# 農作物葉片健康及病態識別分類

Project 3

林呈昱 108034558

# CONTENTS

# ONE

## 主題說明

# 主題說明-問題描述

1

## 作物葉片上的資訊

農作物的健康狀態與否，其農作物上的葉片常常能反映出一些資訊給農民，以讓農民可以察知而及時做出相應措施。

2

## 觀光農場的盛行

農業的轉型進而成立了許多的觀光農場帶動了旅遊產業，成為觀光客嘉日休閒或放鬆的好去處。

1. **假想一個觀光農場**
2. **觀光客對某些農作物有興趣想了解其狀態**

# 主題說明-5W1H

| | |
|---|---|
| Why | 由於普遍大眾對農作物不甚了解，所以觀光客也不易得知其有興趣之農作物之狀態。 |
| What | 建立一個辨識農作物葉片的系統，讓觀光客能透過這個系統理解他們在觀光時有興趣之農作物的狀態。 |
| Where | 本專案所假想之觀光農場內，或擁有相同農作物的場合中。 |
| When | 觀光客對觀光農場內某些農作物有興趣了解時。 |
| Who | 舉凡對農作物狀態有興趣或需了解其狀態者，可以是農民、此觀光農場之觀光客或管理者等等。 |
| How | 收集大量農作物之葉片照片，利用卷積神經網路訓練這些照片，以判斷觀光客有興趣之農作物的狀態，得知其是否健康或者屬於何種病態類別的知識。 |

# 執行過程

1

## 資料收集

由Kaggle公開數據
集取得資料

2

## 資料前處理

3

## 模型設定

初始模型設定
後續模型調整

4

## 訓練集訓練

5

## 測試集測試

6

## 輸出結果

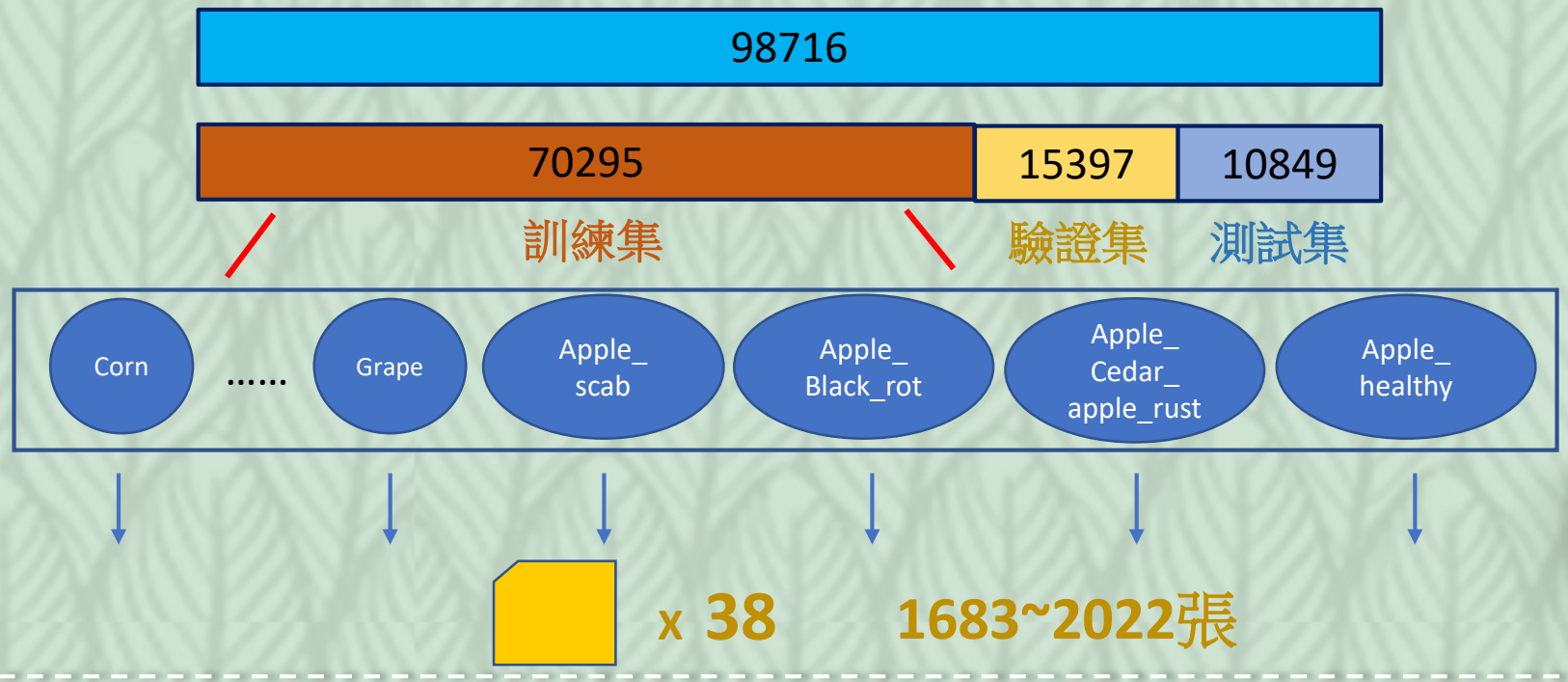
# Two

# 資料前處理

# 資料前處理-(1)

## 1 資料集分類及劃分

由Kaggle公開數據集中取得農作物葉片圖像資料集，包含了13種不同的農作物，其中每種作物的狀態又可分為健康及幾種可能的病態性質，因此每種作物又可分為若干類別，而此13種作物共包含了38種類別, 依類別分別分成29個資料夾, 總共98,716張照片。



# 資料前處理-(2)

2

## ImageDataGenerator

使用ImageDataGenerator生成更多樣的圖像

(1) rescale=1./255 :
將圖像做歸一化，使其像素值從 [0-255] 縮放到 [0-1]

(2)width_shift_range=0.2
圖片寬度的某個比例，生成更多圖像時圖片水平偏移的幅度

(3)height_shift_range=0.2,
圖片高度的某個比例，生成更多圖像時圖片垂直變換的幅度

```
83 from keras.preprocessing.image import ImageDataGenerator
84
85 train_datagen = ImageDataGenerator(rescale=1./255,
86                                     shear_range=0.2,
87                                     zoom_range=0.2,
88                                     width_shift_range=0.2,
89                                     height_shift_range=0.2,
90                                     fill_mode='nearest')
```

# 資料前處理-(3)

3

## flow_from_directory

使用flow_from_directory處理每張圖像

(1) target_size=(224, 224)
所有的圖像將被調整到224*224的尺寸

(2) color_mode='rgb'
圖像轉換成 1 或 3 個顏色通道

```
103 training_set = train_datagen.flow_from_directory(train_dir,
104                                                  target_size=(224, 224),
105                                                  color_mode='rgb',
106                                                  classes=None,
107                                                  batch_size=batch_size,
108                                                  class_mode='categorical')
109
110 valid_set = valid_datagen.flow_from_directory(valid_dir,
111                                                target_size=(224, 224),
112                                                color_mode='rgb',
113                                                classes=None,
114                                                batch_size=batch_size,
115                                                class_mode='categorical')
```

# 資料前處理-(4)

## 4 Labeling & one-hot encoding

讀取各個資料夾，為同類別資料夾內的圖片進行labeling，其值為0-37共38種，並進行 2D one-hot encoding

(1) classes=None
(2) class_mode='categorical'
(3) class_indices

```
103 training_set = train_datagen.flow_from_directory(train_dir,
104                                                  target_size=(224, 224),
105                                                  color_mode='rgb',
106                                                  classes=None,
107                                                  batch_size=batch_size,
108                                                  class_mode='categorical')
109
110 valid_set = valid_datagen.flow_from_directory(valid_dir,
111                                                target_size=(224, 224),
112                                                color_mode='rgb',
113                                                classes=None,
114                                                batch_size=batch_size,
115                                                class_mode='categorical')
```

```
In [27]: class_dict = training_set.class_indices
    ...: print(class_dict)
{'Apple___Apple_scab': 0, 'Apple___Black_rot': 1,
'Apple___Cedar_apple_rust': 2, 'Apple___healthy': 3, 'Blueberry___healthy':
4, 'Cherry_(including_sour)___Powdery_mildew': 5,
'Cherry_(including_sour)___healthy': 6, 'Corn_(maize)___Cercospora_leaf_spot
Gray_leaf_spot': 7, 'Corn_(maize)___Common_rust_': 8,
'Corn_(maize)___Northern_Leaf_Blight': 9, 'Corn_(maize)___healthy': 10,
'Grape___Black_rot': 11, 'Grape___Esca_(Black_Measles)': 12,
'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)': 13, 'Grape___healthy': 14,
'Orange___Haunglongbing_(Citrus_greening)': 15, 'Peach___Bacterial_spot':
16, 'Peach___healthy': 17, 'Pepper,_bell___Bacterial_spot': 18,
'Pepper,_bell___healthy': 19, 'Potato___Early_blight': 20,
'Potato___Late_blight': 21, 'Potato___healthy': 22, 'Raspberry___healthy':
23, 'Soybean___healthy': 24, 'Squash___Powdery_mildew': 25,
'Strawberry___Leaf_scorch': 26, 'Strawberry___healthy': 27,
'Tomato___Bacterial_spot': 28, 'Tomato___Early_blight': 29,
'Tomato___Late_blight': 30, 'Tomato___Leaf_Mold': 31,
'Tomato___Septoria_leaf_spot': 32, 'Tomato___Spider_mites Two-
spotted_spider_mite': 33, 'Tomato___Target_Spot': 34,
'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 35,
'Tomato___Tomato_mosaic_virus': 36, 'Tomato___healthy': 37}
```

# Three

## 模型設定

# 模型設定

## 1 初始模型架構

0. conv2d-1
1. max_pooling2d-1
2. batch_normalization-1
3. conv2d-2
4. max_pooling2d-2
7. batch_normalization-2
8. conv2d-4
9. batch_normalization-3
10. conv2d-5
11. max_pooling2d-3-
12. batch_normalization-4
13. flatten
14. dense-1
15. dropout-1
16. batch_normalization-5
17. dense-2
18. dropout-2
19. batch_normalization-6
20. dense-3
21. dropout-3
22. batch_normalization-7
23. dense-4

```
In [29]: for i, layer in enumerate(classifier.layers):
    ...:     print(i, layer.name)
0 conv2d_36
1 max_pooling2d_22
2 batch_normalization_57
3 conv2d_37
4 max_pooling2d_23
5 batch_normalization_58
6 conv2d_38
7 batch_normalization_59
8 conv2d_39
9 batch_normalization_60
10 conv2d_40
11 max_pooling2d_24
12 batch_normalization_61
13 flatten_8
14 dense_29
15 dropout_22
16 batch_normalization_62
17 dense_30
18 dropout_23
19 batch_normalization_63
20 dense_31
21 dropout_24
22 batch_normalization_64
23 dense_32
```

# 模型設定

| | Loss | 激活函數 | 最後一層全連接層激活函數 | 優化器 | Epoch |
|---|---|---|---|---|---|
| **Model 1** | categorical | relu | softmax | SGD | 10 |

```python
# Initializing the CNN
classifier = Sequential()

# Convolution Step 1
classifier.add(Convolution2D(96, 11, strides = (4, 4), padding = 'valid', input_shape=(224, 224, 3), activation = 'relu'))
# Max Pooling Step 1
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())
# Convolution Step 2
classifier.add(Convolution2D(256, 11, strides = (1, 1), padding='valid', activation = 'relu'))
# Max Pooling Step 2
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding='valid'))
classifier.add(BatchNormalization())
# Convolution Step 3
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())
# Convolution Step 4
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())
# Convolution Step 5
classifier.add(Convolution2D(256, 3, strides=(1,1), padding='valid', activation = 'relu'))
# Max Pooling Step 3
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())

# Flattening Step
classifier.add(Flatten())

# Full Connection Step
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 1000, activation = 'relu'))
classifier.add(Dropout(0.2))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 38, activation = 'softmax'))

classifier.summary()
```

```python
# Compiling the Model
from keras import optimizers
classifier.compile(optimizer= 'adam',
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])
```

```python
# Compiling the Model
from keras import optimizers
classifier.compile(optimizer=optimizers.SGD(lr=0.001, momentum=0.9, decay=0.005),
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])
```

# Four

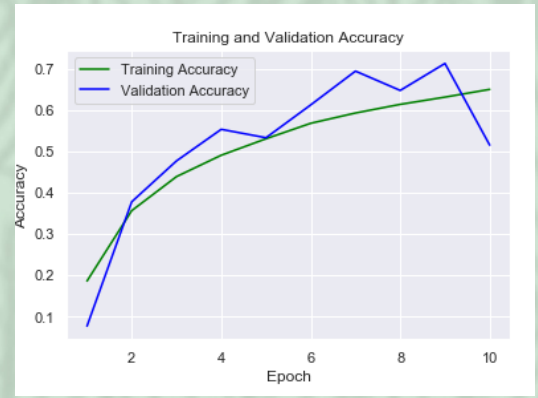
## 模型
## 訓練過程

# 訓練過程

## 1 初始模型(Model 1)結果

| | Loss | 激活函數 | 最後一層全連接層激活函數 | 優化器 | Epoch |
|---|---|---|---|---|---|
| **Model 1** | categorical | relu | softmax | SGD | 10 |

```
Epoch 1/10
274/274 [==============================] - 28444s 104s/step - loss: 3.2248 - accuracy: 0.1857 - val_loss: 3.8209 - val_accuracy:
0.0766
Epoch 2/10
274/274 [==============================] - 14269s 52s/step - loss: 2.2944 - accuracy: 0.3562 - val_loss: 2.2215 - val_accuracy: 0.3776
Epoch 3/10
 98/274 [=========>....................] - ETA: 32:53 - loss: 2.0293 - accuracy: 0.4162
274/274 [==============================] - 3220s 12s/step - loss: 1.9387 - accuracy: 0.4385 - val_loss: 1.5970 - val_accuracy: 0.4766
Epoch 4/10
274/274 [==============================] - 3180s 12s/step - loss: 1.7286 - accuracy: 0.4903 - val_loss: 1.4950 - val_accuracy: 0.5530
Epoch 5/10
274/274 [==============================] - 3188s 12s/step - loss: 1.5676 - accuracy: 0.5301 - val_loss: 1.5124 - val_accuracy: 0.5326
Epoch 6/10
274/274 [==============================] - 3338s 12s/step - loss: 1.4431 - accuracy: 0.5676 - val_loss: 1.1932 - val_accuracy: 0.6122
Epoch 7/10
274/274 [==============================] - 3056s 11s/step - loss: 1.3499 - accuracy: 0.5925 - val_loss: 0.8109 - val_accuracy: 0.6939
Epoch 8/10
274/274 [==============================] - 3066s 11s/step - loss: 1.2764 - accuracy: 0.6134 - val_loss: 1.2128 - val_accuracy: 0.6471
Epoch 9/10
274/274 [==============================] - 3077s 11s/step - loss: 1.2130 - accuracy: 0.6310 - val_loss: 0.8747 - val_accuracy: 0.7125
Epoch 10/10
274/274 [==============================] - 3095s 11s/step - loss: 1.1513 - accuracy: 0.6498 - val_loss: 1.6453 - val_accuracy: 0.5148
```




```
361/361 [==============================] - 926s 3s/step
Test_Accuracy: 43.91%
```

```
155 #saving model
156 filepath="model_1"
157 classifier.save(filepath)
```

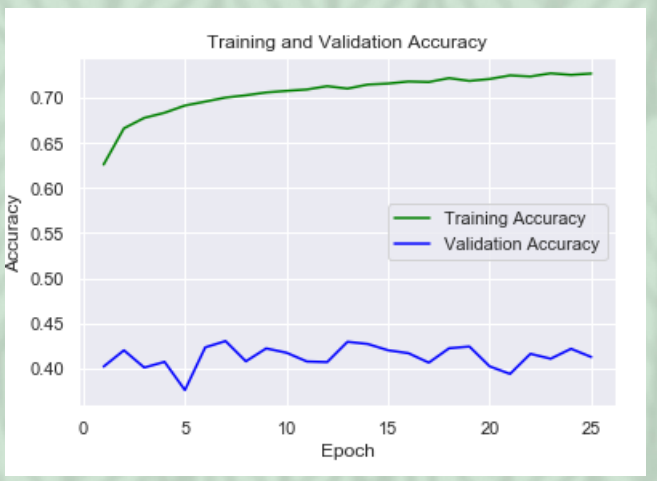
# 訓練過程

## 2 將保存的 model 1 繼續訓練

```
62 classifier.load_weights('C:/Users/qaz74/Desktop/project3/model_1')
```

| | Loss | 激活函數 | 最後一層全連接層激活函數 | 優化器 | Epoch |
|---|---|---|---|---|---|
| **Model 2** | categorical | relu | softmax | SGD | 25 |

```
Epoch 1/25
274/274 [==============================] - 1852s 7s/step - loss: 1.2402 - accuracy: 0.6259 - val_loss: 2.9623 - val_accuracy: 0.4018
Epoch 2/25
274/274 [==============================] - 1749s 6s/step - loss: 1.0697 - accuracy: 0.6661 - val_loss: 3.2971 - val_accuracy: 0.4199
Epoch 3/25
274/274 [==============================] - 1786s 7s/step - loss: 1.0175 - accuracy: 0.6777 - val_loss: 2.6515 - val_accuracy: 0.4007
Epoch 4/25
274/274 [==============================] - 1826s 7s/step - loss: 0.9956 - accuracy: 0.6833 - val_loss: 3.1368 - val_accuracy: 0.4071
Epoch 5/25
274/274 [==============================] - 1778s 6s/step - loss: 0.9698 - accuracy: 0.6914 - val_loss: 3.0172 - val_accuracy: 0.3757
Epoch 6/25
274/274 [==============================] - 1797s 7s/step - loss: 0.9508 - accuracy: 0.6957 - val_loss: 2.5361 - val_accuracy: 0.4231
Epoch 7/25
274/274 [==============================] - 1803s 7s/step - loss: 0.9333 - accuracy: 0.7002 - val_loss: 2.7280 - val_accuracy: 0.4301
Epoch 8/25
274/274 [==============================] - 1741s 6s/step - loss: 0.9267 - accuracy: 0.7027 - val_loss: 2.7332 - val_accuracy: 0.4077
accuracy: 0.7175 - val_loss: 3.1355 - val_accuracy: 0.4062
Epoch 18/25
274/274 [==============================] - 1664s 6s/step - loss: 0.8639 - accuracy: 0.7218 - val_loss: 2.7810 - val_accuracy: 0.4222
Epoch 19/25
274/274 [==============================] - 1649s 6s/step - loss: 0.8677 - accuracy: 0.7187 - val_loss: 2.7626 - val_accuracy: 0.4241
Epoch 20/25
274/274 [==============================] - 1623s 6s/step - loss: 0.8618 - accuracy: 0.7208 - val_loss: 3.0297 - val_accuracy: 0.4020
Epoch 21/25
274/274 [==============================] - 1689s 6s/step - loss: 0.8533 - accuracy: 0.7248 - val_loss: 3.1380 - val_accuracy: 0.3937
Epoch 22/25
274/274 [==============================] - 1647s 6s/step - loss: 0.8543 - accuracy: 0.7236 - val_loss: 2.5496 - val_accuracy: 0.4160
Epoch 23/25
274/274 [==============================] - 1641s 6s/step - loss: 0.8393 - accuracy: 0.7271 - val_loss: 3.3058 - val_accuracy: 0.4105
Epoch 24/25
274/274 [==============================] - 1674s 6s/step - loss: 0.8476 - accuracy: 0.7254 - val_loss: 2.9179 - val_accuracy: 0.4216
Epoch 25/25
274/274 [==============================] - 1651s 6s/step - loss: 0.8400 - accuracy: 0.7269 - val_loss: 3.0704 - val_accuracy: 0.4124
```




```
361/361 [==============================] - 926s 3s/step
Test_Accuracy: 69.10%
```

```
155 #saving model
156 filepath="model_2"
157 classifier.save(filepath)
```

# 訓練過程

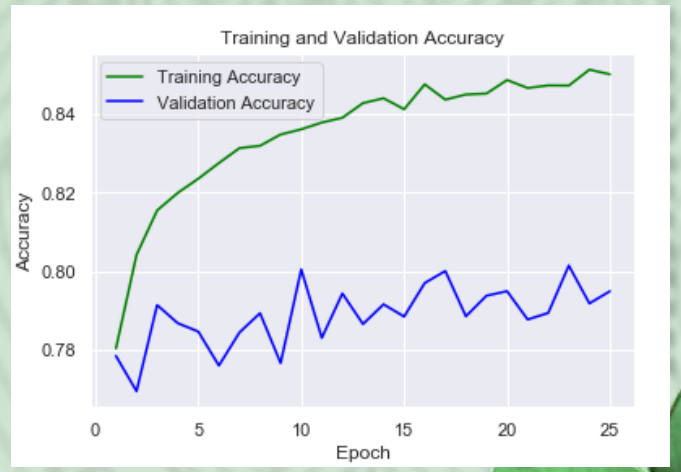
3

## 將優化器改成 adam

```
62 classifier.load_weights('C:/Users/qaz74/Desktop/project3/model_2')
```

| | Loss | 激活函數 | 最後一層全連接層激活函數 | 優化器 | Epoch |
|---|---|---|---|---|---|
| **Model 3** | categorical | relu | softmax | adam | 25 |

```
Epoch 1/25
274/274 [==============================] - 2018s 7s/step - loss: 0.7015 - accuracy: 0.7804 - val_loss: 0.9415 - val_accuracy: 0.7784
Epoch 2/25
274/274 [==============================] - 2017s 7s/step - loss: 0.6058 - accuracy: 0.8042 - val_loss: 0.7185 - val_accuracy: 0.7695
Epoch 3/25
274/274 [==============================] - 2041s 7s/step - loss: 0.5739 - accuracy: 0.8155 - val_loss: 0.7353 - val_accuracy: 0.7914
Epoch 4/25
274/274 [==============================] - 2028s 7s/step - loss: 0.5519 - accuracy: 0.8199 - val_loss: 0.6906 - val_accuracy: 0.7868
Epoch 5/25
274/274 [==============================] - 2019s 7s/step - loss: 0.5374 - accuracy: 0.8236 - val_loss: 0.6165 - val_accuracy: 0.7847
Epoch 6/25
274/274 [==============================] - 2019s 7s/step - loss: 0.5288 - accuracy: 0.8276 - val_loss: 0.7519 - val_accuracy: 0.7760
Epoch 7/25
274/274 [==============================] - 2024s 7s/step - loss: 0.5140 - accuracy: 0.8314 - val_loss: 0.8420 - val_accuracy: 0.7844
Epoch 8/25
274/274 [==============================] - 2060s 8s/step - loss: 0.5111 - accuracy: 0.8320 - val_loss: 0.8329 - val_accuracy: 0.7893
Epoch 9/25
274/274 [==============================] - 2043s 7s/step - loss: 0.4999 - accuracy: 0.8348 - val_loss: 0.7863 - val_accuracy: 0.7766
Epoch 10/25
274/274 [==============================] - 2050s 7s/step - loss: 0.4957 - accuracy: 0.8361 - val_loss: 0.5406 - val_accuracy: 0.8005
Epoch 11/25
274/274 [==============================] - 2046s 7s/step - loss: 0.4948 - accuracy: 0.8379 - val_loss: 0.6910 - val_accuracy: 0.7830
Epoch 15/25
274/274 [==============================] - 1951s 7s/step - loss: 0.4775 - accuracy: 0.8412 - val_loss: 0.7685 - val_accuracy: 0.7885
Epoch 16/25
274/274 [==============================] - 1955s 7s/step - loss: 0.4653 - accuracy: 0.8476 - val_loss: 0.7344 - val_accuracy: 0.7970
Epoch 17/25
274/274 [==============================] - 1985s 7s/step - loss: 0.4686 - accuracy: 0.8437 - val_loss: 0.6843 - val_accuracy: 0.8001
Epoch 18/25
274/274 [==============================] - 1973s 7s/step - loss: 0.4680 - accuracy: 0.8450 - val_loss: 0.6433 - val_accuracy: 0.7885
Epoch 19/25
274/274 [==============================] - 1967s 7s/step - loss: 0.4674 - accuracy: 0.8452 - val_loss: 0.7827 - val_accuracy: 0.7938
Epoch 20/25
274/274 [==============================] - 2036s 7s/step - loss: 0.4565 - accuracy: 0.8487 - val_loss: 0.8171 - val_accuracy: 0.7949
Epoch 21/25
274/274 [==============================] - 2037s 7s/step - loss: 0.4631 - accuracy: 0.8467 - val_loss: 0.6373 - val_accuracy: 0.7877
Epoch 22/25
274/274 [==============================] - 1982s 7s/step - loss: 0.4619 - accuracy: 0.8473 - val_loss: 0.7672 - val_accuracy: 0.7894
Epoch 23/25
274/274 [==============================] - 1966s 7s/step - loss: 0.4510 - accuracy: 0.8473 - val_loss: 0.5738 - val_accuracy: 0.8015
Epoch 24/25
274/274 [==============================] - 1956s 7s/step - loss: 0.4484 - accuracy: 0.8513 - val_loss: 0.6403 - val_accuracy: 0.7918
Epoch 25/25
274/274 [==============================] - 1960s 7s/step - loss: 0.4486 - accuracy: 0.8501 - val_loss: 0.5826 - val_accuracy: 0.7949
```




```
361/361 [==============================] - 1761s 5s/s
Test_Accuracy: 80.50%
```

# 訓練過程

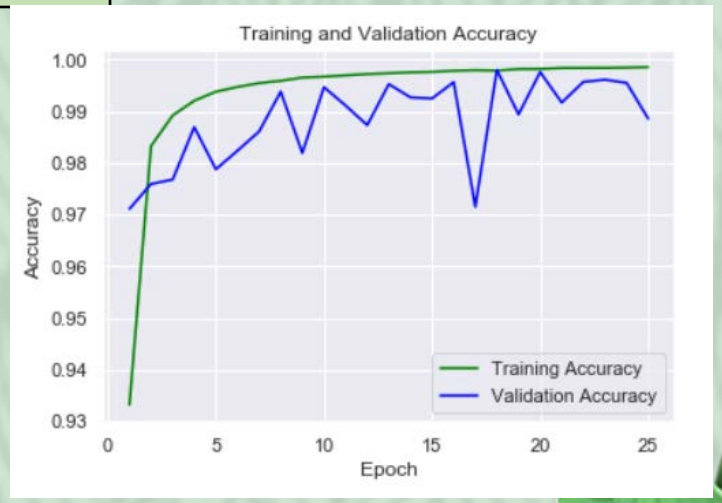
## 4 更改損失函數為binary 、激活函數為 sigmoid

```
62 classifier.load_weights('C:/Users/qaz74/Desktop/project3/model_2')
```

| | Loss | 激活函數 | 最後一層全連接層激活函數 | 優化器 | Epoch |
|---|---|---|---|---|---|
| **Model 4** | binary | relu | sigmoid | adam | 25 |

```
Epoch 1/25
274/274 [==============================] - 893s 3s/step - loss: 0.2028 - accuracy: 0.9332 - val_loss: 0.1096 - val_accuracy:
0.9711
Epoch 2/25
274/274 [==============================] - 831s 3s/step - loss: 0.0497 - accuracy: 0.9832 - val_loss: 0.0892 - val_accuracy:
0.9759
Epoch 3/25
274/274 [==============================] - 654s 2
0.9767
Epoch 4/25
274/274 [==============================] - 608s 2
0.9869
Epoch 5/25
274/274 [==============================] - 605s 2
0.9787
Epoch 6/25
274/274 [==============================] - 603s 2
0.9823
```

```
274/274 [==============================] - 598s 2s/step - loss: 0.0054 - accuracy: 0.9981 - val_loss: 0.0408 - val_accuracy:
0.9893
Epoch 20/25
274/274 [==============================] - 597s 2s/step - loss: 0.0054 - accuracy: 0.9982 - val_loss: 0.0063 - val_accuracy:
0.9976
Epoch 21/25
274/274 [==============================] - 598s 2s/step - loss: 0.0048 - accuracy: 0.9983 - val_loss: 0.0294 - val_accuracy:
0.9916
Epoch 22/25
274/274 [==============================] - 826s 3s/step - loss: 0.0047 - accuracy: 0.9983 - val_loss: 0.0103 - val_accuracy:
0.9957
Epoch 23/25
274/274 [==============================] - 603s 2s/step - loss: 0.0048 - accuracy: 0.9984 - val_loss: 0.0146 - val_accuracy:
0.9961
Epoch 24/25
274/274 [==============================] - 591s 2s/step - loss: 0.0046 - accuracy: 0.9984 - val_loss: 0.0113 - val_accuracy:
0.9955
Epoch 25/25
274/274 [==============================] - 597s 2s/step - loss: 0.0043 - accuracy: 0.9985 - val_loss: 0.0432 - val_accuracy:
0.9885
```




```
361/361 [==============================] - 124s 344ms/st
Test_Accuracy: 99.03%
```

# 訓練過程

## 5 套用model 4 設置在model 1 上

```
62 classifier.load_weights('C:/Users/qaz74/Desktop/project3/model_1')
```

```
Epoch 1/25
274/274 [==============================] - 656s 2s/step - loss: 0.1962 - accuracy: 0.9344 - val_loss: 0.2735 - val_accuracy: 0.
9733
Epoch 2/25
274/274 [==============================] - 844s 3s/step - loss: 0.0519 - accuracy: 0.9831 - val_loss: 0.1168 - val_accuracy: 0.
9708
Epoch 3/25
274/274 [==============================] - 615s 2s/step - loss: 0.0343 - accuracy: 0.9885 - val_loss: 0.0455 - val_accuracy: 0.
9850
Epoch 4/25
274/274 [==============================] - 612s 2s/step - loss: 0.0243 - accuracy: 0.9914 - val_loss: 0.1574 - val_accuracy: 0.
9669
Epoch 5/25
274/274 [==============================]
9676
Epoch 6/25
274/274 [==============================]
9721
Epoch 7/25
274/274 [==============================]
9878
Epoch 8/25
274/274 [==============================]
9888
Epoch 9/25
274/274 [==============================]
9935
Epoch 10/25
274/274 [==============================]
9906
Epoch 11/25
274/274 [==============================]
9816
Epoch 12/25
274/274 [==============================]
9717
Epoch 15/25
274/274 [==============================] - 593s 2s/step - loss: 0.0071 - accuracy: 0.9975 - val_loss: 0.0529 - val_accuracy: 0.
9858
Epoch 16/25
274/274 [==============================] - 593s 2s/step - loss: 0.0067 - accuracy: 0.9977 - val_loss: 0.0261 - val_accuracy: 0.
9906
Epoch 17/25
274/274 [==============================] - 593s 2s/step - loss: 0.0062 - accuracy: 0.9978 - val_loss: 0.0505 - val_accuracy: 0.
9871
Epoch 18/25
274/274 [==============================] - 593s 2s/step - loss: 0.0061 - accuracy: 0.9979 - val_loss: 0.0789 - val_accuracy: 0.
9837
Epoch 19/25
274/274 [==============================] - 594s 2s/step - loss: 0.0060 - accuracy: 0.9979 - val_loss: 0.0215 - val_accuracy: 0.
9930
Epoch 20/25
274/274 [==============================] - 593s 2s/step - loss: 0.0053 - accuracy: 0.9981 - val_loss: 0.0129 - val_accuracy: 0.
9950
Epoch 21/25
274/274 [==============================] - 593s 2s/step - loss: 0.0053 - accuracy: 0.9982 - val_loss: 0.0271 - val_accuracy: 0.
9917
Epoch 22/25
274/274 [==============================] - 593s 2s/step - loss: 0.0053 - accuracy: 0.9982 - val_loss: 0.1791 - val_accuracy: 0.
9711
Epoch 23/25
274/274 [==============================] - 593s 2s/step - loss: 0.0051 - accuracy: 0.9982 - val_loss: 0.0280 - val_accuracy: 0.
9936
Epoch 24/25
274/274 [==============================] - 597s 2s/step - loss: 0.0047 - accuracy: 0.9983 - val_loss: 0.0936 - val_accuracy: 0.
9791
Epoch 25/25
274/274 [==============================] - 594s 2s/step - loss: 0.0048 - accuracy: 0.9983 - val_loss: 0.0329 - val_accuracy: 0.
9910
```


```
361/361 [==============================] - 1728s 5s/step
Test_Accuracy: 97.48%
```

# Five

# 結論與未來展望

# 結論

1. 準確率大幅提升

| | Loss | 激活函數 | 最後一層全連接層激活函數 | 優化器 | Epoch | Test Accuracy |
|---|---|---|---|---|---|---|
| Model 1 | categorical | relu | softmax | SGD | 10 | 43.91 % |
| Model 2 | categorical | relu | softmax | SGD | 25 | 69.10 % |
| Model 3 | categorical | relu | softmax | adam | 25 | 80.05 % |
| Model 4 | binary | relu | sigmoid | adam | 25 | 99.03 % |

2. 為觀光客在此觀光農場提供可信的辨識系統

3. 為後續系統建立奠定基礎

# 未來展望

1 加入更多物種

目前只有13種的農作物，加入更多的物種以因應更大規模的觀光農場 。

3 加入更多圖像

2 開發不同適用範圍的系統

農作物的種植類型，可能會因國家、氣候.....等等因素，而有不同區域性分布的種植類型，可針對此開發專屬某區域的系統 。

# THANK YOU