花卉辨識

110034559　劉兆原

指導教授：邱銘傳　教授

大綱

01 研究背景

02 研究過程

03 模型建構_CNN

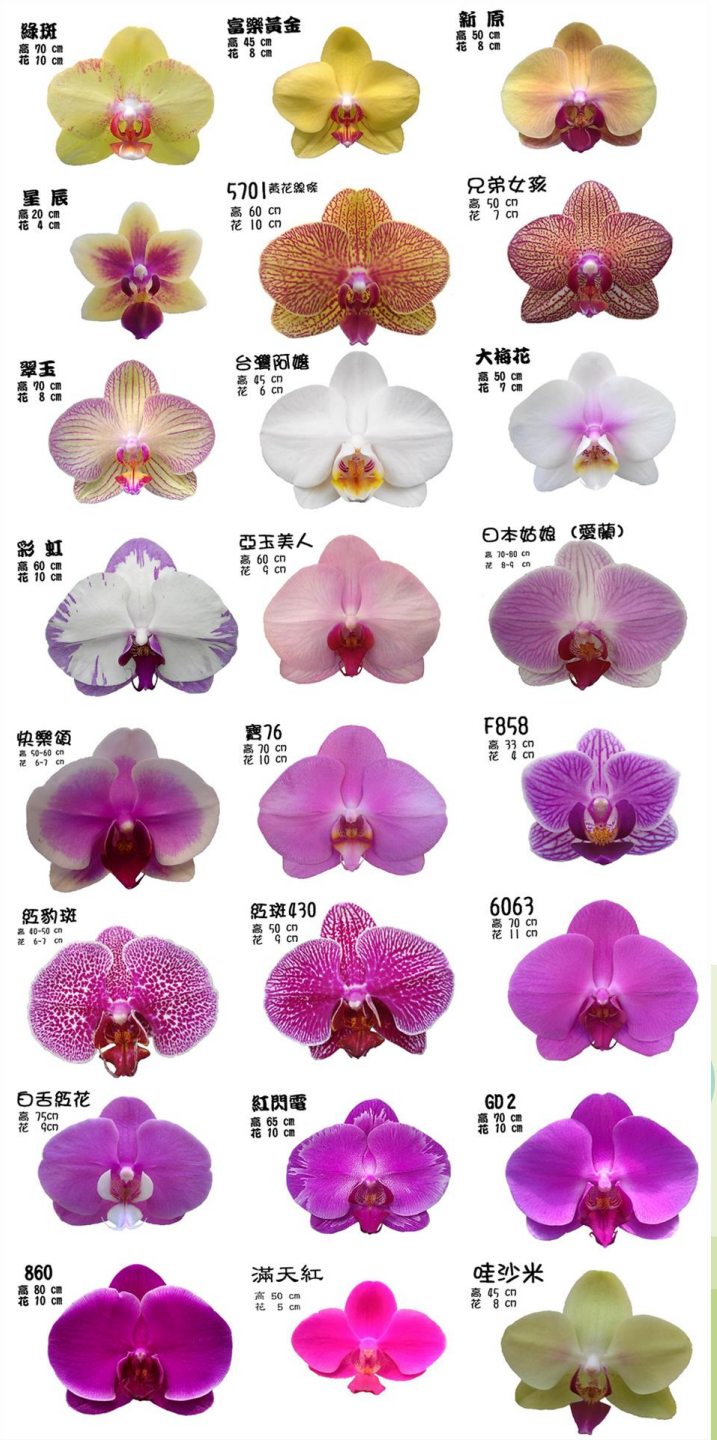
04 模型建構_VGG16

05 結論

1.研究背景

情境說明

- 農民辨識花卉種類費時費力



5W1H

WHAT
農民辨識花卉
種類費時費力

WHO
農民

WHERE
花卉種植場地

WHY
節省人力辨識
之時間與精力

WHEN
農民辨識花卉
種類時

HOW
以CNN與
VGG16訓練

2.研究過程

資料說明

雛菊　764筆

蒲公英　1052筆

玫瑰　784筆

向日葵　733筆

鬱金香　984筆

載入資料

- 於colab載入資料集
- 載入所需套件

```
[1]  from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ]  # import all required libraries for reading, analysing and visualizing data
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from tensorflow import keras

     import os
     import random
     import cv2
     from tqdm import tqdm

     import matplotlib.pyplot as plt
     import PIL
     import tensorflow as tf
     import numpy as np
     import os

     from tensorflow.keras.models import Model, Sequential
     from tensorflow.keras.layers import Dense, Flatten, Dropout
     from tensorflow.keras.applications import VGG16
     from tensorflow.keras.applications.vgg16 import preprocess_input, decode_predictions
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.optimizers import Adam, RMSprop
```

資料前處理1

- 標準化與正規化

```
[ ]  #  normalize  the  data
     X  =  X  /  255  #標準化，[0-255]調整至[0-1]

     #  reshape  the  data
     X  =  X.reshape(-1,  IMG_SIZE,  IMG_SIZE,  3)
     Y  =  Y.reshape(-1,  1)
     Y  =  keras.utils.to_categorical(Y,  5)  #正規化，one-hot  encoding
```

資料前處理2

- 資料分割

```
[ ]  X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2) #資料分割（訓練0.8，測試0.2）

     # Explore the dataset
     print("X_train shape:" + str(X_train.shape))
     print("Y_train shape:" + str(Y_train.shape))
     print("X_test shape:" + str(X_test.shape))
     print("Y_test shape:" + str(Y_test.shape))
```

```
X_train shape:(3453, 124, 124, 3)
Y_train shape:(3453, 5)
X_test shape:(864, 124, 124, 3)
Y_test shape:(864, 5)
```

3.模型建構
_CNN

CNN

```
[ ] from keras.models import Sequential
    from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten
    from keras.layers import Dense
```

```
model = Sequential(
        [
                Conv2D(filters = 32, kernel_size = (3, 3), padding = 'same', activation = 'relu', input_shape = (IMG_SIZE, IMG_SIZE, 3)),
                MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

                Conv2D(filters = 64, kernel_size = (3, 3), padding = 'same', activation = 'relu'),
                MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

                Conv2D(filters = 128, kernel_size = (3, 3), padding = 'same', activation = 'relu'),
                MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

                Conv2D(filters = 256, kernel_size = (3, 3), padding = 'same', activation = 'relu'),
                MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

                Conv2D(filters = 512, kernel_size = (3, 3), padding = 'same', activation = 'relu'),
                MaxPool2D(pool_size = (2, 2), strides = (2, 2)),

                Flatten(),
                Dense(1024, activation = 'relu'),
                Dropout(0.5),
                Dense(5, activation = 'softmax')
        ]
)
```

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

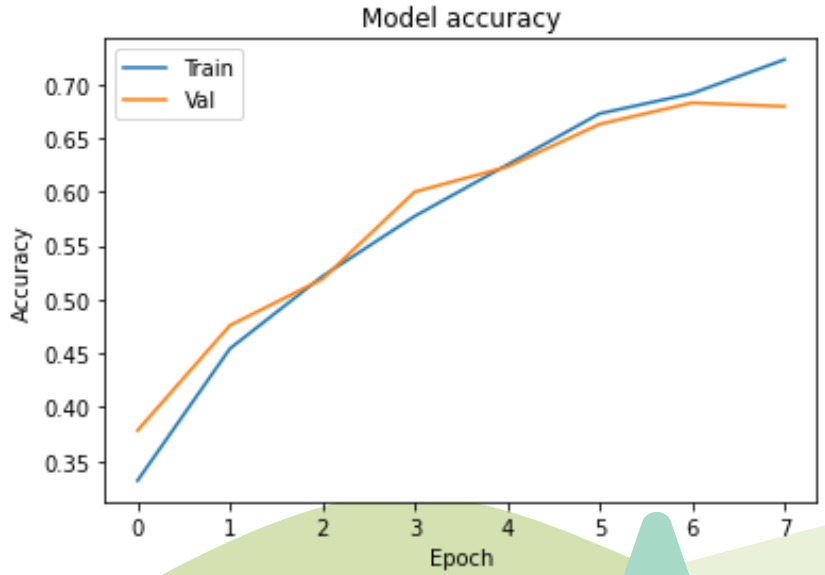
CNN_訓練過程

```
[ ] history = model.fit(X_train, Y_train, epochs = 8, validation_split = 0.2)
```

```
Epoch 1/8
76/76 [==============================] - 128s 2s/step - loss: 1.4554 - accuracy: 0.3328 - val_loss: 1.2128 - val_accuracy: 0.4430
Epoch 2/8
76/76 [==============================] - 126s 2s/step - loss: 1.1799 - accuracy: 0.4851 - val_loss: 1.0403 - val_accuracy: 0.5835
Epoch 3/8
76/76 [==============================] - 122s 2s/step - loss: 1.0481 - accuracy: 0.5832 - val_loss: 1.0110 - val_accuracy: 0.5636
Epoch 4/8
76/76 [==============================] - 122s 2s/step - loss: 0.9435 - accuracy: 0.6267 - val_loss: 0.8861 - val_accuracy: 0.6595
Epoch 5/8
76/76 [==============================] - 122s 2s/step - loss: 0.8559 - accuracy: 0.6701 - val_loss: 0.7879 - val_accuracy: 0.6992
Epoch 6/8
76/76 [==============================] - 122s 2s/step - loss: 0.7972 - accuracy: 0.6974 - val_loss: 0.7977 - val_accuracy: 0.6727
Epoch 7/8
76/76 [==============================] - 122s 2s/step - loss: 0.7520 - accuracy: 0.7210 - val_loss: 0.8312 - val_accuracy: 0.6876
Epoch 8/8
76/76 [==============================] - 125s 2s/step - loss: 0.6879 - accuracy: 0.7405 - val_loss: 0.7150 - val_accuracy: 0.7289
```

- 訓練準確率：0.7405

CNN_訓練結果




```
[28] # find the accuracy on test set
     test_loss, test_acc = model.evaluate(X_test, Y_test)
     print("Accuracy on test set is %f" %(test_acc * 100)  + "%")

     41/41 [==============================] - 180s 4s/step - loss: 0.8159 - accuracy: 0.8210
     Accuracy on test set is 82.098764%
```

- 測試準確率：0.8209

4.模型建構
_VGG16

VGG16

```
[ ]  model = VGG16(weights='imagenet', include_top=True)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_
553467904/553467096 [==============================] - 5s 0us/step
553476096/553467096 [==============================] - 5s 0us/step
```

```
[ ]  pre_trained_model = VGG16(input_shape = (IMG_SIZE, IMG_SIZE, 3), include_top = False, weights = 'imagenet'

     for layer in pre_trained_model.layers[:16]:
         layer.trainable = False

     model = Sequential(
         [
             pre_trained_model,
             Flatten(),
             Dense(5, activation = 'softmax')
         ]
     )
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_
58892288/58889256 [==============================] - 1s 0us/step
58900480/58889256 [==============================] - 1s 0us/step
```

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

VGG16_訓練過程

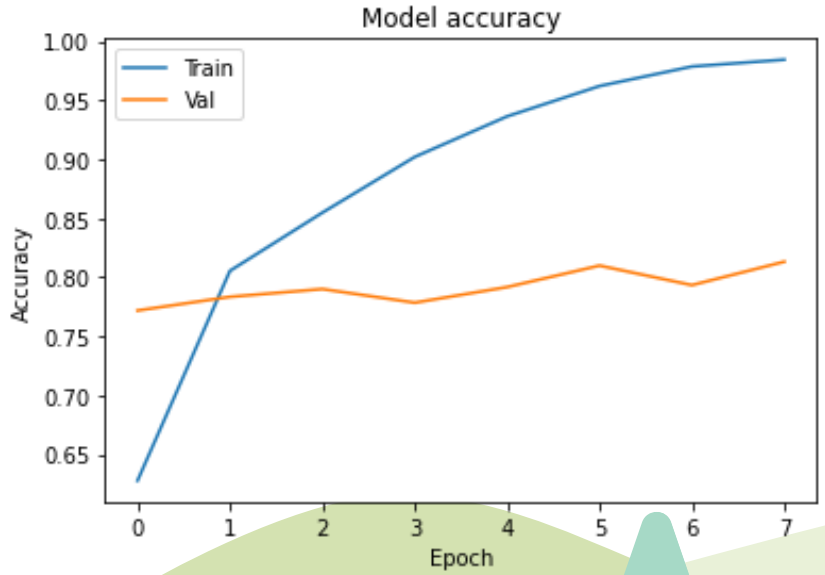
```
[25] model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
[26] history = model.fit(X_train, Y_train, epochs = 8, validation_split = 0.2)
```

```
Epoch 1/8
76/76 [==============================] - 459s 6s/step - loss: 0.9684 - accuracy: 0.6279 - val_loss: 0.6630 - val_accuracy: 0.7719
Epoch 2/8
76/76 [==============================] - 461s 6s/step - loss: 0.5328 - accuracy: 0.8055 - val_loss: 0.5755 - val_accuracy: 0.7835
Epoch 3/8
76/76 [==============================] - 457s 6s/step - loss: 0.3877 - accuracy: 0.8547 - val_loss: 0.5896 - val_accuracy: 0.7901
Epoch 4/8
76/76 [==============================] - 461s 6s/step - loss: 0.2590 - accuracy: 0.9019 - val_loss: 0.6776 - val_accuracy: 0.7785
Epoch 5/8
76/76 [==============================] - 472s 6s/step - loss: 0.1870 - accuracy: 0.9363 - val_loss: 0.7518 - val_accuracy: 0.7917
Epoch 6/8
76/76 [==============================] - 458s 6s/step - loss: 0.1180 - accuracy: 0.9619 - val_loss: 0.8933 - val_accuracy: 0.8099
Epoch 7/8
76/76 [==============================] - 458s 6s/step - loss: 0.0724 - accuracy: 0.9785 - val_loss: 0.8523 - val_accuracy: 0.7934
Epoch 8/8
76/76 [==============================] - 465s 6s/step - loss: 0.0588 - accuracy: 0.9843 - val_loss: 0.9178 - val_accuracy: 0.8132
```

- 訓練準確率：0.9843

VGG16_訓練結果



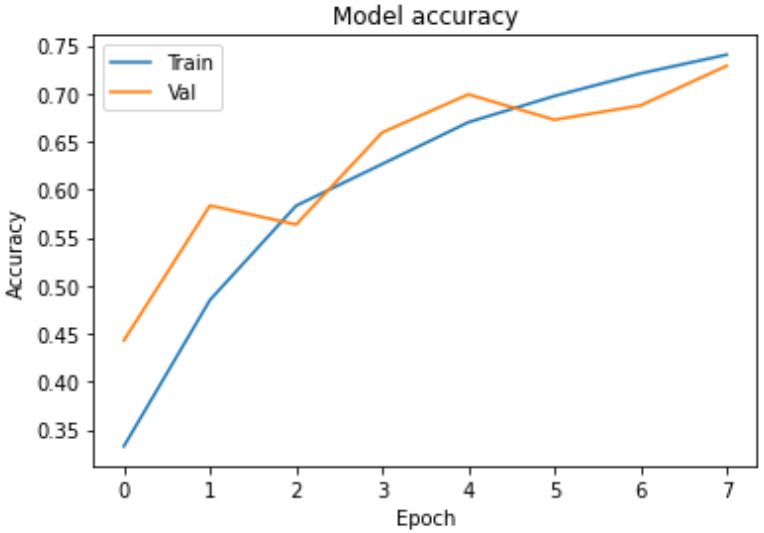

```
[29] # find the accuracy on test set
     test_loss, test_acc = model.evaluate(X_test, Y_test)
     print("Accuracy on test set is %f" %(test_acc * 100) + "%")

     41/41 [==============================] - 180s 4s/step - loss: 0.8159 - accuracy: 0.8210
     Accuracy on test set is 82.098764%
```

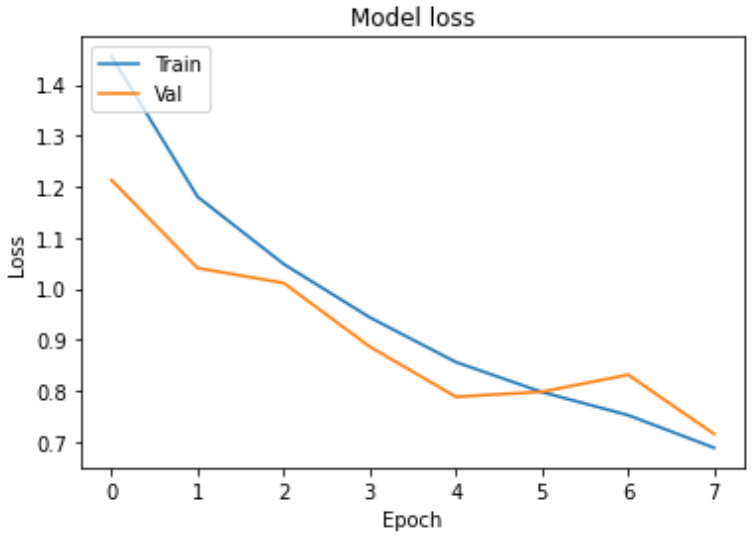
- 測試準確率：0.8209

5.結論

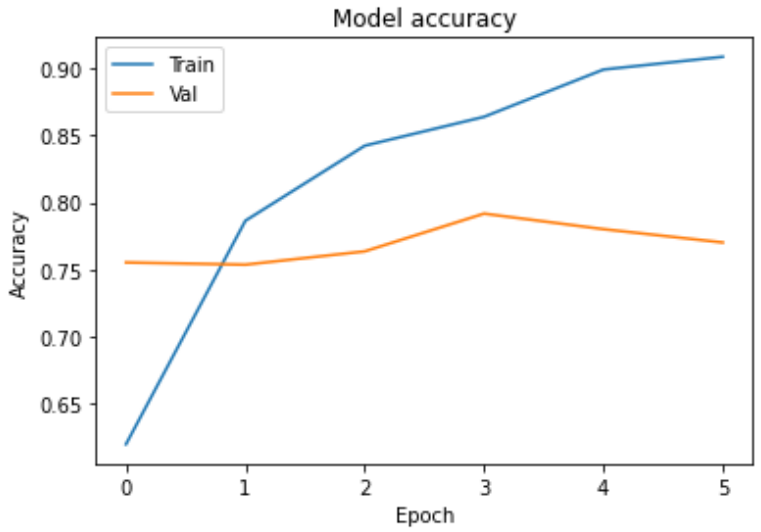
結論

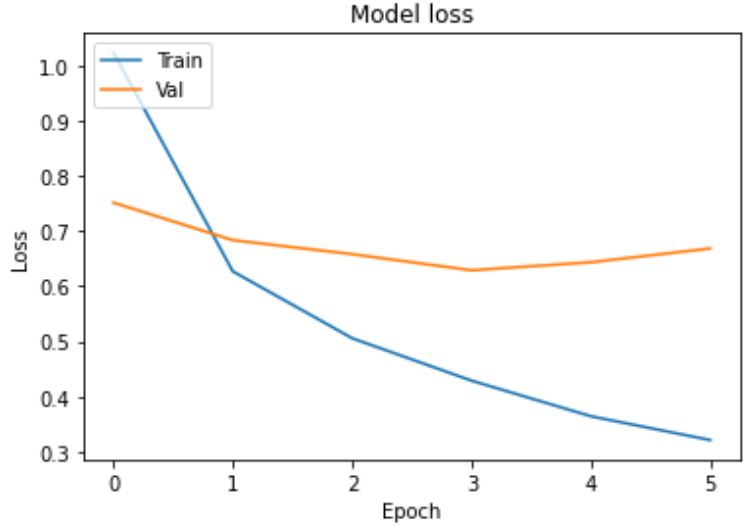
CNN(0.7405)





VGG16(0.9843)





Overfitting可能原因：

- VGG16未設置dropout
- VGG16模型較複雜

以更多參數組合測試VGG16

未來展望

- 以VGG16為基礎設計模型，協助農民辨識花卉種類

參考資料

- https://reurl.cc/bkRZ0E
- https://reurl.cc/g08WVR
- https://reurl.cc/oe9gWM
- https://reurl.cc/MbZArv


感 謝 聆 聽