cv2.CASCADE_SCALE_IMAGE：按比例正常檢測。

cv2.CASCADE_DO_CANNY_PRUNING：利用 Canny 邊緣檢測器來排除一些邊緣很少或很多的圖像區域。

cv2.CASCADE_FIND_BIGGEST_OBJECT：只檢測最大的物體。

cv2.CASCADE_DO_ROUGH_SEARCH：只做初略檢測。

例如設定辨識最小區域為 (10,10)、誤檢率為 5、模式為正常偵測，用於偵測 image 圖片，並將偵測結果存於 faces 變數中：

```
faces = faceCascade.detectMultiScale(image, scaleFactor=1.1,
    minNeighbors=5, minSize=(10,10), flags = cv2.CASCADE_SCALE_IMAGE)
```

detectMultiScale 方法可偵測圖片中多張臉部，所以傳回值是串列，串列元素是由臉部圖形左上角 x 坐標、y 坐標、臉部寬度、臉部高度組成的元組，以下列程式就可取得每一張臉部圖形的資料：

```
for (x,y,w,h) in faces:
```

x、y 為臉部圖形左上角 x、y 坐標，w、h 為臉部圖形的寬及高，有了這些資料就可對臉部做各種處理，如標識出臉部位置、擷取臉部圖形等。

範例：標識臉部位置

標識出圖片中臉部位置，並在左下角顯示辨識出臉部的數量。

程式碼：detectFace1.py

```
import cv2
casc_path = cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(casc_path)
```


碁峰 www.gotop.com.tw

```
4   image = cv2.imread("media\\person1.jpg")
5   faces = faceCascade.detectMultiScale(image, scaleFactor=1.1,
      minNeighbors=5, minSize=(30,30), flags = cv2.CASCADE_SCALE_IMAGE)
6   imgheight=image.shape[0] #圖片高度
7   imgwidth=image.shape[1]  #圖片寬度
8   cv2.rectangle(image, (10,imgheight-20), (110,imgheight),
      (0,0,0), -1) #左下角黑色矩形
9   cv2.putText(image,"Find " + str(len(faces)) + " face!", (10,imgheight-5),
      cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255), 2)
10  for (x,y,w,h) in faces: #臉部矩形
11      cv2.rectangle(image,(x,y),(x+w, y+h),(128,255,0),2)
12  cv2.namedWindow("facedetect")
13  cv2.imshow("facedetect", image)
14  cv2.waitKey(0)
15  cv2.destroyWindow("facedetect")
```

程式說明

- 2-3　建立偵測物件。
- 4　讀取要偵測的圖片。
- 5　進行偵測，辨識結果存於 `faces` 變數中。
- 6-7　取得圖片的高度和寬度，請注意：`image.shape[0]` 是圖片高度、`image.shape[1]` 是圖片寬度。
- 8　在圖片左下角畫一個黑色矩形，做為顯示文字的區域。
- 9　以白色文字顯示臉部數量。
- 10-11　以迴圈逐一在臉部位置畫矩形以標識出臉部位置。

detectMultiScale 方法可偵測圖片中多張臉部，讀者可修改第 4 列程式讀取 <person3.jpg> 或 <person5.jpg> 觀看標識多張臉部結果。

▲ detectFace3.py

▲ detectFace5.py

碁峰 www.gotop.com.tw

範例：擷取臉部及存檔

使用 OpenCV 取得臉部區域後，擷取臉部圖形並存檔。

程式碼：saveFace1.py

```
……
7   count = 1
8   for (x,y,w,h) in faces:
9       cv2.rectangle(image, (x,y), (x+w,y+h), (128,255,0), 2) # 框選臉部
10      filename = "media\\face" + str(count)+ ".jpg"  # 存檔名稱
11      image1 = image[y: y + h, x: x + w]        # 取得臉部圖形
12      image2 = cv2.resize(image1, (400, 400))  # 重置圖形大小
13      cv2.imwrite(filename, image2)
14      count += 1
……
```

程式說明

- 1-6　以 OpenCV 偵測臉部，程式與 <detectFace1.py> 相同。
- 7,10　count 為檔名計數器：第一張圖檔名為 <face1.jpg>、第二張圖檔名為 <face2.jpg>，依此類推。
- 11　擷取臉部圖形。
- 12　將圖形解析度轉成為 400x400。
- 13　儲存圖形檔案。
- 14　計數器增加 1。

修改第 4 列程式為 <person3.jpg> 或 <person5.jpg>，可一次擷取多張臉部圖形。(<saveFace3.py> 及 <saveFace5.py>)

碁峰 www.gotop.com.tw

2.2 圖片特效處理

很多 OCR 辨識，為了達到較佳的辨識效果，通常會將彩色的圖片先轉換為灰階圖片，再將灰階圖片轉換為黑白圖色，再進行 OCR 辨識。

2.2.1 認識圖片的維度

OpenCV 的維度中，索引第 0 維表示高度 (或 y 坐標)、索引第 1 維則表示寬度 (或 x 坐標)，因為它和其他圖形處理軟體坐標使用的習慣正好相反，筆者也經常被搞得暈頭轉向。

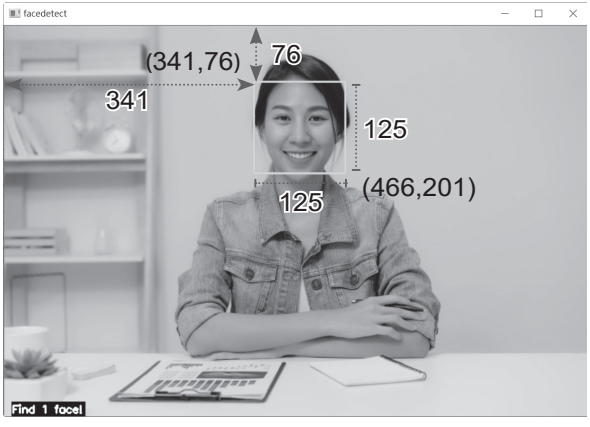
彩色圖片的維度

如果我們開啟一張彩色的圖片，以 shape 屬性觀察其維度，以範例說明如下：

```
img = cv2.imread('media\\person1.jpg')  # 讀取要辨識的圖形
print("img.shape=",img.shape) # (533, 800, 3)
```

代表圖片高度為 533、寬度為 800，一般會說這張圖片為 (800*533)，即寬度為 800、高度為 533，但 OpenCV 和我們一般使用的習慣不同，請讀者一定要多加留意和體會。

索引第 2 維度為 3，代表它包含有 3 個值，即 B、G、R (注意：不是 R、G、B)，也就是使用這 3 個值組成該像素的顏色。



碁峰 www.gotop.com.tw

設定像素

設定 (x,y) 坐標點像素語法如下：

```
圖形 [y,x]= 顏色值
```

由於彩色圖片的像素值是一個 [B、G、R] 串列，使用範例如下：

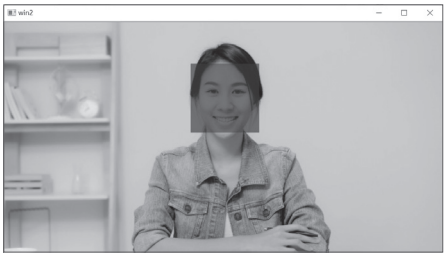
```
img[row,col]=0        # B、G、R 都是 0，所以是黑色
img[row,col]=[0,0,255]   # B=0、G=0、R=255 ，所以是紅色
```

改變每個像素的 B、G、R 值，例如：設定 B=0、G=50、R 的值不變，就可以達到圖片特效處理的效果。

```
img[row,col][0]=0    # 設定 B 值為 0
img[row,col][1]=50   # 設定 G 值為 50
```

範例：圖形擷取與特效處理

讀取並顯示圖片和維度，擷取圖片中的人臉，顯示其像素值，並將人臉做特效處理。



shape.py

```
import cv2
img = cv2.imread('media\\person1.jpg') # 讀取要辨識的圖形
print("img.shape=",img.shape) # (533, 800, 3)
cv2.imshow("win1", img)

x,y,w,h=341,76,125,125
face = img[y: y + h, x: x + w]     # 取得臉部圖形

x,y,w,h=341,76,125,125
for row in range(y,y+y):      # 總共有 h 列 (125 列 )
    for col in range(x,x+w): # 每一列有 w 個像素 (125 個像素 )
        print(img[row,col],end=" ")
    print()
```


碁峰 www.gotop.com.tw

5.2.3 在命令提示字元視窗執行 Yolo

終於完成 Yolo 安裝，可以執行 Yolo 了！

偵測圖片中物體

dartnet 架構可用於 Yolov3 及 Yolov4。

在命令提示字元視窗偵測圖片中物體的語法為：

```
darknet.exe 程式路徑 detect 組態檔路徑 權重檔路徑 圖片檔路徑
```

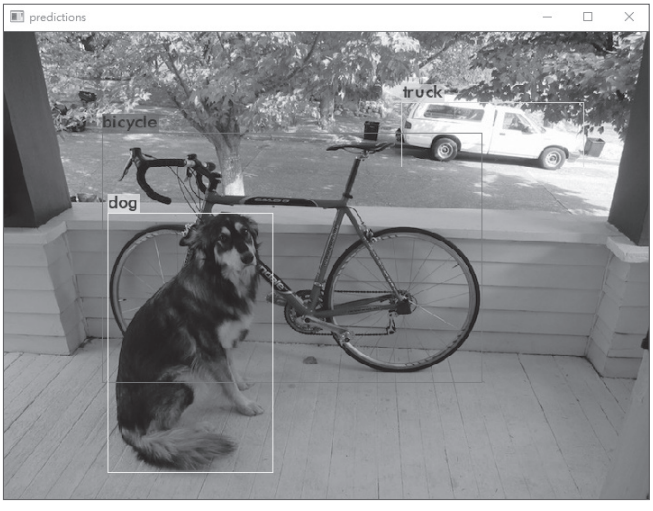
- **darknet.exe 程式路徑**：預設在 <dartnet> 資料夾的根目錄，「.exe」可以省略。
- **組態檔路徑**：預設在 <dartnet> 資料夾的 <cfg> 目錄，Yolov3 的組態檔為 <yolov3.cfg>，Yolov4 的組態檔為 <yolov4.cfg>。
- **權重檔路徑**：Yolov3 的權重檔為 <yolov3.weights>，Yolov4 的組態檔為 <yolov3.weights>。這兩個權重檔在前一節已複製到 <dartnet> 資料夾的根目錄中。

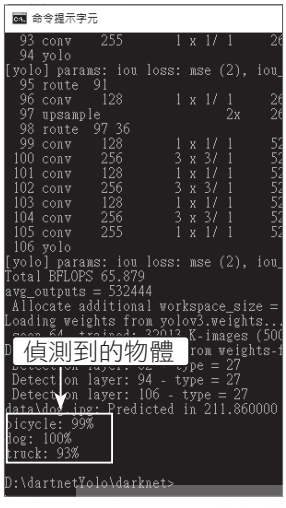
開啟命令提示字元視窗，切換到 <D:\dartnetYolo\darknet> 資料夾，先測試 Yolov3，執行下列命令：

```
darknet.exe detect cfg\yolov3.cfg yolov3.weights data\dog.jpg
```

「data\dog.jpg」是 dartnet 架構提供的示範圖片。

若出現下列結果就表示 Yolo 正確安裝了！





碁峰 www.gotop.com.tw

5.3 訓練自訂 Yolo 偵測模型

雖然 Yolo 已經可以偵測 80 種物體，但終究會有偵測自己物體的需求，例如玫瑰花、火龍果等，這時就必須訓練自己的 Yolo 模型來進行偵測。

5.3.1 標記工具

建立自己 Yolo 模型的第一步是對圖片標記，即框選出圖片中要偵測的物體，因此首先要安裝標記工具。

安裝標記工具

先安裝 PyQt5 模組，在命令提示字元視窗執行下面命令：

```
pip install pyqt5 --user
```

接著安裝標記工具 labelimg 模組：

```
pip install labelimg
```

如此就完成標記工具的安裝。

使用標記工具

我們使用 5 張口罩圖片做為標記示範：請將本章範例 <masktest> 資料夾複製到 <D:\dartnetYolo> 資料夾中。<masktest> 資料夾的 <images> 資料夾有 5 張口罩圖片，<labels> 資料夾為空的，用來儲存標記後的文字檔。

在命令提示字元視窗執行下面命令即可啟動標記工具：

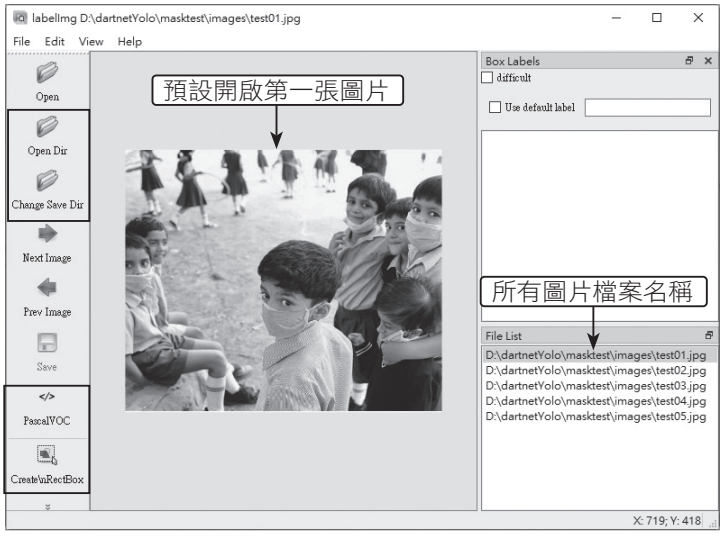
```
labelimg
```

按 **Open Dir** 鈕，選擇要標記的圖片資料夾 (<D:\dartnetYolo\masktest\images>)，此時右下方會顯示所有圖片檔名。按 **Change Save Dir** 鈕，選擇存放標記文字檔的資料夾 (<D:\dartnetYolo\masktest\labels>)。

此標記工具可以儲存「PascalVOC」及「Yolo」格式，**PascalVOC** 鈕可以切換不同儲存格式。由於本章將使用一個現成的資料來訓練，該資料為 PascalVOC 格式，所以此處使用 PascalVOC 格式。

碁峰 www.gotop.com.tw

按 **Create RectBox** 鈕即可開始標記。



本章口罩圖片標記分為三種：口罩正確戴好標記為「with_mask」，未戴口罩標記為「without_mask」，有戴口罩但未戴好標記為「mask_weared_incorrect。」

軟體最上方會顯示目前正在標記的圖片檔名。拖曳滑鼠框選出要標記的物件後放開滑鼠左鍵，在 **labelimg** 對話方塊中輸入標籤名稱 (此處為「with_mask」)，按 **OK** 鈕就完成第一個標記。

碁峰 www.gotop.com.tw

6.3 使用自行訓練的 Yolo 模型

自己的 Yolo 模型訓練完成後，就可使用模型來辨識物體了！

下面範例皆以自行訓練的 Yolov3-Tiny 模型為例。

6.3.1 在 Colab 中使用自行訓練 Yolo 模型

在 Colab 中使用自行訓練 Yolo 模型，需用到自行訓練的權重檔及要偵測口罩的圖形，因為僅使用兩個檔，只需將檔案上傳到 Google Drive 雲端硬碟即可，不必複製到 Colab 虛擬機器中。

上傳自行訓練的權重檔：以訓練 10000 次的權重檔為例，將本章範例 <weights> 資料夾的 <yolov3-tiny-obj_10000.weights> 檔上傳到 Google Drive 雲端硬碟 <Colab Notebooks> 資料夾。

上傳圖片：將本章範例 <masktest\images> 資料夾上傳到 Google Drive 雲端硬碟 <Colab Notebooks> 資料夾，內含 5 張圖片，此 5 張圖片皆與訓練模型的圖片不同。

在 Colab 中使用 Yolov3 模型偵測圖片中戴口罩狀況的程式如下：(如果使用者已執行訓練 Yolo 模型程式，就只要執行最後 3 列程式即可)

```
!git clone https://github.com/AlexeyAB/darknet.git
%cd darknet
!sed -i "s/GPU=0/GPU=1/g" Makefile
!sed -i "s/CUDNN=0/CUDNN=1/g" Makefile
!sed -i "s/OPENCV=0/OPENCV=1/g" Makefile
!make
%cd /content
!ln -s "/content/drive/My Drive/Colab Notebooks" /godrive
!cp /godrive/maskyolov3_colab.zip /content
!unzip maskyolov3_colab.zip
%cd maskyolov3_colab/
!/content/darknet/darknet detector test cfg/obj.data cfg/yolov3-tiny-obj.cfg
  /godrive/yolov3-tiny-obj_10000.weights /godrive/masktest/images/test01.jpg
from PIL import Image
Image.open('predictions.jpg')
```

執行後會將執行結果儲存於目前所在目錄的 <predictions.jpg> 圖片中，最後 2 列程式為顯示圖片。

碁峰 www.gotop.com.tw

執行結果：

綠色框為口罩配戴正確，紫色框為口罩配戴不正確，橙色框為未配戴口罩。

6.3.2 在 Python 中使用自行訓練 Yolo 模型

使用自有模型與前一章「以 Python 程式執行 Yolov3- 偵測圖片中物體」範例的方式雷同，只是使用的模型是自己訓練的模型而已。

複製權重檔：將本章範例 <weights> 資料夾的 <yolov3-tiny-obj_1000.weights>、<yolov3-tiny-obj_5000.weights> 及 <yolov3-tiny-obj_10000.weights> 檔複製到 <D:\dartnetYolo\maskyolov3\cfg\weights> 資料夾中。

複製圖片：將本章範例 <masktest> 資料夾複製到 <D:\dartnetYolo> 資料夾中。

將本章範例 <maskdemo.py> 複製到 <D:\dartnetYolo\maskyolov3> 資料夾中。<maskdemo.py> 可使用自有模型偵測圖片中物體，程式碼為：

程式碼：maskdemo.py

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

net = cv2.dnn.readNetFromDarknet("cfg/yolov3-tiny-obj.cfg","cfg/weights/yolov3-tiny-obj_10000.weights")  # 讀模型
layer_names = net.getLayerNames()
```


碁峰 www.gotop.com.tw

```
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
classes = [line.strip() for line in open("cfg/obj.names")]
colors = [(0,0,255), (255,0,0), (0,255,0)]

img = cv2.imread("../masktest/images/test01.jpg")  # 讀圖片
img = cv2.resize(img, None, fx=0.4, fy=0.4)
height, width, channels = img.shape
blob = cv2.dnn.blobFromImage(img, 1/255.0, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
outs = net.forward(output_layers)
class_ids = []
confidences = []
boxes = []
for out in outs:
    for detection in out:
        tx, ty, tw, th, confidence = detection[0:5]
        scores = detection[5:]
        class_id = np.argmax(scores)
        if confidence > 0.3:
            center_x = int(tx * width)
            center_y = int(ty * height)
            w = int(tw * width)
            h = int(th * height)
            # 取得箱子方框座標
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.3, 0.4)
font = cv2.FONT_HERSHEY_PLAIN
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[class_ids[i]]
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 1)
        cv2.putText(img, label, (x, y - 5), font, 1, color, 1)
plt.rcParams['figure.figsize'] = [15, 10]
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```


碁峰 www.gotop.com.tw

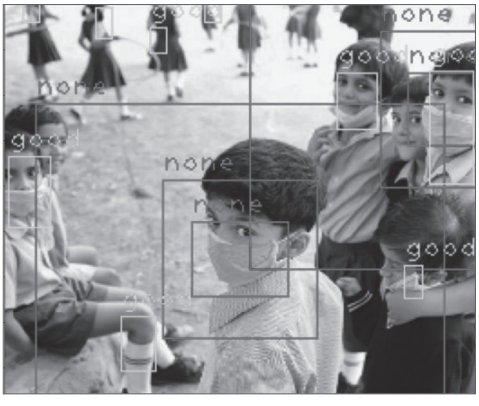
程式說明

- 5　　使用自行訓練的設定檔和權重檔。
- 8　　使用自己的標籤文字檔。
- 11　　讀取要偵測物體的圖片。

其餘程式碼說明請參閱前一章。

執行結果：

(使用者可將第 5 列程式的權重檔改為 <yolov3-tiny-obj_1000.weights> 或 <yolov3-tiny-obj_5000.weights> 比較執行結果)



▲ yolov3-tiny-obj_1000.weights

▲ yolov3-tiny-obj_5000.weights

碁峰 www.gotop.com.tw

```
16
17 cv2.namedWindow("win", cv2.WINDOW_AUTOSIZE)
18 cv2.imshow("win", img)
19 cv2.waitKey(0)
20 cv2.destroyAllWindows()
```

程式說明

- 4　　讀入自行訓練的模型。
- 6　　讀取圖片檔。
- 7　　進行偵測。
- 8　　顯示偵測到的汽車數量。
- 9-15　　逐一處理偵測到的汽車資訊。
- 10-13　　取得汽車矩形的坐標。
- 15　　繪出汽車矩形。

執行後 Console 視窗顯示偵測到的汽車數量及各汽車的位置資料：

```
偵測到的汽車數：5
第 1 輛車：左 688; 上 94; 右 750; 下 156
第 2 輛車：左 498; 上 44; 右 550; 下 95
第 3 輛車：左 325; 上 156; 右 414; 下 245
第 4 輛車：左 645; 上 31; 右 688; 下 74
第 5 輛車：左 517; 上 10; 右 553; 下 46
```

顯示圖形為：

碁峰 www.gotop.com.tw

8.1.5 自行訓練模型應用：車流偵測

本節自行訓練的模型功能是偵測汽車，我們可以利用此模型對道路上的汽車進行流量偵測。

程式碼：carCount.py

```
import cv2
import dlib

detector = dlib.simple_object_detector("carimg.svm")  # 讀取模型檔
video = cv2.VideoCapture('media\\selfcars1.mp4')  # 讀入影片

rectangleColor = (0, 255, 0)  # 框線顏色
currentCarID = 0  # 最新追蹤車輛編號
carTracker = {}  # 記錄所有追蹤車輛
WIDTH = 640  # 設定影片寬度
HEIGHT = 480  # 設定影片高度
# 輸出影片檔
out = cv2.VideoWriter('media\\output.avi',cv2.VideoWriter_fourcc(
    'M','J','P','G'), 10, (WIDTH, HEIGHT))  # 設定輸出影片
while True:
    _, image = video.read()
    if type(image) == type(None):  # 影片播完
        break
    image = cv2.resize(image, (WIDTH, HEIGHT))
    carIDtoDelete = []  # 記錄要移除車輛
    # 追蹤車輛
    for carID in carTracker.keys():
        trackingQuality = carTracker[carID].update(image) # 更新追蹤車輛
        if trackingQuality < 7:  # 車輛離開範圍就移除
            carIDtoDelete.append(carID)
    # 移除追蹤車輛
    for carID in carIDtoDelete:
        print ('移除追蹤車輛 carID ' + str(carID))
        carTracker.pop(carID, None)
    # 偵測車子
    dets = detector(image)  # 偵測車輛
    for det in dets:
        x = det.left()
        y = det.top()
        w = int(det.right() - det.left())
        h = int(det.bottom() - det.top())
```


碁峰 www.gotop.com.tw