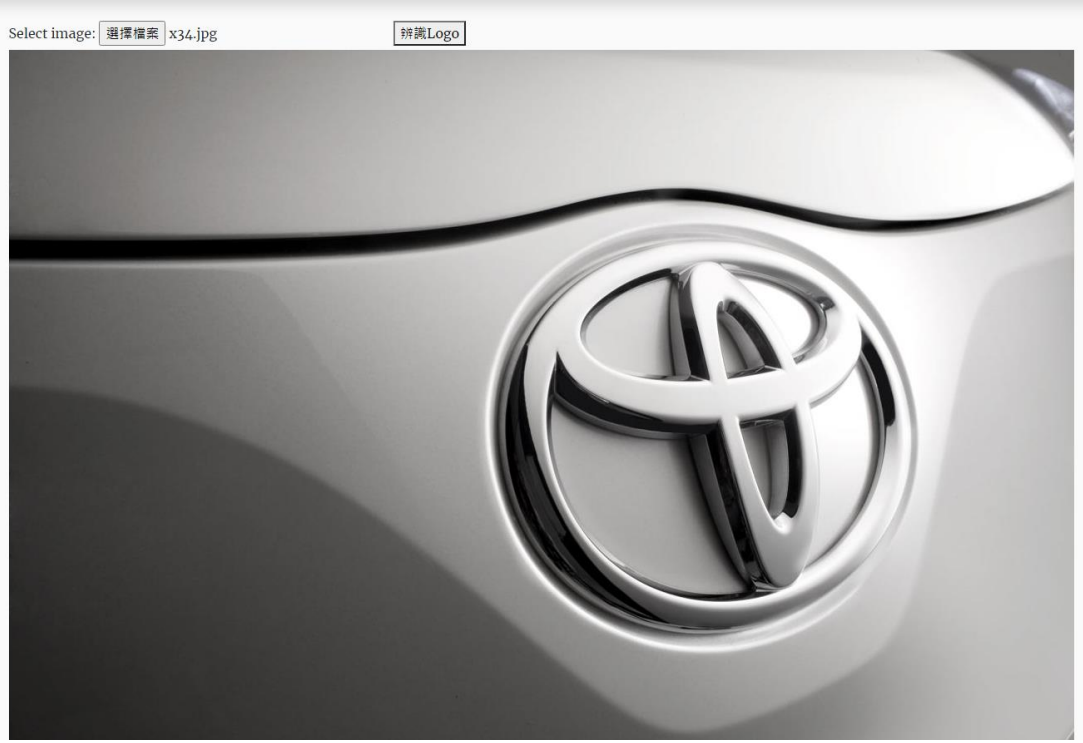
4-3 網頁介面



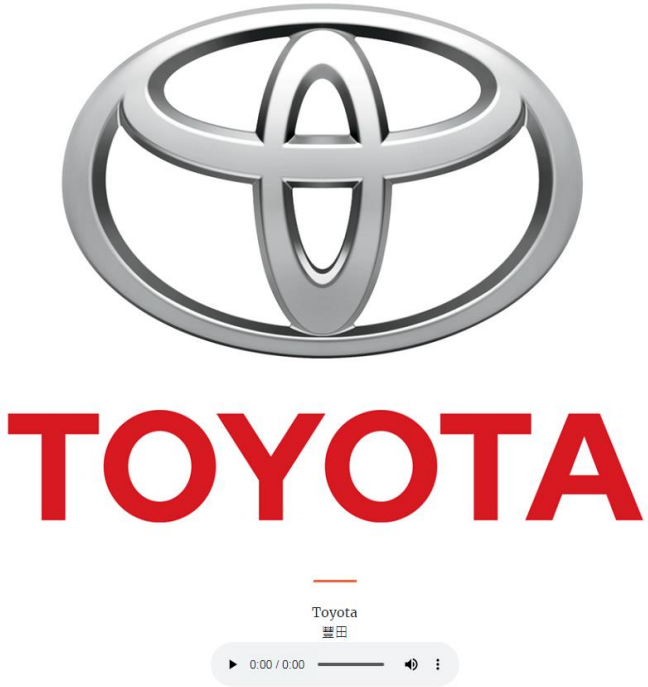
主頁面

汽車展示介面

4-3 網頁介面



匯入圖片



辨識後結果

05

結論與未來展望

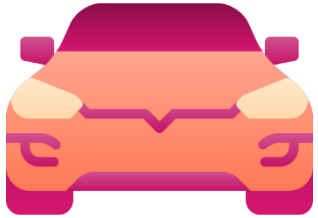
5-1 整 體 改 善

- 從模型結果可以得出，嘗試不同模型組合後，本研究在**ResNet**的模型中表現最佳，訓練準確率能達0.946，測試的準確率能達0.9598

- 本模型的訓練結果可以結合辨識王網站中，準確的辨識所上傳的logo照片。

5-2 未 來 展 望

- 需要Logo較大的圖示較可成功辨識
- 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
- 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




Thank you for listening