

HW6

自定義資料分類訓練

組別：為什麼要醬組

組員一：陳胤璫 B103012001

組員二：林凡皓 B103012002

組員三：洪漢霖 B103012011

組員四：吳尚恩 B103040033

環境建置

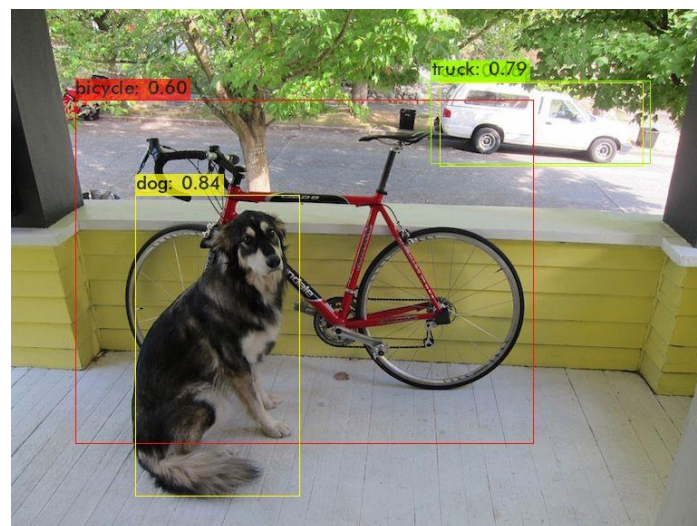
先前將資料處理完後，接下來我們要開始對模型進行訓練，但在那之前我們得先檢查環境是否建立正確。

照著講義內步驟一個一個安裝好函示庫(如)和所需要的檔案時，我們直接跳到測試環境的部分，在進入工作站，確認已經啟用 YOLO 環境後，我們輸入以下指令：

```
./darknet detector test cfg/coco.data cfg/yolov4-tiny.cfg
```

```
yolov4-tiny.weights data/dog.jpg
```

可以在 dark 找到 prediction.jpg 檔案，可見以下圖片和講義內同，因此在這邊可以確定環境建立完畢。



一、 資料處理

本次作業我們選擇做人臉偵測，並判斷是否有戴口罩。

資料來源為部分，我們從 kaggle 上尋找各式各樣的圖片來做訓練。

資料來源：

<https://www.kaggle.com/datasets/andrewmvd/face-mask-detection>

<https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection>

<https://www.kaggle.com/datasets/aditya276/face-mask-dataset>

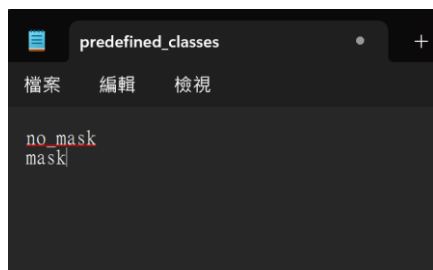
部分照片如下



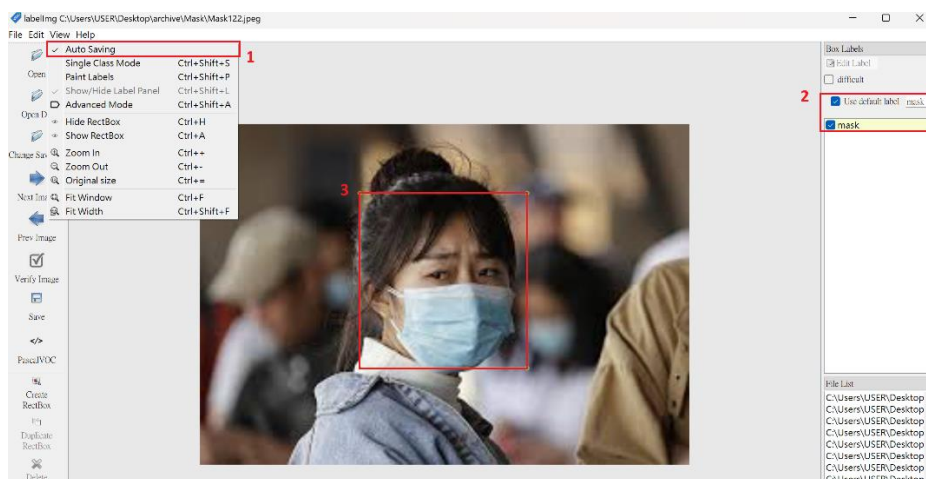
我們將照片分成 training set、validation set、testing set 來做訓練，以及公平的評估我們的模型。

資料準備完成後，我們使用 labellmg 來標註資料，操作步驟如下

1. 創建一個 predefined_classes.txt 檔案，存放要用到的 label。



2. 進入 labellmg 設定自動儲存，並使用 Use default label 以加速標記(對於 no_mask 之部分再調整 label 即可)。將要標記的部分選起來就完成一張圖片的標記。



3. 進入存檔資料夾檢視存檔結果，可以發現到一張圖片會對應到一個.txt 檔案。

_111510370_060683565.jpg	2024/5/15 下午 12:05	JPG 檔案	60 KB
_111510370_060683565.txt	2024/5/15 下午 12:05	文字文件	1 KB
_111512783_mediaitem111512782.jpg	2024/5/15 下午 12:05	JPG 檔案	13 KB
_111512783_mediaitem111512782.txt	2024/5/15 下午 12:05	文字文件	1 KB
_111550872_gettyimages-112816256...	2024/5/15 下午 12:05	JPG 檔案	21 KB
_111550872_gettyimages-112816256...	2024/5/15 下午 12:05	文字文件	2 KB
_112342927_airportfacemask.jpg	2024/5/15 下午 12:05	JPG 檔案	84 KB
_112342927_airportfacemask.txt	2024/5/15 下午 12:05	文字文件	1 KB
_112393674_gettyimages-121225388...	2024/5/15 下午 12:05	JPG 檔案	13 KB
_112393674_gettyimages-121225388...	2024/5/15 下午 12:05	文字文件	1 KB

4. 解析.txt 檔案

```
0 0.162500 0.500000 0.316667 0.891358
0 0.502083 0.422222 0.290278 0.656790
0 0.836111 0.435802 0.316667 0.718519
```

由此資料可以看出，這張照片中有三個沒有戴口罩的人(label = 0)，後面四個 column 值為方框位置。

5. 創建存放路徑的 train.txt、valid.txt。由於照片數量很多，我們自己寫了一個 python 程式幫助我們自動化處理。

Python 程式部分，首先讀取存放資料的路徑，並將此路徑寫入.txt 檔案中，接著再修改.txt 檔案中路徑，將其改成符合工作站上的路徑，code 如下

```
import os

# 資料夾路徑
folder_path = 'Data'

# 獲取資料夾中的所有文件名
all_files = os.listdir(folder_path)

# 篩選出所有的 .png 文件
files = [f for f in all_files]

# 獲取所有 .png 文件的絕對路徑
paths = [os.path.abspath(os.path.join(folder_path, f)) for f in files]

# 將這些絕對路徑寫入到一個 .txt 文件中
file_path = 'train.txt'
with open(file_path, 'w') as file:
    for path in paths:
        file.write(path + '\n')

print(f"所有 .png 文件的絕對路徑已存儲到 {file_path} 文件中。")

# 指定 .txt 文件的路徑
file_path = 'train.txt'

# 讀取 .txt 文件中的所有內容
with open(file_path, 'r', encoding='utf-8') as file:
    lines = file.readlines()

# 替換每一行中的 'C:\\Users\\User\\Desktop\\yolo_train\\Data' 為 '/home/DPAML112/DPAML112a4/lil_train/yolo_train'
updated_lines = [line.replace('C:\\Users\\User\\Desktop\\yolo_train\\Data', '/home/DPAML112/DPAML112a4/lil_train/yolo_train') for line in lines]

# 將修改後的內容寫回 .txt 文件
with open(file_path, 'w', encoding='utf-8') as file:
    file.writelines(updated_lines)

print(f"{file_path} 文件中的 'C:\\Users\\User\\Desktop\\yolo_train\\Data' 已替換成 '/home/DPAML112/DPAML112a4/lil_train/yolo_train'。")
```

產生之存放路徑檔案如下

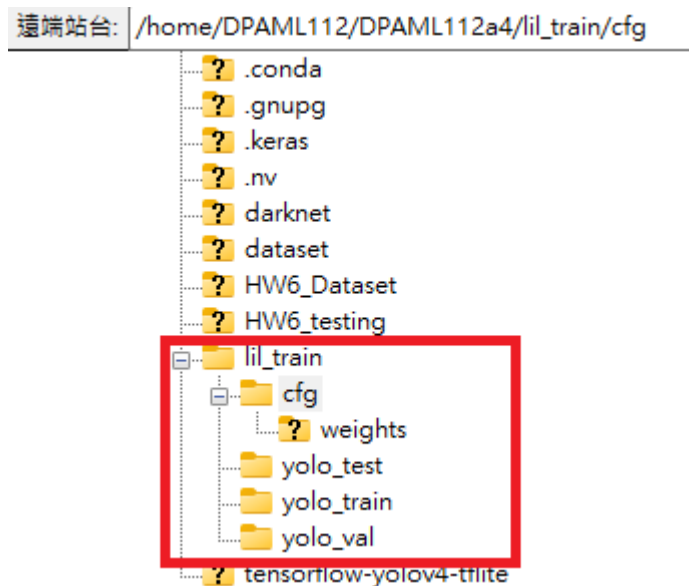
```

/home/DPAML112/DPAML112a4/lil_train/yolo_train/article-5e3a3b262ca4f.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/coronavirus_wuhan_yunfei_li.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/coronavirus (2).jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/b_20200210192750.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/ckeditor-5e4507060b13f.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/d3b67fe6-0529-46fd-bd99-75b44439a249.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/b7a52faa-2962-4bf0-a359-1358d0c47d76.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/article-5e450331d87b6.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/banale_mask6.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/bajo_la_sombra_del_coronavirus_llega_el_ano_nuevo_lunar_h
ay_30_millones_de_personas_con_restricciones.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/coronavirus-1_1580220542.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/coronavirus-china-gty-aa-200130_hpMain_16x9_992.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/asian-girl-wearing-a-dust-mask-pm25-on-bed-health-care-
concept-T6BBWK.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/ca_coronaviruscontagious_012720getty.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/china-virus.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/depositphotos_196900844-stock-photo-mother-wearing-
protective-mask-daughter.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/chinese-girl-with-face-mask-yinchuan-ningxia-china-
J61449.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/airport-mask-1200x830.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/cff8244482ede545b1d81c9be7f5b6ed_original.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/article-5e2d7e375fbff.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/coronavirus-chinois-est-encore-trop-tot-pour-parler-
pandemie.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/abstrax_art_13120-copy.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/cba5afe7-67d8-48a4-ad60-d1d21948c393_size190_w1024
_h795.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/asian-child-wearing-mask-prevent-protection-factor-for-
n95-filtering-face-mask-safty-white-mask-T5WP83.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/b83abacd915ca1914599d05b4d62592a.jpg
/home/DPAML112/DPAML112a4/lil_train/yolo_train/asian-mom-carrying-her-baby-hipseat-walking-wearing-
protection-mask-against-air-pollution-bangkok-city-thailand-asian-147889817.jpg

```

由於資料來源為多個 kaggle 資料集，因此路徑有些雜亂。

6. 將準備好的資料以及存放路徑之檔案上傳到 FTP 上

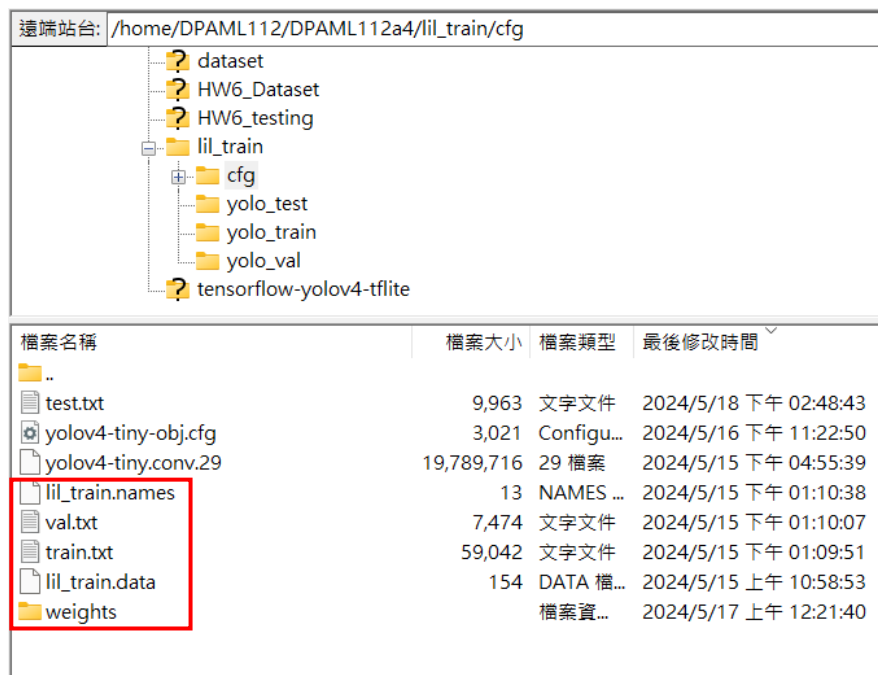


二、開始訓練

建立好環境、下載所需檔案和準備好訓練資料後，接下來我們會先對訓練進行設置，包括安置好訓練檔、修改檔案內參數等等。

訓練設置

首先我們在先前創立的 cfg 資料夾中再創建資料夾及檔案：



接下來一一對所創立的檔案/資料夾進行檢查：

1.lil_train.data

我們利用 notepad++修改此檔案的內容如下：

```
1 classes = 2
2 train = ./lil_train/cfg/train.txt
3 valid = ./lil_train/cfg/val.txt
4 names = ./data/lil_train.names
5 backup = ./lil_train/cfg/weights
```

其中 line 1 的種類設置，由於我們本次要訓練的是人臉辨識是否有戴口罩的模型，因此種類只會有兩種，分別是有戴口罩和沒有戴口罩，因此 #classes = 2。

2.lil_train.names

我們利用 notepad++修改此檔案的內容如下：

```
1 no_mask
2 mask
```

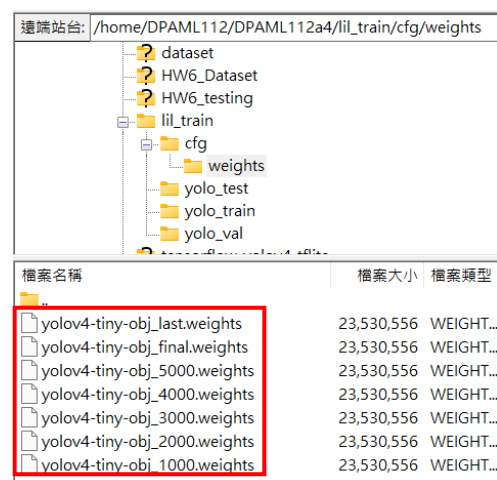
在人臉口罩辨識中，只有”有戴口罩”和”沒有戴口罩”這兩個種類，因此我們將種類名稱 no_mask、mask 修改於 lil_train.names 中。

3.train.txt、val.txt

於上述資料準備中，我們已用自動化程式將所有資料的存放路徑寫入這兩個檔案中，而分別這兩個 txt 檔內容為每個訓練資料、驗證資料的路徑，圖片可以參考資料準備的部分。

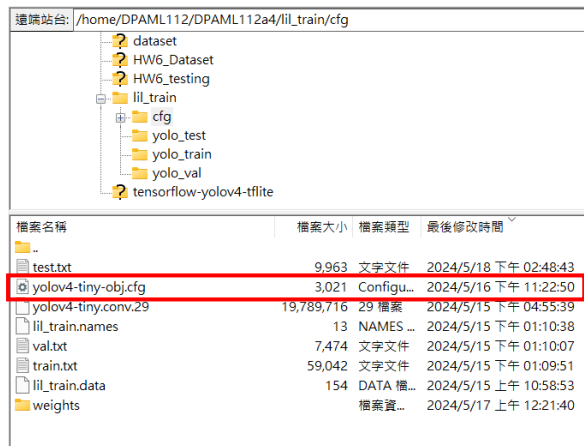
4.weights

這個資料夾在訓練時，會將訓練中、最終訓練結果的權重值存在這個資料夾中，如下圖：

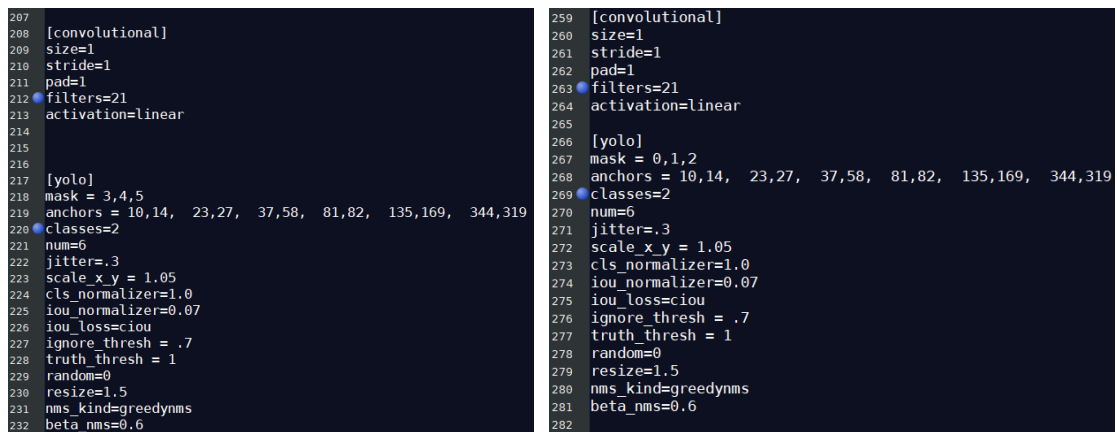


檢查完上述檔案/資料夾內容沒有問題後，接下來我們設定訓練模型 yolov4-tiny 的 config。

首先從 darknet/cfg 中找到這次要使用的 yolov4-tiny-custom.cfg 將此檔複製到 lil_train/cfg 中：

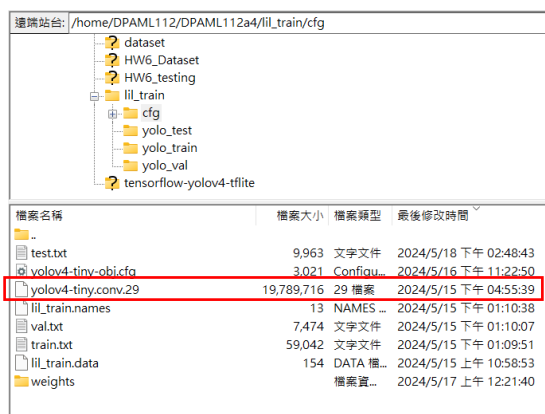


接下來我們以 notepad++ 來修改 config 中的參數：



在 line 212、220、263、269 中，根據講義中的公式 $(C+5)*B$ ，由於我們口罩辨識種類為 2，使用 B 預設為 3，因此這邊的 filters 設置為 $(2 + 5) * 3 = 21$ 、classes 設置為 2。

將預訓練權重 yolov4-tiny.conv.29 丟入 lil_train/cfg 中：



設置完畢後，我們就可以開始進行訓練。

訓練過程

啟用 YOLO 環境後，我們輸入以下訓練指令：

```
(base) (15:16:16)DPAML112a4@Vodka:~ $ activate YOLO
(base) (15:16:21)DPAML112a4@Vodka:~ $ conda activate YOLO
(YOLO) (15:16:24)DPAML112a4@Vodka:~ $ ./darknet/darknet detector train ./lil_train/cfg/lil_train.data ./lil_train/cfg/yolov4-tiny-obj.cfg ./lil_train/cfg/yolov4-tiny.conv.29 - dont_show
```

就會開始進行訓練：

```
6/500200: loss=307.2 hours left=416.4
6: 307.166779, 307.752167 avg loss, 0.000000 rate, 0.286950 seconds, 192 images, 416.384466 hours left
OpenCV exception: draw_train_loss()
Loaded: 0.985298 seconds - performance bottleneck on CPU or Disk HDD/SSD
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.287512), count: 22, class_loss = 107.555634, iou_loss = 0.050850, total_loss = 107.606483
```

在這個訓練中，訓練的 epoches 為 500200，是一個非常大的數字，起初是有讓他訓練，但一陣子之後我們發現 loss 值只有在 0.5~0.9 的跳，loss 值已經到非常低了，可能已經達到 Overfitting 的情況，因此當時我們直接中斷訓練，重新修改 yolov4.cfg 中的內容：

```
17
18 learning_rate=0.00261
19 burn_in=1000
20 max_batches = 5000
21 policy=steps
22 steps=400000,450000
23 scales=.1,.1
```

在這邊我們做一個初步的測試，將原本的 500200 改為 5000 即可，儲存之後我們重新訓練：

```
4/5000: loss=332.4 hours left=4.7
4: 332.407959, 331.946747 avg loss, 0.000000 rate, 0.356169 seconds, 128 images, 4.740118 hours left
OpenCV exception: draw_train_loss()
Loaded: 0.211751 seconds - performance bottleneck on CPU or Disk HDD/SSD
```

可以看到 max_epochs 已經改為 5000，經過一段時間訓練後，我們可以在 lil_train/weights 中看到這次訓練系統幫我們儲存的檔案：

遠端站點: /home/DPAML112/DPAML112a4/lil_train/cfg/weights					
<ul style="list-style-type: none"> results scripts src dataset HW6_Dataset HW6_testing lil_train <ul style="list-style-type: none"> cfg weights 					
檔案名稱	檔案大小	檔案類型	最後修改時間	權限	擁有人/...
..					
yolov4-tiny-obj_last.weights	23,530,556	WEIGHT...	2024/5/20 下午 03:42:59	-rw-rw-...	DPAML1...
yolov4-tiny-obj_final.weights	23,530,556	WEIGHT...	2024/5/17 上午 12:21:40	-rw-rw-...	DPAML1...
yolov4-tiny-obj_5000.weights	23,530,556	WEIGHT...	2024/5/17 上午 12:21:40	-rw-rw-...	DPAML1...
yolov4-tiny-obj_4000.weights	23,530,556	WEIGHT...	2024/5/17 上午 12:09:59	-rw-rw-...	DPAML1...
yolov4-tiny-obj_3000.weights	23,530,556	WEIGHT...	2024/5/16 下午 11:58:19	-rw-rw-...	DPAML1...
yolov4-tiny-obj_2000.weights	23,530,556	WEIGHT...	2024/5/16 下午 11:46:37	-rw-rw-...	DPAML1...
yolov4-tiny-obj_1000.weights	23,530,556	WEIGHT...	2024/5/16 下午 11:35:01	-rw-rw-...	DPAML1...

四、訓練結果

有了訓練出的權重，我們將使用測試集資料進行測試，在接下來的部分中，會使用多個圖片才進行測試，而為了檢驗模型正確率是否有隨著訓練而上升，我們會套入不同次數(ex:1000、2000...)的權重來檢驗，測試時我們是輸入以下指令：

`./darknet detector test cfg/lil_train.data cfg/yolov4-tiny-obj.cfg + (預測試權重)`

+ (圖片)

而造成當我們要測試多筆權重、圖片時，總共要輸入 `#權重 * #圖片` 次，會相當麻煩，因此我們自己撰寫了一個 python 腳本來自動化輸入上述指令，內容僅為簡單的雙層迴圈，並且將輸出檔案改名為對應圖片和對應權重，以方便管理輸出內容。

```




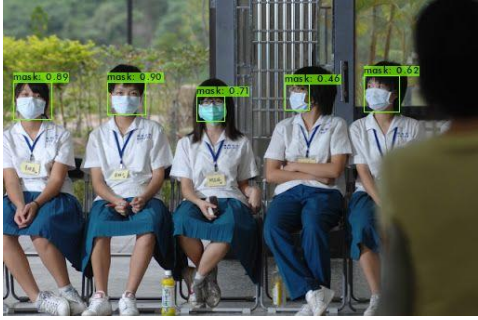




1 import os
2 import subprocess
3
4 # 圖片和權重文件的目錄
5 image_dir = 'data/test'
6 weights_dir = 'weights'
7 output_dir = 'predictions'
8
9 # 確保輸出目錄存在
10 os.makedirs(output_dir, exist_ok=True)
11
12 # 遍歷權重文件夾
13 for weight in os.listdir(weights_dir):
14     if weight.endswith('.weights'):
15         weight_path = os.path.join(weights_dir, weight)
16         print(f'正在使用權重文件: {weight_path}')
17         # 遍歷圖片文件夾
18         for image in os.listdir(image_dir):
19             if image.endswith('.jpg'):
20                 image_path = os.path.join(image_dir, image)
21                 print(f'正在處理圖片: {image_path}')
22                 # 構造輸出圖片名稱
23                 output_image_name = f'{os.path.splitext(image)[0]}_{os.path.splitext(weight)[0]}.jpg'
24                 output_image_path = os.path.join(output_dir, output_image_name)
25                 # 執行Darknet命令
26                 subprocess.run(['./darknet', 'detector', 'test', 'cfg/lil_train.data', 'cfg/yolov4-tiny-obj.cfg', weight_path, image_path])
27                 # 將預測的圖片移動到指定的輸出文件夾
28                 if os.path.exists('predictions.jpg'):
29                     os.rename('predictions.jpg', output_image_path)

```

要判斷模型表現的好壞，我們將會測試以下場合：

1.全部人皆為正臉對視螢幕

為了要可以測試最基本的功能，檢驗此模型是否能夠正常辨識是否有口罩。

	Test_image1	Test_image2
原圖		
epoch 1000		
epoch 2000		
epoch 3000		

epoch 4000		
epoch 5000		

2.部分人拍攝到側臉





有時候我們得到的照片中，有些人並不會正視鏡頭，因此測試側臉是否也能被偵測。

	Test_image1	Test_image2
原圖		
epoch 1000		
epoch 2000		

epoch 3000		
epoch 4000		
epoch 5000		

3.除了主要聚焦在部分人身上，背景含有未對焦的人

一樣為特殊情況，有時候照片中並不會每個人臉都是清晰，同時用這個例子來檢測若照片為模糊時模型是否可以運作。

	Test_image1	Test_image2
原圖		
epoch 1000		

epoch 2000		
epoch 3000		
epoch 4000		
epoch 5000		

4.照片內有一大群民眾

測驗此模型的能耐，若照片內有非常多目標物件要被偵測時，我們想知道模型的效能會是如何。

	Test_image1	Test_image2
原圖		

epoch 1000		
epoch 2000		
epoch 3000		
epoch 4000		
epoch 5000		

五、困難與討論