

Chapter 2 : YOLOv5 X 物體辨識

本章重點

介紹如何使用 YOLOv5 預訓練模型來辨識物件。並自行訓練物件偵測模型，以訓練和辨識交通號誌為範例。

準備材料



可上網的電腦 (含攝影鏡頭)

學習目標

1. 使用 YOLOv5 預模型來偵測物件。
2. 自行訓練 YOLOv5 模型來偵測物件。

2-1. YOLOv5 物件偵測

安裝 YOLOv5 環境

首先，建立一個 YOLOv5 專用的 Python 3.9 虛擬環境，可由 **Chapter 1** 完成的 `myenv_39` 來複製，名稱為 `yolov5env`：

```
conda create -n yolov5env_39 --clone myenv_39
```

載入 Python 虛擬環境 `yolov5env`：

```
activate yolov5env_39
```

下載 YOLOv5 檔案

先安裝 [git for Windows](https://gitforwindows.org/) 的下載指令：<https://gitforwindows.org/>

下載 YOLOv5 原始碼：

```
# 下載 YOLOv5 原始碼  
(yolov5env_39) git clone https://github.com/ultralytics/yolov5
```

在 Python 虛擬環境之下，安裝 YOLOv5 所需要的各種 Python 的 Package：

```
# 安裝 YOLOv5 所需環境  
(yolov5env_39) cd yolov5  
(yolov5env_39) pip install -r requirements.txt
```

使用 YOLOv5 模型偵測物件

`detect.py` 指令稿

在 YOLOv5 的原始碼中，有附帶一個偵測物件用的 Python 指令稿 `detect.py`，可以使用指定的 YOLOv5 模型來進行物件偵測。例如：拿官方的 `zidane.jpg` 圖檔（位於 `data/images` 資料夾）進行物件偵測：

以 YOLOv5 模型偵測圖片物件

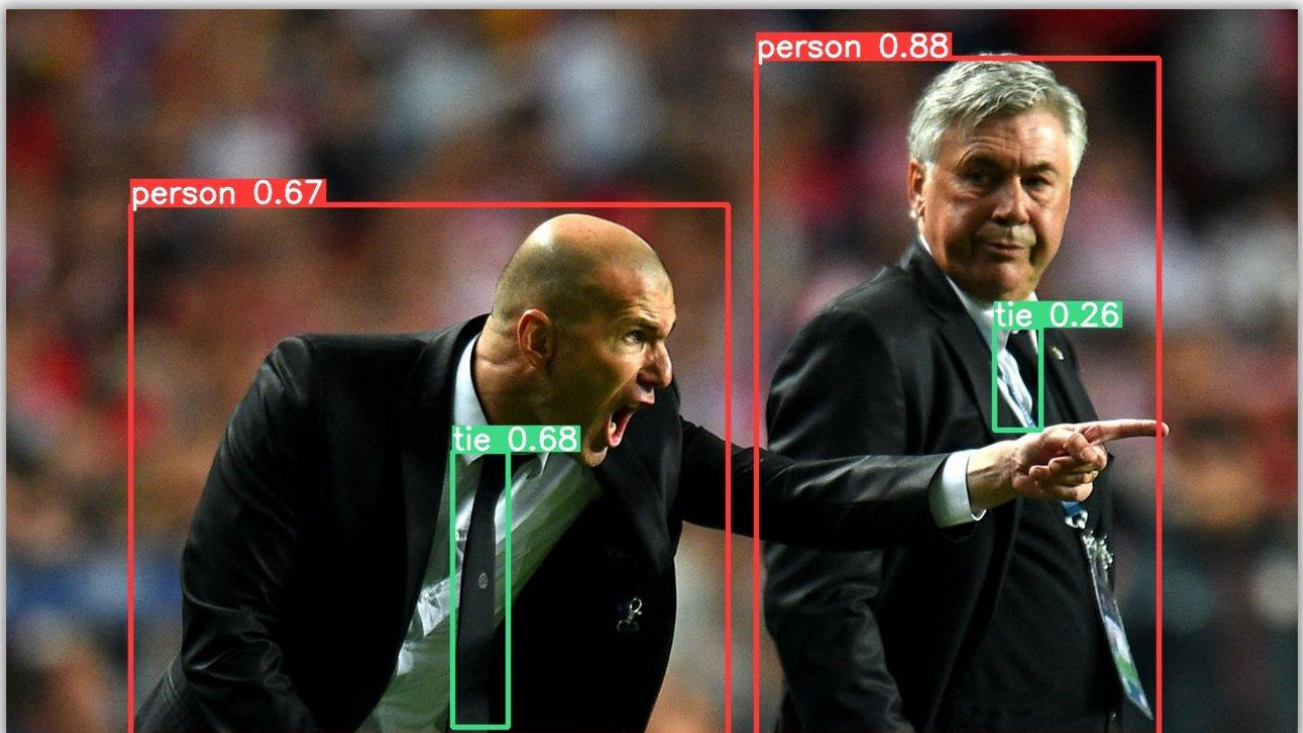
```
(yolov5env_39) python detect.py --source data/images/zidane.jpg
```

```
Anaconda Prompt (yolov5env)
(yolov5env) C:\Users\allow\yolov5>python detect.py --source ./data/images/zidane.jpg
detect: weights=yolov5s.pt, source=./data/images/zidane.jpg, data=data\coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5 v6.1-250-g6adc53b Python-3.9.12 torch-1.11.0+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
image 1/1 C:\Users\allow\yolov5\data\images\zidane.jpg: 384x640 2 persons, 2 ties, Done. (0.257s)
Speed: 1.0ms pre-process, 256.9ms inference, 3.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\exp

(yolov5env) C:\Users\allow\yolov5>
```

其中 `--source` 參數可以用來指定影像來源，支援的格式非常多，包含：電腦攝影機、圖片檔案 (jpg、png)、影片檔案 (mp4)、包含圖片的目錄、YouTube 網址、RTSP 串流位址等，而偵測結果預設會擺放在 `runs/detect/exp` 目錄之下，以下是偵測的結果。



detect.py 指令稿：辨識 YouTube

來源可為 YouTube 的影片（目前只能直接串接部分直播的影片）：

```
# 以 YOLOv5 模型偵測攝影機物件
(yolov5env_39) python detect.py --source https://www.youtube.com/
watch?v=1EiC9bvVGnk
```

解決方式：可將 YouTube 影片先使用 yt-dlp，在終端機下載 YouTube 影片。

Step 1 安裝 yt-dlp。

```
(yolov5env_39) pip install yt-dlp
```

Step 2 使用 yt-dlp 指令下載 YouTube。

```
(yolov5env_39) yt-dlp -o test1.mp4 https://www.youtube.com/
watch?v=te09w4vbQiY
```

Step 3 使用 detect.py 指令稿辨識 .mp4 檔。

```
(yolov5env_39) python detect.py --source test1.mp4
```

detect.py 指令稿：辨識即時攝影機

進行攝影機偵測，第一台攝影機參數為 --source 0，第二台為 --source 1，依此類推：

```
# 以 YOLOv5 模型偵測攝影機物件
(yolov5env_39) python detect.py --source 0
```

YOLOv5 實際上又依據模型複雜度分為 yolov5n、yolov5s（預設）、yolov5m、yolov5l、yolov5x，我們可以使用 --weight 參數來指定要採用的模型，例如：使用 yolov5n，副檔名為 .pt。

```
# 以 yolov5n 的模型偵測攝影機物件
(yolov5env_39) python detect.py --source 0 --weight yolov5n.pt
```

detect.py 指令稿還有支援其他相當多的參數，可以使用 --help 查詢：

```
# 查詢參數用法
(yolov5env_39) python detect.py --help
```

自行撰寫 Python 偵測物件

除了使用既有的指令稿之外，我們也可以自行撰寫 Python 來使用 YOLOv5 模型偵測物件，YOLO 的使用已經內建在 torch 的深度學習框架中。

2-1 匯入 torch 來使用 YOLOv5 模型偵測物件。(2-1.py)

```
import torch

# 從 PyTorch Hub 下載 YOLOv5s 預訓練模型，可選用的模型有 yolov5s, yolov5m,
yolov5x 等
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
# 影像來源，支援檔案、路徑、PIL、OpenCV, NumPy, list 等
img = 'https://ultralytics.com/images/zidane.jpg'
# 進行物件偵測
results = model(img)
# 顯示結果摘要
results.print()
```

輸出結果：

```
image 1/1: 720x1280 2 persons, 2 ties
Speed: 1071.2ms pre-process, 186.9ms inference, 2.0ms NMS per image at
shape (1, 3, 384, 640)
```

偵測出物件之後，可以顯示或儲存結果圖片：

```
# 顯示結果圖片
results.show()
# 儲存結果圖片
results.save()
```

- results.show()：會顯示在 spyder 的 Plots 視窗裡。
- results.save()：會儲存在 runs/detect/exp 目錄之下。

亦可顯示所有偵測到的物件列表，包含位置、信心 (confidence)、類別編號與名稱：

```
# 顯示物件列表
print(results.pandas().xyxy[0])
```

輸出結果：

	xmin	ymin	xmax	ymax	confidence	class	name
0	742.974792	48.395599	1141.844604	720.000000	0.881052	0	person
1	442.007660	437.522461	496.653961	709.973511	0.675214	27	tie
2	123.024261	193.287292	715.662109	719.723877	0.665813	0	person
3	982.803162	308.417358	1027.365845	419.987000	0.260076	27	tie

也可以使用 OpenCV 模組來實現更多的功能。例如：即時影像辨識與存檔。(建議使用 OpenCV)

2-2 使用 YOLOv5 即時偵測。(2-2.py)

```
import torch
import cv2

# 從 PyTorch Hub 加載 YOLOv5 模型
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

# 初始化攝影機
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print('請檢查攝影機，無法開啟')
    exit()

# 設定影像解析度
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while True:
    # 從攝影機讀取影像
    ret, frame = cap.read()
    if not ret:
        print('無法取得影像')
        break
```

```

# OpenCV 的影像格式為 BGR，需要轉換為 RGB
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

# 使用 YOLOv5 模型進行偵測
results = model(rgb_frame)

# 獲取辨識後的影像與標註資訊
output_frame = results.render()[0] # 在影像上繪製標註

# 顯示影像
bgr_frame = cv2.cvtColor(output_frame, cv2.COLOR_RGB2BGR)
cv2.imshow('YOLOv5 Detection', bgr_frame)

# 按下 'q' 鍵退出
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# 釋放攝影機並關閉視窗
cap.release()
cv2.destroyAllWindows()

```

YOLOv5 預設使用的數據集

YOLOv5 預設使用的數據集不是 COCO128，而是 COCO 數據集的完整版（2017 版本），這是一個大型的目標檢測數據集，包含約 118,000 張訓練圖片和 5,000 張驗證圖片，涵蓋 80 個類別。但 YOLOv5 提供了 COCO128 作為一個小型數據集，用於快速測試和調試模型。以下是兩者的主要區別：

1. COCO128

- 內容：COCO 的子集，包含 128 張標註完整的圖片。
- 用途：快速測試和驗證模型的基本功能（例如，檢查模型是否能成功訓練、推論）。
- 特性：體積小，下載和處理速度快。

2. COCO（完整版）

- 內容：包含超過 200,000 張圖片（118,000 張訓練圖片，5,000 張驗證圖片，以及其他未標註的圖片）。
- 用途：用於訓練大規模、通用的目標檢測模型。
- 特性：涵蓋更多場景和物體類型，能產生更準確、更通用的模型。

預設的 `train.py` 和 `detect.py` 支持用戶選擇自己的數據集。在官方來源檔案的配置中，訓練完整模型時會假定使用 `coco` 數據集，但用戶可以使用 `coco128` 或自定義數據集來訓練或測試。如果你計畫用 `YOLOv5` 進行自定義任務，`coco128` 是一個很好的起點，用於確認模型是否正常運行，然後再切換到更大或更具挑戰性的數據集進行完整訓練。

在 `data/coco128.yaml` 的設定檔中，定義了這 80 個生活中常用的物件標籤。

練習 1

1. 參考 `data/coco128.yaml` 的物件標籤後，將程式 `2-2.py` 修改成只框出辨識人 (`person`) 與手機 (`cell phone`)。(解答為 `2-2a.py`)

2-2. 自行訓練 YOLO 模型

自行訓練模型

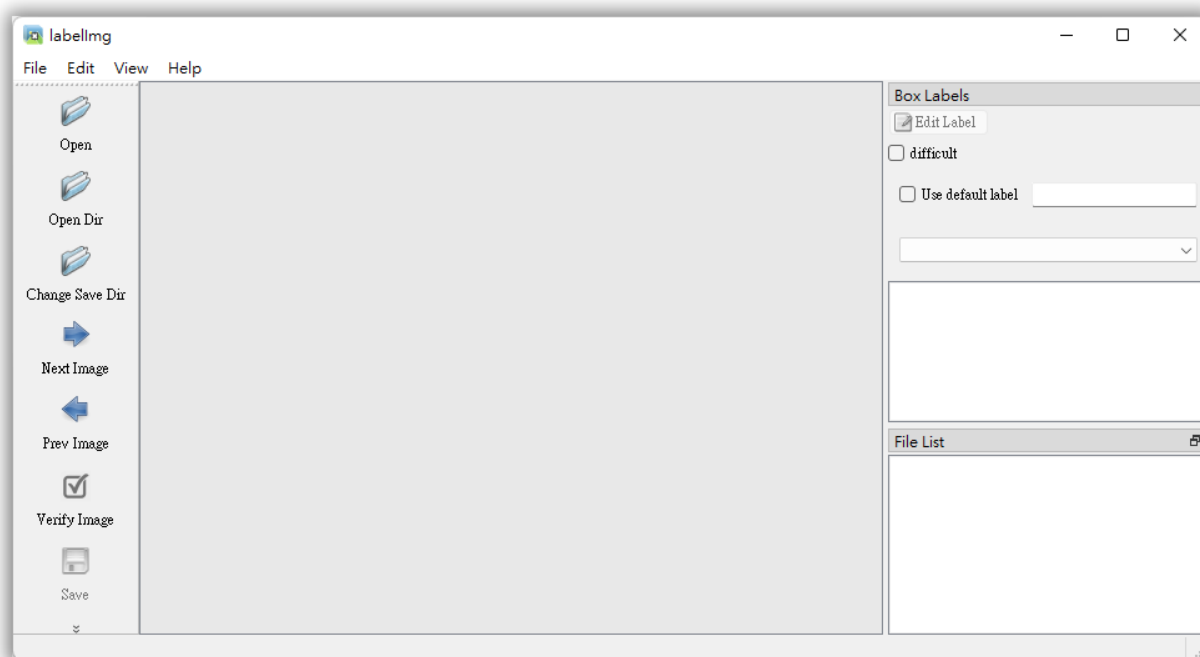
若想要針對自己的資料，訓練出自己的 YOLO 模型，可以先準備好標註的圖片之後，再用預訓練的 YOLO 模型，進行遷移式學習 (transfer learning)。遷移式學習使用已有問題的解決模型，並將其利用在其他不同但相關問題上。例如：用來辨識汽車的知識 (或者是模型) 也可以被用來提升識別卡車的能力。

安裝 labelImg 標註工具

實作影像的物件偵測時，需要大量的已知資料集，也就是照片加上物件的所在位置 (特徵) 以及物件的名稱 (標籤)，初期都會使用人工的方式來手動標註，而 labelImg 就是用來標註照片中物體位置與名稱的小工具。此工具是以 Python 所開發的，可以用 pip 安裝：

```
# 從 PyPI 下載與安裝 labelImg  
(yolov5env_39) pip install labelImg  
# 執行 labelImg，start /min 為背景執行  
(yolov5env_39) start /min labelImg
```

開啟畫面如下：



標註圖片

Step 1 自行拍攝或下載圖片數據集。

你需要自行拍攝或下載圖片數據集 (dataset)。

例如：到 Kaggle (<https://www.kaggle.com/datasets/andrewmvd/road-sign-detection>) 下載訓練照片檔 (檔案名稱為 RoadSignsPascalVOC.zip)。

以下是照片的規格：

- 數據集包括以下的類別：Traffic Light (紅綠燈)、Stop (停車)、Speedlimit (限速)、Crosswalk (人行橫道)。
- 圖片數量：877 個 .png。



Traffic Light



Stop



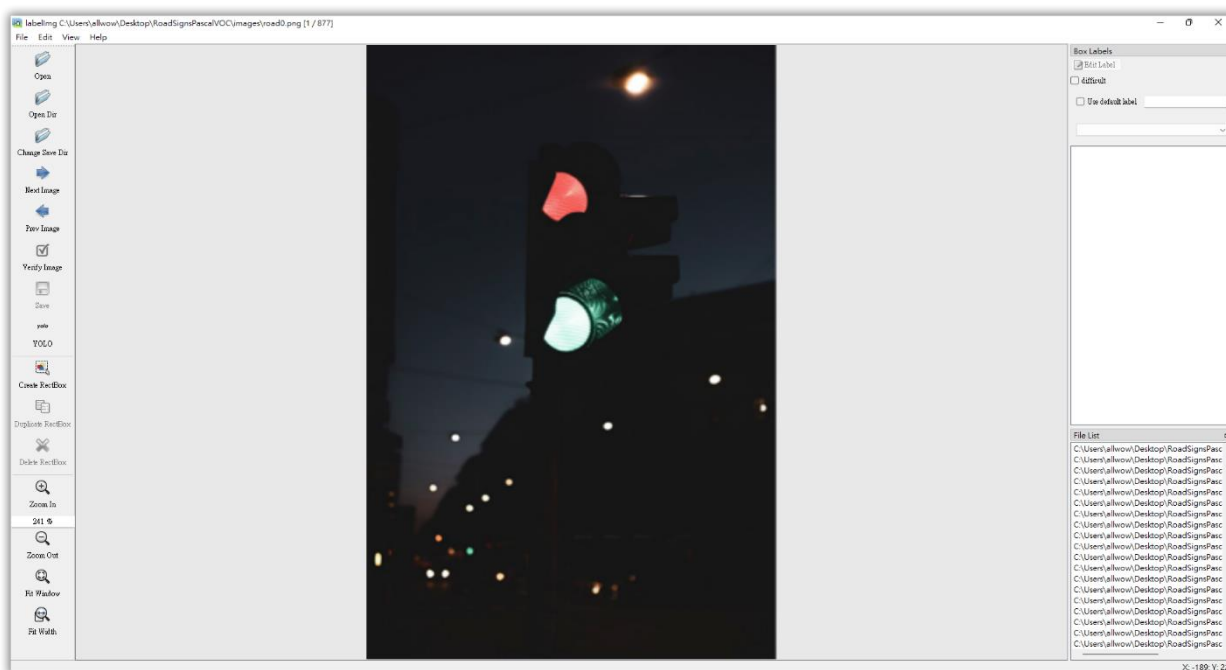
Speedlimit



Crosswalk

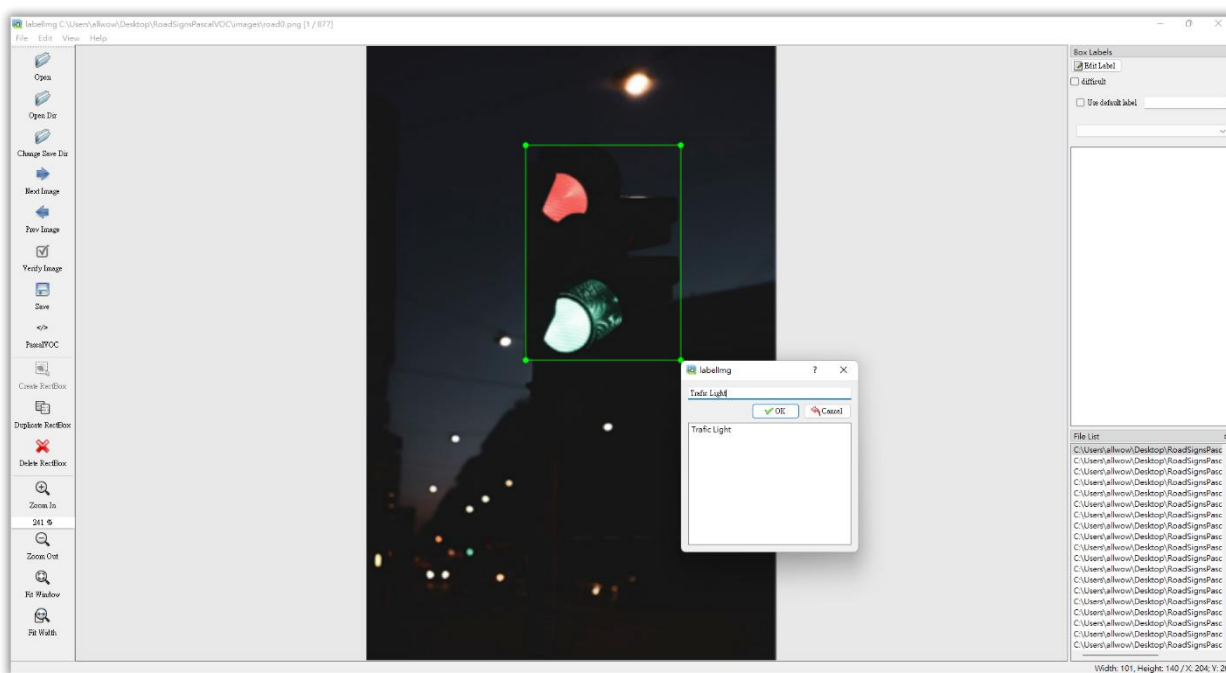
Step 2 開啟圖片

開啟 labelImg 之後，首先開啟想要進行標註的圖片檔，開啟時可以選擇『 Open 』開啟單張圖檔，或是以『 Open Dir 』開啟整個目錄中所有的圖檔。例如：解壓縮 RoadSignsPascalVOC.zip 後，以『 Open Dir 』開啟 images 資料夾：

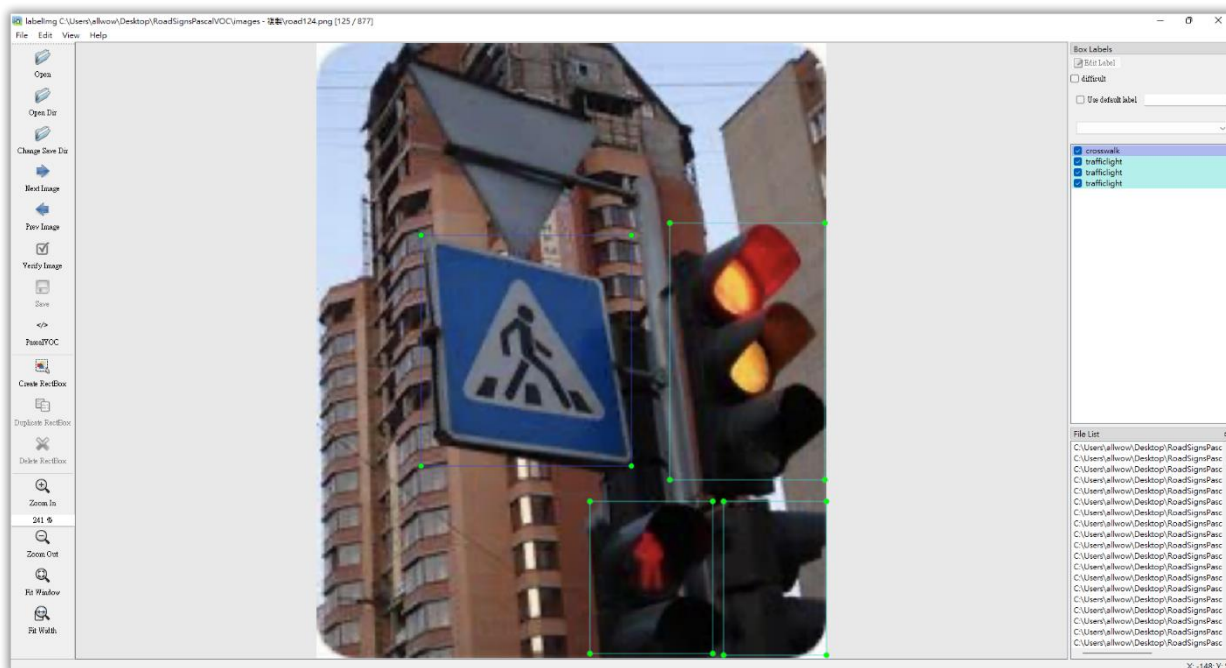


Step 3 標示出物件的位置與標籤。

選擇『 Create RectBox 』，使用滑鼠將物件框起來，並輸入這個物件的標籤。



一張圖有多少想要分類的類別，都要標註。標示完成後，在視窗的右方會列出所有標示好的物件，我們可以對任何的物件進行修改或刪除等動作。



Step 4 轉檔與儲存。

當一張圖片上的物件都標註完成後，記得按下左側的『 Save 』鍵儲存，labelImg 會使用 XML 格式 (PascalVOC) 來儲存標註資訊。



介紹兩種常用的深度學習引擎的圖片標註格式：

- PascalVOC：`.xml` 檔。使用 LabelImg 開啟照片時，會自動檢查是否有對應的標註檔，如果它發現同目錄下有對應的 `.xml` 標註檔，就會自動載入它並顯示出來。例如：

```
<annotation>
  <folder>images</folder>
  <filename>road1.png</filename>
  <size>
    <width>400</width>
    <height>283</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>trafficlight</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <bndbox>
      <xmin>154</xmin>
      <ymin>63</ymin>
      <xmax>258</xmax>
      <ymax>281</ymax>
    </bndbox>
  </object>
</annotation>
```

- YOLO：`.txt` 檔。訓練 YOLO 模型時可以直接使用的文字檔，例如：

```
1 0.515000 0.607774 0.260000 0.770318
```

YOLO 的格式是普通文字檔，每一行代表一個標註物件。

格式為：`class x_center y_center width height`

5 個欄位以空白分隔，意義如下：

1. `class`：類別 ID。
2. `x_center`：中心點 X 座標（標準化）。
3. `y_center`：中心點 Y 座標（標準化）。
4. `width`：方框寬度（標準化）。
5. `height`：方框高度（標準化）。

Step 5

按 YOLO 的目錄管理方式存儲圖片與 TXT 檔。

將所有的圖片標註好之後，參考 [coco128 資料集](#) 的目錄結構，將自己的資料也依照同樣的目錄結構存放，類似像這樣：(可見一個目錄 datasets，存放 road)

```
road
├── images
│   └── train
│       ├── road0.png
│       ├── road1.png
│       └── road2.png
└── labels
    └── train
        ├── road0.txt
        ├── road1.txt
        └── road2.txt
```

YOLOv5 訓練程式在讀取圖檔所對應的標註檔案時，會將圖檔路徑中最後一個 images 替換為 labels，然後在這個路徑中尋找對應的標註檔案。

這裡只是舉例來說明圖片與標註檔案的目錄結構，實際上不會只用三筆資料就進行模型訓練，依照 [YOLOv5 官方網頁](#) 上的建議，每個類別最好收集到 1,500 張影像，並包含 10,000 個標註物件。

資料集 YAML 設定檔

接著為資料集準備一個給訓練程式用的 road.yaml 設定檔，並儲存至 data 目錄，內容如下：

```
# 設定圖檔路徑
path: ./datasets/road # 資料根目錄
train: images/train # 訓練用資料集 (相對於 path)
val: images/train # 驗證用資料集 (相對於 path)
test: # 測試用資料集 (相對於 path, 可省略)

# 物件類別設定
nc: 4 # 類別數量
names: ['trafficlight', 'speedlimit', 'crosswalk', 'stop'] # 類別名稱
```

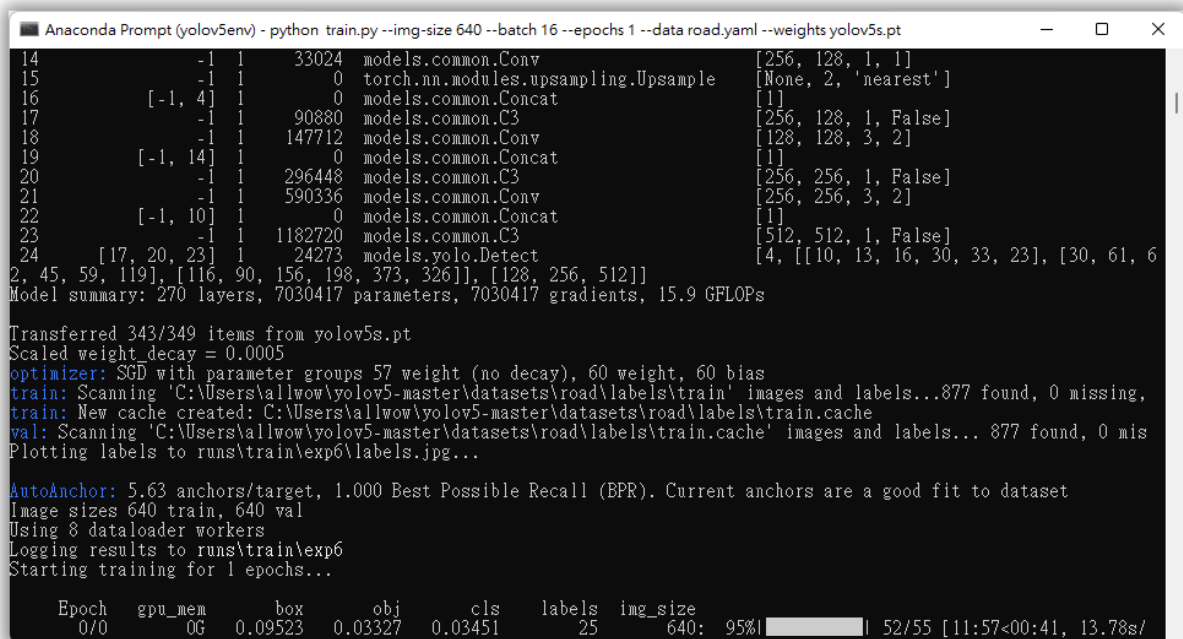
其他資料集的 yam1 設定檔，可以從 YOLOv5 原始碼的 data 目錄中查看。

訓練 YOLOv5 模型

準備好標註資料與設定檔之後，執行 YOLOv5 原始碼中附帶的模型訓練指令稿（執行時間要靠 GPU，請自行上網搜尋相關技術文，或使用 Google Colab 來訓練）：

```
# 進行 YOLOv5s 模型訓練
(yolov5env_39) python train.py --img-size 640 --batch 16 --epochs 500 \
--data road.yam1 --weights yolov5s.pt
```

- `--img-size`：可以指定影像的大小，對於偵測比較小的物件時，可以加大影像的大小，預設的影像大小為 640。
- `--weights`：可用來指定預訓練的模型。



```
Anaconda Prompt (yolov5env) - python train.py --img-size 640 --batch 16 --epochs 1 --data road.yam1 --weights yolov5s.pt
14      -1 1      33024 models.common.Conv [256, 128, 1, 1]
15      -1 1      0      torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
16      [-1, 4] 1      0      models.common.Concat [1]
17      -1 1      90880 models.common.C3 [256, 128, 1, False]
18      -1 1      147712 models.common.Conv [128, 128, 3, 2]
19      [-1, 14] 1      0      models.common.Concat [1]
20      -1 1      296448 models.common.C3 [256, 256, 1, False]
21      -1 1      590336 models.common.Conv [256, 256, 3, 2]
22      [-1, 10] 1      0      models.common.Concat [1]
23      -1 1      1182720 models.common.C3 [512, 512, 1, False]
24      [17, 20, 23] 1      24273 models.yolo.Detect [4, [[10, 13, 16, 30, 33, 23], [30, 61, 6
2, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
Model summary: 270 layers, 7030417 parameters, 7030417 gradients, 15.9 GFLOPs

Transferred 343/349 items from yolov5s.pt
Scaled weight_decay = 0.0005
optimizer: SGD with parameter groups 57 weight (no decay), 60 weight, 60 bias
train: Scanning 'C:\Users\allow\yolov5-master\datasets\road\labels\train' images and labels...877 found, 0 missing,
train: New cache created: C:\Users\allow\yolov5-master\datasets\road\labels\train.cache
val: Scanning 'C:\Users\allow\yolov5-master\datasets\road\labels\train.cache' images and labels... 877 found, 0 mis
Plotting labels to runs\train\exp6\labels.jpg...

AutoAnchor: 5.63 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to runs\train\exp6
Starting training for 1 epochs...

Epoch  0/0  gpu_mem  0G  box  0.09523  obj  0.03327  cls  0.03451  labels  25  img_size  640: 95% | 52/55 [11:57<00:41, 13.78s/
```

將參數改為 `--epochs 1`，只訓練一個回合來暫時完成所有的結果，並且檢查以下的輸出結果：

- 訓練完成之後，新的模型與各種訓練過程的資訊都會放在 `runs/train/exp` 目錄中，其中 `result.png` 可以查看模型的收斂狀況，如果模型收斂的情況不理想，可能就要考慮調整參數並重新訓練。
- 而在 `runs/train/exp/weights` 目錄中存放了最佳的模型參數 `best.pt` 以及訓練到最後的模型參數 `last.pt`，通常我們會使用 `best.pt` 這個最佳的模型參數來做後續的預測。
- 若發生 **OMP: Error #15**，請在 `train.py` 加入以下程式碼：

```
os.environ["KMP_DUPLICATE_LIB_OK"] = "TRUE"
```

使用 Google Colab 來訓練模型

以下使用 Colab 來訓練模型：

- 請先上傳我們整理好的 `yolov5-master.zip` 的壓縮檔至自己的 Google Drive，這樣上傳單一個壓縮檔時間才不會花太多時間。
- 登入你的 [Colab](#)，使用 Colab 進行解壓縮後，再進行模型訓練。

首先，登入 Google Drive，並掛載到 Colab 預設的 `content` 目錄下：

```
# 登入 Google Drive
from google.colab import drive
drive.mount('/content/gdrive') # 此處需要登入 google 帳號
```

進入雲端硬碟目錄，因為 Colab 的作業系統是 Linux，必須使用 Linux 的指令，而 Colab 在 Linux 指令前要加上『%』。(注意：%是 Colab 的語法，未來可能會更動)

```
# 進入掛載的雲端硬碟資料夾
%cd gdrive/MyDrive/
```

解壓縮 `yolov5-master.zip`，而 Colab 在某些 Linux 指令前要加上驚嘆號『!』。

```
!unzip yolov5-master.zip
```

進入 `yolov5-master` 目錄：

```
%cd yolov5-master
```

安裝 `yolov5` 所需模組：

```
pip install -r requirements.txt
```

使用模型訓練指令稿：

```
# 進行 YOLOv5s 模型訓練
!python train.py --img-size 640 --batch 16 --epochs 500
--data road.yaml --weights yolov5s.pt
```

- 以筆者的為例，訓練 500 個 epochs 的時間為 5 個小時。
- 若有錯誤，請善用 ChatGPT 除錯。

使用自己的模型進行預測

使用自己的模型進行預測：detect.py 指令稿

訓練出來的 YOLO 模型可在 Anaconda Prompt 中，直接套用 detect.py 進行預測：

```
# 使用自行訓練的 YOLO 模型進行預測
(yolov5env_39) python detect.py --weight runs/train/exp/weights/best.pt
--source road0.png --conf-thres 0.5 --iou-thres 0.5
```

- `--iou-thres`：IoU (Intersection over Union) 是用來衡量模型預測的邊界框 (bounding box) 與實際標註框 (ground truth box) 之間重疊程度的指標，通常會設為 0.5。但在更高精度要求的任務中，IoU 門檻可能會設置為 0.75 或更高。
- `--conf-thres`：信心分數 (confidence score) 是對某個預測框中有目標物體的置信程度，數值範圍在 0 到 1 之間。若信心門檻值設為 0.5，則只有信心分數 ≥ 0.5 的框才會被保留，常見的設置範圍是 0.25 到 0.5。
- 偵測的結果同樣會放在 `runs/detect/exp` 目錄之下。

以下是偵測攝影機物件：

```
# 以自行訓練的 YOLO 模型偵測攝影機物件
(yolov5env_39) python detect.py --weight runs/train/exp/weights/best.pt
--source 0
```

以下是拿 Youtube 的影片來測試：會失敗，請自行改以手動下載 Youtube 後，再進行辨識。

```
# 以自行訓練的 YOLO 模型偵測 Youtube 的影片
(yolov5env_39) python detect.py --weight runs/train/exp/weights/best.pt
--source https://www.youtube.com/watch?v=7BwW2dH6ZI0
```

使用自己的模型進行預測：自行撰寫 Python

自行訓練的 YOLOv5 模型，也可以用於自行撰寫的 Python 指令稿中，使用方式與前述標準的 YOLOv5 模型類似：

2-3 使用自己的 YOLOv5 模型偵測物件。(2-3.py)

```
import torch

# 載入自行訓練的 YOLOv5 模型
model = torch.hub.load('ultralytics/yolov5', 'custom',
                       path = 'runs/train/exp/weights/best.pt')

# 影像來源
img = 'datasets/road/images/train/road6.png'

model.iou = 0.5 # 設定 IoU 門檻值
model.conf = 0.5 # 設定信心門檻值

# 進行物件偵測
results = model(img)

results.print() # 顯示結果摘要
results.show() # 顯示結果圖片
results.save() # 儲存結果圖片

# 顯示結果摘要
print(results.pandas().xyxy[0]) # 顯示物件列表
```

輸出結果：

```
image 1/1: 400x267 1 trafficlight
Speed: 19.6ms pre-process, 245.2ms inference, 0.0ms NMS per image at
shape (1, 3, 640, 448)
Saved 1 image to runs\detect\exp2
      xmin      ymin      xmax      ymax  confidence  class \
0  94.884598  151.530502  175.620407  327.117279    0.964944    0

      name
0  trafficlight
```

```
import torch
import cv2

model = torch.hub.load('ultralytics/yolov5', 'custom',
                       path = 'runs/train/exp/weights/best.pt')

cap = cv2.VideoCapture('test2.mp4')

while True:
    ret, frame = cap.read()
    if not ret:
        print('無法取得影像')
        break

    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = model(rgb_frame)

    output_frame = results.render()[0] # 在影像上繪製標註

    bgr_frame = cv2.cvtColor(output_frame, cv2.COLOR_RGB2BGR)
    cv2.imshow('YOLOv5 Detection', bgr_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

練習 2

1. 請將 2-3.py 改成辨識即時攝影機。(解答為 2-3a.py)
2. 專案：請收集自己的照片，請開發成一個實際應用的辨識輔助，並說明你的情境。

常見的終端機命令

常用的指令	Windows	Linux
列出目前所有的檔案/目錄	dir	ls
變更目錄到指定位置	cd 目錄路徑	cd 目錄路徑
回到到上一層	cd ..	cd ..
中斷程序	Ctrl + C	Ctrl + C