Intelligent Integration of Enterprise

Project 3

植物學家—植物照片辨識分類模型之建構與驗證

108034545 黃唯真

指導教授：邱銘傳 教授

目錄

一、研究主題說明 2

二、資料分析與前處理 2

1. 資料來源與結構分析 2
2. 資料前處理 4

三、預測模型架構 6

1. 模型架構與應用方法論 6
2. 參數設定與模型訓練過程.....7

四、模型效度驗證 9

五、小結與未來展望 10

一、 研究主題說明

植物學家時常需要至野外拍攝記錄各種類新舊品種的植物，作為後續研究分析的數據資訊，但植物學家外出拍攝後回到基地，需要將拍攝的照片首先進行讀取，存到電腦中後，再進行手動的操作去分類各植物的品種，整理照片，成千上萬的照片分類會耗費龐大的作業時間。本團隊將自己定位為提供照片分類技術的企業，使用深度學習建模，使照片能智能化的自動辨識並進行植物品種的分類歸檔，如此可以大大減少植物學家的負擔與時間。植物學家只需連接電腦等待照片上傳後，系統便會自動做完後續的分類歸檔。植物學家就可直接從分類歸檔好的資料集中取用需要研究的照片資料，進行後續的研究作業。

針對此研究主題，5W1H 分析說明如下表：

表一、5W1H 分析

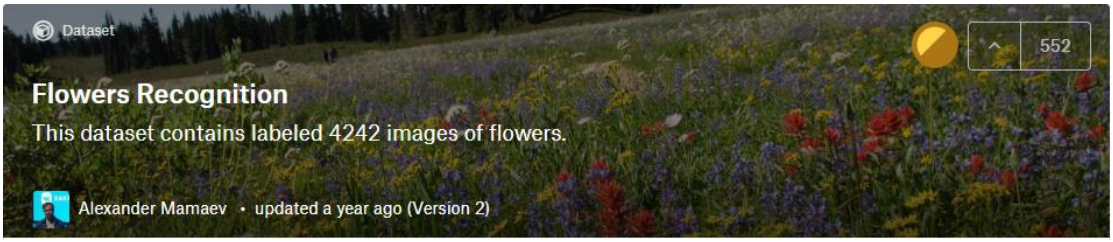
項目	內容
What	植物學者需要手動將搜集的照片進行分類歸檔，耗時龐大且不具效率，屬於時間浪費的行為
Where	研究基地，需上傳拍攝後的照片並進行分類歸檔作業
Who	植物學者、智能照片分類歸檔團隊
When	野外搜集植物照片返回基地整理照片時
Why	透過深度學習的模型，建構植物品種辨識並自行分類歸檔的功能，提供植物學者便捷有效率的技術，節省人員手動操作的時間與精力，改善植物學者的時間利用率
How	植物各類物種照片的資料前處理及 CNN 模型訓練

二、 資料分析與前處理

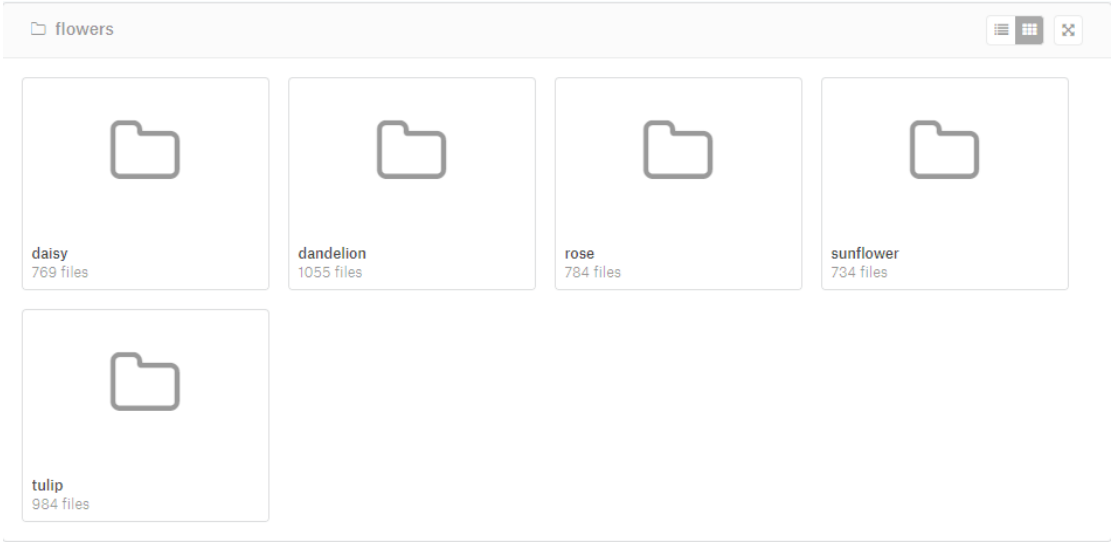
1. 資料來源與結構分析

團隊由 Kaggle 資料科學社群平台網站(圖一)獲取雛菊、蒲公英、玫瑰、向日葵與鬱金香五種植物品種的各式照片資料，共 4323 筆照片，資料檔案分類格式如圖二所示。其中雛菊 769 筆，蒲公英 1052 筆，玫瑰 784 筆，向日葵 734 筆，鬱金香 984 筆。

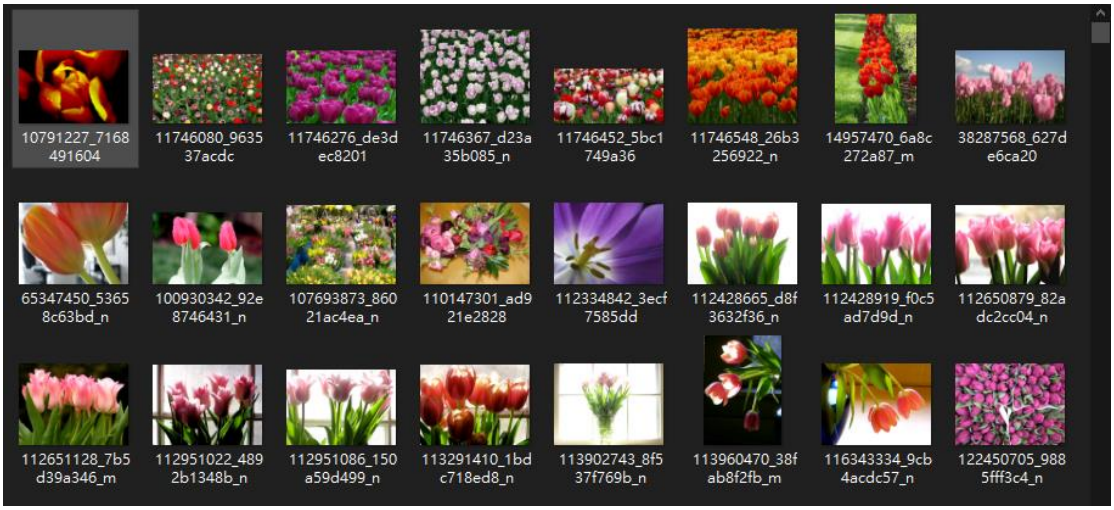
每個品種的植物資料夾下有數量不等的該品種植物各種角度、時期、光線、遠近、環境下的照片，以鬱金香照片為例，如圖三所示。



圖一、Kaggle 資料科學社群平台網站



圖二、植物品種檔案分類資料



圖三、鬱金香資料夾照片資料

結合以上照片的資料來源，我們將透過資料前處理並應用 CNN 的深度學習方法來訓練模型，進行植物品種的辨識分類，並期望最後準確率有超過八成的表現。

2. 資料前處理

(1) **Step1**：定義取得照片來源資料的函數程式，並以此程式讀取照片資料及標籤，將照片重新尺寸統一化，獲得每個類別植物照片張數

■ Code >

```
X=[]
Z=[]
IMG_SIZE=150
FLOWER_DAISY_DIR='/content/drive/My Drive/Colab Notebooks/flowers/flowers/daisy'
FLOWER_SUNFLOWER_DIR='/content/drive/My Drive/Colab Notebooks/flowers/flowers/sunflower'
FLOWER_TULIP_DIR='/content/drive/My Drive/Colab Notebooks/flowers/flowers/tulip'
FLOWER_DANDI_DIR='/content/drive/My Drive/Colab Notebooks/flowers/flowers/dandelion'
FLOWER_ROSE_DIR='/content/drive/My Drive/Colab Notebooks/flowers/flowers/rose'
```

```
def assign_label(img,flower_type):
    return flower_type
```

```
def make_train_data(flower_type,DIR):
    for img in tqdm(os.listdir(DIR)):
        label=assign_label(img,flower_type)
        path = os.path.join(DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))

        X.append(np.array(img))
        Z.append(str(label))
```

■ Output >

```
make_train_data('Daisy',FLOWER_DAISY_DIR)
print(len(X))
```

```
100%|██████████| 769/769 [00:04<00:00, 179.72it/s]769
```

```
make_train_data('Sunflower',FLOWER_SUNFLOWER_DIR)
print(len(X))
```

```
100%|██████████| 734/734 [00:04<00:00, 157.44it/s]1503
```

```
make_train_data('Tulip',FLOWER_TULIP_DIR)
print(len(X))
```

```
100%|██████████| 988/988 [00:05<00:00, 169.58it/s]2491
```

```
make_train_data('Dandelion',FLOWER_DANDI_DIR)
print(len(X))
```

```
100%|██████████| 1052/1052 [00:06<00:00, 170.99it/s]3543
```

```
make_train_data('Rose',FLOWER_ROSE_DIR)
print(len(X))
```

```
100%|██████████| 784/784 [00:04<00:00, 177.97it/s]4327
```

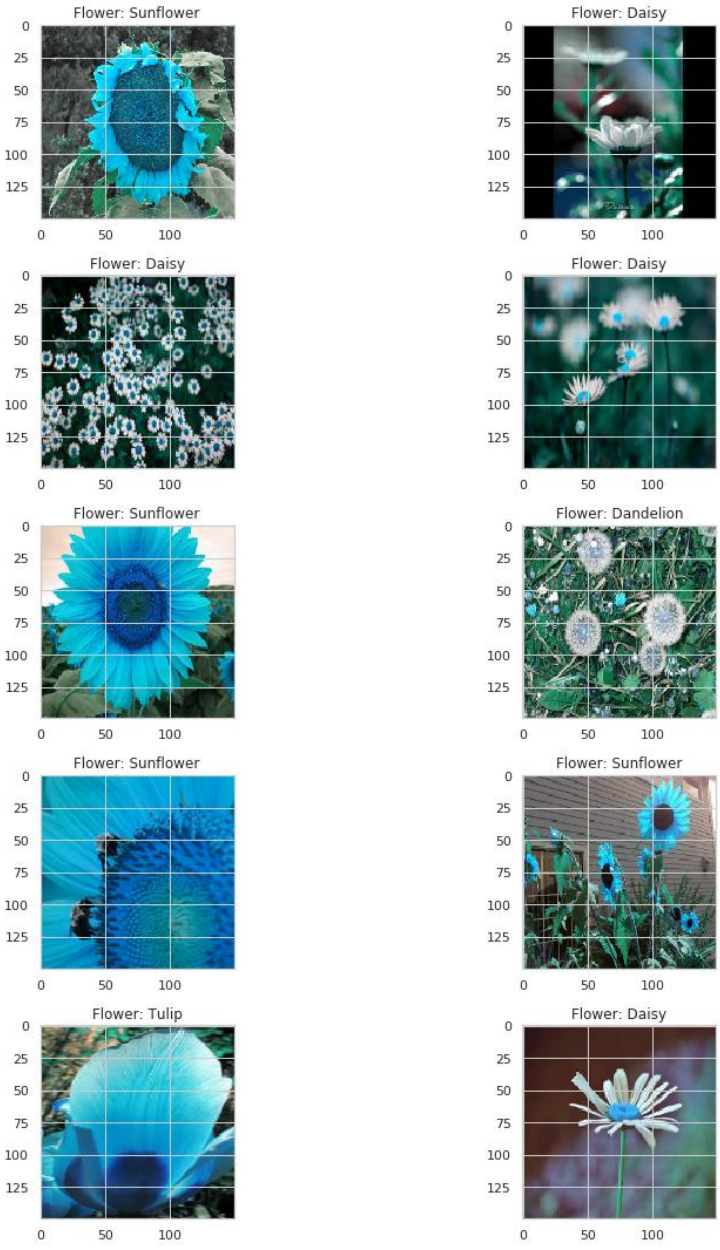
(2) Step2：隨機印出部分讀取後的照片資料，如下圖

■ Code >

```
fig,ax=plt.subplots(5,2)
fig.set_size_inches(15,15)
for i in range(5):
    for j in range (2):
        l=rn.randint(0,len(Z))
        ax[i,j].imshow(X[l])
        ax[i,j].set_title('Flower: '+Z[l])

plt.tight_layout()
```

■ Output >



(3) Step3：數據修正與調整

1. Label Encoding the Y array (i.e. Daisy->0, Rose->1 etc...) & then One Hot Encoding. 將植物品種數據化編碼從 0 開始，以此類推，灰階的圖片數值為 0~255 之間，將它縮放到 0~1 之間，並對類別資料做 onehot-encoding 處理
2. Splitting into Training and Validation Sets. 將照片資料分為訓練與驗證組
3. Setting the Random Seeds.

■ Code >

```
le=LabelEncoder()
Y=le.fit_transform(Z)
Y=to_categorical(Y,5)
X=np.array(X)
X=X/255
```

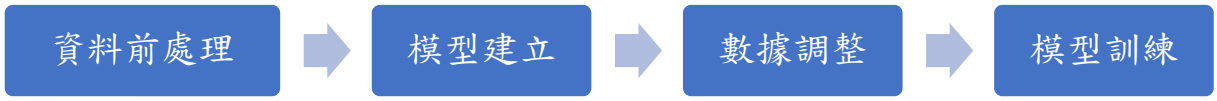
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,random_state=42)
```

```
np.random.seed(42)
rn.seed(42)
tf.set_random_seed(42)
```

三、 模型架構

1. 模型架構與應用方法論

模型建構步驟如下：



本模型使用 CNN (Convolutional Neural Network，卷積神經網絡)演算法建構，此演算法對影像識別方面的威力非常強大，許多影樣辨識的模型也都是以 CNN 的架構為基礎去做延伸。下段將對模型訓練及參數調整做進一步說明。

2. 參數設定與模型訓練過程

(1) 模型建立

以下將詳述模型建構過程。

Step 1：filter 為 32，kernel size 為(5,5), padding 為(same)，max pooling size 為(2,2)

Step 2：filter 為 64，kernel size 為(3,3), padding 為(same)，max pooling size 為(2,2)，步長(Strides)為(2,2)

Step 3：filter 為 96，kernel size 為(3,3), padding 為(same)，max pooling size 為(2,2), 步長(Strides)為(2,2)

Step 4：filter 為 96，kernel size 為(3,3), padding 為(same)，max pooling size 為(2,2), 步長(Strides)為(2,2)

Step 5：平坦化

```
# # modelling starts using a CNN.

model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu', input_shape = (150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))


model.add(Conv2D(filters =96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dense(5, activation = "softmax"))
```

(2) 數據調整

Using a LR Annealer

```
batch_size=128
epochs=50

from keras.callbacks import ReduceLROnPlateau
red_lr= ReduceLROnPlateau(monitor='val_acc',patience=3,verbose=1,factor=0.1)
```

Data Augmentation to prevent Overfitting

```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False)  # randomly flip images


datagen.fit(x_train)
```

Compiling the Keras Model & Summary

```
model.compile(optimizer=Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 150, 150, 32)      2432
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 75, 75, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 37, 37, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 37, 37, 96)        55392
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 18, 18, 96)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 18, 18, 96)        83040
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 9, 9, 96)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 7776)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               3981824
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 2565
=================================================================
Total params: 4,143,749
Trainable params: 4,143,749
Non-trainable params: 0
```

(3) 模型訓練

在訓練過程中，設定模型 fitting 跑 50 個 epoch。

```
History = model.fit_generator(datagen.flow(x_train,y_train, batch_size=batch_size),
                              epochs = epochs, validation_data = (x_test,y_test),
                              verbose = 1, steps_per_epoch=x_train.shape[0] // batch_size)
```

```
Epoch 12/50
25/25 [==============================] - 176s 7s/step - loss: 0.7386 - acc: 0.7229 - val_loss: 0.7621 - val_acc: 0.7116
Epoch 13/50
25/25 [==============================] - 175s 7s/step - loss: 0.7204 - acc: 0.7199 - val_loss: 0.7729 - val_acc: 0.7079
Epoch 14/50
25/25 [==============================] - 172s 7s/step - loss: 0.6819 - acc: 0.7321 - val_loss: 0.7753 - val_acc: 0.7237
Epoch 15/50
25/25 [==============================] - 179s 7s/step - loss: 0.7408 - acc: 0.7162 - val_loss: 0.7631 - val_acc: 0.7116

Epoch 48/50
25/25 [==============================] - 178s 7s/step - loss: 0.3416 - acc: 0.8734 - val_loss: 0.6384 - val_acc: 0.8013
Epoch 49/50
25/25 [==============================] - 175s 7s/step - loss: 0.3126 - acc: 0.8746 - val_loss: 0.6493 - val_acc: 0.8059
Epoch 50/50
25/25 [==============================] - 175s 7s/step - loss: 0.3273 - acc: 0.8796 - val_loss: 0.6102 - val_acc: 0.8244
```

四、 模型校度驗證

取得測試集準確率：

```
score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)

score1, acc1 = model.evaluate(x_train, y_train,
                            batch_size=batch_size)
#print('Test score:', score)
print('Test accuracy:', acc)
print('Train accuracy:', acc1)
```

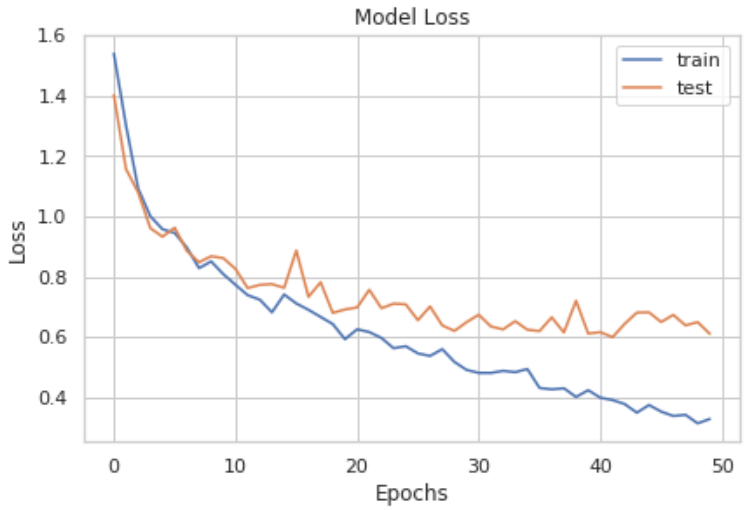
模型分類準確率，如下圖。整體平均預測準確率達 **82%**。

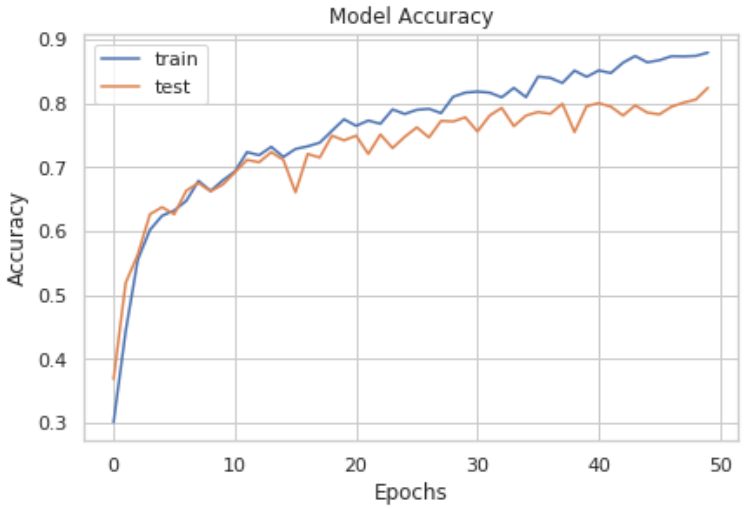
```
1082/1082 [==============================] - 15s 14ms/step
3245/3245 [==============================] - 46s 14ms/step
Test accuracy: 0.8243992600775911
Train accuracy: 0.9334360555434263
```

繪製 Loss 與 Accuracy 對 Epochs 的關係折線圖

```
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```

```
plt.plot(History.history['acc'])
plt.plot(History.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```





五、 小結與未來展望

由模型調整訓練的結果發現植物品種分類的準確率確實達到了八成以上，未來或許可以從再對模型的訓練做更多的調整有望使分類準確度達到九成五以上，更能減少植物學家發現分類歸檔錯誤後修正的時間。並且可以更廣泛的應用到其他領域，像是其他生物物種，海底動物，鳥類等等。或者擴張到影像的物種辨識裡，在影片中辨識出動植物的種類自行剪輯出影像或圖片。