以VGG16進行鳥類辨識

110034550 黃喆志
指導教授：邱銘傳

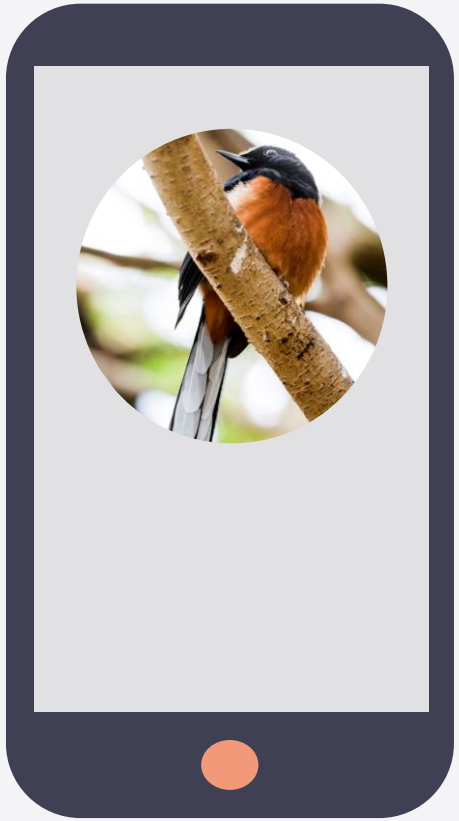
Outline

01 背景介紹

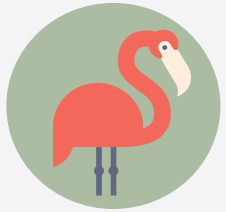
背景與動機

- 大多數民眾對於野鳥認識甚少，鳥類的搜索方式通常是靠粗淺的外觀描述，如毛色、體型大小……等進行關鍵字搜索，相當不便。

- 本研究以VGG16 建立鳥類識別器，幫助使用者直接從影像就可以辨識鳥類。



5W1H



Who

一般民眾、學術單位

What

鳥類品種辨識

When

需要辨識鳥類品種時

Where

鳥類出沒的地點

Why

鳥類種類繁多而且特徵
難以辨識

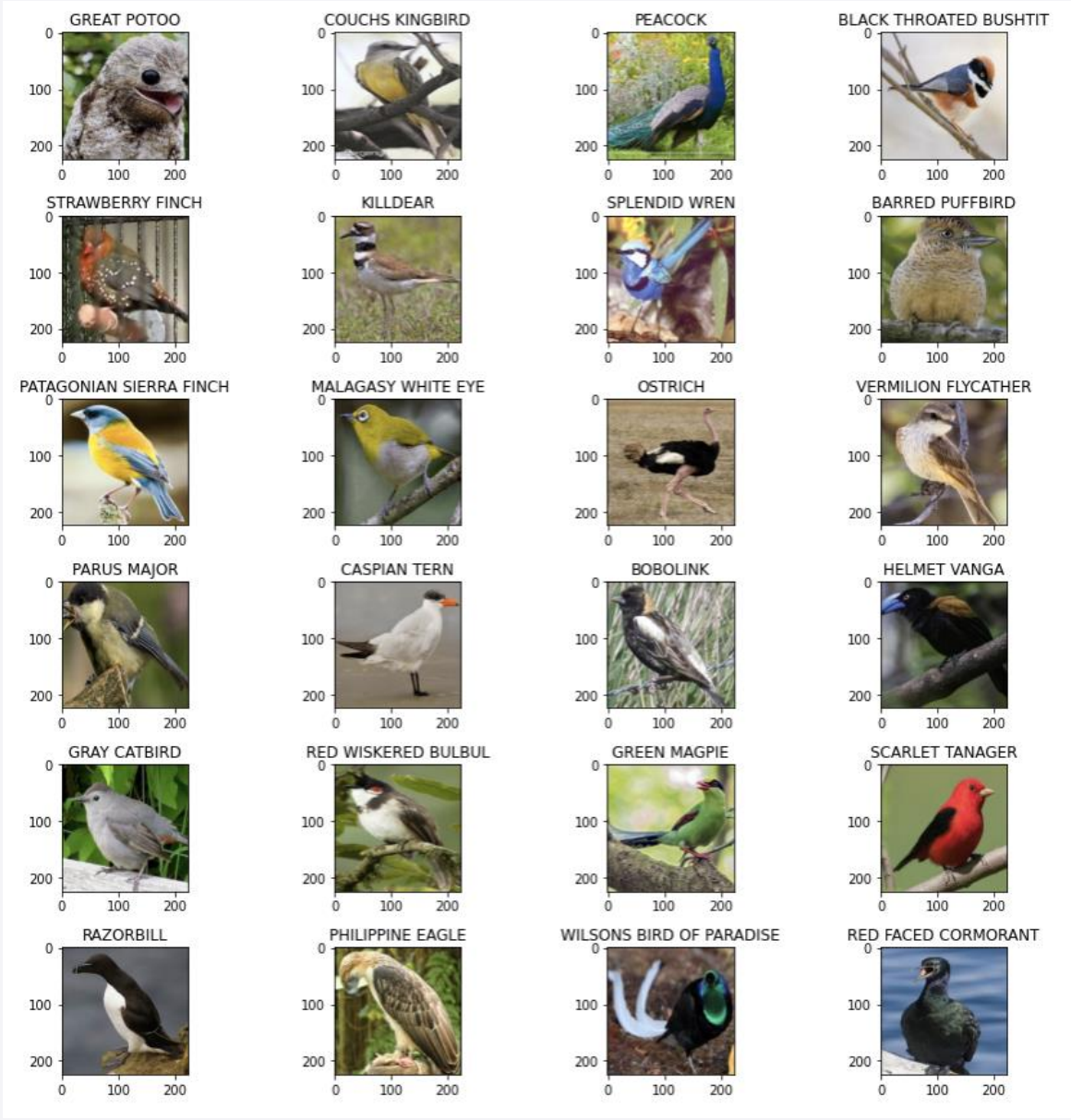
How

深度學習、機器學習、
資料分析

02 資料搜集與整理

資料來源

- 本資料集使用 Kaggle網站中的325 Bird Species Classification資料集，資料集包含325種鳥類照片，被劃分為訓練集477332張、驗證集以及測試集皆1625張。此外，所有圖像都是 jpg 格式的 224 X 224 X 3 彩色圖像



資料前處理

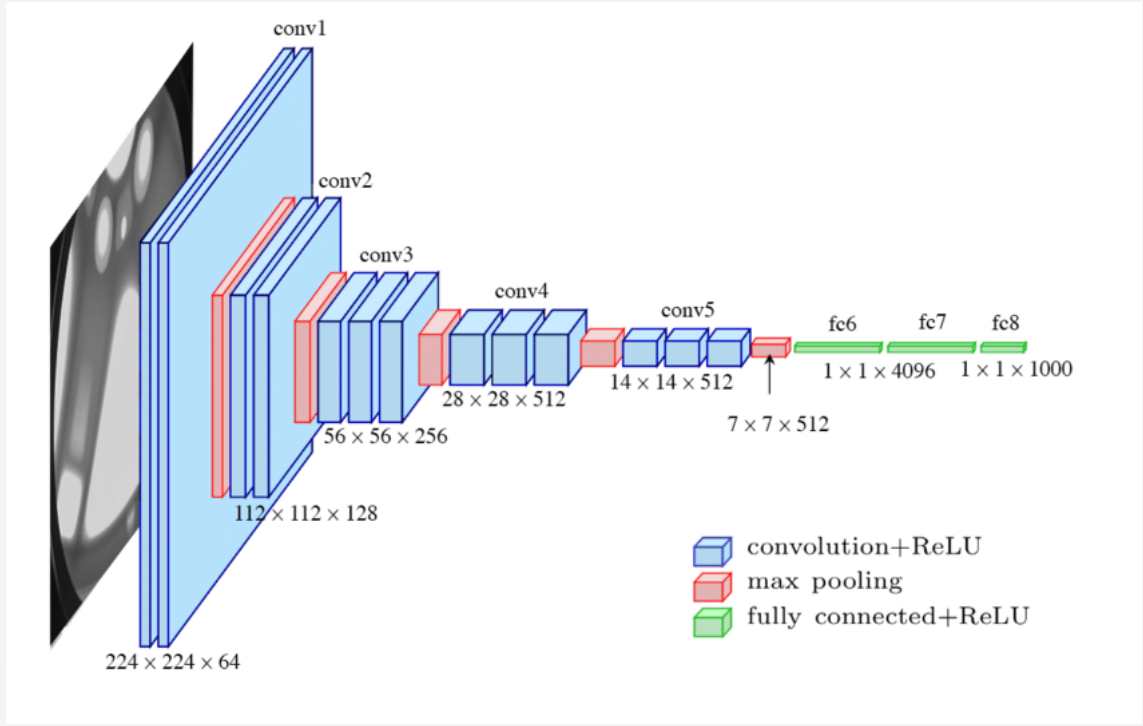
```
#Creating generator for Training DataSet
train_datagen = ImageDataGenerator(
        preprocessing_function=preprocess_input,
        shear_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True,rescale=1/255)
```

使用ImageDataGenerator來讓資料集之圖片進行標準化、裁切、縮放和水平翻轉，以利模型學習。

03 模型建構

VGG16 (1/2)

本研究使用之深度學習模型為VGG16。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，大量的圖片訓練提高準確率。共為16層(13個卷積層及3個全連接層)



VGG16 (2/2)

特點

- 卷積核大小(kernel size)統一為3x3
- 最大池化層(maxpool)統一為2x2
- 利用較小的卷積核來替代較大的卷積核

優/缺點

- VGG的架構簡單統一
- 證明較深的層數能提高效能
- 參數量龐大，計算資源需求高
- 訓練時間過長，難以調整參數

模型建立與訓練 (1/2)

Pre-trained model:

```
base_model = keras.applications.vgg16.VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.summary()
```

VGG16 模型從 ImageNet 上預訓練加載權重

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 224, 224, 3)]     0
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
flatten (Flatten)            (None, 25088)             0
fc1 (Dense)                  (None, 4096)              102764544
fc2 (Dense)                  (None, 4096)              16781312
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

模型建立與訓練 (2/2)

Freezing base layer:

```
[24] base_model.trainable = False
```

優點

- 減少訓練模型時間
- 僅對幾個層進行反向傳播和更新權重，節省計算時間。

Adding layers:

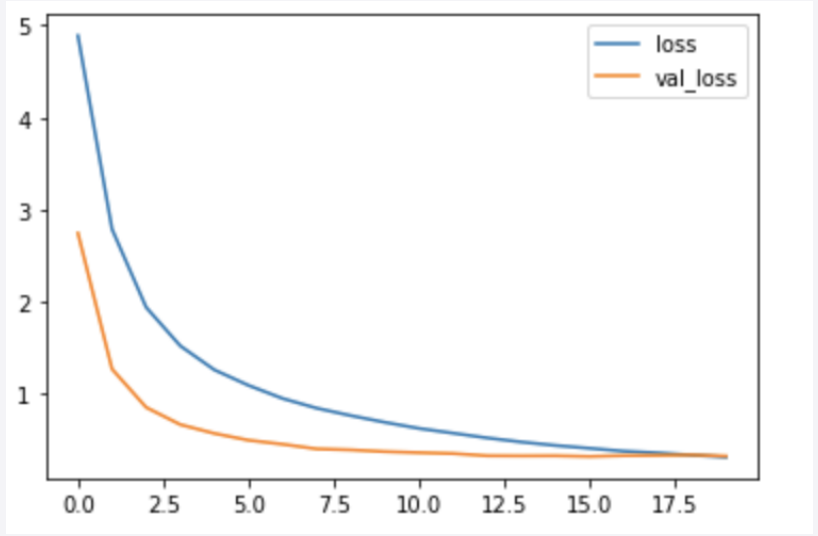
```
#Create new model on top
from keras.models import Sequential
from keras.layers import Dense,Flatten,Dropout
model=Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(2048,activation='relu',kernel_initializer='he_normal'))
model.add(Dropout(0.3))
model.add(Dense(2048,activation='relu',kernel_initializer='he_normal'))
model.add(Dropout(0.3))
model.add(Dense(325,activation='softmax',kernel_initializer='glorot_normal'))
model.summary()
```

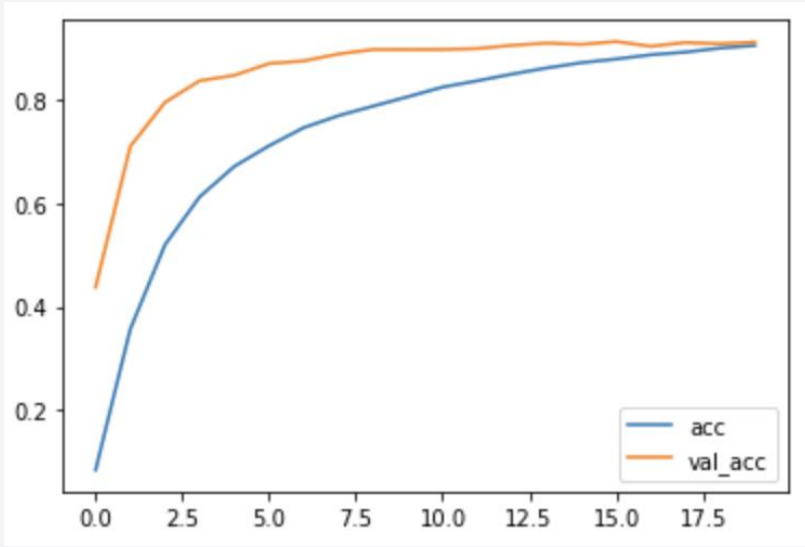
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_2 (Flatten)         (None, 25088)             0

 dense_6 (Dense)             (None, 2048)              51382272

 dropout_4 (Dropout)         (None, 2048)              0

 dense_7 (Dense)             (None, 2048)              4196352

 dropout_5 (Dropout)         (None, 2048)              0

 dense_8 (Dense)             (None, 325)               665925

=================================================================
Total params: 70,959,237
Trainable params: 56,244,549
Non-trainable params: 14,714,688
_________________________________________________________________
```

04 模型分析

確認Epoch次數

- 確認一個固定的Epoch，往後的實驗設計都將套用這個Epoch數值
- 當Epoch=25右圖呈現收斂的狀態，故實驗設計Epoch都將設為25。





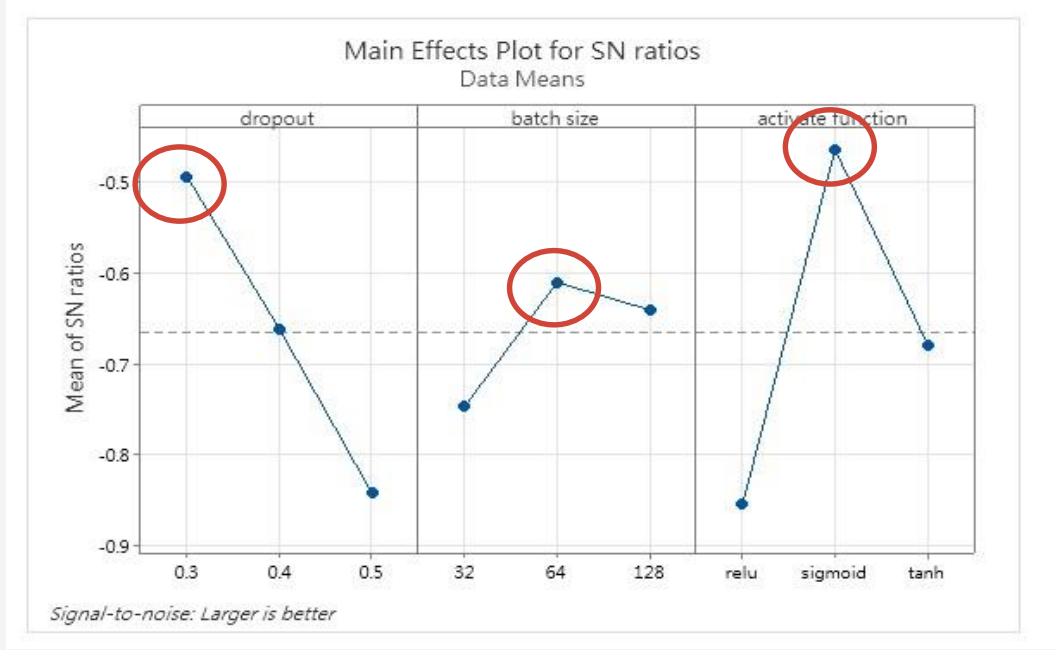
參數優化(1/3)

Factor\Level	Level1	Level2	Level3
Dropout	0.3	0.4	0.5
Batch Size	32	64	128
Activate Function	Relu	Sigmoid	Tanh

參數優化(2/3)

	Dropout	Batch Size	Activate function	Train acc	Test acc
1	0.3	32	Relu	0.9242	0.9206
2	0.3	64	Sigmoid	0.9847	0.9348
3	0.3	128	Tanh	0.9876	0.9225
4	0.4	32	Sigmoid	0.9748	0.9212
5	0.4	64	Tanh	0.9487	0.9274
6	0.4	128	Relu	0.8732	0.9206
7	0.5	32	Tanh	0.8535	0.9194
8	0.5	64	Relu	0.8874	0.9145
9	0.5	128	Sigmoid	0.9455	0.9311

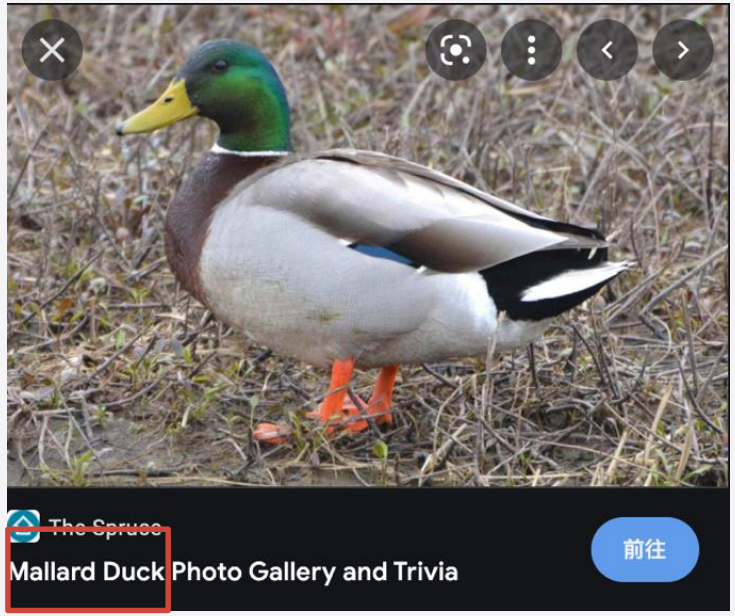
參數優化(3/3)



	Dropout	Batch Size	Activate function	Train acc	Test acc
最佳組合	0.3	64	Sigmoid	0.9868	0.9354

05 結果與討論

泛化性測試




```
from tensorflow.keras.preprocessing import image
from skimage import transform
img_width,img_height=224,224
img=image.load_img('/content/drive/MyDrive/mallard1-59511cd33df78cae81214312.jpeg')
img=image.img_to_array(img)
img=transform.resize(img,(224,224,3))
img=np.expand_dims(img,axis=0)
ans=model.predict(img)
print(ans.shape)
print(type(ans))
```

```
(1, 325)
<class 'numpy.ndarray'>
```

```
[22] print(np.argmax(ans))
```

```
188
```

187	MALEO
188	MALLARD DUCK
189	MANDRIN DUCK

結論與未來展望

- 本研究透過VGG16對325種鳥類進行分析，並得到良好的準確度得以見得VGG16優良的訓練成效
- 透過實驗設計L9直交表並使用統計軟體統計出最佳組合，最後本研究用資料集以外的圖片進行辨識，且成功識別。
- 未來希望可以使用特有種鳥類的資料集進行訓練，讓本模型可以支援本土特有種鳥類的辨識。

Thank you for listening