SDC Final Competition : Radar Detection Report

312512005 黃名諄

1. Pipeline of my Detection program:

Training data

YOLOv8

model

training

Transform to yolo.txt format

my detection model

Transform to .json

format

my detection model

image

detection result

prediction.json

Predict

Visualization with

ground truth

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>。

1. 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。

1. 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。

1. 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。

1. Contribution:
   1. 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:

w

h

Red bbox : original training data annotation

Blue bbox : for YOLO format

而在實作上，我使用此function:

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:

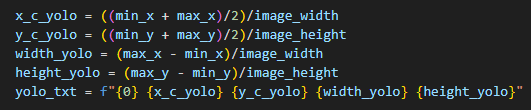
分別對應到[class , x\_center , y\_center , w(normalized) , h(normalized)]

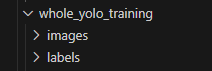
其中x\_center , y\_center , w , h 都是以圖片長寬歸一化後的結果，

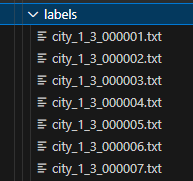
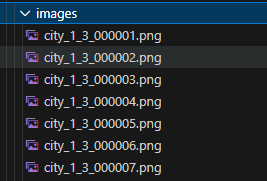
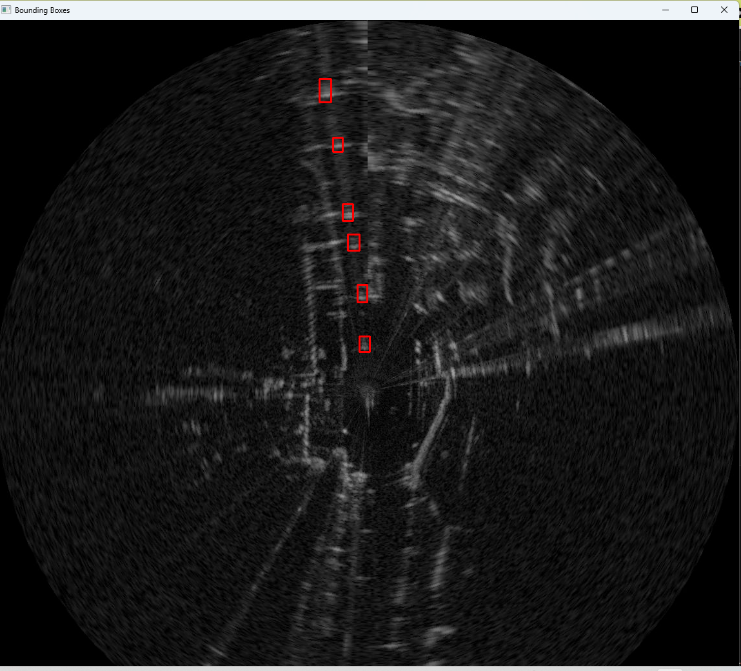
依照我們先前所得bounding box之xyxy值，可如下列公式簡單計算出YOLO格式所需的x\_center , y\_center , w(normalized) , h(normalized) :

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

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:

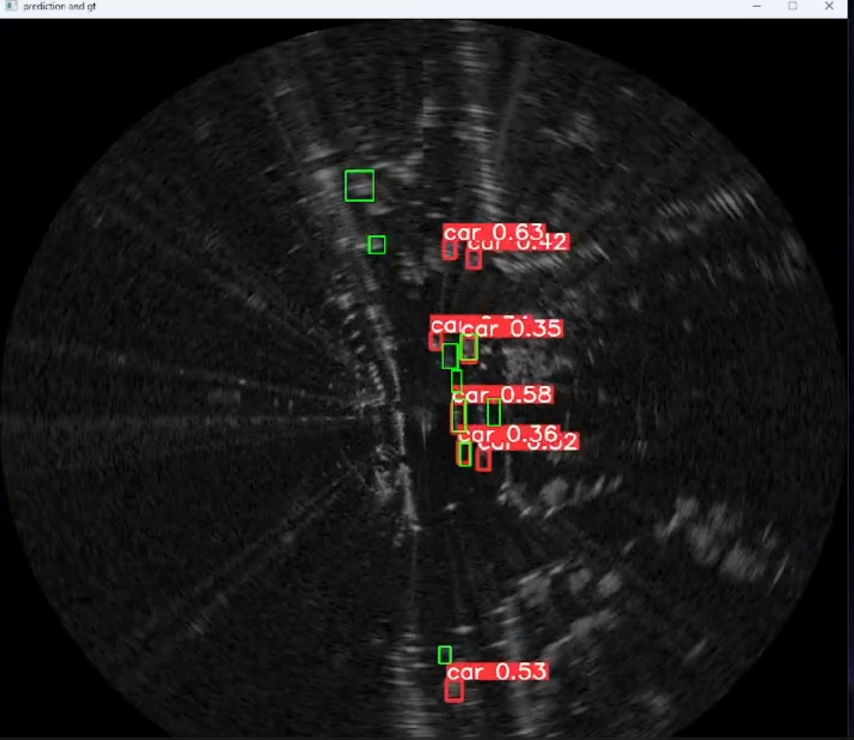
至此我完成了將所有data都轉為此YOLOv8需要的YOLO.txt format而可以送入做YOLOv8模型的訓練，而我還有另外寫一個project\_bboxes\_radar.py的程式，能將轉好的labels投影到radar image上去稍微進一步確認自己的轉換是正確的。

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:



* 1. 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。

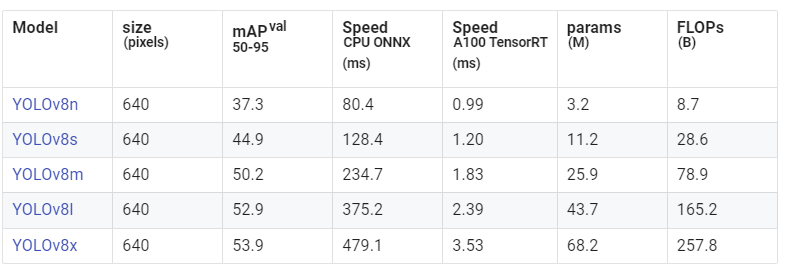
圖中綠色是ground truth，紅色是model的detection，而這部份的程式首先是在先前的detection.py中做prediction時，將YOLOv8的參數save設為true，就能將帶有prediction投影框的圖片存起來，接著將gt.json中對於相應每張radar image的bounding box找出來並繪製於對應的圖上，就能做到同時顯示出prediction和ground truth的功能，這在原來的範例程式中應該也沒有，我認為此舉能更直接的讓人清楚的明白detection的好壞，及搭配mAP去解讀會有更好的結果展現能力。<https://youtu.be/gMMutw0m8M8?si=Wff9QQNC3B_wzZog>

1. Problems and Solutions:
   1. 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。



* 1. 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:

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 |
| YOLOv8s | 480 | 0.8911 |
| YOLOv8m | 640 | 0.8524 |

My best

overfitting?

1. 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。