Deep Learning圖像辨識
Flower Recognition

學生:楊恆軒

指導老師:邱銘傳

National Tsing Hua University, R.O.C.

January, 2019

目錄

簡介&研究方法

- 透過建立CNN模型，利用圖像辨識之技術辨識出 Dataset 之花朵的種類，並計算其準確率。
- Dataset:

1. 雛菊:769張照片
2. 蒲公英:1055張照片
3. 玫瑰:784張照片
4. 向日葵:734張照片
5. 鬱金香:984張照片

使用工具&套件

- OpenCV

```
# OpenCV
!apt-get -qq install -y libsm6 libxext6 && pip install -q -U opencv-python
```

- Import the necessary libraries

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models

# Helper libraries
import os
from os import listdir
from os.path import join
import numpy as np
import matplotlib.pyplot as plt
import cv2
from google.colab import drive
import pandas
import random
```

使用工具&套件

- Accessing My Google Drive

```
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9

Enter your authorization code:
··········
Mounted at /content/drive
```

- Location of my dataset on My Google Drive

```
# defining global variable path
# Location of my dataset on My Google Drive
image_path = 'drive/My Drive/flowers'

def loadImages(path):
    '''Put files into lists and return them as one list with all images
     in the folder'''
    image_files = sorted([os.path.join(path, 'train', file)
                          for file in os.listdir(path + '/train')
                          if file.endswith('.png')])
    return image_files
```

```
# Set the path of the input folder
data = 'drive/My Drive/flowers'

# List out the directories inside the main input folder
folders = os.listdir(data)
print(folders)
```

```
['dandelion', 'tulip', 'daisy', 'sunflower', 'rose']
```

資料前處理

- 調整尺寸

```
# Import the images and resize them to a 64*64 size
# Also generate the corresponding labels

image_names = []
train_labels = []
train_images = []

size = 64,64

for folder in folders:
    for file in os.listdir(os.path.join(data,folder)):
        if file.endswith("jpg"):
            image_names.append(os.path.join(data,folder,file))
            train_labels.append(folder)
            img = cv2.imread(os.path.join(data,folder,file))
            im = cv2.resize(img,size)
            train_images.append(im)
        else:
            continue
```

- 轉換成矩陣&標準化

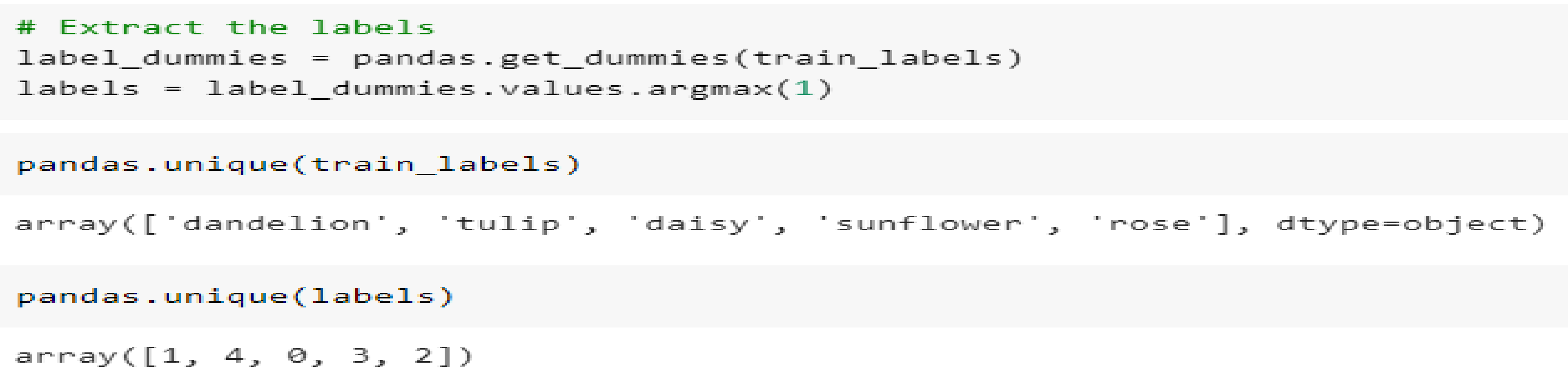
```
# Transform the image array to a numpy type
train = np.array(train_images)
train.shape
```

```
(4323, 64, 64, 3)
```

```
# Reduce the RGB values between 0 and 1
train = train.astype('float32') / 255.0
```

資料前處理

- 提取標籤&給定編號



```
# Extract the labels
label_dummies = pandas.get_dummies(train_labels)
labels = label_dummies.values.argmax(1)
```

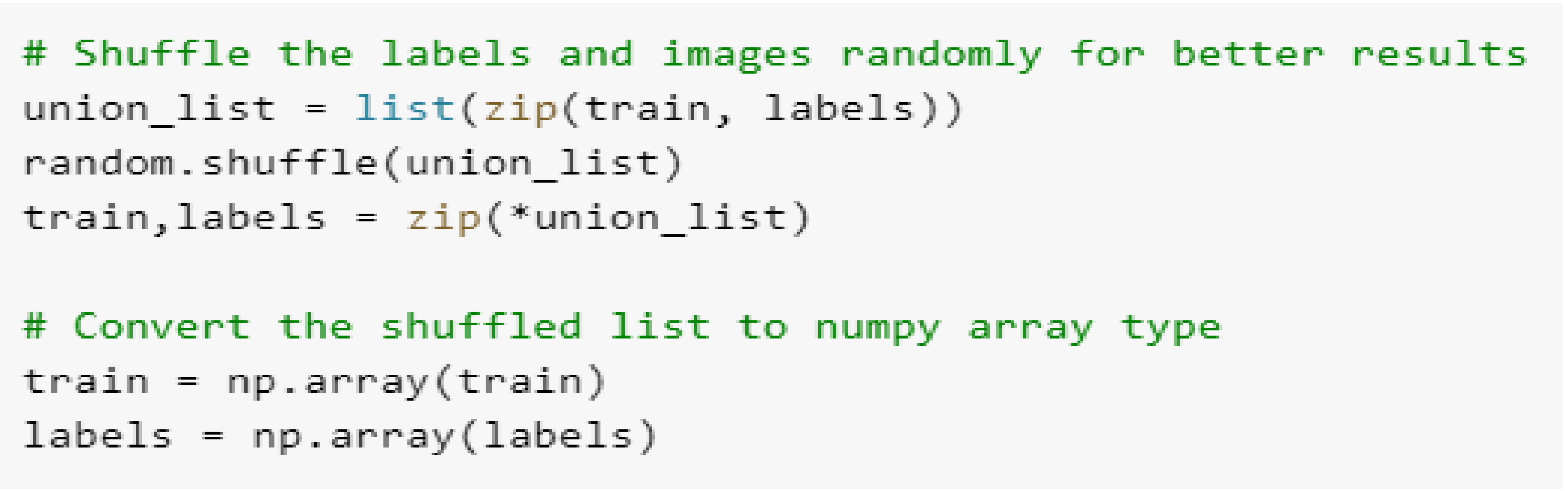
```
pandas.unique(train_labels)
```

```
array(['dandelion', 'tulip', 'daisy', 'sunflower', 'rose'], dtype=object)
```

```
pandas.unique(labels)
```

```
array([1, 4, 0, 3, 2])
```

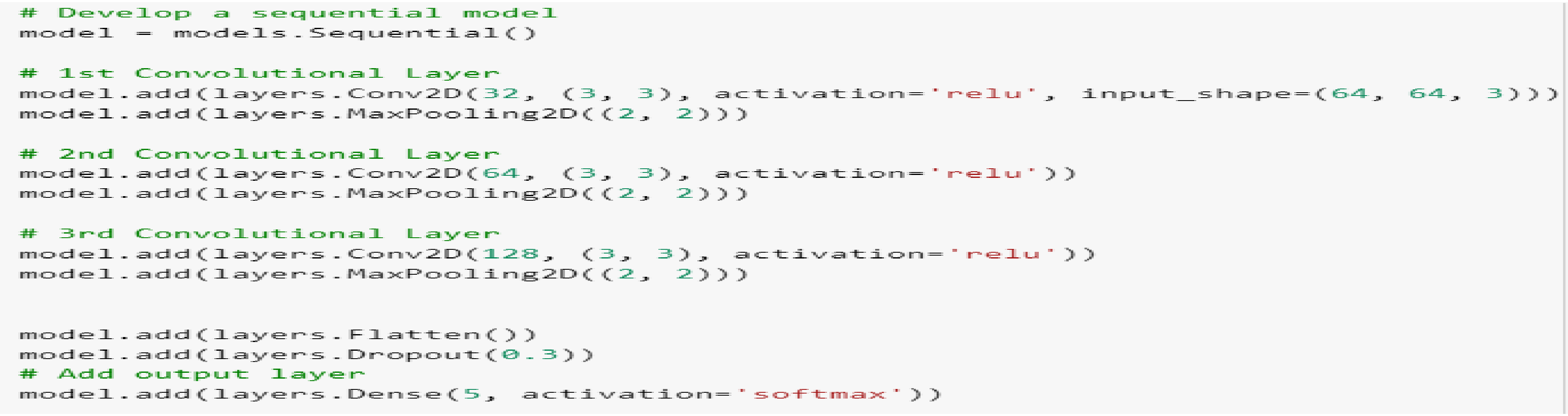
- 隨機打亂圖像&編號



```
# Shuffle the labels and images randomly for better results
union_list = list(zip(train, labels))
random.shuffle(union_list)
train,labels = zip(*union_list)

# Convert the shuffled list to numpy array type
train = np.array(train)
labels = np.array(labels)
```

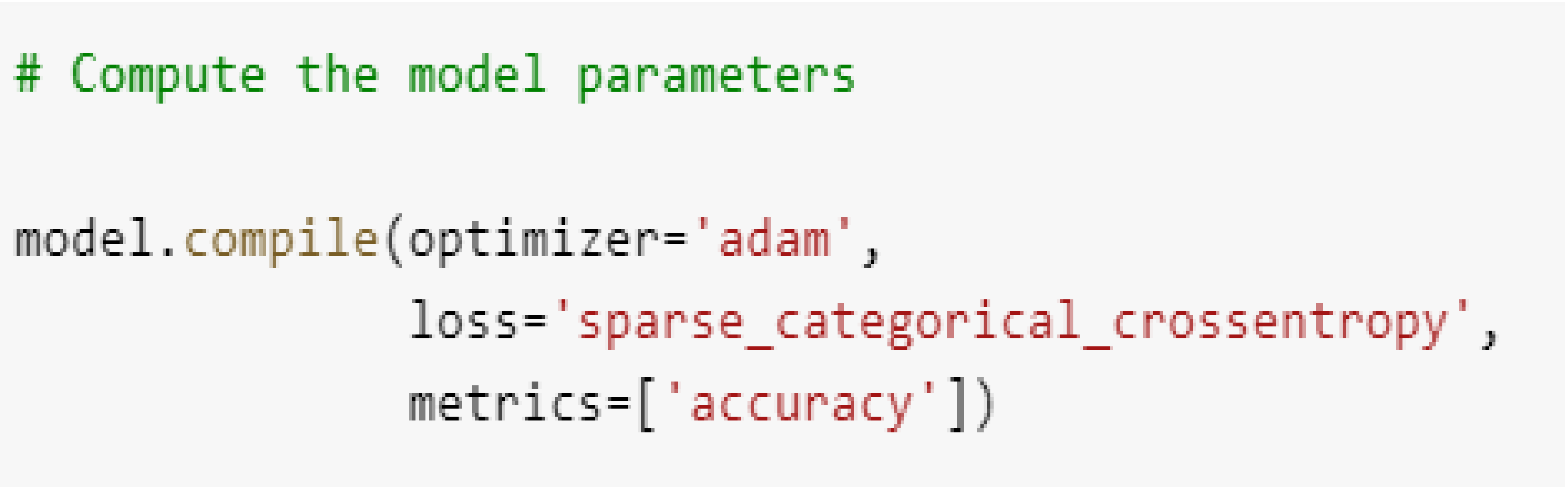
模型建立

- 發展模型



```
# Develop a sequential model
model = models.Sequential()

# 1st Convolutional Layer
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)))
model.add(layers.MaxPooling2D((2, 2)))

# 2nd Convolutional Layer
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# 3rd Convolutional Layer
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))


model.add(layers.Flatten())
model.add(layers.Dropout(0.3))
# Add output layer
model.add(layers.Dense(5, activation='softmax'))
```

- 設定模型參數



```
# Compute the model parameters


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

模型建立

- 訓練模型&計算準確率

```
# Train the model  with 5 epochs
model.fit(train,labels, epochs=5)
```

```
Train on 4323 samples
Epoch 1/5
4323/4323 [==============================] - 25s 6ms/sample - loss: 1.2681 - acc: 0.4481
Epoch 2/5
4323/4323 [==============================] - 25s 6ms/sample - loss: 1.0544 - acc: 0.5785
Epoch 3/5
4323/4323 [==============================] - 25s 6ms/sample - loss: 0.9615 - acc: 0.6234
Epoch 4/5
4323/4323 [==============================] - 25s 6ms/sample - loss: 0.8711 - acc: 0.6620
Epoch 5/5
4323/4323 [==============================] - 25s 6ms/sample - loss: 0.8201 - acc: 0.6944
<tensorflow.python.keras.callbacks.History at 0x7f5c12d9ff98>
```

```
 #get score acording to train datas
print("Test Accuracy: {0:.2f}%".format(model.evaluate(train,labels)[1]*100))
```

```
4323/4323 [==============================] - 7s 2ms/sample - loss: 0.7483 - acc: 0.7095
Test Accuracy: 70.95%
```

加入新圖片

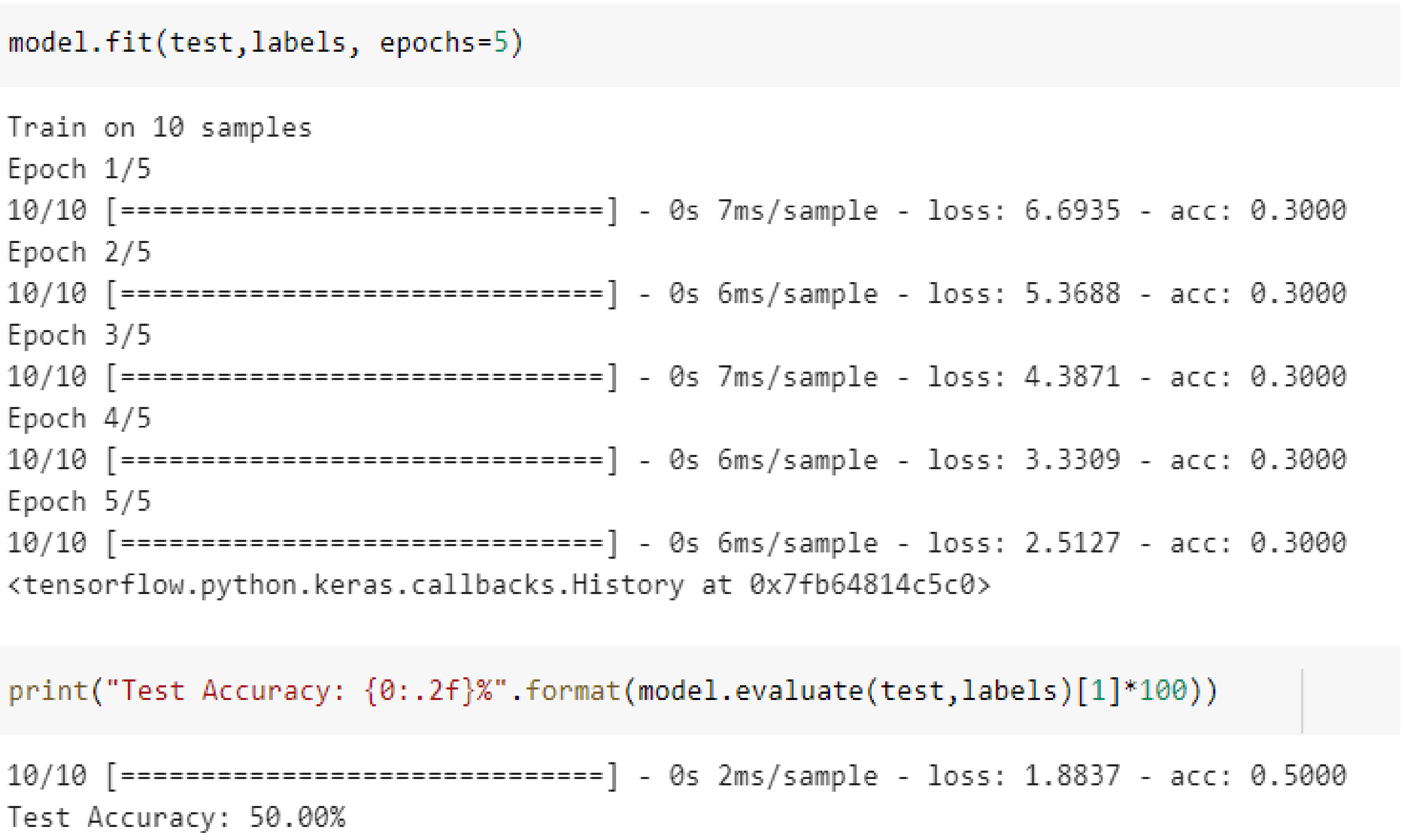
- New image

```
test = np.array(test_images)
test.shape
```

```
(10, 64, 64, 3)
```

加入新圖片

- 加入模型



```
model.fit(test,labels, epochs=5)
```

```
Train on 10 samples
Epoch 1/5
10/10 [==============================] - 0s 7ms/sample - loss: 6.6935 - acc: 0.3000
Epoch 2/5
10/10 [==============================] - 0s 6ms/sample - loss: 5.3688 - acc: 0.3000
Epoch 3/5
10/10 [==============================] - 0s 7ms/sample - loss: 4.3871 - acc: 0.3000
Epoch 4/5
10/10 [==============================] - 0s 6ms/sample - loss: 3.3309 - acc: 0.3000
Epoch 5/5
10/10 [==============================] - 0s 6ms/sample - loss: 2.5127 - acc: 0.3000
<tensorflow.python.keras.callbacks.History at 0x7fb64814c5c0>
```

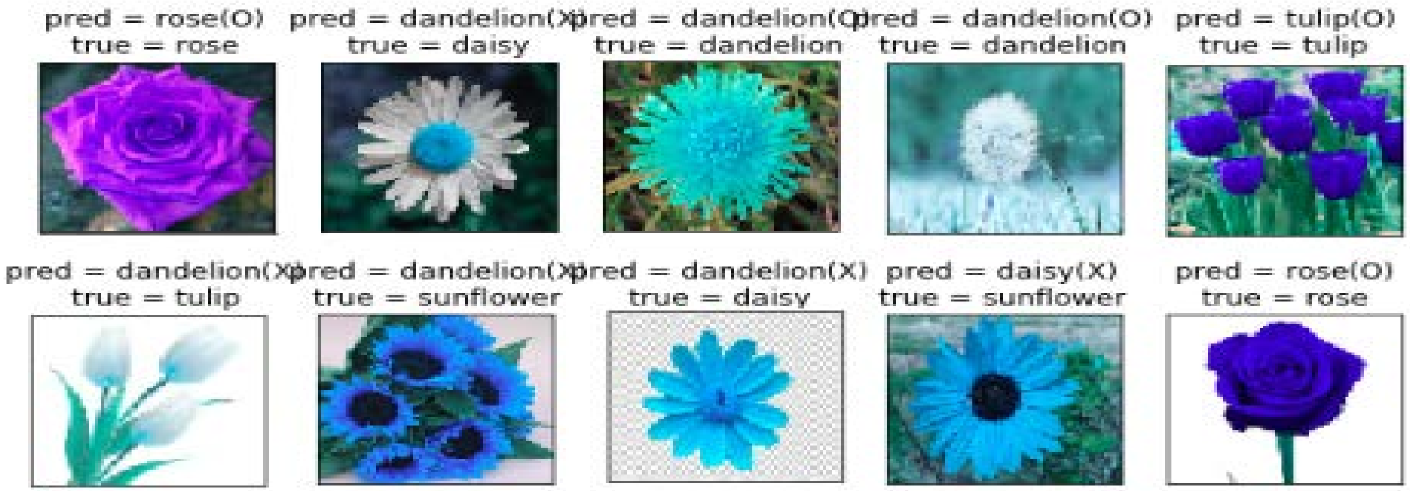
```
print("Test Accuracy: {0:.2f}%".format(model.evaluate(test,labels)[1]*100))
```

```
10/10 [==============================] - 0s 2ms/sample - loss: 1.8837 - acc: 0.5000
Test Accuracy: 50.00%
```

加入新圖片

- 顯示結果

```
import matplotlib.pyplot as plot
import math
def show_test_label_prediction( test, labels, predictions, indexList ) :

    num = len(indexList)
    plot.gcf().set_size_inches( 2*5, (2+0.4)*math.ceil(num/5) )

    loc = 0
    for i in indexList :
      loc += 1
      subp = plot.subplot( math.ceil(num/5), 5, loc )
      subp.imshow( test[i], cmap='binary' )
      if( len(predictions) > 0 ) :
            title = 'pred = ' + label_desc[ predictions[i] ]
            title += ('(O)' if predictions[i]==labels[i] else '(X)')
            title += '\ntrue = ' + label_desc[ labels[i] ]
      else :
            title = 'true = ' + label_desc[ labels[i] ]
      subp.set_title( title, fontsize=12 )
      subp.set_xticks( [] )
      subp.set_yticks( [] )
plot.show()
show_test_label_prediction( test, labels, prediction, range(0, 10) )
```



結論&未來方向

- 提高準確度

-蒐集更多圖片，增加Dataset數量。

- 增加花朵種類

-例如增加牡丹花、牽牛花...等等，使模型更加強大。

- 結合手機

-特過APP照相上傳，讓模型辨別花朵種類。

參考資料

- Kaggle資料集

https://www.kaggle.com/alxmamaev/flowers-recognition

- Model

https://www.kaggle.com/rajmehra03/flower-recognition-cnn-keras/notebook

- 顯示圖片結果

https://tomkuo139.blogspot.com/2018/05/aicnn-cifar10.html

Thank You