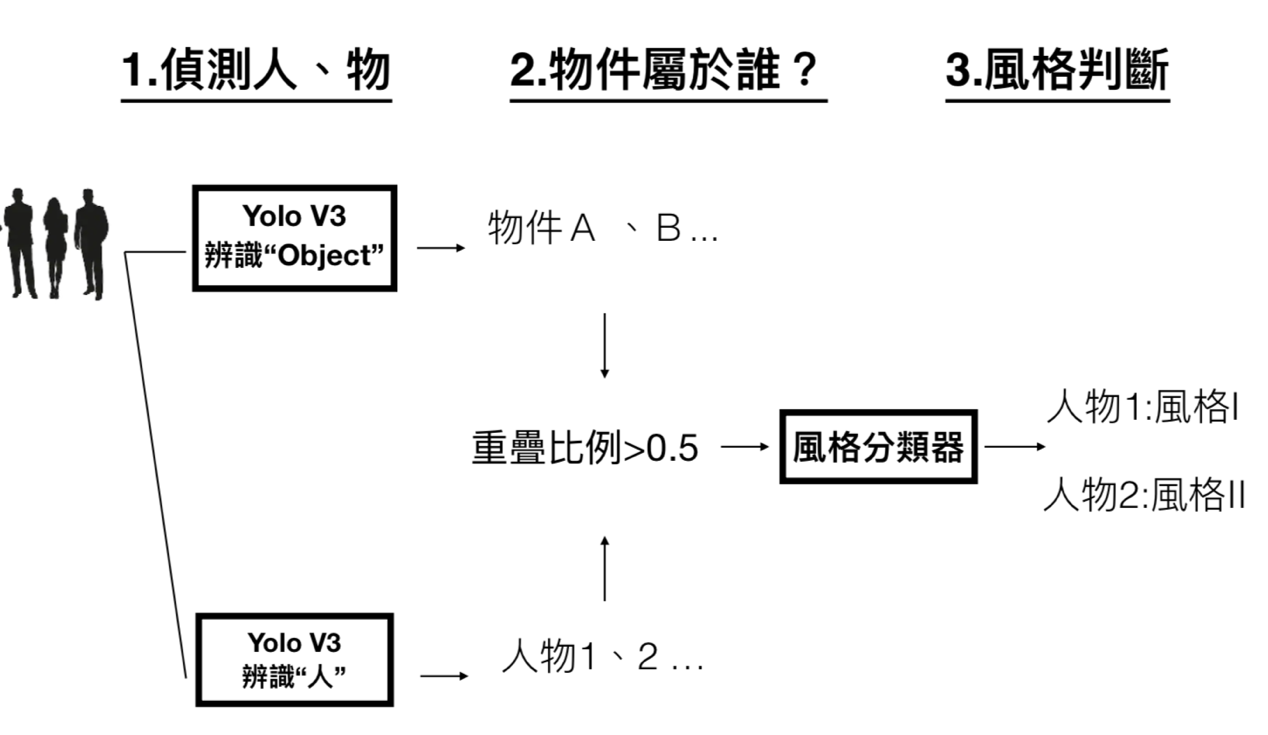
**風格辨識程式說明文件**

**一、方法說明**

(1)模型邏輯

為達成辨識「多人多風格」之目的，此風格辨識程式由3個分類器組成：服裝物件辨識、人物辨識與風格分類器。其中人物辨識的目的在於判斷畫面中的服裝物件歸屬於誰，接著再依序每個人的所屬物件進行風格的判斷。

****

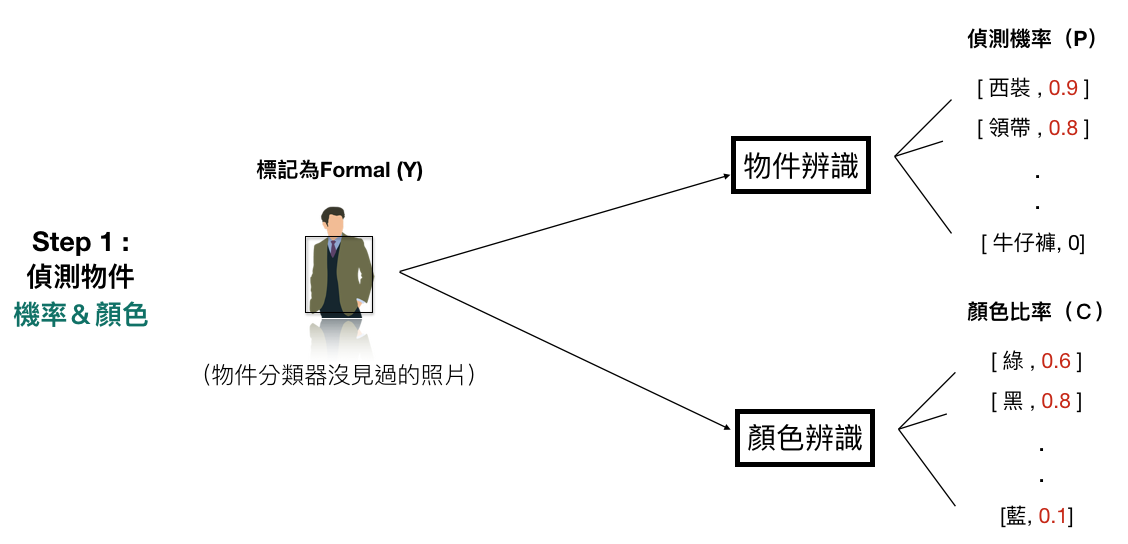
（2）物件辨識

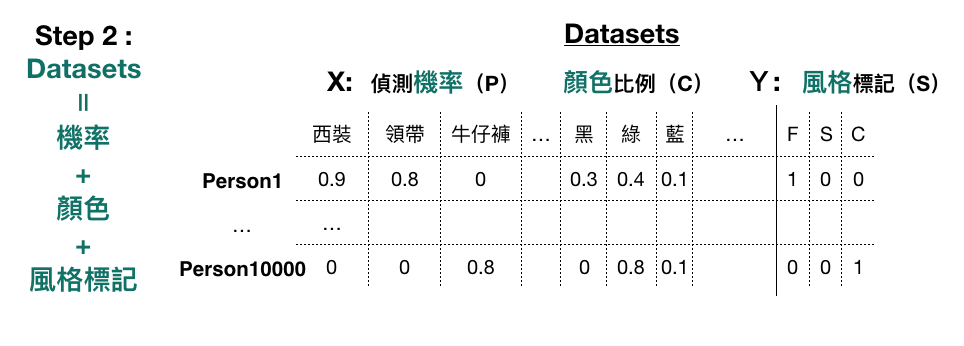
本模型使用Yolo V3作為物件和人物辨識模型，詳細流程會於後面章節介紹。

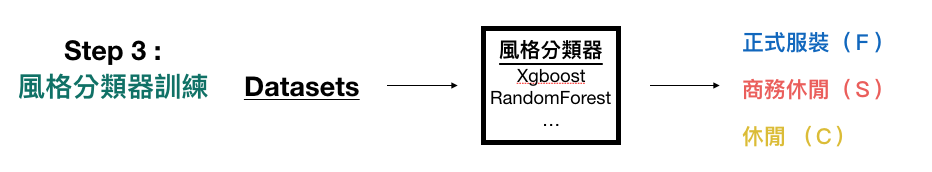
（3）風格辨識

在變數開發過程中發現「服裝物件」、「物件顏色」為「風格」判斷的主要依據，故本程式會以「Yolo V3物件偵測機率」、「OpenCV計算之顏色比例」為自變數（Independent variable）;人工斷定之「風格標記」為依變數（Dependent variable）來建立模型，樣本為1萬多張已標記好風格的照片。

詳細步驟圖示如下：

****

****

****

**二、資料夾＆檔案介紹**

由於檔案眾多，下表為風格辨識訓練流程中重要檔案位置及功能簡介，更多細節在後面章節詳細說明：

|  |  |  |
| --- | --- | --- |
| 流程 | 資料夾/檔案 | 功能 |
| 照片標籤標記 | 1.LabelImg / LabelImg.py | 標籤標記 |
| 資料清理 | 2.All\_xml\_img /1.Add\_label\_to\_picture.py | 將照片檔名加上label名稱，以利於在資料夾中用label名稱搜尋被標記之照片。 |
| 2.All\_xml\_img /2.Select\_wrong\_file.py | 將要修正之照片從原資料夾，移到暫存資料夾，以供LabelImg工具重新標記，或利用Change\_tag.py批量修改標籤名稱 |
| 2.All\_xml\_img /3.Change\_tag.py | 批量修改標籤名稱 |
| 2.All\_xml\_img/Anno | 標籤檔 |
| 2.All\_xml\_img/Picture | 照片檔 |
| 2.All\_xml\_img/check\_label | 加上label名稱之照片檔 |
| 2.All\_xml\_img/wrong | 從check\_label中挑出之要修改的照片檔 |
| 2.All\_xml\_img/temp\_anno | 待處理之標籤檔 |
| 2.All\_xml\_img/temp\_picture | 待處理之照片檔 |
| 物件辨識訓練 | 3.YoloV3/darknet-master/darknet | 執行YoloV3物件辨識模型主程式 |
| 3.YoloV3/darknet-master /Deepfashion/voc/VOC2007 /VOCdevkit/JPEGImages  3.YoloV3/darknet-master /Deepfashion/voc/VOC2007 /VOCdevkit/Annotations  3.YoloV3/darknet-master /Deepfashion/voc/VOC2007 /VOCdevkit/test.py | 欲訓練之照片存放資料夾  欲訓練之標籤檔存放資料夾  切割train、validation、test 資料之程式 |
| 3.YoloV3/darknet-master /Deepfashion/voc/voc\_label.py  3.YoloV3/darknet-master /Deepfashion/voc/deepfashion.names  3.YoloV3/darknet-master /Deepfashion/voc/deepfashion.data  3.YoloV3/darknet-master /Deepfashion/voc/yolov3.cfg  3.YoloV3/darknet-master /Deepfashion/voc/backup | 將test.py切出之txt檔轉換成可供darknet訓練之格式  欲訓練之標籤名稱  .name (標籤檔)、train.txt & val.txt (訓練集、測試集列表)、backup (權重檔存放地址)、之檔案地址，供darknet讀取  YoloV3網路設定檔  訓練完成之權重存放資料夾 |
| 風格辨識模型Pipeline | 4.Pipeline/Picture/Casual  4.Pipeline/Picture/Formal  4.Pipeline/Picture/Smart casual  4.Pipeline/Picture/Total\_xml | 3種風格（休閒、正式、商務休閒）照片檔之存放資料夾  3種風格照片之標籤檔存資料夾 |
| 4.Pipeline/cfg/  4.Pipeline/weights/ | 訓練好之YoloV3模型設定檔(.cfg檔)、標籤檔(.names檔)  訓練好之YoloV3模型權重檔 |
| 4.Pipeline/Performane/Object/  4.Pipeline/ Performane/Style/  4.Pipeline/test/ | 存放物件辨識成效結果  存放風格辨識成效結果  欲測試資料之名稱列表(txt檔) |
| 4.Pipeline/1.Object\_Confusion\_maxtrix.py | 測試YoloV3訓練好之權重檔效果。  Input為：  上方說明之照片、標籤檔  、.name、.cfg、.weights  Output為：   1. Label\_performance.xlsx:每個標籤的Confusion matrix 2. 顯示Truth positive、False positive、False Nagetive的照片於Performane/Object/資料夾中 |
| 4.Pipeline/2.Datasets.py  4.Pipeline/Form | 2.Datasets.py產生風格訓練模型之Datasets於Form資料夾中 |
| 4.Pipeline/3.Classification.py  4.Pipeline/Pickle | 3.Classification.py產生風格分類器之權重檔於Pickle資料夾中 |
| 4.Pipeline/4.Style\_Confusion\_maxtrix.py | 測試風格分類器模型效果Input為：  1.上方說明之照片、標籤檔  2.物件偵測之  .name、.cfg、.weights檔  3.風格辨識模型之權重檔  Output為：   1. Style\_performance.xlsx:每個標籤的Confusion matrix 2. 顯示False positive的照片於Performane/Style/資料夾中 |
| 4.Pipeline/5.Style\_detect.py  4.Pipeline/Picture\_for\_test | 1.打開Camera測試物件、風格分類器成效  2.讀取Picture\_for\_test資料夾下之照片辨識 |
| Line bot Demo | 5.Line\_bot /app.py | app.py 為Line bot主程式 |
| 5.Line\_bot /static/tmp  5.Line\_bot/static/processed | 接收照片存放位置  處理後照片存放位置 |
| 5.Line\_bot/style/style.py | style.py執行風格辨識並將結果存放於processed |
| 5.Line\_bot/template/dashboard.py | dashboard.py 管理line bot 回應之介面 |

＊原始權重檔由此連結下載：[權重下載連結](https://drive.google.com/drive/folders/1njgF9dutD93s-trFvDK8kUp8Df_G4Smz?usp=sharing)

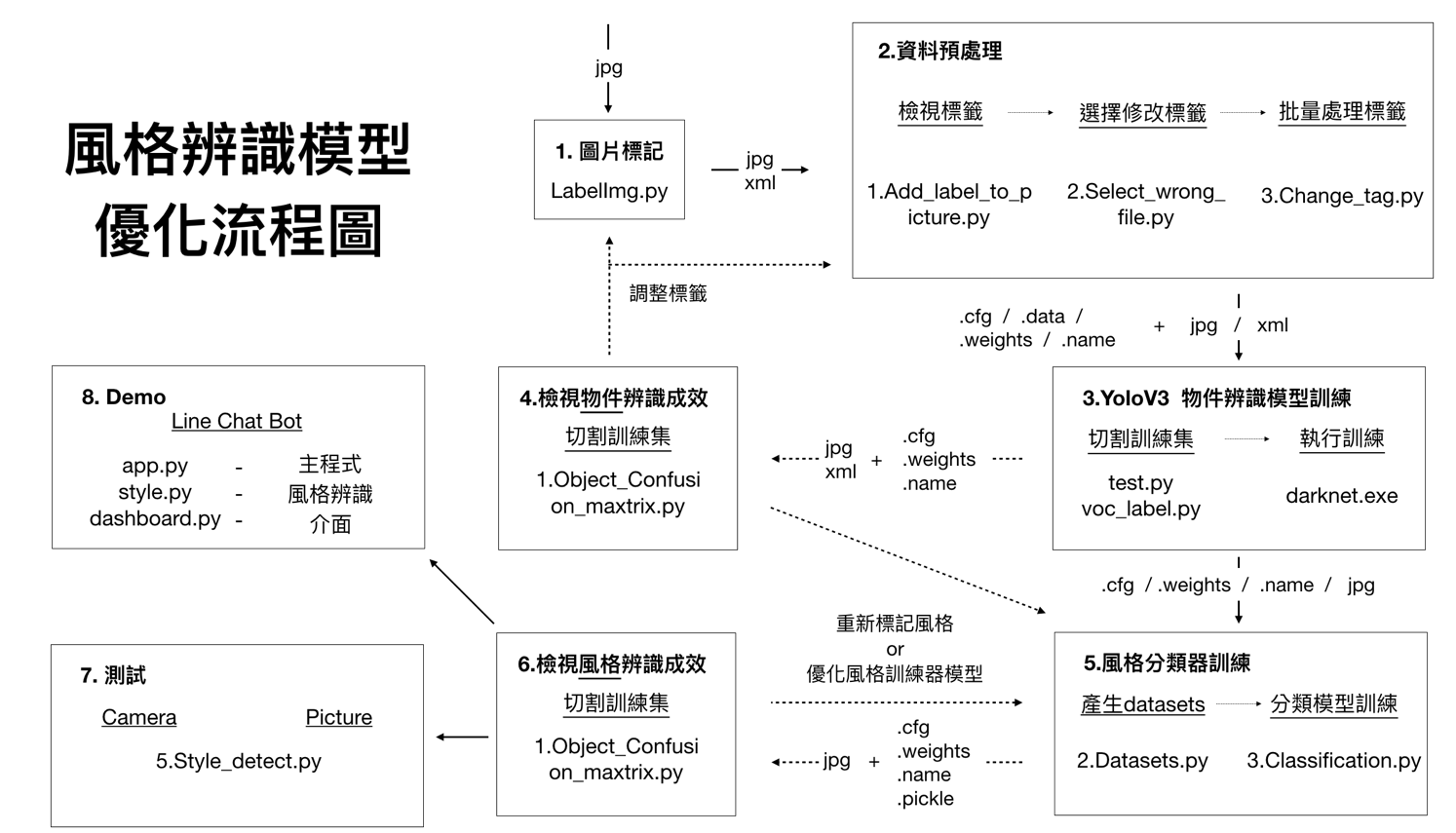
其中，

yolov3.weights ＆ darknet53.conv.74

請置於 /4.Pipeline/weights/ 與 /5.Line\_bot/config/weights/底下

yolov3\_best.weights

請置於/4.Pipeline/weights/日期/ 與 /5.Line\_bot/config/weights/日期/ 底下

**三、優化流程說明**

以下將會依序介紹圖中的程式碼，其中提及的檔案絕對位置，可參考前章：

1. **圖片標記**

標記工具：LabelImg

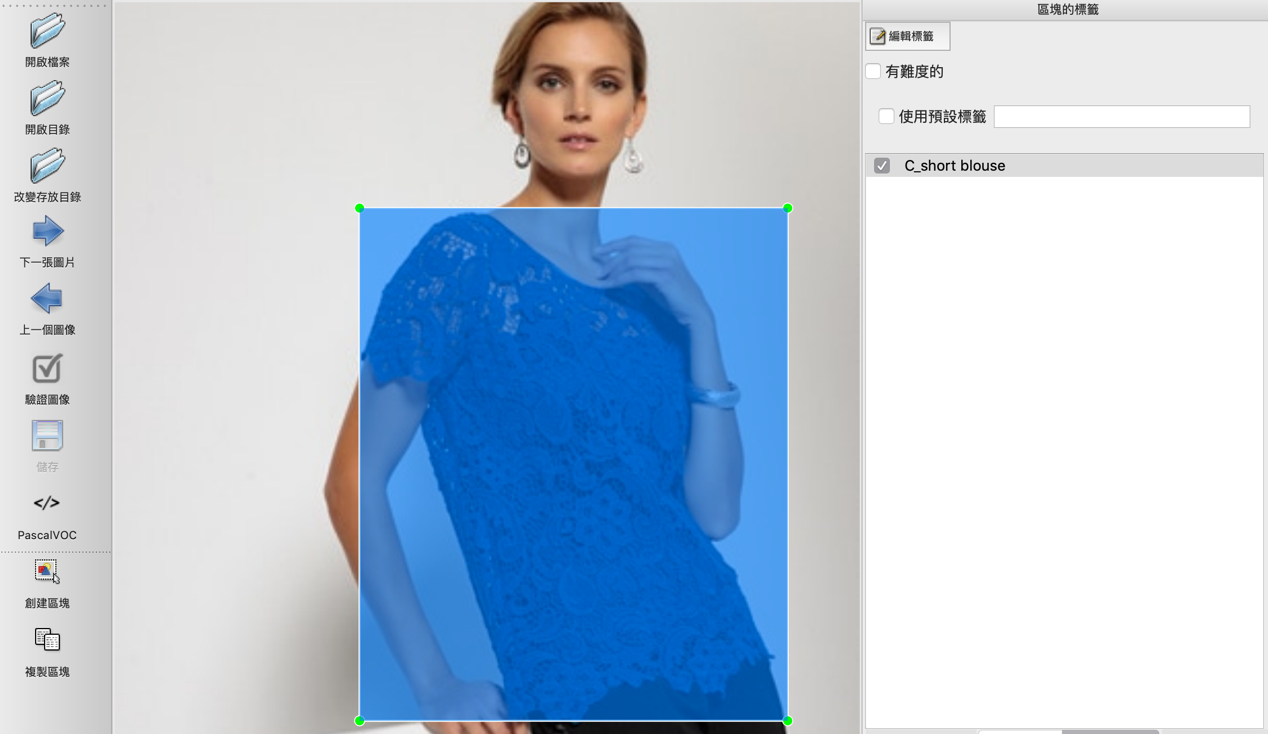
工具下載＆環境安裝介紹：<https://github.com/tzutalin/labelImg>

使用方法：

1. 在Terminal中切到「1.LabelImg」目錄下執行

“python3 labelImg.py” 指令。





1. 跳出視窗後，將「開啟目錄」位置更改為圖檔位置（/2.All\_xml\_img/Picture）、「改變存放目錄」更改為標籤檔位置（/2.All\_xml\_img/Anno）
2. 點擊Ｗ鍵，開始拉框進行標記動作

小訣竅：

1. 可至/1.LabelImg/data/predefined\_classes.txt 更改預設標籤名稱
2. 可至View點選自動儲存模式
3. Ａ、Ｄ字母鍵可以左右切換照片
4. **資料預處理**
5. 檢視標籤：

在Terminal中至2.All\_xml\_img資料夾底下執行以下指令後，可以至“2.All\_xml\_img/check\_label”位置，以標籤搜尋的方式查看圖片



1. 選擇修改標籤

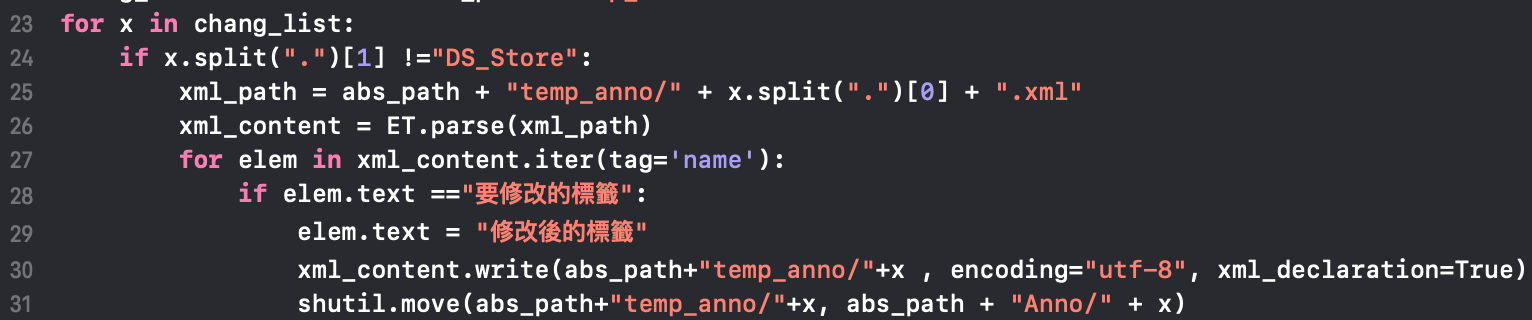
將欲修改的圖片從“check\_label”資料夾底下拖拉至同位置之“wrong”資料夾，執行以下指令後



可將“Picture”、“Anno”資料夾底下與“wrong”資料夾底下圖片名稱相同者，轉移至“temp\_picture”“temp\_anno”資料夾，利用labelImg工具或批量處理標籤工具進行修改。

1. 批量處理標籤

開啟2.All\_xml\_img 資料夾底下的“3.Change\_tag.py”檔

填入“要修改的標籤”與“修改後的標籤”名稱後，執行本代碼即可批量修改“temp\_anno”底下之xml檔內的label名稱

1. **Yolo V3物件模型訓練**

-環境安裝介紹：<https://github.com/AlexeyAB/darknet>

-YoloV3有五個設定檔，分別為：.names、.data、train.txt、test.txt、yolov3.cfg ，設定方式以及參數介紹可以參考以下連結:

[建立自己的YOLO辨識模型 – 以柑橘辨識為例](https://chtseng.wordpress.com/2018/09/01/%E5%BB%BA%E7%AB%8B%E8%87%AA%E5%B7%B1%E7%9A%84yolo%E8%BE%A8%E8%AD%98%E6%A8%A1%E5%9E%8B-%E4%BB%A5%E6%9F%91%E6%A9%98%E8%BE%A8%E8%AD%98%E7%82%BA%E4%BE%8B/)

[深度學習-物件偵測YOLOv1、YOLOv2和YOLOv3 cfg 檔解讀(一)](https://medium.com/@chih.sheng.huang821/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E7%89%A9%E4%BB%B6%E5%81%B5%E6%B8%ACyolov1-yolov2%E5%92%8Cyolov3-cfg-%E6%AA%94%E8%A7%A3%E8%AE%80-75793cd61a01)

-在本程式中，.names 、.data .cfg 放置於voc在資料夾底下

（可參考第二章說明）

訓練流程如下：

1. 切分train\_val\_test訓練集:

至VOC2007資料夾底下開啟 test.py 修改trainval\_percent、train\_percent兩比例後執行test.py檔，會在 / ImageSets/Main/ 底下產生test.txt 、train.txt、val.txt、trainval.txt4個檔案

trainval\_percent = 0.3

train\_percent = 0.7

以上述比例設定，切分結果如下：

Total data

trainval.txt

train.txt

：

3

7

val.txt

test.txt

：

7

3

1. 由於之前用LabelImg標記好的標籤檔為VOC格式（以xml檔儲存），但Yolo 訓練需要將標籤檔轉成Yolo的格式。且darknet讀取train.txt 、test.txt、val.txt內容需用絕對路徑的形式

執行voc資料夾底voc\_label.py可以幫我們完成這些事情



執行完上述指令後，會在 /VOCdevkit/VOC2007/labels 下將所有xml檔轉成yolo格式，並在voc資料夾底下建立test.txt、train.txt、val.txt 三個檔案

\*\*\*注意\*\*\*

voc\_label.py中的classes列表，需改成所要訓練之標籤，且順序要與.names檔一致

1. 完成voc資料夾下 .data .cfg .names檔設定
2. 在Terminal 中切換到 darknet-master資料夾

執行 ./darknet detector train /.data路徑 /.cfg路徑 /.預訓練權重的路徑

即可開始進行訓練，訓練好之權重，會存放在backup資料夾

＊預訓練權重可放前幾次訓練之權重

1. **檢視Yolo V3物件訓練成效**
2. 將訓練Yolo模型之.cfg、.names檔放置/4.Pipeline/cfg/ 底下，並將訓練之最優權重 .weights檔放置在/4.Pipeline/weights資料夾下
3. 至1.Object\_Confusion\_maxtrix.py 內修改檔案名稱



1. 執行 1.Object\_Confusion\_maxtrix.py 檔後，會在/4.Pipeline/Performance/Object/ 下 ，會在產出訓練結果

“Label\_performance.xlsx”裡面會有每個物件confusion matrix的資訊，另外會在4.Pipeline/Performance/ 下的三個資料夾顯示False positve、 False nagetive、Truth positive之圖片，其中FP\_picture\_o資料夾內檔名為「 index\_辨識到但ground truth沒有的物件1\_機率1\_辨識到但ground truth沒有的物件2\_機率2..etc 」 ; FN\_picture\_o資料夾內檔名為「 index\_沒辨識到的物件1\_沒辨識到的物件2\_..etc 」 ; TP\_picture\_o資料夾內檔名為「 index\_辨識對的物件1\_機率1\_辨識對的物件2\_機率2..etc 」 可供標籤搜尋以檢視成效、以及優化方向。

＊可更改/test/ 中的txt檔內容，決定要測試的照片範圍

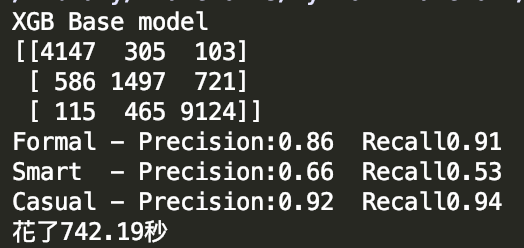
1. **風格分類器訓練**
2. 將訓練Yolo模型之.cfg、.names檔放置/4.Pipeline/cfg/ 底下，並將訓練之最優權重 .weights檔放置在/4.Pipeline/weights資料夾下
3. 至2.Datasets.py 內修改檔案名稱



1. 執行2.Datasets.py ，會在Form資料夾中生成 X 欄位有物件名稱、顏色 ; Ｙ欄位有風格類型之Datasets xlsx檔
2. 執行風格分類優化模型 3.Classification.py 檔

此程式碼目的為測試何種模型 & Resampling方法效果最優，主要分成兩個部分：

* 1. 154～215行：測試何種模型 & Resampling方法效果最優，成效會以Confusion Matrix的形式呈現 如下



* 1. 272～284行：將最好的模型和Resampling結果經調參後輸出，其格式為pickle檔，存放在 /4.Pipeline /Pickle/中

1. **檢視分類器成效**

(I)開啟/4.Pipeline/4.Style\_Confusion\_maxtrix.py ，

更新 .pickle、.cfg、.names、 .weights 、test.txt 內容







(II)執行4.Style\_Confusion\_matrix.py



程式執行完畢後會有兩個結果：

* + 1. 輸出Confusion Matrix 在/4.Pipeline/Performance/Style/ 資料夾底下，檔名為Style\_performance.xlsx
    2. 在/4.Pipeline/Performance/Style/FP\_picture\_s 資料夾底下，會顯示預測錯誤之照片以供檢視，其圖片命名規則如下：

index\_P?\_A?\_機率.jpg

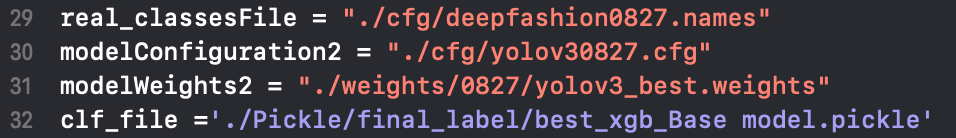
P?為預測為？風格 ; A?為真實為?風格

正式:?=0 商務休閒:?=1 休閒:?=2

1. **測試**

設定：

開啟5.Style\_detect.py 檔，更新物件辨識 .names 、 .cfg 、.weights檔，以及用於風格分類之.pickle檔後，即可進行測試



1. Camera

直接執行 5.Style\_detect.py ，則會開啟攝像頭進行辨識

1. 照片

利用 –i 參數 輸入照片進行測試，結果會輸出至Picture\_for\_test資料夾中，檔名為原檔名+\_yolo.jpg

範例：





＊最上方顯示風格的為人物辨識的框，底下會顯示服裝物件主要3個色系

1. 影片

利用 –v 參數 輸入影片進行測試，會輸出結果會輸出至Picture\_for\_test資料夾中，檔名為原檔名+\_yolo.avi

1. **Line Bot Demo**

待更新