AI 技術班期中考_人臉辨識

題目說明

簡述

本試題之目標為辨識圖片中的人物是誰，資料是助教用手機自行拍攝蒐集的。

進行辦法

利用深度學習模型進行圖像辨識。

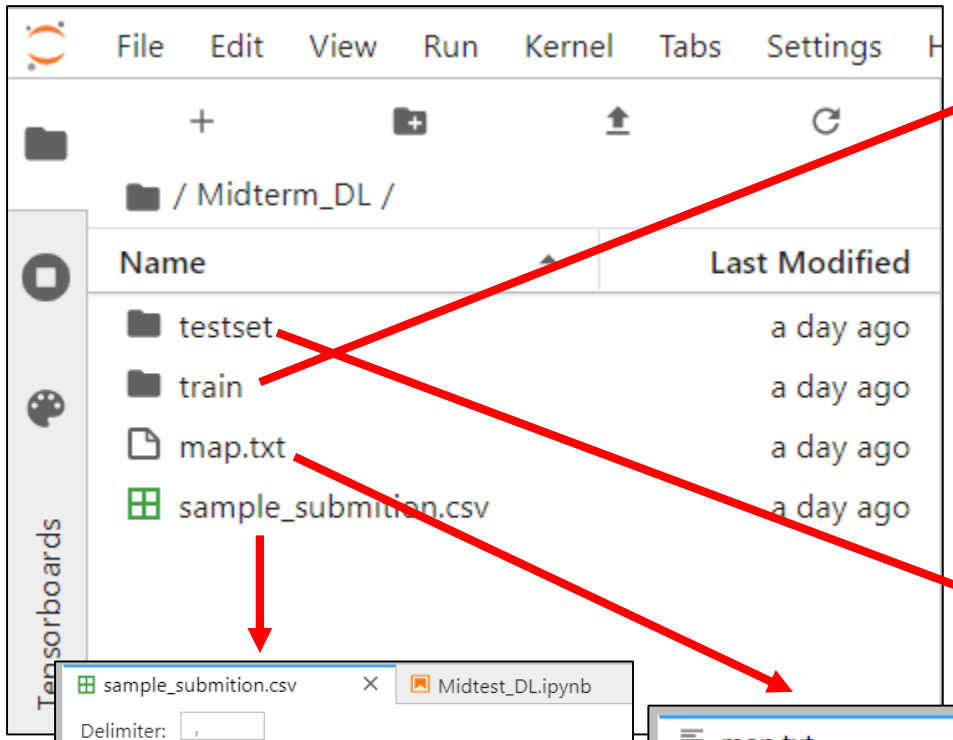
評量標準

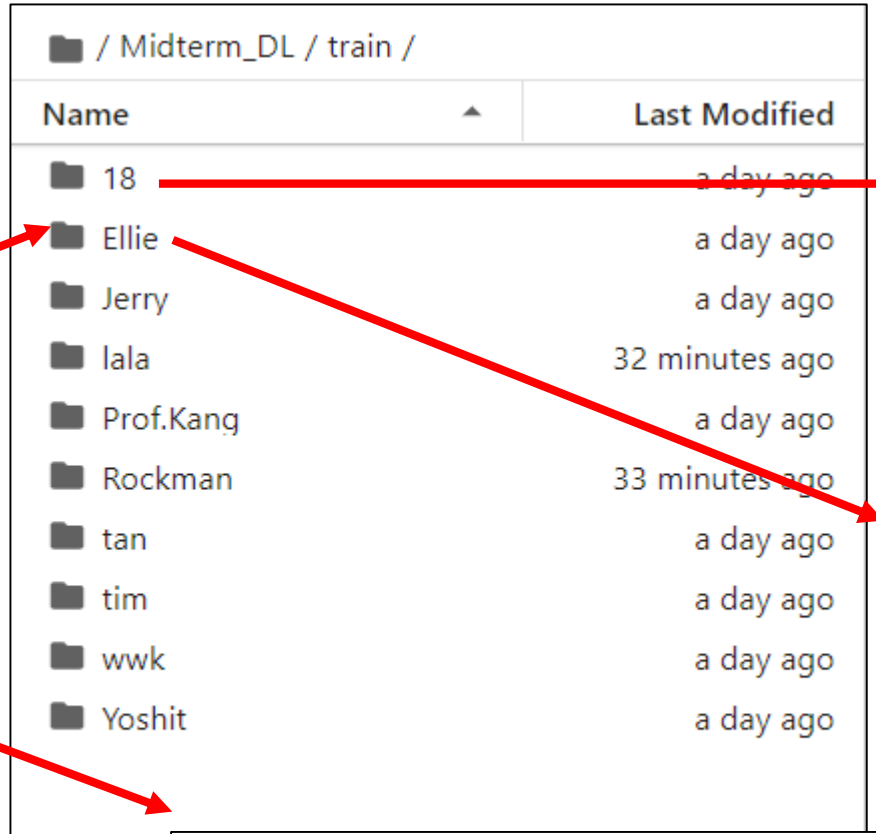
- 以test資料集分類結果的準確率,做為評比標準。
- 合格成績需比 Leaderboard 的 Baseline 有更高的 Accuracy。

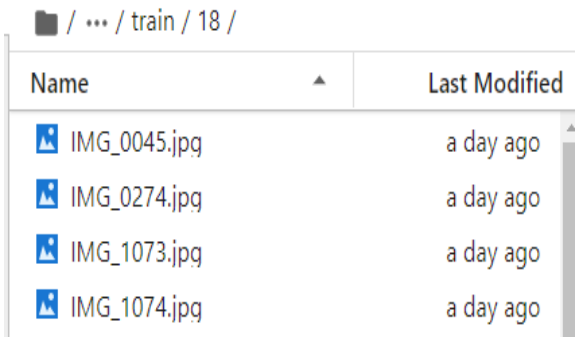
其他規定

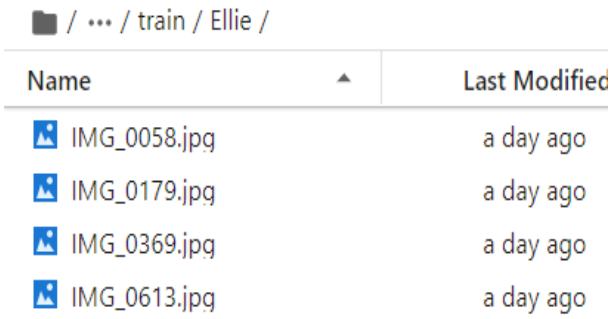
- 一人一組, 嚴禁共享、抄襲程式碼之情事。
- 請保留自己每一次提交的程式碼，助教將會進行Code Review,
  以確認大家對整體流程與程式碼有一定掌握程度。

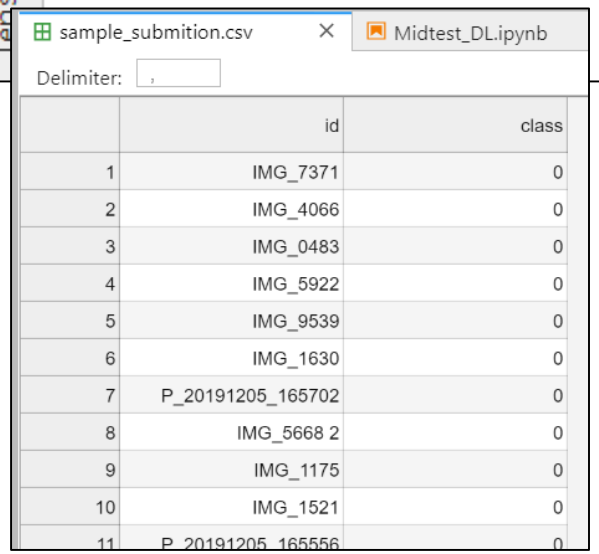
- 資料存放

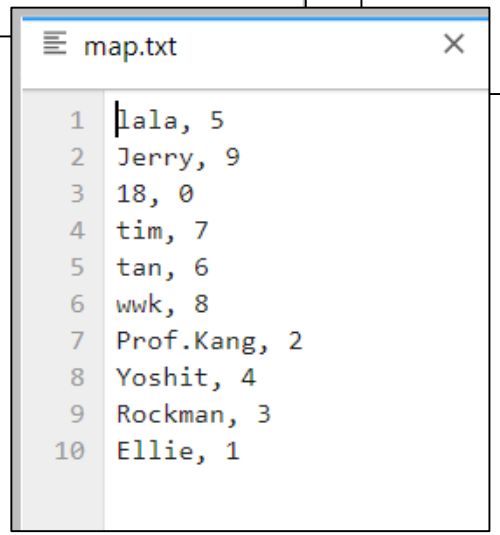

File Edit View Run Kernel Tabs Settings
/ Midterm_DL /
Name
Last Modified
testset
a day ago
train
a day ago
map.txt
a day ago
sample_submition.csv
a day ago
Tensorboards

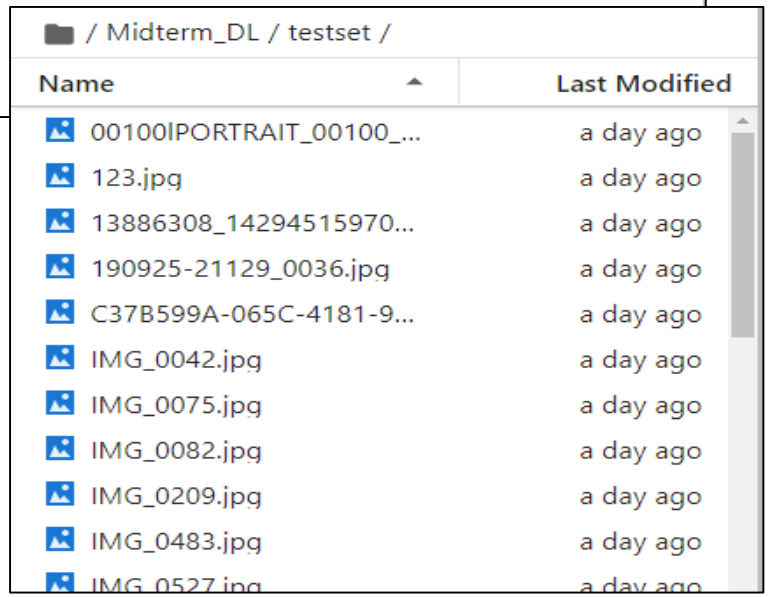

/ Midterm_DL / train /
Name
Last Modified
18
a day ago
Ellie
a day ago
Jerry
a day ago
lala
32 minutes ago
Prof.Kang
a day ago
Rockman
33 minutes ago
tan
a day ago
tim
a day ago
wwk
a day ago
Yoshit
a day ago


/ ••• / train / 18 /
Name
Last Modified
IMG_0045.jpg
a day ago
IMG_0274.jpg
a day ago
IMG_1073.jpg
a day ago
IMG_1074.jpg
a day ago


/ ••• / train / Ellie /
Name
Last Modified
IMG_0058.jpg
a day ago
IMG_0179.jpg
a day ago
IMG_0369.jpg
a day ago
IMG_0613.jpg
a day ago


sample_submition.csv
Midtest_DL.ipynb
Delimiter: ,
id
class
1 IMG_7371 0
2 IMG_4066 0
3 IMG_0483 0
4 IMG_5922 0
5 IMG_9539 0
6 IMG_1630 0
7 P_20191205_165702 0
8 IMG_5668 2 0
9 IMG_1175 0
10 IMG_1521 0
11 P_20191205_165556 0


map.txt
1 lala, 5
2 Jerry, 9
3 18, 0
4 tim, 7
5 tan, 6
6 wwk, 8
7 Prof.Kang, 2
8 Yoshit, 4
9 Rockman, 3
10 Ellie, 1


/ Midterm_DL / testset /
Name
Last Modified
00100lPORTRAIT_00100_...
a day ago
123.jpg
a day ago
13886308_14294515970...
a day ago
190925-21129_0036.jpg
a day ago
C37B599A-065C-4181-9...
a day ago
IMG_0042.jpg
a day ago
IMG_0075.jpg
a day ago
IMG_0082.jpg
a day ago
IMG_0209.jpg
a day ago
IMG_0483.jpg
a day ago

- 資料

class_mapping = {'18': 0,'Ellie': 1,'Prof.Kang': 2,'Rockman': 3,'Yoshit': 4,'lala': 5, 'tan': 6,'tim': 7, 'wwk': 8,'Jerry': 9}

18 : 61 張

Ellie : 64 張

Prof.Kang: 53 張

Rockman : 79 張

Yoshit : 42 張

lala : 40 張

tan : 80 張

tim : 54 張

wwk : 46 張

Jerry : 24 張

- 針對各類別資料量少的，增加資料量

```
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
import numpy as np
datagen = ImageDataGenerator(
            rotation_range=1,
            width_shift_range=0.01,
            height_shift_range=0.01,
            shear_range=0.02,
            #zoom_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest')

gener=datagen.flow_from_directory(r'C:\Users\user\Desktop\Gen_yoshit', #類別子文件夾的上一級文件夾
                                  batch_size=20,
                                  shuffle=False,
                                  save_to_dir=r'C:\Users\user\Desktop\yoshit',
                                  save_prefix='trans_',
                                  save_format='jpg')
for i in range(5):
    gener.next()
```

本例對 Jerry， lala，Yoshit 做資料擴增

程 式 說 明

輸入會用到的套件

```
import glob
import matplotlib.pyplot as plt
import numpy as np
import cv2
from keras.utils import np_utils
%matplotlib inline
```

```
Using TensorFlow backend.
```

定義類別與其整數對應

```
#定義類別與其整數對應

class_mapping = {'18': 0,'Ellie': 1,'Prof.Kang': 2,'Rockman': 3,'Yoshit': 4,'lala': 5, 'tan': 6,'tim': 7, 'wwk': 8,'Jerry': 9}
```

讀取每張圖片，與其標籤

```
#讀取每張圖片，與其標籤
# OpenCV 本身就有提供讀取圖片檔的函數可用，讀取一般的圖片檔，只要呼叫 cv2.imread 即可將圖片讀取進來
# 以 cv2.imread 讀進來的資料，會儲存成一個 NumPy 的陣列

#   在OpenCV中，图像不是用常规的RGB颜色通道来存储的，它们用的是BGR顺序。当读取一幅图像后，
#   默认的是BGR，不过有很多转换方式是可以利用的。颜色空间转换可以用函数cvtColor()函数

# # 使用 OpenCV 讀取圖檔
# img_bgr = cv2.imread('image.jpg')

# # 將 BGR 圖片轉為 RGB 圖片
# img_rgb = img_bgr[:,:,::-1]

# # 或是這樣亦可
# # img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

# # 使用 Matplotlib 顯示圖片
# plt.imshow(img_rgb)
# plt.show()


img_paths = glob.glob('/home/jovyan/Midterm_DL/train/*/*.jpg')
images, labels = [], []
img_sizes = []
for img_path in img_paths:
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (299, 299))
    if img.shape not in img_sizes:
        img_sizes.append(img.shape)
    label = img_path.split('/')[-2]
    label = class_mapping.get(label)
    images.append(img)
    labels.append(label)
images = np.array(images)
labels = np.uint8(labels)
```

```
images.shape
```

```
(588, 299, 299, 3)
```

把照片 reshape 與 標準化

```
x_Train4D = images.reshape(images.shape[0],299, 299,3).astype('float32')
x_Train4D.shape
```

```
(535, 299, 299, 3)
```

```
x_Train4D_normalize = x_Train4D/255
```

把資料拆成訓練集與驗證集

```
from sklearn.model_selection import train_test_split
X_train, X_valid, y_train_1, y_valid_1 = train_test_split(x_Train4D_normalize,
                                                          labels,
                                                          test_size=0.2,
                                                          )
```

對 Y 做 one-hot encoding

```
y_train = np_utils.to_categorical(y_train_1)
y_valid = np_utils.to_categorical(y_valid_1)
```

結果:

```
X_train.shape
```

```
(470, 299, 299, 3)
```

```
y_train.shape
```

```
(470, 10)
```

```
X_valid.shape
```

```
(118, 299, 299, 3)
```

```
y_valid.shape
```

```
(118, 10)
```

輸入建模會用到的套件

```
import tensorflow.keras as keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization, GlobalAveragePooling2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications.xception import Xception

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

使用 Transfer Learning，其模型為: Xception，每層參數會做 Fine-Tune

```
model_name = 'Xception-Fine-Tune'
num_classes = 10 本例共 10 類

img_rows, img_cols, img_channel = 299, 299, 3
base_model = Xception(weights='imagenet', include_top=False,
                      input_shape=(img_rows, img_cols, img_channel))

x = base_model.output
x = GlobalAveragePooling2D(data_format='channels_last')(x)
x = Dropout(0.5)(x)
predictions = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

model.summary()
```

使用 GlobalAveragePooling2D，不使用 Flatten
=>減少參數量 (避免 overfitting)

```
Model: "model"
_________________________________________________________________________________________
Layer (type)                 Output Shape         Param #     Connected to
```

建模

資料增強

```
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest')

optimizer = keras.optimizers.Adam(lr=10e-6)

model_path = './saved_models/{}.h5'.format(model_name)

checkpoint = ModelCheckpoint(model_path, monitor='val_loss', save_best_only=True, verbose=1)
earlystop = EarlyStopping(monitor='val_loss', patience=5, verbose=1)

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

epochs = 100
batch_size = 10

model_history = model.fit_generator(datagen.flow(X_train, y_train, batch_size = batch_size),
                                    steps_per_epoch = 100 ,
                                    epochs = epochs,
                                    validation_data = (X_valid, y_valid),
                                    callbacks = [checkpoint, earlystop])
```

資料增強目的:
若訓練資料只有舉右手照片，則預測舉左手照片一定不佳。所以要資料增強，生成舉左手的照片

儲存最佳 Model (val_loss 最小)

跑演算時，若連續 5 回 val_loss 未下降，則停止

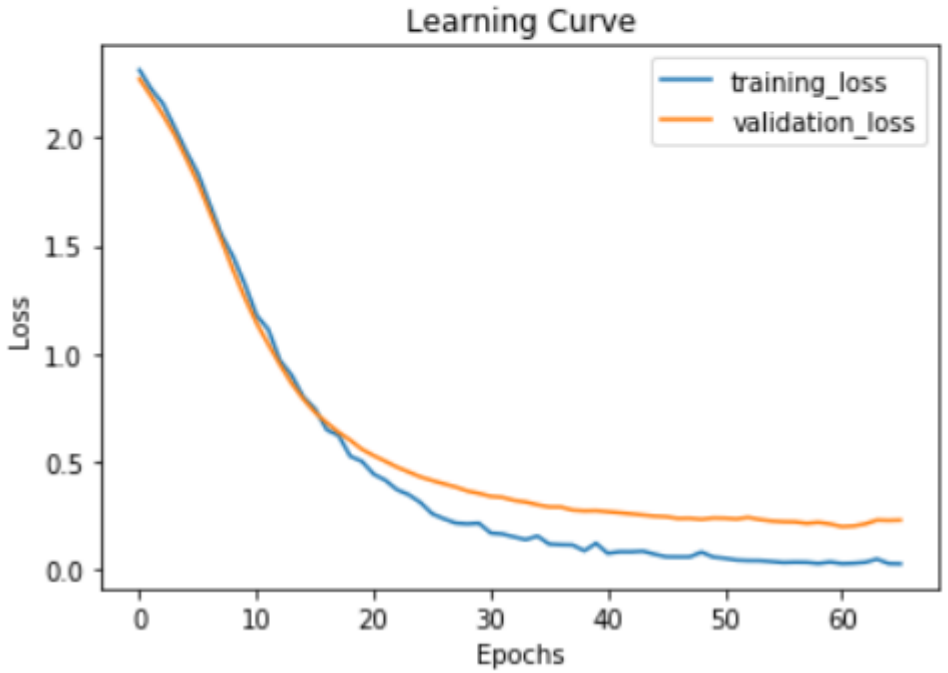
使用 1000 張照片來 training

```
Epoch 1/100
 99/100 [============================>.] - ETA: 0s - loss: 2.3070 - accuracy: 0.1332
Epoch 00001: val_loss improved from inf to 2.19266, saving model to ./saved_models/Xception-Fine-Tune.h5
100/100 [==============================] - 36s 360ms/step - loss: 2.3052 - accuracy: 0.1329 - val_loss: 2.1927 - val_accuracy: 0.2250
Epoch 2/100
 99/100 [============================>.] - ETA: 0s - loss: 2.1103 - accuracy: 0.2446
Epoch 00002: val_loss improved from 2.19266 to 2.05945, saving model to ./saved_models/Xception-Fine-Tune.h5
100/100 [==============================] - 37s 374ms/step - loss: 2.1088 - accuracy: 0.2472 - val_loss: 2.0595 - val_accuracy: 0.4375
```

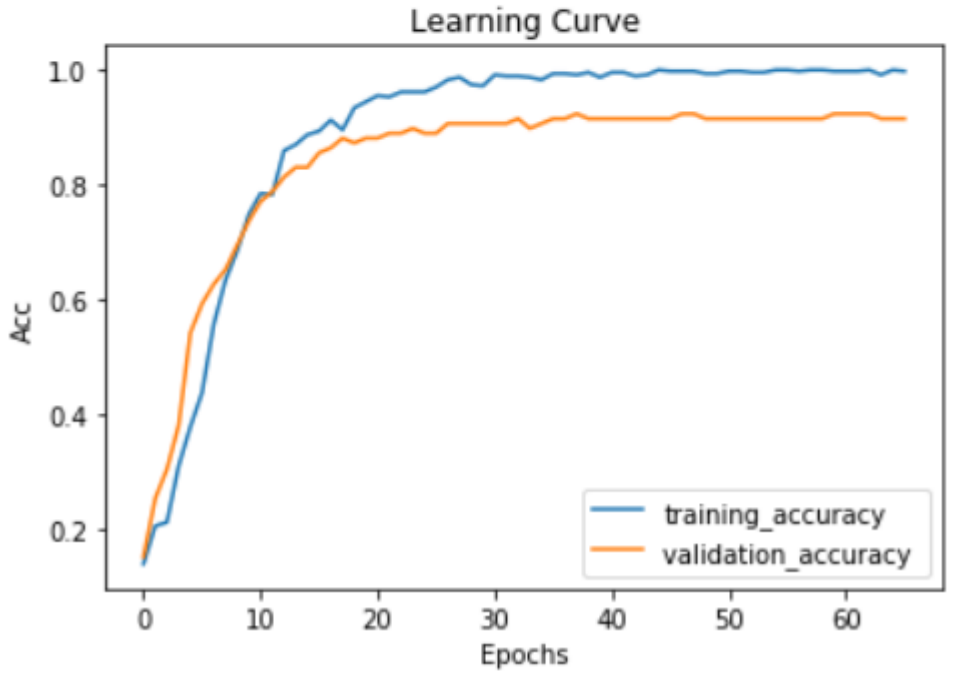
畫 Training 與 Validation 的 Loss & Accuracy

目的: 判斷 Model 有無 Overfitting (high variance) or Underfitting (high bias)

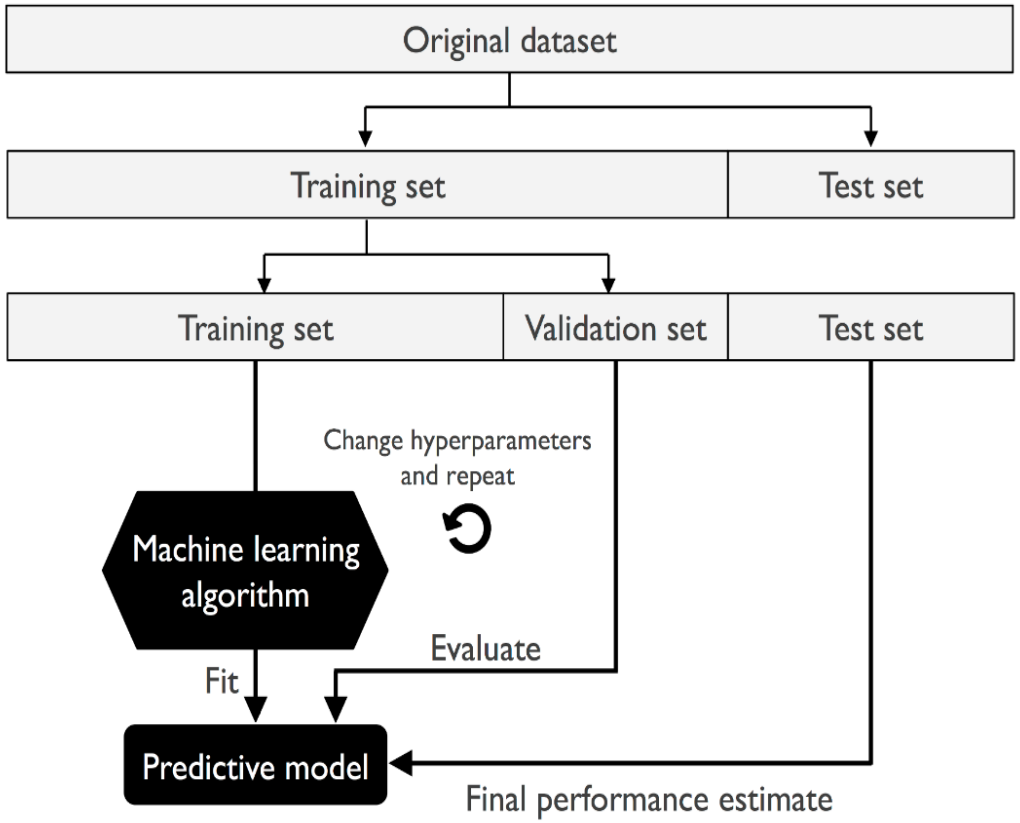
```
training_loss = model_history.history['loss']
val_loss = model_history.history['val_loss']

plt.plot(training_loss, label="training_loss")
plt.plot(val_loss, label="validation_loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Learning Curve")
plt.legend(loc='best')
plt.show()
```




```
training_acc = model_history.history['accuracy']
val_acc = model_history.history['val_accuracy']

plt.plot(training_acc, label="training_accuracy")
plt.plot(val_acc, label="validation_accuracy ")
plt.xlabel("Epochs")
plt.ylabel("Acc")
plt.title("Learning Curve")
plt.legend(loc='best')
plt.show()
```



Validation Set 預測錯誤分析



正確方式為對 Test Set 做預測錯誤分析

本例因為資料量少，資料只拆成 Training Set 與 Validation Set，因此對 Validation Set 做預測錯誤分析

Check 所儲存的 model 為最佳 Model (val_loss 最小)

```
model = load_model(model_path)

scores = model.evaluate(X_valid, y_valid, verbose=1)
print('Validation loss:', scores[0])
print('Validation accuracy:', scores[1])
```

```
118/1 [========================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
===============================================================
Validation loss: 0.20058701035834975
Validation accuracy: 0.9237288
```

Show 出 Vilidation Set 的預測值

```
X_valid_prediction = model.predict(X_valid)
prediction_X_valid = X_valid_prediction.argmax(axis=-1)
```

```
prediction_X_valid
```

```
array([4, 3, 6, 4, 7, 8, 5, 3, 8, 3, 6, 3, 0, 5, 3, 2, 0, 2, 0, 3, 9, 1,
       4, 3, 1, 6, 1, 1, 0, 6, 3, 9, 2, 3, 9, 9, 9, 0, 9, 5, 9, 6, 7, 6,
       0, 1, 9, 9, 5, 7, 0, 7, 1, 8, 7, 0, 1, 5, 4, 8, 6, 3, 1, 0, 3, 4,
       1, 1, 1, 0, 0, 4, 8, 6, 3, 6, 3, 2, 3, 3, 3, 5, 4, 1, 1, 5, 6, 8,
       8, 6, 7, 5, 2, 6, 2, 7, 2, 3, 5, 1, 7, 2, 0, 8, 7, 6, 9, 0, 1, 9,
       2, 7, 1, 8, 0, 4, 4, 6])
```

畫出 crosstab

```
import pandas as pd
pd.crosstab(y_valid_1,prediction_X_valid, rownames=['label'],colnames=['predict'])
```

predict	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
label										
0	13	0	0	0	0	0	0	0	0	0
1	0	14	0	0	0	0	0	0	0	0
2	0	0	8	2	0	0	1	0	0	0
3	0	1	0	15	1	0	0	0	0	0
4	1	0	0	0	8	0	0	0	0	0
5	0	1	0	0	0	9	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0
7	0	0	0	0	0	0	0	9	0	0
8	0	0	0	0	0	0	0	1	9	0
9	0	0	1	0	0	0	0	0	0	11

把 class_mapping 做 inverse mapping

```
df = pd.DataFrame({'label':y_valid_1 , 'predict':prediction_X_valid})
```

```
class_mapping = {'18': 0,'Ellie': 1,'Prof.Kang': 2,'Rockman': 3,'Yoshit': 4,'lala': 5, 'tan': 6,'tim': 7, 'wwk': 8,'Jerry': 9}
inv_mapping = {v: k for k , v in class_mapping.items()}
inv_mapping
```

```
{0: '18',
 1: 'Ellie',
 2: 'Prof.Kang',
 3: 'Rockman',
 4: 'Yoshit',
 5: 'lala',
 6: 'tan',
 7: 'tim',
 8: 'wwk',
 9: 'Jerry'}
```

```
label_1 = []
for i in range(len(df)) :
    aa = inv_mapping[df['label'][i]]
    label_1.append(aa)
```

```
label_1 = pd.DataFrame(label_1 , columns = ['label_1'])
df = pd.concat([df , label_1] , axis=1)
df
```

	label	predict	label_1
0	1	1	Ellie
1	4	4	Yoshit
2	4	4	Yoshit

```
predict_1 = []
for i in range(len(df)) :
    aa = inv_mapping[df['predict'][i]]
    predict_1.append(aa)
```

```
predict_1 = pd.DataFrame(predict_1 , columns = ['predict_1'])
df = pd.concat([df , predict_1] , axis=1)
df
```

	label	predict	label_1	predict_1
0	1	1	Ellie	Ellie
1	4	4	Yoshit	Yoshit
2	4	4	Yoshit	Yoshit

畫出 inverse mapping 的 crosstab

```
pd.crosstab(df['label_1'],df['predict_1'], rownames=['label'],colnames=['predict'])
```

predict	18	Ellie	Jerry	Prof.Kang	Rockman	Yoshit	lala	tan	tim	wwk
label										
18	13	0	0	0	0	0	0	0	0	0
Ellie	0	14	0	0	0	0	0	0	0	0
Jerry	0	0	11	1	0	0	0	0	0	0
Prof.Kang	0	0	0	8	2	0	0	1	0	0
Rockman	0	1	0	0	15	1	0	0	0	0
Yoshit	1	0	0	0	0	8	0	0	0	0
lala	0	1	0	0	0	0	9	0	0	0
tan	0	0	0	0	0	0	0	13	0	0
tim	0	0	0	0	0	0	0	0	9	0
wwk	0	0	0	0	0	0	0	0	1	9

找出 Vilidation Set 預測錯誤的 item

```
df_wrong = df[df.label_1!=df.predict_1]
df_wrong
```

	label	predict	label_1	predict_1
1	2	3	Prof.Kang	Rockman
24	3	1	Rockman	Ellie
30	2	3	Prof.Kang	Rockman
37	4	0	Yoshit	18
51	8	7	wwk	tim
67	5	1	lala	Ellie
75	2	6	Prof.Kang	tan
94	9	2	Jerry	Prof.Kang
116	3	4	Rockman	Yoshit

找出 Vilidation Set 預測錯誤的照片

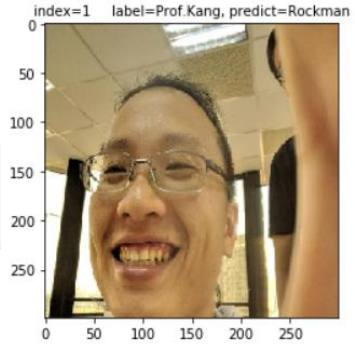
```
wrongindex = df_wrong.index
wrongindex
```

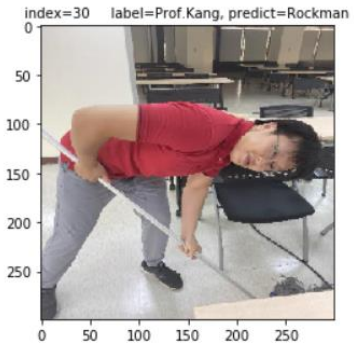
```
Int64Index([1, 24, 30, 37, 51, 67, 75, 94, 116], dtype='int64')
```

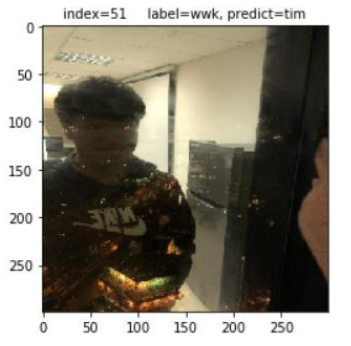
```
df_wrong['label_1'].iloc[1] , df_wrong['predict_1'].iloc[1] , wrongindex[1]
```

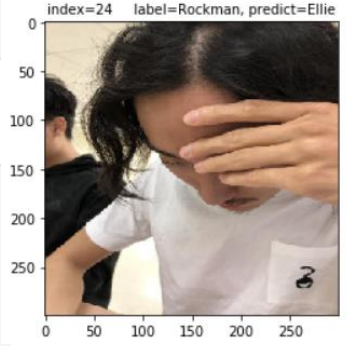
```
('Rockman', 'Ellie', 24)
```

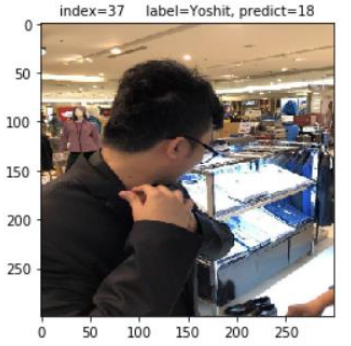
```
import matplotlib.pyplot as plt
fig = plt.gcf()

for i in range(len(wrongindex)) :
    plt.figure()
    plt.subplot(2,1+len(wrongindex)/2,1+i)

    fig.set_size_inches(3, 3)
    ax = plt.axes()
    plt.imshow(X_valid[wrongindex[i]])

    title ="index=" +str(wrongindex[i])
    ax.set_title(title,fontsize=10)
    title+="     label=" +str(df_wrong['label_1'].iloc[i])
    ax.set_title(title,fontsize=10)
    title+=", predict=" +str(df_wrong['predict_1'].iloc[i])
    ax.set_title(title,fontsize=10)
```

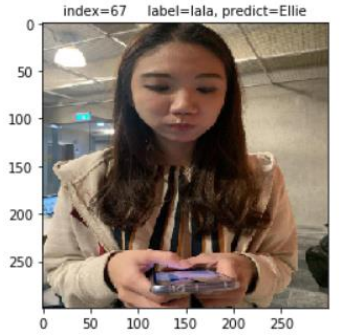


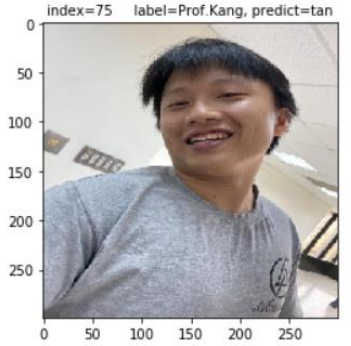


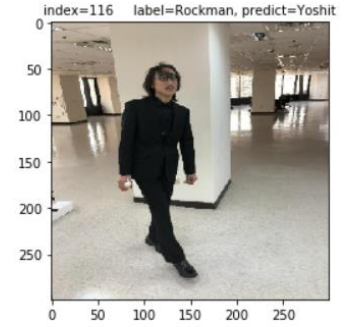


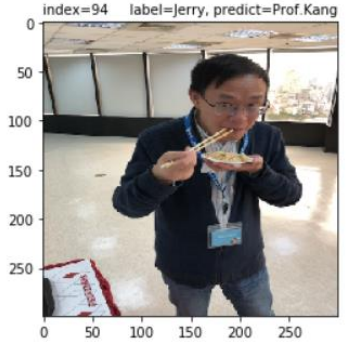












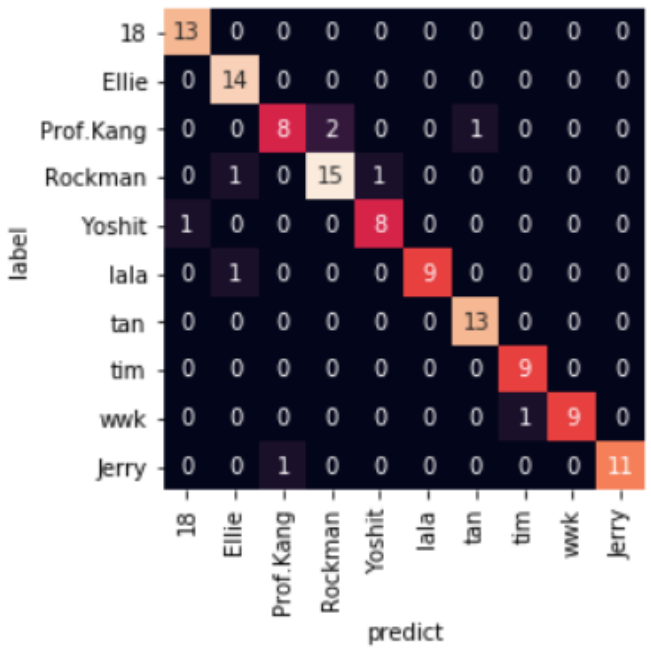
找出 Vilidation Set 預測錯誤的照片 (Subplot)

```
import matplotlib.pyplot as plt
plt.figure(figsize=(30,30))
for i in range(len(wrongindex)) :
    plt.subplot(int(len(wrongindex)/5)+1 , 5, i+1 )
    plt.imshow(X_valid[wrongindex[i]])
    plt.gca().set_title(["index=" +str(wrongindex[i]),"label=" +str(df_wrong['label_1'].iloc[i]),"predict=" +str(df_wrong['predict_1'].iloc[i])])
```

做出 Confusion Matrix , Classification Report

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
mat = confusion_matrix(y_valid_1 , prediction_X_valid)
sns.heatmap(mat,square=True,annot=True,fmt='d',cbar=False,
            xticklabels=mapping , yticklabels=mapping)
plt.xlabel('predict')
plt.ylabel('label')
```

```
Text(91.68, 0.5, 'label')
```




```
from sklearn.metrics import classification_report
print(classification_report(y_valid_1 , prediction_X_valid , target_names=mapping))
```

```
              precision    recall  f1-score   support

          18       0.93      1.00      0.96        13
       Ellie       0.88      1.00      0.93        14
   Prof.Kang       0.89      0.73      0.80        11
     Rockman       0.88      0.88      0.88        17
      Yoshit       0.89      0.89      0.89         9
        lala       1.00      0.90      0.95        10
         tan       0.93      1.00      0.96        13
         tim       0.90      1.00      0.95         9
         wwk       1.00      0.90      0.95        10
       Jerry       1.00      0.92      0.96        12

    accuracy                           0.92       118
   macro avg       0.93      0.92      0.92       118
weighted avg       0.93      0.92      0.92       118
```

預測 Test Set

Test Set 資料讀取，整理，預測

```
img_paths = glob.glob('/home/jovyan/Midterm_DL/testset/*.jpg')
images = []
img_sizes = []
ids = []
for img_path in img_paths:
    ids.append(img_path.split("/")[-1].split(".")[0])
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (299, 299))
    if img.shape not in img_sizes:
        img_sizes.append(img.shape)

    images.append(img)

images_1 = np.array(images)
```

```
x_Test4D = images_1.reshape(images_1.shape[0],299,299,3).astype('float32')
```

```
x_Test4D_normalize = x_Test4D/255
```

```
y_test_pred_prob = model.predict(x_Test4D_normalize)
prediction = y_test_pred_prob.argmax(axis=-1)
```

```
prediction[:10]
```

```
array([5, 5, 2, 2, 3, 8, 7, 3, 8, 0])
```

Summit to Kaggle

Summit to Kaggle

```
submit = pd.read_csv("/home/jovyan/Midterm_DL/sample_submition.csv")
```

```
import pandas as pd
submit = pd.DataFrame({"id":ids, "class":prediction})
```

```
print(submit)
```

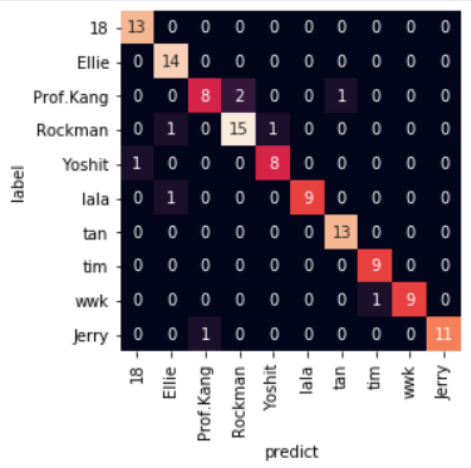
```
              id  class
0       IMG_5922      5
1       IMG_4066      5
2       IMG_1194      2
3    S__20275217      2
4       IMG_7365      3
..            ...    ...
127     IMG_0082      8
128     IMG_4833      4
129     IMG_0042      1
130     IMG_8804      3
131     IMG_7282      2

[132 rows x 2 columns]
```

```
submit.to_csv('1214.csv', index = False)
```

Majority Vote

各模型 Confusion Matrix , Classification Report

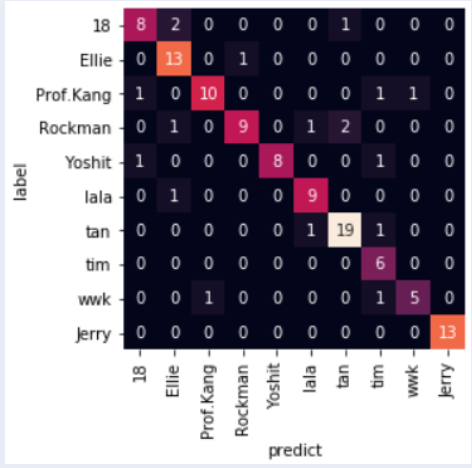
Xception



	precision	recall	f1-score	support
18	0.93	1.00	0.96	13
Ellie	0.88	1.00	0.93	14
Prof.Kang	0.89	0.73	0.80	11
Rockman	0.88	0.88	0.88	17
Yoshit	0.89	0.89	0.89	9
lala	1.00	0.90	0.95	10
tan	0.93	1.00	0.96	13
tim	0.90	1.00	0.95	9
wwk	1.00	0.90	0.95	10
Jerry	1.00	0.92	0.96	12
accuracy			0.92	118
macro avg	0.93	0.92	0.92	118
weighted avg	0.93	0.92	0.92	118

Prof.Kang 會判成 Rockman

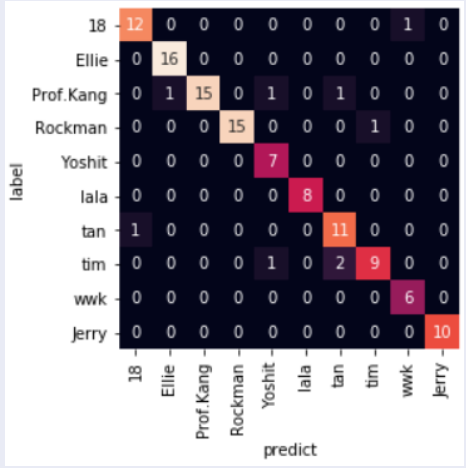
VGG



	precision	recall	f1-score	support
18	0.80	0.73	0.76	11
Ellie	0.76	0.93	0.84	14
Prof.Kang	0.91	0.77	0.83	13
Rockman	0.90	0.69	0.78	13
Yoshit	1.00	0.80	0.89	10
lala	0.82	0.90	0.86	10
tan	0.86	0.90	0.88	21
tim	0.60	1.00	0.75	6
wwk	0.83	0.71	0.77	7
Jerry	1.00	1.00	1.00	13
accuracy			0.85	118
macro avg	0.85	0.84	0.84	118
weighted avg	0.86	0.85	0.85	118

18 會判成 Ellie
Rockman 會判成 tan

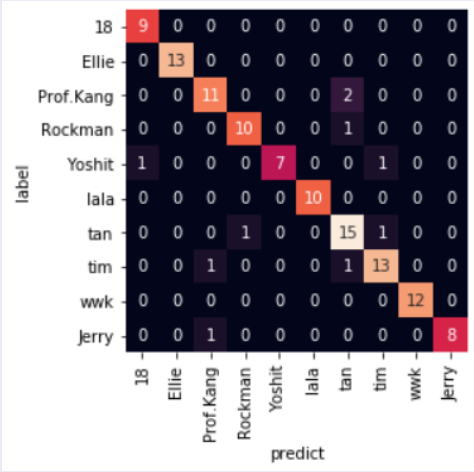
ResNet50



	precision	recall	f1-score	support
18	0.92	0.92	0.92	13
Ellie	0.94	1.00	0.97	16
Prof.Kang	1.00	0.83	0.91	18
Rockman	1.00	0.94	0.97	16
Yoshit	0.78	1.00	0.88	7
lala	1.00	1.00	1.00	8
tan	0.79	0.92	0.85	12
tim	0.90	0.75	0.82	12
wwk	0.86	1.00	0.92	6
Jerry	1.00	1.00	1.00	10
accuracy			0.92	118
macro avg	0.92	0.94	0.92	118
weighted avg	0.93	0.92	0.92	118

tim 會判成 tan

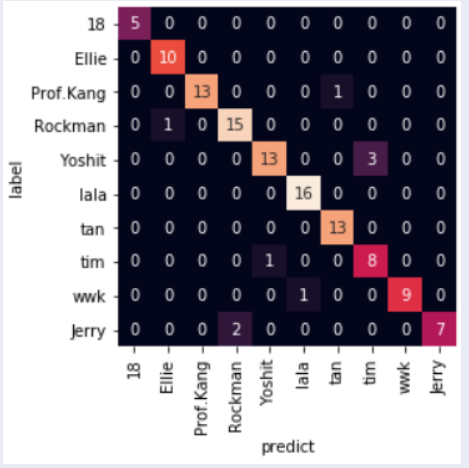
DenseNet201



	precision	recall	f1-score	support
18	0.90	1.00	0.95	9
Ellie	1.00	1.00	1.00	13
Prof.Kang	0.85	0.85	0.85	13
Rockman	0.91	0.91	0.91	11
Yoshit	1.00	0.78	0.88	9
lala	1.00	1.00	1.00	10
tan	0.79	0.88	0.83	17
tim	0.87	0.87	0.87	15
wwk	1.00	1.00	1.00	12
Jerry	1.00	0.89	0.94	9
accuracy			0.92	118
macro avg	0.93	0.92	0.92	118
weighted avg	0.92	0.92	0.92	118

Prof.Kang 會判成 tan

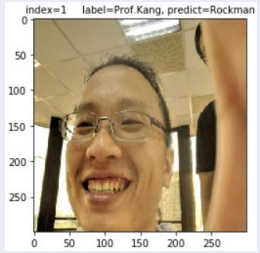
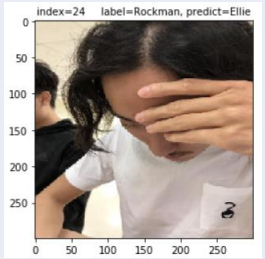
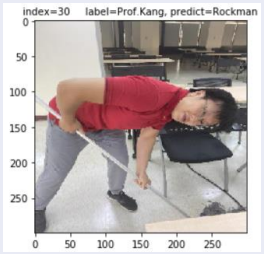
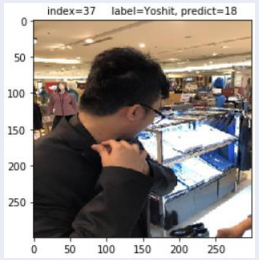
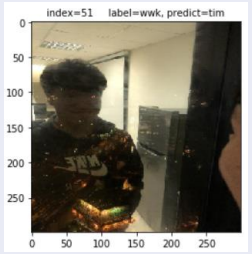
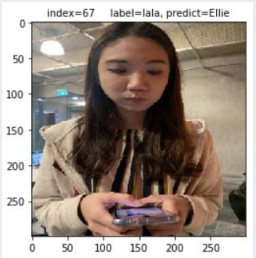
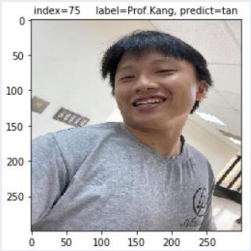
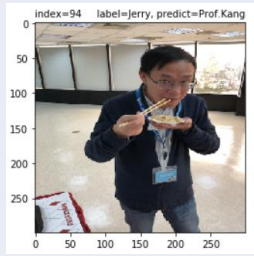
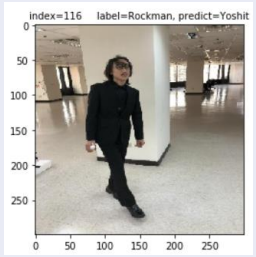
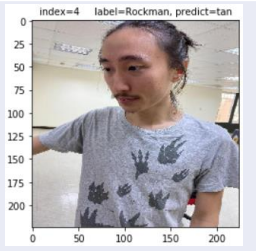
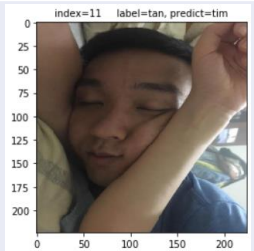
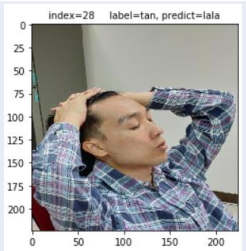
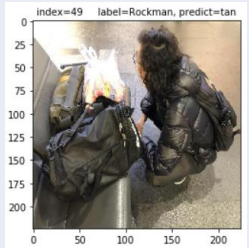
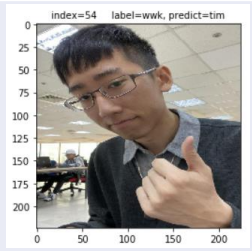
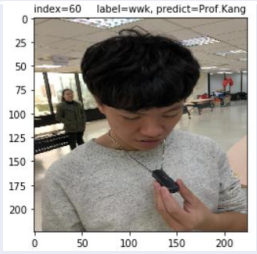
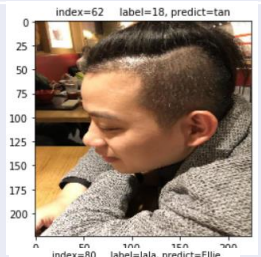
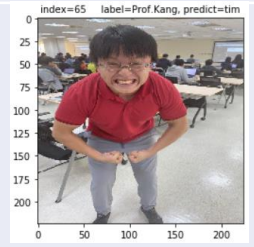
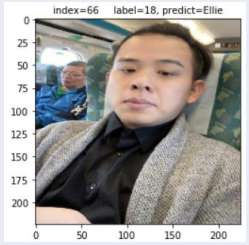
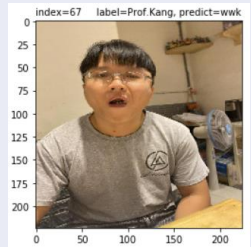
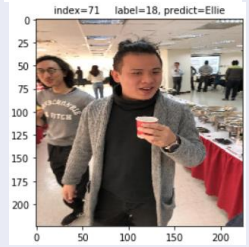
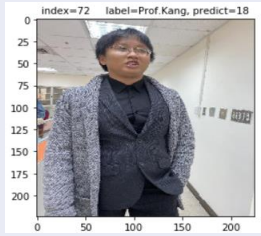
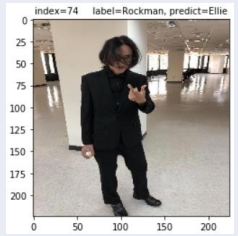
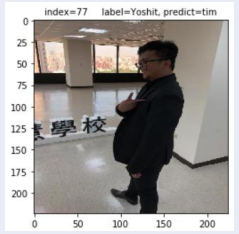
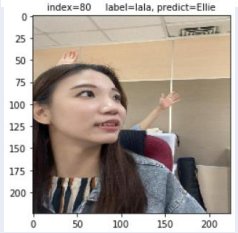
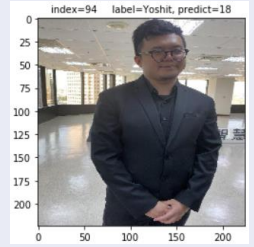
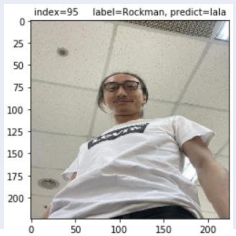
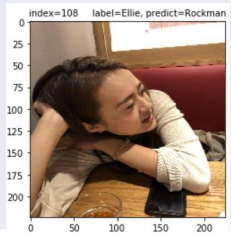
InceptionResNetV2

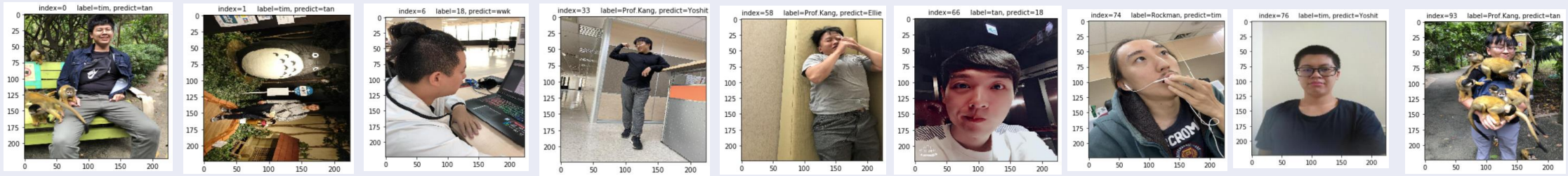
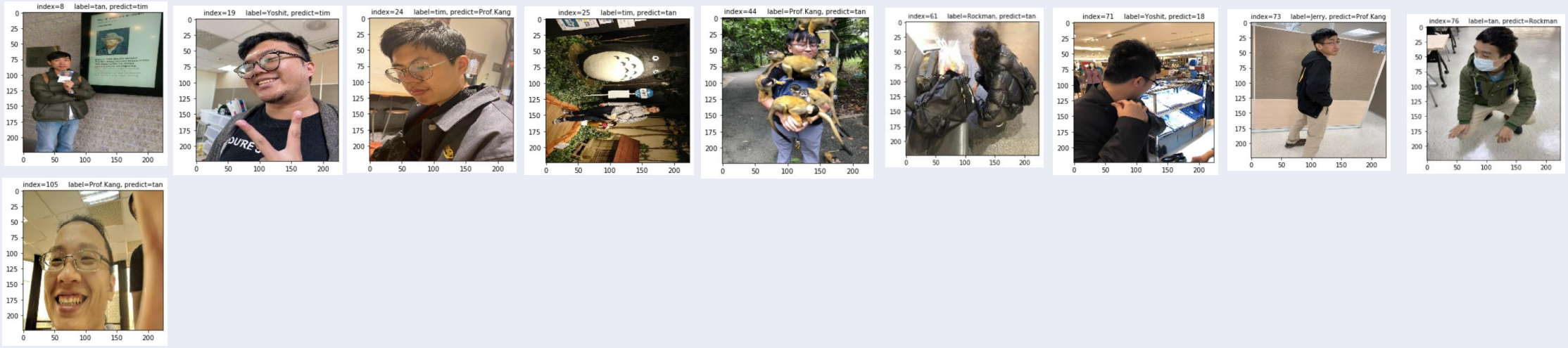
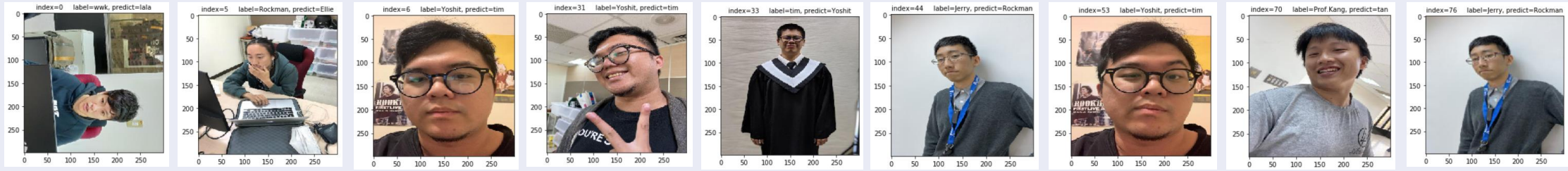


	precision	recall	f1-score	support
18	1.00	1.00	1.00	5
Ellie	0.91	1.00	0.95	10
Prof.Kang	1.00	0.93	0.96	14
Rockman	0.88	0.94	0.91	16
Yoshit	0.93	0.81	0.87	16
lala	0.94	1.00	0.97	16
tan	0.93	1.00	0.96	13
tim	0.73	0.89	0.80	9
wwk	1.00	0.90	0.95	10
Jerry	1.00	0.78	0.88	9
accuracy			0.92	118
macro avg	0.93	0.92	0.92	118
weighted avg	0.93	0.92	0.92	118

Yoshit 會判成 tim

各模型 Validation Set 預測錯誤整理

Model	Picture
Xception	        
VGG	                 

Model	Picture
ResNet50	
DenseNet201	
InceptionResNetV2	

程式碼

```
import numpy as np
import pandas as pd
```

```
Xception = pd.read_csv("Xception.csv")
ResNet50 = pd.read_csv("ResNet50.csv")
VGG = pd.read_csv("VGG.csv")
DenseNet201 = pd.read_csv("DenseNet201.csv")
InceptionResNetV2 = pd.read_csv("InceptionResNetV2.csv")
```

```
Majority_Vote = Xception
Majority_Vote['Xception'] = Xception['class']
Majority_Vote = Majority_Vote.drop(columns=["class"]).copy()
```

```
Majority_Vote = Majority_Vote.merge(ResNet50,how="left",on="id")

# 把沒有匹配到的NaN 改為 0
# tables = Majority_Vote.merge(ResNet50,how="left",on="id").fillna('--')
# Majority_Vote = Majority_Vote.merge(ResNet50,how="left",on="id").fillna('--')
Majority_Vote = Majority_Vote.rename(columns={'class':'ResNet50'})
```

```
Majority_Vote = Majority_Vote.merge(VGG,how="left",on="id")
Majority_Vote = Majority_Vote.rename(columns={'class':'VGG'})
```

```
Majority_Vote = Majority_Vote.merge(DenseNet201,how="left",on="id")
Majority_Vote = Majority_Vote.rename(columns={'class':'DenseNet201'})
```

```
Majority_Vote = Majority_Vote.merge(InceptionResNetV2,how="left",on="id")
Majority_Vote = Majority_Vote.rename(columns={'class':'InceptionResNetV2'})
```

```
Majority_Vote
```

	id	Xception	ResNet50	VGG	DenseNet201	InceptionResNetV2
0	IMG_5922	5	5	5	5	5
1	IMG_4066	5	5	5	5	5
2	IMG_1194	2	2	2	2	2
3	S__20275217	2	2	7	2	2
4	IMG_7365	3	3	3	3	3
...	...	...	...	...	...	...
127	IMG_0082	8	8	2	8	8
128	IMG_4833	4	4	4	4	4
129	IMG_0042	5	1	1	1	1
130	IMG_8804	3	3	3	3	3
131	IMG_7282	2	2	2	2	2

132 rows × 6 columns

程式碼

```
all_Majority_Vote = []
from scipy import stats
for i in range(len(Majority_Vote)) :
    bb = stats.mode(Majority_Vote.iloc[i,[1,2,3,4,5]])
    bb = bb[0][0]
    all_Majority_Vote.append(bb)
```

```
all_Majority_Vote = pd.DataFrame(all_Majority_Vote , columns = ['class'])
Majority_Vote = pd.concat([Majority_Vote , all_Majority_Vote] , axis=1)
```

```
Majority_Vote.to_csv('Majority_Vote.csv', index = False)
Majority_Vote
```

	id	Xception	ResNet50	VGG	DenseNet201	InceptionResNetV2	class
0	IMG_5922	5	5	5	5	5	5
1	IMG_4066	5	5	5	5	5	5
2	IMG_1194	2	2	2	2	2	2
3	S__20275217	2	2	7	2	2	2
4	IMG_7365	3	3	3	3	3	3
...	...	...	...	...	...	...	...
127	IMG_0082	8	8	2	8	8	8
128	IMG_4833	4	4	4	4	4	4
129	IMG_0042	5	1	1	1	1	1
130	IMG_8804	3	3	3	3	3	3
131	IMG_7282	2	2	2	2	2	2

132 rows × 7 columns

```
Majority_Vote = Majority_Vote.drop(columns=["ResNet50"]).copy()
Majority_Vote = Majority_Vote.drop(columns=["DenseNet201"]).copy()
Majority_Vote = Majority_Vote.drop(columns=["InceptionResNetV2"]).copy()
Majority_Vote = Majority_Vote.drop(columns=["Xception"]).copy()
Majority_Vote = Majority_Vote.drop(columns=["VGG"]).copy()
```

```
Majority_Vote.to_csv('Majority_Vote.csv', index = False)
Majority_Vote
```

	id	class
0	IMG_5922	5
1	IMG_4066	5
2	IMG_1194	2
3	S__20275217	2
4	IMG_7365	3
...	...	...
127	IMG_0082	8
128	IMG_4833	4
129	IMG_0042	1
130	IMG_8804	3
131	IMG_7282	2

132 rows × 2 columns

Kaggle Result:

Submission and Description	Private Score
Majority_Vote.csv 4 days ago by 楊惇昱 add submission details	0.92424
VGG.csv 4 days ago by 楊惇昱 add submission details	0.87878
Xception.csv 4 days ago by 楊惇昱 add submission details	0.87878
ResNet50.csv 4 days ago by 楊惇昱 add submission details	0.90909
DenseNet201.csv 4 days ago by 楊惇昱 add submission details	0.92424
InceptionResNetV2.csv 4 days ago by 楊惇昱 add submission details	0.89393