蘭花種類辨識及分類競賽

TEAM_733

1080661傅晧、1080617孫培瑁、1080646韓育銘

OUTLINE

- 1.try
- 2.final approach & score
- 3.分工

TRY 1

- Datasets(一類40張):

左右翻轉、順逆轉10度

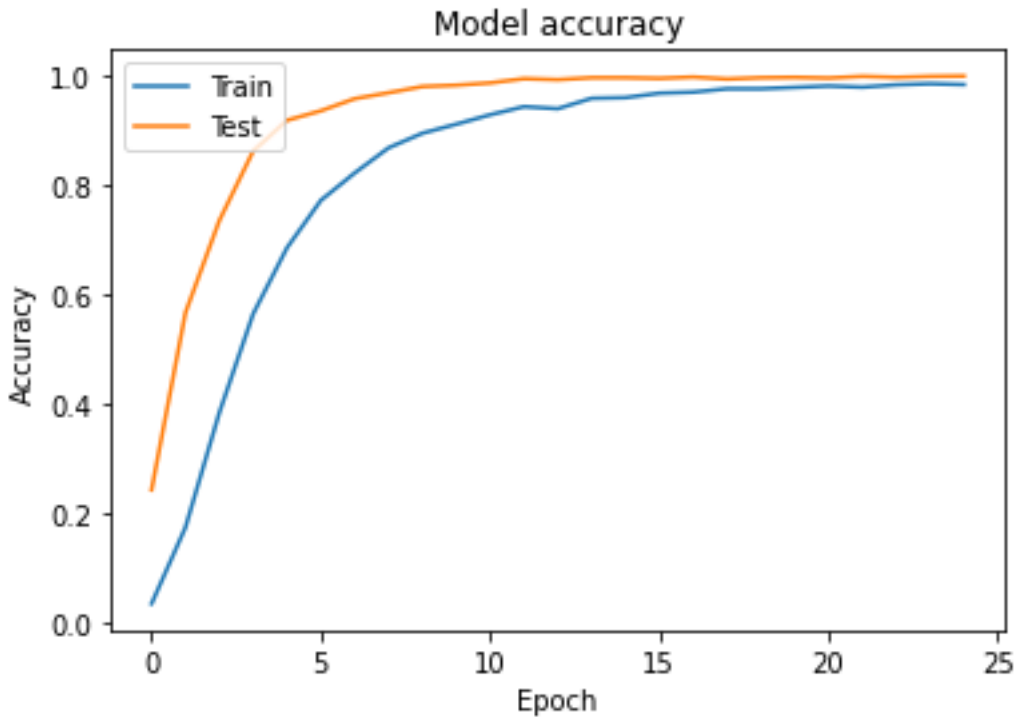
- Model:

ResNet50、 ResNet101、 ResNet152 、 InceptionV3 、 EfficientNetB0-B7、 EfficientNetV2B0-V2L

- Test accuracy(EffiN_B0):

1.0

```
global_average_pooling2d_1 (Gl  (None, 1280)        0           ['top_activation[0][0]']
obalAveragePooling2D)

flatten_1 (Flatten)            (None, 1280)        0           ['global_average_pooling2d_1[0][0
                                                                ]']

dense_2 (Dense)                (None, 512)         655872      ['flatten_1[0][0]']

dropout_1 (Dropout)            (None, 512)         0           ['dense_2[0][0]']

dense_3 (Dense)                (None, 219)         112347      ['dropout_1[0][0]']

==================================================================================================
Total params: 4,817,790
Trainable params: 4,775,767
Non-trainable params: 42,023
```



TRY 2

- Datasets(一類295張):

寫網路爬蟲以圖搜圖，一類爬285張 + 比賽原data

- Model:

EfficientNetB0

- Test accuracy:

慘不忍睹

TRY 2 DATASETS


nzt9hr5se2.jpg8.jpg


nzt9hr5se2.jpg7.jpg


nzt9hr5se2.jpg6.jpg


nzt9hr5se2.jpg5.jpg


nzt9hr5se2.jpg4.jpg


nzt9hr5se2.jpg3.jpg


nzt9hr5se2.jpg2.jpg


nzt9hr5se2.jpg1.jpg


nzt9hr5se2.jpg0.jpg


niky1zwp74.jpg29.jpg


niky1zwp74.jpg28.jpg


niky1zwp74.jpg27.jpg


niky1zwp74.jpg26.jpg


niky1zwp74.jpg25.jpg


niky1zwp74.jpg24.jpg


niky1zwp74.jpg23.jpg


niky1zwp74.jpg22.jpg


niky1zwp74.jpg21.jpg


niky1zwp74.jpg20.jpg


niky1zwp74.jpg19.jpg


niky1zwp74.jpg18.jpg


niky1zwp74.jpg17.jpg


niky1zwp74.jpg16.jpg


niky1zwp74.jpg15.jpg


niky1zwp74.jpg14.jpg


niky1zwp74.jpg13.jpg


niky1zwp74.jpg12.jpg


niky1zwp74.jpg11.jpg


niky1zwp74.jpg10.jpg


niky1zwp74.jpg9.jpg


niky1zwp74.jpg8.jpg


niky1zwp74.jpg7.jpg


niky1zwp74.jpg6.jpg


niky1zwp74.jpg5.jpg


niky1zwp74.jpg4.jpg


niky1zwp74.jpg3.jpg


niky1zwp74.jpg2.jpg


niky1zwp74.jpg1.jpg


niky1zwp74.jpg0.jpg


me3uqlixjn.jpg29.jpg

NOT WORK


dyk3prs289.jpg 7.jpg


dyk3prs289.jpg 8.jpg


dyk3prs289.jpg 9.jpg


dyk3prs289.jpg 10.jpg


dyk3prs289.jpg 11.jpg


dyk3prs289.jpg 12.jpg


dyk3prs289.jpg 13.jpg


dyk3prs289.jpg 14.jpg


dyk3prs289.jpg 15.jpg


dyk3prs289.jpg 16.jpg


dyk3prs289.jpg 17.jpg


dyk3prs289.jpg 18.jpg


dyk3prs289.jpg 19.jpg


dyk3prs289.jpg 20.jpg


dyk3prs289.jpg 21.jpg


dyk3prs289.jpg 22.jpg


dyk3prs289.jpg 23.jpg


dyk3prs289.jpg 24.jpg


dyk3prs289.jpg 25.jpg


dyk3prs289.jpg 26.jpg


dyk3prs289.jpg 27.jpg


dyk3prs289.jpg 28.jpg


dyk3prs289.jpg 29.jpg


eqm1tx2069.jpg


eqm1tx2069.jpg0.jpg


eqm1tx2069.jpg1.jpg


eqm1tx2069.jpg2.jpg


eqm1tx2069.jpg3.jpg


eqm1tx2069.jpg4.jpg


eqm1tx2069.jpg5.jpg


eqm1tx2069.jpg6.jpg


eqm1tx2069.jpg7.jpg


eqm1tx2069.jpg8.jpg


eqm1tx2069.jpg9.jpg


eqm1tx2069.jpg10.jpg


eqm1tx2069.jpg11.jpg

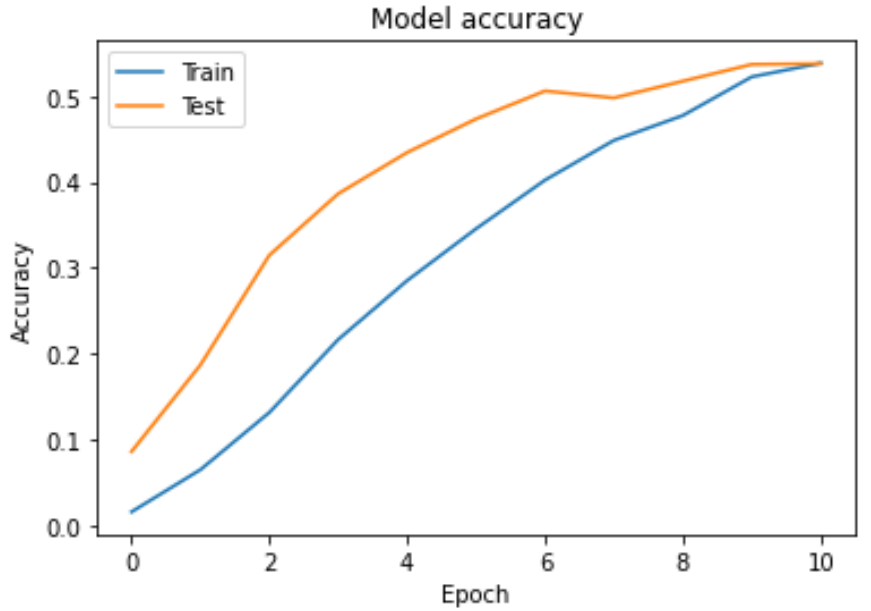

eqm1tx2069.jpg12.jpg


eqm1tx2069.jpg13.jpg


eqm1tx2069.jpg14.jpg


eqm1tx2069.jpg15.jpg

TRY 3



- Datasets(一類30張):

從上頁data中一類挑出20張 + 比賽原data

- Model:

EfficientNetB0

- Test accuracy:

0.60

NOT WORK

Class 1


7m94tnvze5.jpg


7m94tnvze5.jpg0.jpg


7m94tnvze5.jpg4.jpg


7m94tnvze5.jpg6.jpg


7m94tnvze5.jpg8.jpg


98uhpvlfni.jpg


98uhpvlfni.jpg21.jpg


98uhpvlfni.jpg26.jpg


gr8hxnsmju.jpg


gr8hxnsmju.jpg1.jpg


h8q6mjt0kd.jpg0.jpg


h8q6mjt0kd.jpg1.jpg


h8q6mjt0kd.jpg3.jpg


h8q6mjt0kd.jpg28.jpg


hfyk9wvsxa.jpg20.jpg

Class 13


7vj8cfwtgz.jpg


7vj8cfwtgz.jpg0.jpg


7vj8cfwtgz.jpg1.jpg


7vj8cfwtgz.jpg2.jpg


7vj8cfwtgz.jpg3.jpg


7vj8cfwtgz.jpg17.jpg


7vj8cfwtgz.jpg18.jpg


7vj8cfwtgz.jpg19.jpg


7vj8cfwtgz.jpg23.jpg


7vj8cfwtgz.jpg25.jpg


8qxyeg4dmn.jpg25.jpg


8qxyeg4dmn.jpg28.jpg


9pkz40oquj.jpg


au51dlkvj8.jpg


gx30a61mz2.jpg

TRY 4

- 使用DCGAN生成資料
- Newsize:256
- Epoch:10000
- Not work:時間不夠

FINAL APPROACH & SCORE

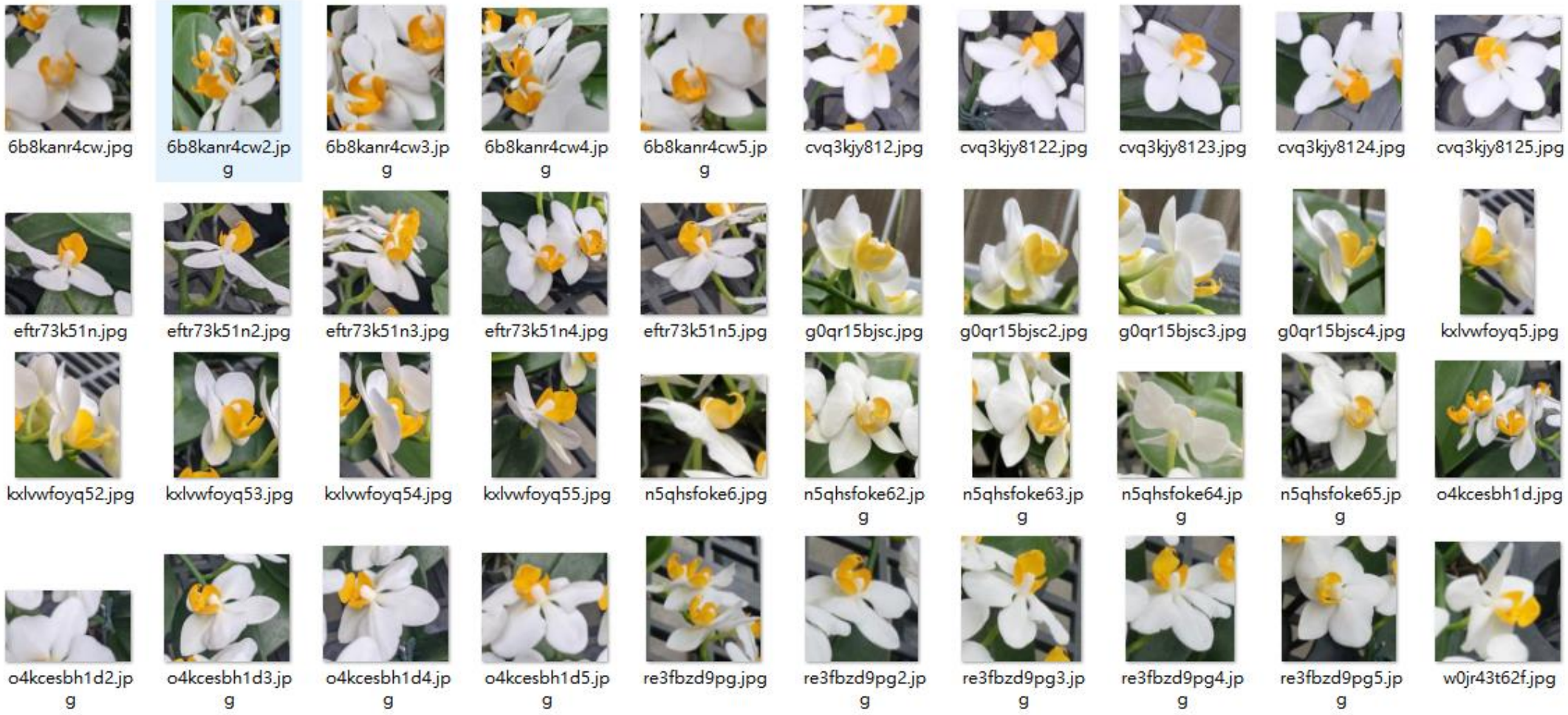
- 利用yolov5框圖，再做訓練；public & private 框完再預測

Label

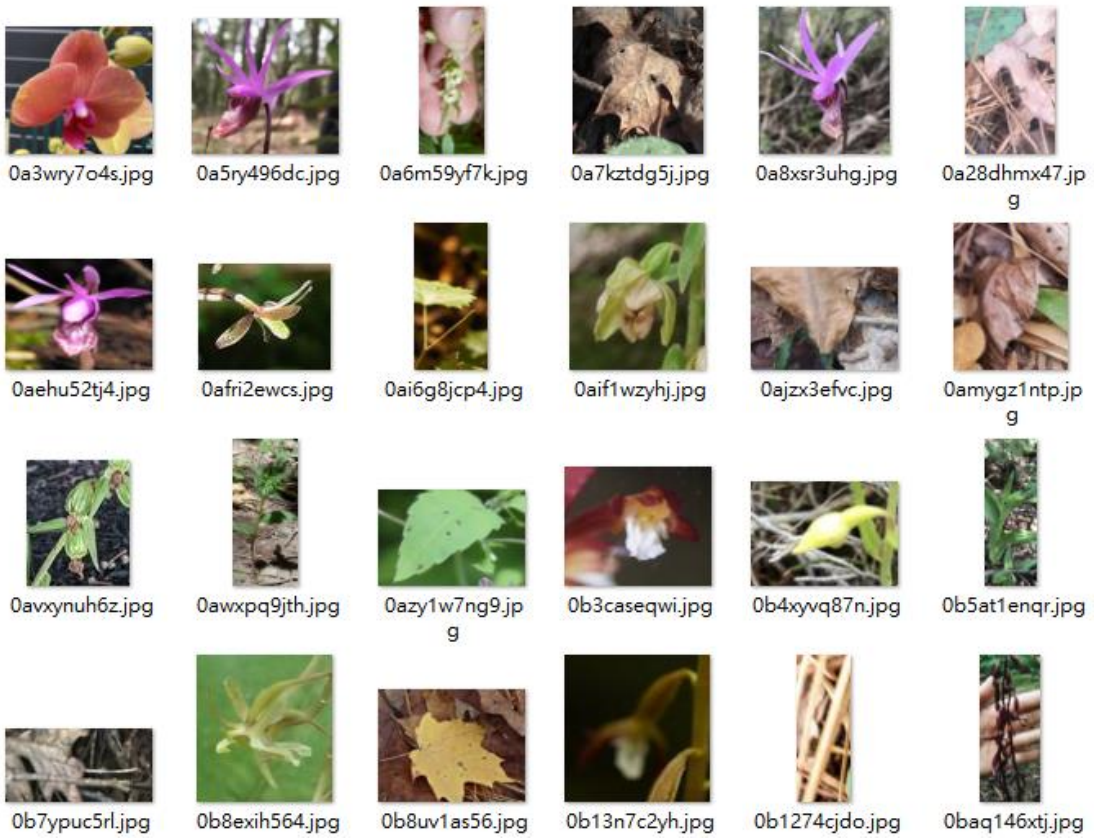
Predict

DATASETS

每類數量不等，總數5482張



Public data



Private data

FINAL APPROACH & SCORE

- Datasets:

同上頁

- Model:

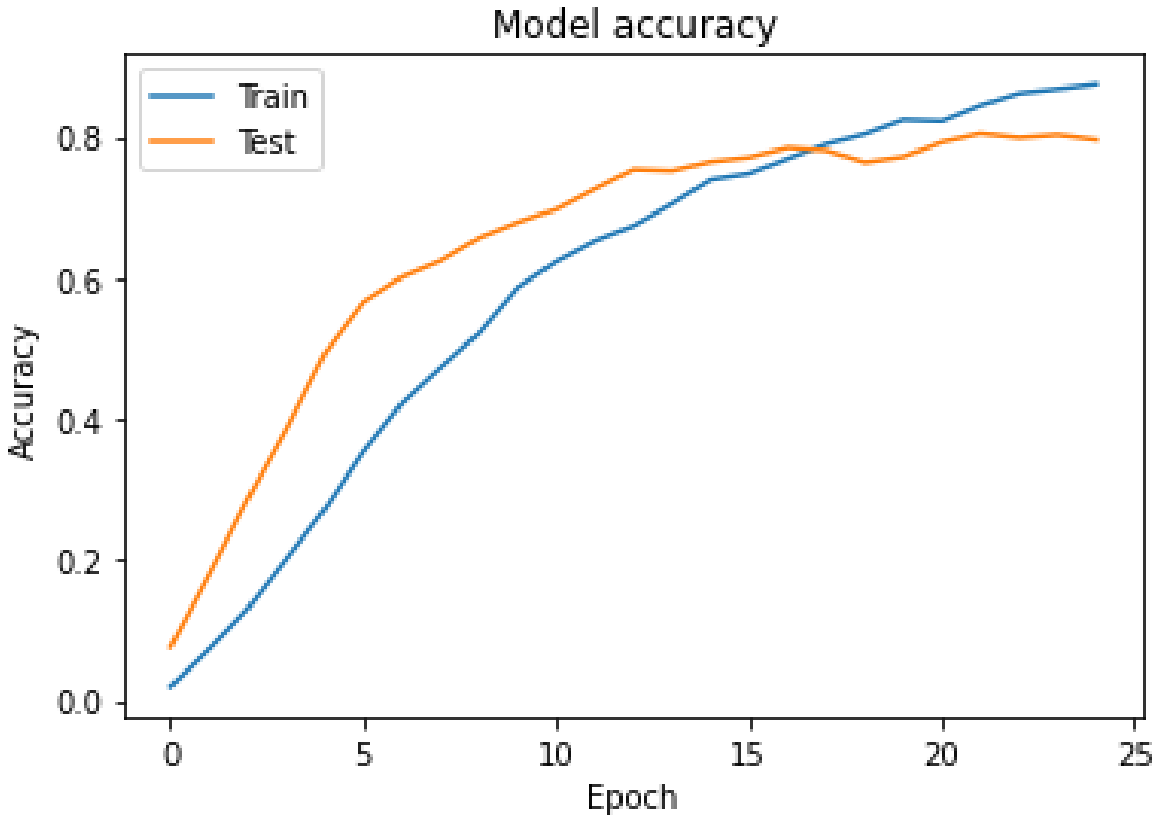
EfficientNetB0

- Test accuracy:

0.80

- Public score:

0.733890



	submission_template.csv				
	first 上傳成員 孫 培瑁	2022-06-06 12:45:34	0.733890		Scoring success.

FINAL APPROACH & SCORE

- Datasets:

同上頁

- Model:

DenseNet121

- Val accuracy:

0.78

- Public score:

0.780869

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, None, None, 1024)  7037504

 global_average_pooling2d_8   (None, 1024)             0
 (GlobalAveragePooling2D)

 dense_8 (Dense)             (None, 219)               224475

=================================================================
Total params: 7,261,979
Trainable params: 7,178,331
Non-trainable params: 83,648
_________________________________________________________________
```

	submission_template3.csv DenseNet 上傳成員 孫 培瑁	2022-06-06 03:14:52	0.780869		Scoring success.

FINAL APPROACH & SCORE

- Datasets:

同上頁

- Model:

將EfficientNetB0、EfficientNetB1、EfficientNetB2、DenseNet121做concatenate

- Val accuracy:

0.75

- Public score:

0.779285

	submission_template2.csv second 上傳成員 孫 培瑁	2022-06-06 02:11:47	0.779285		Scoring success.

架構圖

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnet-b0 (Functional  (None, 10, 10, 1280)     4049564
 )

 global_average_pooling2d_9   (None, 1280)            0
 (GlobalAveragePooling2D)

 dense_9 (Dense)             (None, 219)               280539

=================================================================
Total params: 4,330,103
Trainable params: 4,288,087
Non-trainable params: 42,016
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnet-b2 (Functional  (None, 10, 10, 1408)     7768562
 )

 global_average_pooling2d_11  (None, 1408)            0
  (GlobalAveragePooling2D)

 dense_11 (Dense)            (None, 219)               308571

=================================================================
Total params: 8,077,133
Trainable params: 8,009,565
Non-trainable params: 67,568
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnet-b1 (Functional  (None, 10, 10, 1280)     6575232
 )

 global_average_pooling2d_10  (None, 1280)            0
  (GlobalAveragePooling2D)

 dense_10 (Dense)            (None, 219)               280539

=================================================================
Total params: 6,855,771
Trainable params: 6,793,723
Non-trainable params: 62,048
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, None, None, 1024)  7037504

 global_average_pooling2d_8   (None, 1024)            0
 (GlobalAveragePooling2D)

 dense_8 (Dense)             (None, 219)               224475

=================================================================
Total params: 7,261,979
Trainable params: 7,178,331
Non-trainable params: 83,648
_________________________________________________________________
```

架構圖

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
=================================================================================================
 input_16 (InputLayer)          [(None, 320, 320, 3  0           []
                                )]

 sequential_16 (Sequential)     (None, None, None,   7037504     ['input_16[0][0]']
                                1024)

 sequential_17 (Sequential)     (None, 10, 10, 1280  4049564     ['input_16[0][0]']
                                )

 sequential_18 (Sequential)     (None, 10, 10, 1280  6575232     ['input_16[0][0]']
                                )

 sequential_19 (Sequential)     (None, 10, 10, 1408  7768562     ['input_16[0][0]']
                                )

 concatenate_1 (Concatenate)    (None, 10, 10, 4992  0           ['sequential_16[0][0]',
                                )                                 'sequential_17[0][0]',
                                                                  'sequential_18[0][0]',
                                                                  'sequential_19[0][0]']

 global_average_pooling2d_12 (G  (None, 4992)        0           ['concatenate_1[0][0]']
 lobalAveragePooling2D)

 dropout (Dropout)              (None, 4992)         0           ['global_average_pooling2d_12[0][
                                                                 0]']

 dense_12 (Dense)               (None, 219)          1093467     ['dropout[0][0]']

=================================================================================================
Total params: 26,524,329
Trainable params: 1,093,467
Non-trainable params: 25,430,862
```

FINAL APPROACH & SCORE

- 將此三份csv檔做加權
- Public score(85名):
0.794806
- Private score(76名):
0.680559772

	submission.csv				
	3 models 上傳成員 傅 晧	2022-06-06 03:19:41	0.794806	0.680559772	Scoring success.

```
import csv
import pandas as pd

sub_files = [
                '../input/flower/submission_template.csv',
                '../input/flower/submission_template2.csv',
                '../input/flower2/submission_template3.csv',
]

# Weights of the individual subs
sub_weight = [
                0.734**2,
                0.779**2,
                0.781**2,
            ]
```

```
Hlabel = 'filename'
Htarget = 'category'
npt = 1
place_weights = {}
for i in range(npt):
    place_weights[i] = (1 / (i + 1))

print(place_weights)

lg = len(sub_files)
sub = [None]*lg
for i, file in enumerate( sub_files ):
    print("Reading {}: w={} - {}". format(i, sub_weight[i], file))
    reader = csv.DictReader(open(file,"r"))
    sub[i] = sorted(reader, key=lambda d: str(d[Hlabel]))

out = open("submission.csv", "w", newline='')
writer = csv.writer(out)
writer.writerow([Hlabel,Htarget])

for p, row in enumerate(sub[0]):
    target_weight = {}
    for s in range(lg):
        row1 = sub[s][p]
        for ind, trgt in enumerate(row1[Htarget].split(' ')):
            target_weight[trgt] = target_weight.get(trgt,0) + (place_weights[ind]*sub_weight[s])
    tops_trgt = sorted(target_weight, key=target_weight.get, reverse=True)[:npt]
    writer.writerow([row1[Hlabel], " ".join(tops_trgt)])
out.close()
```

分工

- 1080617 孫培瑁：資料前處理、模型優化
- 1080661 傅晗：模型優化、資料後處理
- 1080646 韓育銘：測試只使用原始圖片的模型

THANKS