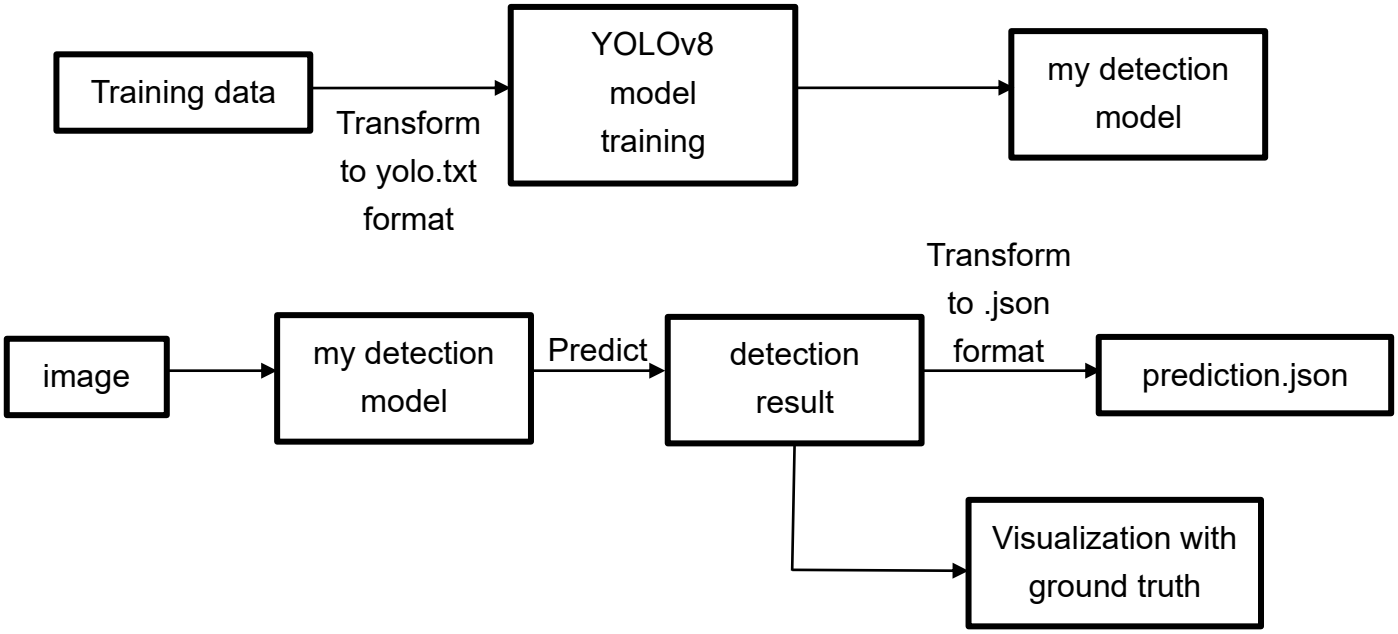
SDC Final Competition : Radar Detection Report

312512005　黃名諄

1. Pipeline of my Detection program:



本次 detection 競賽中，我選擇了 YOLOv8 作為 detection 模型，原因是我第一次接觸深度學習，過去在實際應用上好像比較常聽到 YOLO，工研院的工程師之前來演講也有提到 YOLO，且 YOLOv8 是 2023/01 出的，比起 Faster R-CNN(2015)，是更新的模型，因此我才想藉這個機會嘗試 YOLOv8 看看，這是這次競賽的 detection 影片 https://youtu.be/t1aJATv6Ndw?si=0P6nFDl0vCG-zjTG。

I. Training data 轉成 yolo.txt format:

因為 YOLO v8 訓練時的 annotation 有其自己的.txt 格式，因此需先把 Training data 都轉為 YOLO format，且 bounding box 也只能使用正框，因此需做 data 上的轉換處理才能做後續的 training，詳細會於後面 contribution 處說明。

II. Training a YOLOv8 model:

接著就是將轉好的 Training data 訓練 YOLOv8 model，此處要決定使用的 model 為何，有 5 個不同參數大小的 model 可選，以及一些參數設定，經過此次競賽，我發現較為關鍵的是 image size 及 epoch 兩參數，後續會再做比較說明。

III. Prediction and visualization:

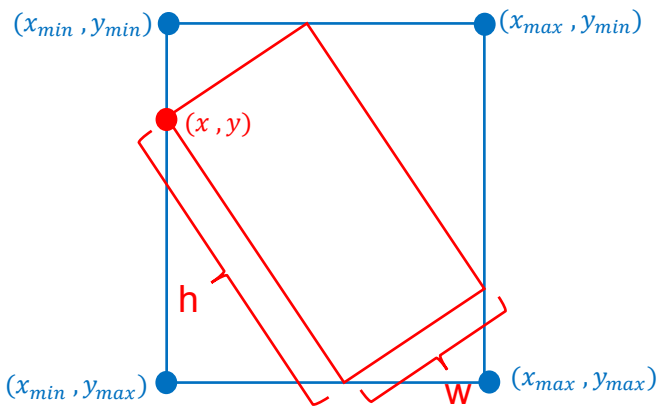
Training 後得到包含 weight 的模型檔案，就能用來對 radar image 做 Prediction，另外為了競賽的格式需求，我還有撰寫將結果用助教

的.json 格式儲存的程式，以利後續能計算 evaluation 的 mAP。最後我還有寫個視覺化程式，將助教給的 test data 中的 ground truth 的 json 檔投影到 radar image 上，並將儲存的 prediction 結果及 ground truth 一起 show 出來，能更清楚以視覺的方式比較我的 model 的 detection 結果好壞。

2. Contribution:

I. Code of Transformation with yolo format:

為了處理檔案格式的問題，我花了很多時間寫這部分轉檔的程式，首先在 training data 的處理上，YOLOv8 只能使用正框，而助教給我們的 training dataset 中是有 rotation 的斜框，一開始我並沒有意識到需處理旋轉的問題，直接使用其 xywh 當正框做計算，後來才發現應該考慮旋轉影響，將原 bounding box 計算出其完全包覆的正框當作我 YOLOv8 使用的 bounding box，如下示意圖所示:



Red bbox : original training data annotation

Blue bbox : for YOLO format

而在實作上，我使用此 function:

```
min_x, min_y, max_x, max_y = gen_boundingbox(bbox['position'], bbox['rotation'])
```

得到新的正框的 bounding box 之 xyxy 值，所有的 bounding box 都轉為這樣的形式去做後續的處理。

得到 bounding box 後，YOLOv8 需要將 label 以其 yolo txt 的格式儲存才能使用，如下 YOLO.txt format 所示:

```
0 0.4956597222222222 0.4357638888888889 0.013888888888888888 0.020833333333333332
```

分別對應到[class , x_center , y_center , w(normalized) , h(normalized)]
其中 x_center , y_center , w , h 都是以圖片長寬歸一化後的結果，

依照我們先前所得 bounding box 之 xyxy 值，可如下列公式簡單計算出 YOLO 格式所需的 x_center , y_center , w(normalized) , h(normalized) :

$$x_{center} = \frac{(x_{min} + x_{max})}{2}\frac{1}{w}$$

$$y_{center} = \frac{(y_{min} + y_{max})}{2}\frac{1}{h}$$

$$w_{normalized} = \frac{(x_{max} - x_{min})}{w}$$

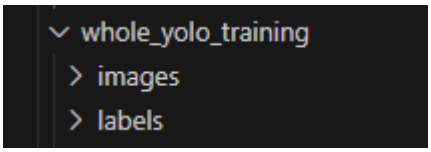
$$h_{normalized} = \frac{(y_{max} - y_{min})}{h}$$

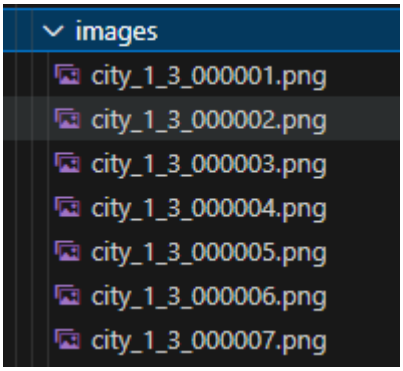
其中 w 是原圖片寬，h 是原圖片高，在這個 mini_train dataset 中圖片都是 1152*1152。
而 class 部分因為我們比賽將所有載具當作 car 一個類別，因此將原 trainin data 中類別不是'pedestrian'或'group_of_pedestrians'的都當作 car 而寫入 0

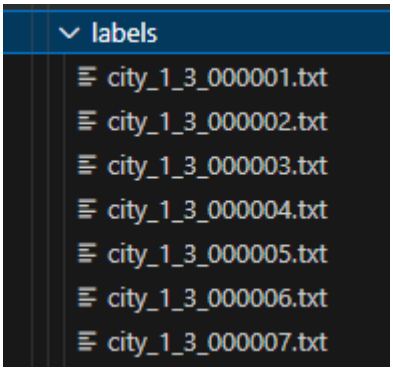
```
if (object["class_name"] != 'pedestrian' and object["class_name"] != 'group_of_pedestrians'):
```

```
x_c_yolo = ((min_x + max_x)/2)/image_width
y_c_yolo = ((min_y + max_y)/2)/image_height
width_yolo = (max_x - min_x)/image_width
height_yolo = (max_y - min_y)/image_height
yolo_txt = f"{0} {x_c_yolo} {y_c_yolo} {width_yolo} {height_yolo}"
```

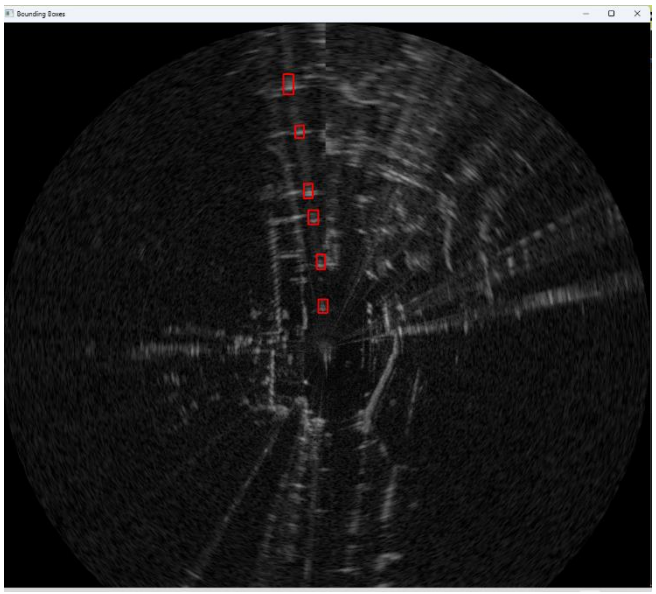
最後 YOLOv8 需要將圖片檔案及 annotation 別存在 images 和 labels 資料夾中，並以對應的檔名儲存，如下所示:







至此我完成了將所有 data 都轉為此 YOLOv8 需要的 YOLO.txt format 而可以送入做 YOLOv8 模型的訓練，而我還有另外寫一個 project_bboxes_radar.py 的程式，能將轉好的 labels 投影到 radar image 上去稍微進一步確認自己的轉換是正確的。

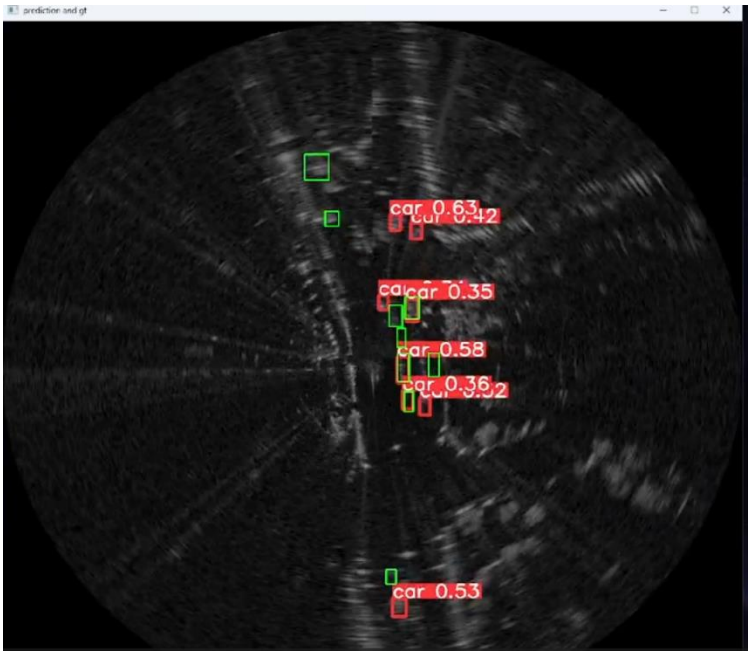


接著是結果的部分，我寫了一個 detection.py 能用訓練好 YOLOv8 模型做 predict，但 YOLOv8 沒有直接輸出 bounding box 檔案為助教的.json 格式，所以我花了一點工夫將 predict 結果的 bounding box 存成助教需要的.json 格式以利後續使用，對每張圖片 YOLOv8 會 predict 後 return 出此圖所有 bounding box 的物件，我再將其儲存進.json 格式中，程式如下:

```python
output_data = []
for image in images_path_list:
    #predict the image by yolo
    results = model.predict(image, save=True)
    image_name =  os.path.splitext(os.path.split(image)[1])[0]
    #Note: only one result object in result list
    for result in results:
        boxes = result.boxes.cpu().numpy()
        boxes_xyxy = boxes.xyxy.tolist()
        boxes_conf = boxes.conf.tolist()
        for box, score in zip(boxes_xyxy,boxes_conf):
            min_x = box[0]
            min_y = box[1]
            max_x = box[2]
            max_y = box[3]
            point1 = [min_x,max_y]
            point2 = [min_x,min_y]
            point3 = [max_x,min_y]
            point4 = [max_x,max_y]
            points = [point1,point2,point3,point4]
            objects = {
                        "sample_token": image_name,
                        "points": points,
                        "name": "car",
                        "score": float(score)
                                }
            output_data.append(objects)
```

II. Visualization with ground truth:

雖然我用 YOLOv8 發現不太能使用原先助教範例程式中的影像視覺化，而畢竟人是視覺化的動物，所以我認為還是需要一個視覺上的表現去評斷自己模型的 prediction 情況，比起直接看 mAP 會更有感覺，而助教給的 mini_test data 中也有 ground truth，因此我自己寫了一個 visualization.py 的 code 來將 ground truth 和 prediction 能同時顯示在對應的 radar image 上，好比較 detection 和 gt 的差異多大。



圖中綠色是 ground truth，紅色是 model 的 detection，而這部份的

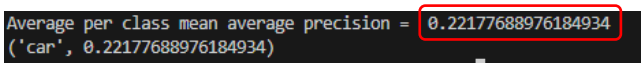
程式首先是在先前的 detection.py 中做 prediction 時，將 YOLOv8 的參數 save 設為 true，就能將帶有 prediction 投影框的圖片存起來，接著將 gt.json 中對於相應每張 radar image 的 bounding box 找出來並繪製於對應的圖上，就能做到同時顯示出 prediction 和 ground truth 的功能，這在原來的範例程式中應該也沒有，我認為此舉能更直接的讓人清楚的明白 detection 的好壞，及搭配 mAP 去解讀會有更好的結果展現能力。https://youtu.be/gMMutw0m8M8?si=Wff9QQNC3B_wzZog

3. Problems and Solutions:

I. 模型泛化能力不佳:

在 evaluation 時發現到，competition 使用的圖片場景是車子靜止且車少的情境，以我表現最好的 model 來說，mAP 有 0.8956 左右，但

後來在用 mini_test 的 data 下去做 detection 及 evaluation，發現 Map 都非常低，且用我的 visualization 程式也看的出來 detection 沒有很準確以 city_7_0 為例，mAP 只有 0.2218 左右，相比競賽使用的 image，



mAP 差了非常多，我因此認為我的模型並不適用於各種場景，換句話說就是泛化能力不佳，只要換到較複雜車多的場景 detection 就會較差，而在其他人的報告中也有聽到這個問題，我的想法是未來可以使用較大的模型、更多的訓練資料以及更加多元而有不同場景的 data 去 training，或許能提升 detection 模型的泛化能力。

II. YOLOv8 不同模型及參數的影響:

在訓練 YOLOv8 模型時，在處理完格式問題後，首先碰到的問題就是模型的選擇，YOLOv8 由參數小到大共有 5 種模型可用:

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

越大的模型訓練時間較長但也可能更準確；另外在參數上，imgsz(image size)是其把原始圖片壓縮至多大去訓練，設越大當然越吃顯卡算力及時間，以及 epochs 是訓練幾輪，而 YOLOv8 在訓練收斂時會自動停止訓練，因此我都想說設 epochs=1000 大一點來讓訓練自己收斂，但後來發現這樣是不太好的。一開始因為我本身的顯卡較差，我選擇先用 YOLOv8s 訓練並考慮到訓練時間將 imgsz 設 480，後來又

試了 YOLOv8n 並嘗試調高 imgsz 至預設的 640，發現用小模型但 imgsz 大結果更好，於是我推論不壓縮 image 應該會有更好的表現，但因為我都讓他跑到收斂需要很長的訓練時間，且顯卡的 ram 也難以負荷原始的 size 1152，因此我試著用更大的 YOLOv8m 搭配 imgsz=640 去 train，原本預期會有更好的表現，但結果反而是最差的，而後我思考了一下原因，發現到 YOLOv8n 都在 epoch 跑到 500 左右就收斂停止了，但 YOLOv8m 跑到 1000 都還沒收斂，因此我認為會不會是跑太多 epoch 導致發生 overfitting 的問題，使表現變差，我後來有拿 YOLOv8m 跑 550 左右 epoch 的 model 去試發現表現比跑 1000 epoch 的 model 表現更好，雖然還是比不上 YOLOv8n，但間接證實了確實跑太多而 overfitting 的問題，在後來報告時看到其他使用 YOLOv8 的同學們，epoch 都只有 50~100 左右，也表明了我確實有 epoch 設太大的問題。

結論第一個就是 imgsz 設的越接近原始尺寸而不壓縮圖片會有更好的表現，第二個是 epoch 不能跑太多輪，會有 overfitting 而降低表現的問題，若有時間及機會我認為可以試看看用 YOLOv8x 大模型加上原始 imgsz=1152 去訓練，並搭配適當的 epoch 輪數去避免 overfitting 問題，相信會有更不錯的 detection 表現

model	Image size	mAP of competition	
YOLOv8n	640	0.8956	→ My best
YOLOv8s	480	0.8911	
YOLOv8m	640	0.8524	→ overfitting?

4. Other Discussion or Findings:

此次競賽很可惜最後沒有得到不錯的名次，雖然我很早就開始做也曾經在第二名好一陣子，但自己做的方向有點錯誤，最後來不及做修正比較可惜，但從中也學到很多做 detection 的經驗，第一名的同學也使用 YOLOv8，從他們身上我也知道了自己哪裡沒做好，學習到很多。另外 radar 的特徵真的不像一般影像特徵明顯，在 detection 上有一定難度，畢竟肉眼都難以分辨，但至少簡單的情境深度學習是能有一定處理能力的，我認為未來的技術發展肯定能處理更多的情況，而另外老師也有提到，bbox

的旋轉框問題，YOLOv8 沒有能力預測斜框是比較可惜的，喪失了更精準的 bbox，我認為這也是未來 YOLO 可以加入的功能之一。我另外也有做 training data 少的 bonus，發現訓練出來的模型辨識能力真的較差，確實在資料量少及情境複雜的狀況下來有很多挑戰，最後感謝各位同學及助教，讓我在最後的競賽上學到非常多寶貴的經驗。