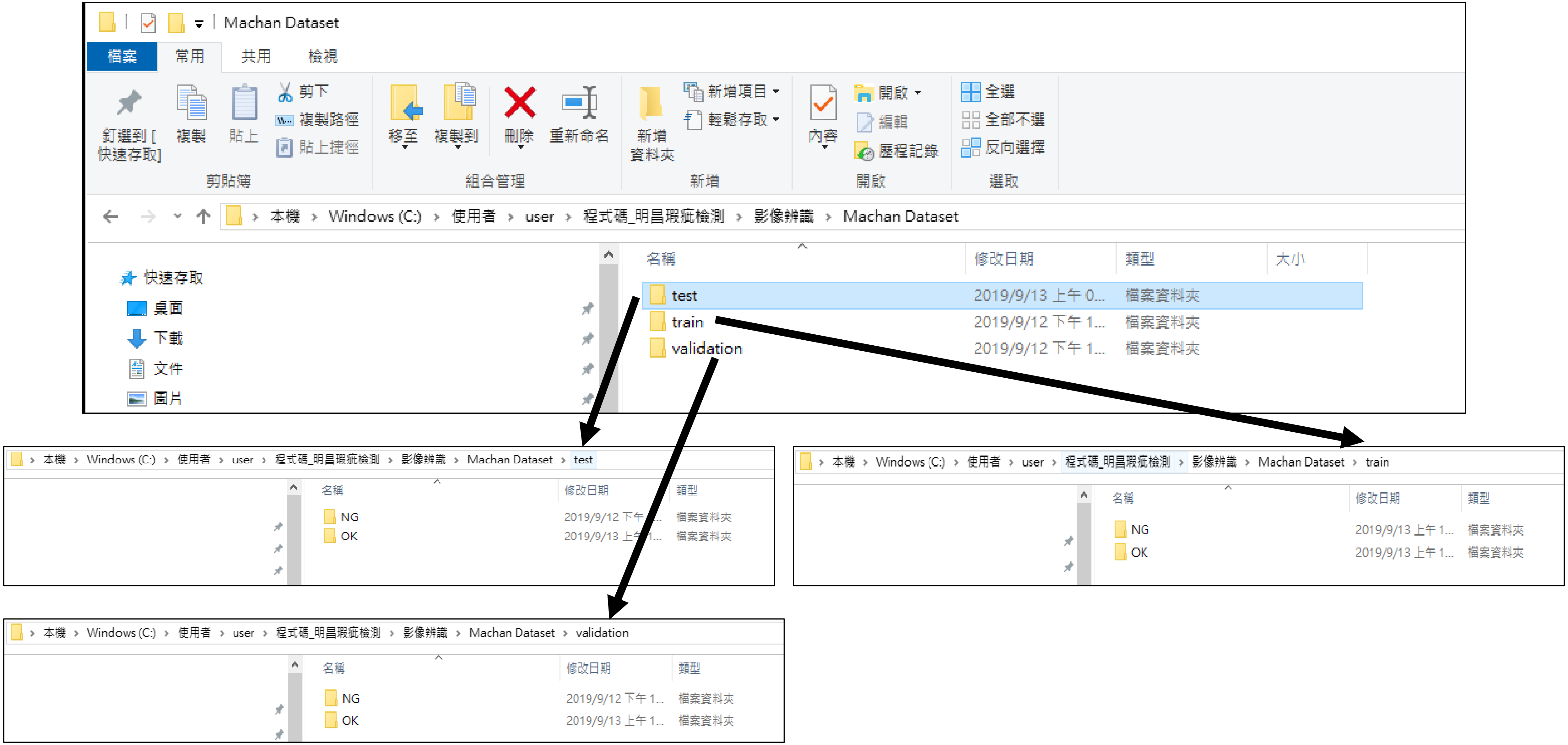
分　類

➢ 資料存放方式:



Machan Dataset
檔案
常用
共用
檢視
釘選到[快速存取]
複製
貼上
剪下
複製路徑
貼上捷徑
剪貼簿
移至
複製到
刪除
重新命名
組合管理
新增資料夾
新增項目
輕鬆存取
新增
內容
開啟
編輯
歷程記錄
開啟
全選
全部不選
反向選擇
選取
本機 › Windows (C:) › 使用者 › user › 程式碼_明昌瑕疵檢測 › 影像辨識 › Machan Dataset
快速存取
桌面
下載
文件
圖片
名稱
修改日期
類型
大小
test
2019/9/13 上午 0...
檔案資料夾
train
2019/9/12 下午 1...
檔案資料夾
validation
2019/9/12 下午 1...
檔案資料夾
本機 › Windows (C:) › 使用者 › user › 程式碼_明昌瑕疵檢測 › 影像辨識 › Machan Dataset › test
NG
2019/9/12 下午...
檔案資料夾
OK
2019/9/13 上午 1...
檔案資料夾
本機 › Windows (C:) › 使用者 › user › 程式碼_明昌瑕疵檢測 › 影像辨識 › Machan Dataset › train
NG
2019/9/13 上午 1...
檔案資料夾
OK
2019/9/13 上午 1...
檔案資料夾
本機 › Windows (C:) › 使用者 › user › 程式碼_明昌瑕疵檢測 › 影像辨識 › Machan Dataset › validation
NG
2019/9/12 下午 1...
檔案資料夾
OK
2019/9/13 上午 1...
檔案資料夾

➢ 程式 VGG16:

In [1]:
```python
import keras
keras.__version__
```

```
Using TensorFlow backend.
```

Out[1]:
```
'2.2.4'
```

In [2]:
```python
# 資料擴增的特徵萃取
from keras import models
from keras import layers
from keras.applications import VGG16

conv_base = VGG16(weights='imagenet',include_top=False,input_shape=(150,150,3))

model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```python
# 把 block5_conv1 之前的層都設定為不可訓練(凍結)
conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

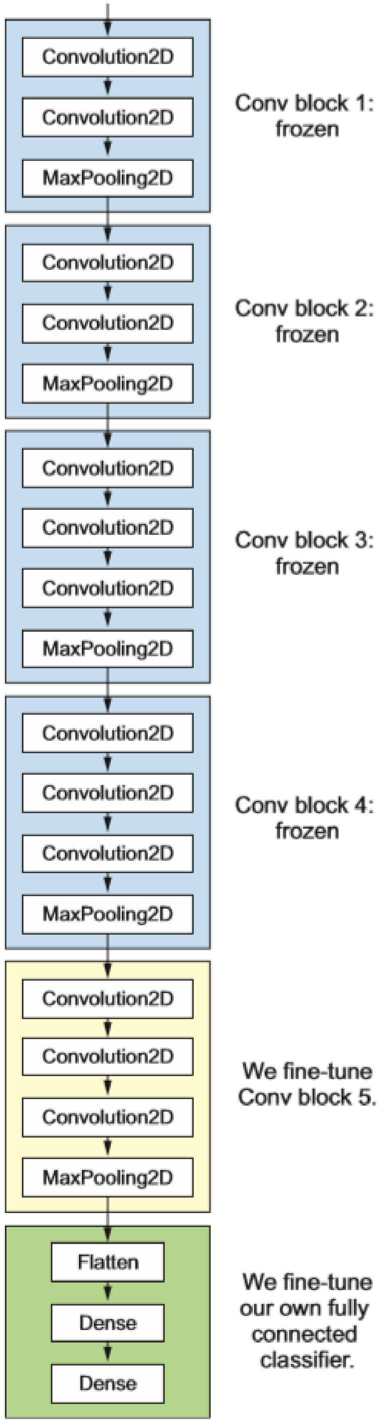


Figure 5.19 Fine-tuning the last convolutional block of the VGG16 network

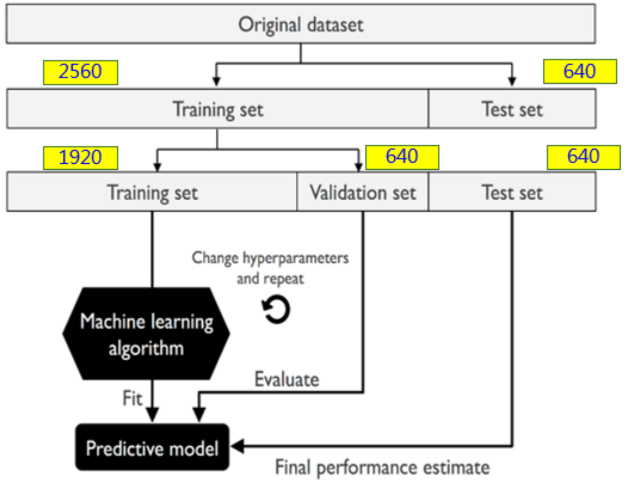
➢ 程式 VGG16:

```python
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers

# 資料路徑
DATASET_PATH  = r'C:\Users\user\程式碼_明昌瑕疵檢測\影像辨識\Machan Dataset'

# 影像大小
IMAGE_SIZE = (150,150)

# 影像類別數
NUM_CLASSES = 2

# 若 GPU 記憶體不足，可調降 batch size 或凍結更多層網路
BATCH_SIZE = 10


# Epoch 數
NUM_EPOCHS = 10

# 模型輸出儲存的檔案
WEIGHTS_FINAL = 'model-VGG16-final.h5'
```

```python
# 擴充訓練資料
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        DATASET_PATH + '/train',
        # All images will be resized to 150x150
        target_size=IMAGE_SIZE,
        batch_size=BATCH_SIZE,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        DATASET_PATH + '/validation',
        target_size=IMAGE_SIZE,
        batch_size=BATCH_SIZE,
        class_mode='binary')

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-5),
              metrics=['acc'])

history = model.fit_generator(
      train_generator,
      #steps_per_epoch=train_generator.samples // BATCH_SIZE,
      steps_per_epoch=train_generator.samples // 20,
      epochs=NUM_EPOCHS,
      validation_data=validation_generator,
      validation_steps=validation_generator.samples // BATCH_SIZE)

# 儲存訓練好的模型
model.save(WEIGHTS_FINAL)
```

```
Found 1920 images belonging to 2 classes.
Found 640 images belonging to 2 classes.
```

資料以 6:2:2 拆為 Training set, Validation set , Test set

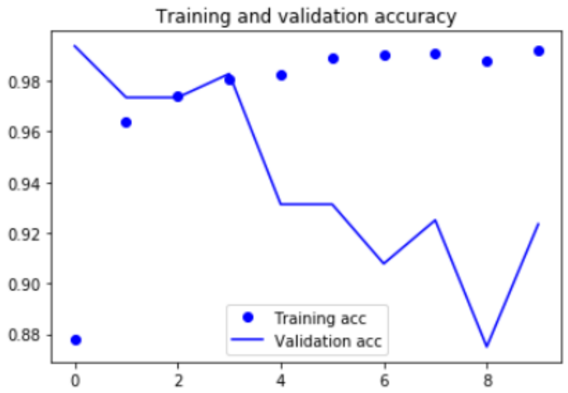



```
Epoch 1/10
192/192 [==============================] - 24s 123ms/step - loss: 0.3513 - acc: 0.8521 - val_loss: 0.2139 - val_acc: 0.9234
Epoch 2/10
192/192 [==============================] - 18s 94ms/step - loss: 0.1090 - acc: 0.9562 - val_loss: 0.0847 - val_acc: 0.9672
Epoch 3/10
192/192 [==============================] - 18s 94ms/step - loss: 0.0675 - acc: 0.9734 - val_loss: 0.0600 - val_acc: 0.9766
Epoch 4/10
192/192 [==============================] - 18s 95ms/step - loss: 0.0515 - acc: 0.9797 - val_loss: 0.0676 - val_acc: 0.9719
Epoch 5/10
192/192 [==============================] - 18s 95ms/step - loss: 0.0391 - acc: 0.9849 - val_loss: 0.1155 - val_acc: 0.9531
Epoch 6/10
192/192 [==============================] - 18s 95ms/step - loss: 0.0348 - acc: 0.9880 - val_loss: 0.0790 - val_acc: 0.9719
Epoch 7/10
192/192 [==============================] - 18s 95ms/step - loss: 0.0283 - acc: 0.9906 - val_loss: 0.3509 - val_acc: 0.8781
Epoch 8/10
192/192 [==============================] - 18s 95ms/step - loss: 0.0277 - acc: 0.9885 - val_loss: 0.0595 - val_acc: 0.9750
Epoch 9/10
192/192 [==============================] - 18s 96ms/step - loss: 0.0302 - acc: 0.9891 - val_loss: 0.0487 - val_acc: 0.9781
Epoch 10/10
192/192 [==============================] - 18s 95ms/step - loss: 0.0234 - acc: 0.9917 - val_loss: 0.0736 - val_acc: 0.9531
```

```
Epoch 1/10
96/96 [==============================] - 21s 221ms/step - loss: 0.4584 - acc: 0.8146 - val_loss: 0.3160 - val_acc: 0.8437
Epoch 2/10
96/96 [==============================] - 13s 140ms/step - loss: 0.2028 - acc: 0.9333 - val_loss: 0.1490 - val_acc: 0.9422
Epoch 3/10
96/96 [==============================] - 13s 135ms/step - loss: 0.1277 - acc: 0.9490 - val_loss: 0.1615 - val_acc: 0.9062
Epoch 4/10
96/96 [==============================] - 13s 135ms/step - loss: 0.0866 - acc: 0.9646 - val_loss: 0.1082 - val_acc: 0.9469
Epoch 5/10
96/96 [==============================] - 13s 135ms/step - loss: 0.0653 - acc: 0.9729 - val_loss: 0.0781 - val_acc: 0.9719
Epoch 6/10
96/96 [==============================] - 13s 135ms/step - loss: 0.0590 - acc: 0.9740 - val_loss: 0.0796 - val_acc: 0.9719
Epoch 7/10
96/96 [==============================] - 13s 136ms/step - loss: 0.0471 - acc: 0.9823 - val_loss: 0.1327 - val_acc: 0.9375
Epoch 8/10
96/96 [==============================] - 13s 136ms/step - loss: 0.0408 - acc: 0.9833 - val_loss: 0.0719 - val_acc: 0.9719
Epoch 9/10
96/96 [==============================] - 13s 136ms/step - loss: 0.0465 - acc: 0.9844 - val_loss: 0.0616 - val_acc: 0.9734
Epoch 10/10
96/96 [==============================] - 13s 136ms/step - loss: 0.0498 - acc: 0.9812 - val_loss: 0.0590 - val_acc: 0.9750
```

➢ 程式 VGG16:

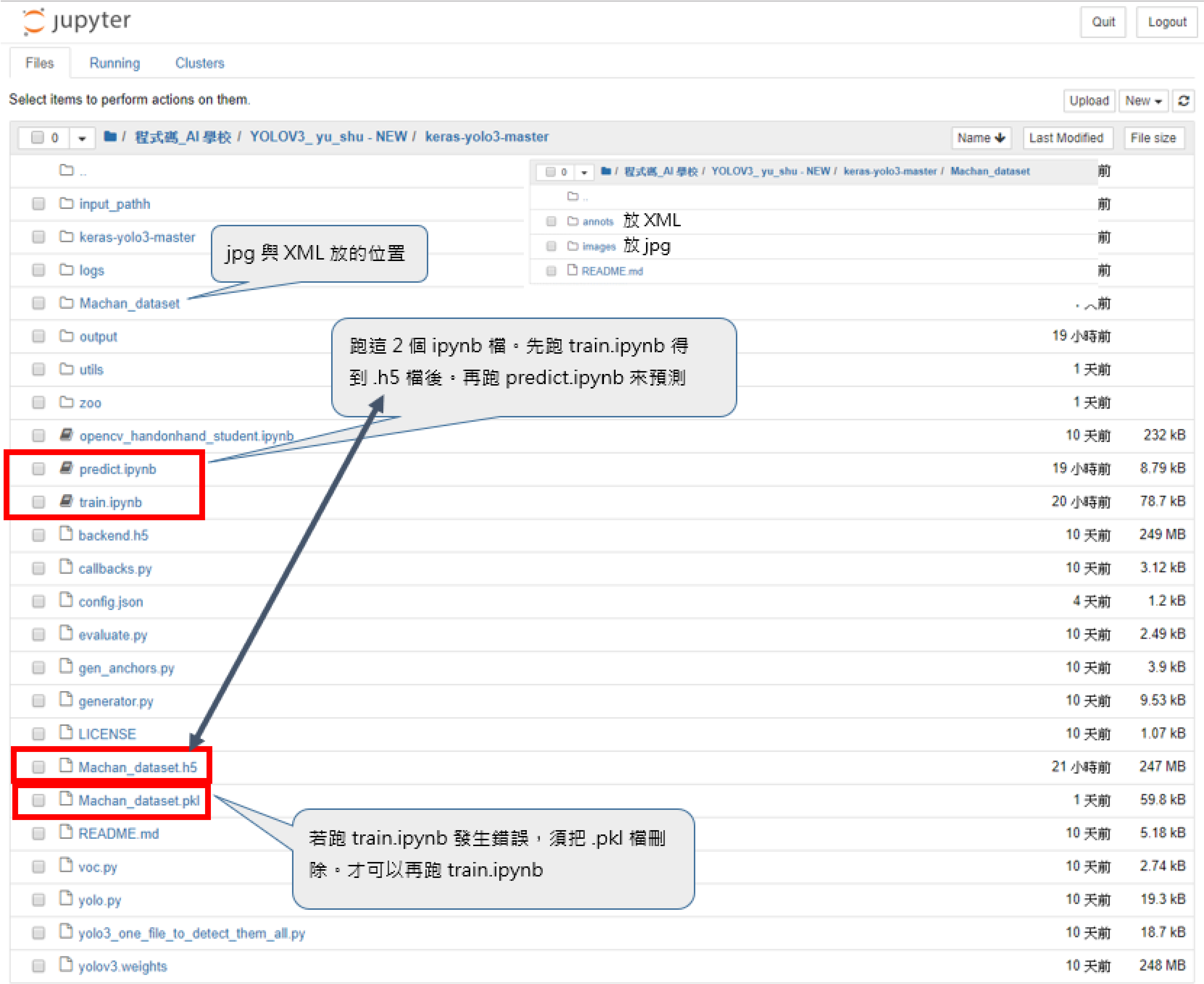
```python
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```




```python
test_generator = test_datagen.flow_from_directory(
        DATASET_PATH + '/test',
        target_size=IMAGE_SIZE,
        batch_size=BATCH_SIZE,
        class_mode='binary')

test_loss, test_acc = model.evaluate_generator(test_generator, steps=50)
print('test acc:', test_acc)
```

```
Found 640 images belonging to 2 classes.
test acc: 0.7940000003576279
```

物 件 偵 測


jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
程式嗎_AI 學校 / YOLOV3_yu_shu - NEW / keras-yolo3-master
Name
Last Modified
File size
程式嗎_AI 學校 / YOLOV3_yu_shu - NEW / keras-yolo3-master / Machan_dataset
annots 放 XML
images 放 jpg
README.md
input_pathh
keras-yolo3-master
jpg 與 XML 放的位置
logs
Machan_dataset
output
跑這 2 個 ipynb 檔，先跑 train.ipynb 得到 .h5 檔後，再跑 predict.ipynb 來預測
utils
zoo
opencv_handonhand_student.ipynb
predict.ipynb
train.ipynb
backend.h5
callbacks.py
config.json
evaluate.py
gen_anchors.py
generator.py
LICENSE
Machan_dataset.h5
Machan_dataset.pkl
若跑 train.ipynb 發生錯誤，須把 .pkl 檔刪除，才可以再跑 train.ipynb
README.md
voc.py
yolo.py
yolo3_one_file_to_detect_them_all.py
yolov3.weights
19 小時前
1 天前
10 天前 232 kB
19 小時前 8.79 kB
20 小時前 78.7 kB
10 天前 249 MB
10 天前 3.12 kB
4 天前 1.2 kB
10 天前 2.49 kB
10 天前 3.9 kB
10 天前 9.53 kB
10 天前 1.07 kB
21 小時前 247 MB
1 天前 59.8 kB
10 天前 5.18 kB
10 天前 2.74 kB
10 天前 19.3 kB
10 天前 18.7 kB
10 天前 248 MB

- train.ipynb

In [5]:

```
#更改訓練資料夾路徑，以及cache_name
# Parse the annotations
train_annot_folder = './Machan_dataset/annots/'
train_image_folder = './Machan_dataset/images/'
cache_name = 'Machan_dataset.pkl'
valid_annot_folder = ''
valid_image_folder = ''
valid_cache_name = ''
labels = ["Orange peel","Scratch","Water mark","Granular","Crater","Pit"]
train_ints, valid_ints, labels, max_box_per_image = create_training_instances(
    train_annot_folder,
    train_image_folder,
    cache_name,
    valid_annot_folder,
    valid_image_folder,
    valid_cache_name,
    labels
)
#print('\nTraining on: \t' + str(labels) + '\n')

# 可在這更改batch_size
# Create the generators
anchors = [55,69, 75,234, 133,240, 136,129, 142,363, 203,290, 228,184, 285,359, 341,260]
batch_size = 1
max_input_size = 448
min_input_size = 228

train_generator = BatchGenerator(
    instances           = train_ints,
    anchors             = anchors,
    labels              = labels,
    downsample          = 32, # ratio between network input's size and network output's size, 32 for YOLOv3
    max_box_per_image   = max_box_per_image,
    batch_size          = batch_size,
    min_net_size        = min_input_size,
    max_net_size        = max_input_size,
    shuffle             = True,
    jitter              = 0.3,
    norm                = normalize
)

valid_generator = BatchGenerator(
    instances           = valid_ints,
    anchors             = anchors,
    labels              = labels,
    downsample          = 32, # ratio between network input's size and network output's size, 32 for YOLOv3
    max_box_per_image   = max_box_per_image,
    batch_size          = batch_size,
    min_net_size        = min_input_size,
    max_net_size        = max_input_size,
    shuffle             = True,
    jitter              = 0.0,
    norm                = normalize
)
```

資料的路徑

Label 的名稱

➢ train.ipynb

```
#更改儲存model名稱
# Create the model
saved_weights_name = 'Machan_dataset.h5'
warmup_epochs = 3
train_times = 8
ignore_thresh = 0.5
learning_rate = 1e-4
grid_scales = [1,1,1]
obj_scale = 5
if os.path.exists(saved_weights_name):
    warmup_epochs = 0
warmup_batches = warmup_epochs * (train_times*len(train_generator))

train_model, infer_model = create_model(
    nb_class            = len(labels),
    anchors             = anchors,
    max_box_per_image   = max_box_per_image,
    max_grid            = [max_input_size, max_input_size],
    batch_size          = batch_size,
    warmup_batches      = warmup_batches,
    ignore_thresh       = 0.5,
    saved_weights_name  = saved_weights_name,
    lr                  = 1e-4,
    grid_scales         = [1,1,1],

)

# Start training
callbacks = create_callbacks(saved_weights_name, infer_model)
nb_epochs = 7
train_model.fit_generator(
    generator        = train_generator,
    steps_per_epoch  = len(train_generator) * train_times,
    epochs           = nb_epochs + warmup_epochs,
    verbose          = 2,
    callbacks        = callbacks,
    workers          = 4,
    max_queue_size   = 8
)
```

儲存的 model 名稱

➢ predict.ipynb

In [1]:

```python
# predict
import os
import argparse
import json
import cv2
from utils.utils import get_yolo_boxes, makedirs
from utils.bbox import draw_boxes
from keras.models import load_model
from tqdm import tqdm
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

#單次檔案input
#input_path   = 'IMG_7562.jpg'
input_path   = 'Dunyu_2.mp4'

output_path  = 'output/'
makedirs(output_path)

# Set some parameter
net_h, net_w = 416, 416 # a multiple of 32, the smaller the faster
obj_thresh, nms_thresh = 0.5, 0.45
anchors = [55,69, 75,234, 133,240, 136,129, 142,363, 203,290, 228,184, 285,359, 341,260]
labels = ["Orange peel","Scratch","Water mark","Granular","Crater","Pit"]

# Load the model
infer_model = load_model('Machan_dataset.h5')
```

預測圖片 or 影片

預測後的存放路徑

Label 的名稱

預測的 model 名稱