

國立清華大學  
計算機視覺  
Computer Vision



國立清華大學  
NATIONAL TSING HUA UNIVERSITY

Homework 4

系所級:電子所二年級

學號:111063548

姓名:蕭方凱

指導老師:孫民教授

## 目錄

### Problems .....3

1. Find the split train val path function in data/preprocess.py and write the code to generate the training YAML file using the Q1 dataset.....3
  - (1) WAY1(random indices to control paths) .....3
  - (2) WAY2(sklearn.model\_selection).....5
2. finish the select images function in data/preprocess.py to generate the training YAML file. ....6
  - (1) WAY1(random shuffle) .....6
  - (2) WAY2(equally distributed path).....7
  - (3) WAY3(Distribute in proportion).....8
3. In the final question, besides utilizing the dataset selected in Q2, you can incorporate another 1,200 unlabeled images. In the following rules, you can try any method to improve performance..... 10

### Discussion.....14

1. Precision and Recall..... 14
2. Q1..... 16
3. Q2..... 17
4. Q3..... 18
  - (1) Freeze backbone..... 18
  - (2) Positive weights ..... 18
  - (3) Focal loss ..... 19

## Problems

1. Find the split train val path function in data/preprocess.py and write the code to generate the training YAML file using the Q1 dataset.

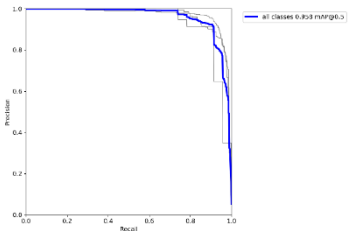
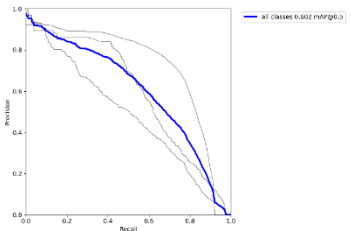
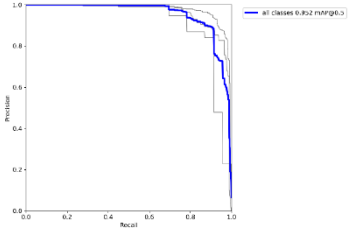
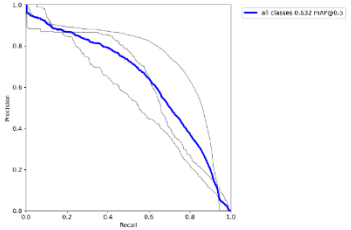
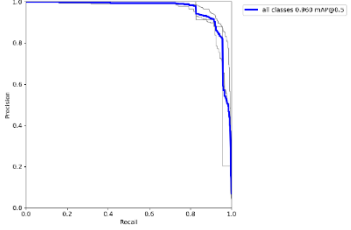
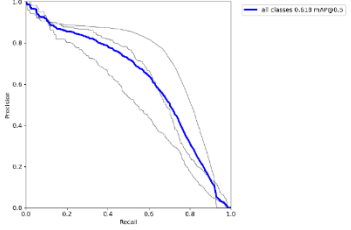
在 split\_train\_val\_path function 中，分別使用兩種方法去將資料分成 training data 及 validation data，下面分別介紹兩種方法及其對應結果。

- (1) WAY1(random indices to control paths)

```
150 import numpy as np
151 from sklearn.metrics import confusion_matrix
152 import matplotlib.pyplot as plt
153
154 rng = np.random.default_rng()
155
156 num_samples = len(all_image_paths)
157 num_train = int(train_val_ratio * num_samples)
158
159 indices = np.arange(num_samples)
160 rng.shuffle(indices)
161
162 train_indices = indices[:num_train]
163 val_indices = indices[num_train:]
164
165 train_image_paths = [all_image_paths[i] for i in train_indices]
166 val_image_paths = [all_image_paths[i] for i in val_indices]
```

先將 all\_image\_path 的數量透過指令 len 求出，即 num\_samples，接著將 num\_samples 乘上比重 train\_val\_ratio，得到 train\_image\_paths 的數量，本次作業 num\_samples 固定為 200，假設乘上 train\_val\_ratio=0.9，最終則會將 180 個 paths 丟入 training\_image\_paths，其餘 20 筆 paths 則丟入 validation\_image\_paths。

Indices 部分是將 all\_image\_paths 順序打散用，先使用 np.arange 產生一 0-199 的一維陣列([0 1 2 ... 198 199])，再用 rng.shuffle 將順序打亂，並按照前面算好的比重數量分配給 train\_indices 和 val\_indices，最後將打亂後的順序分別對應到 all\_image\_paths，產生隨機順序的 train\_image\_paths 及 val\_image\_paths。

	Train result	Test result
<b>Ratio = 0.7</b>		
<b>Ratio = 0.8</b>		
<b>Ratio = 0.9</b>		

上表為三個 ratio 下的 PR Curve result，左邊 column 為 train result，右邊 column 為 test result，在 ratio=0.8 時，test result 有最好的表現，mAP@0.5 為 0.632。

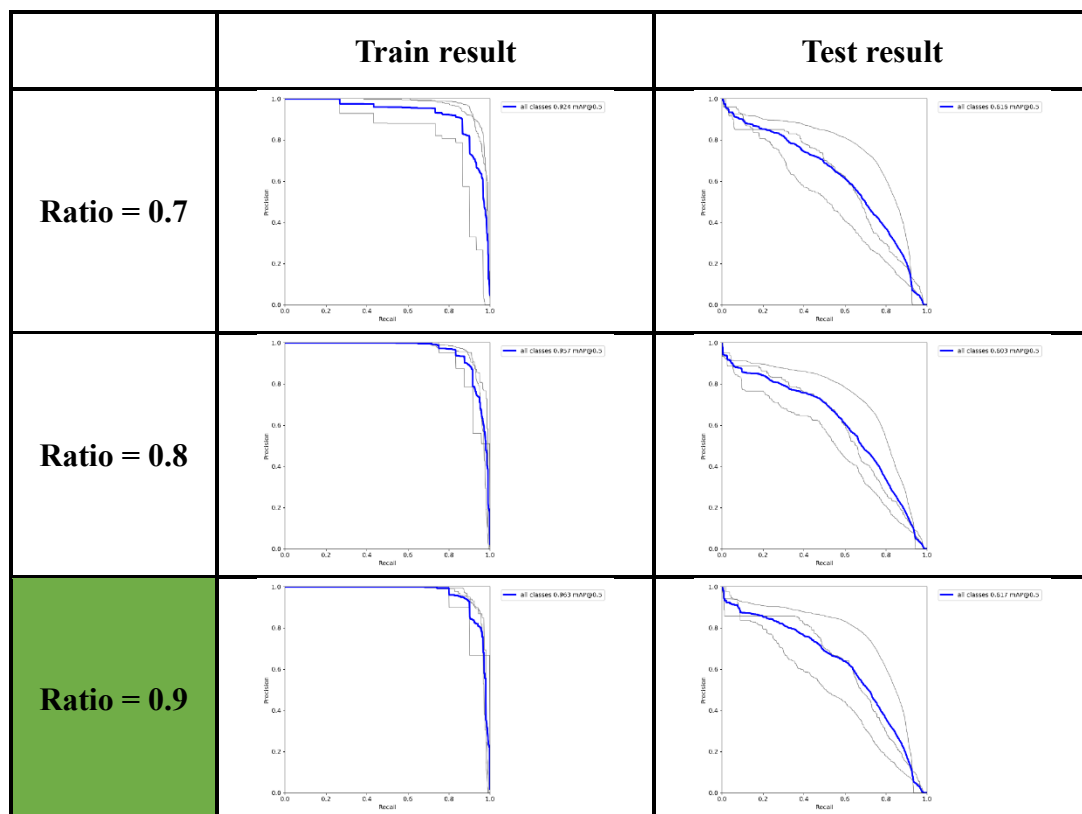
## (2) WAY2(sklearn.model\_selection)

```

169 from sklearn.model_selection import train_test_split
170 train_image_paths, val_image_paths = train_test_split(all_image_paths,
171                                                     random_state=777,
172                                                     train_size=train_val_ratio,
173                                                     shuffle=True)
174
175 return train_image_paths, val_image_paths

```

引用 sklearn.metrics 的 train\_test\_split 功能，輸入 all\_image\_paths，random\_state=777 是為了始終產生相同的隨機分割，train\_size 則直接定義 ratio，會將 all\_image\_path 內的 path 數量乘 ratio 並將此數量的 path 指定給 train\_image\_paths，其餘指定給 val\_image\_paths，suffle=True 則與 WAY1 一樣達到打亂 path 順序的效果。



上表為三個 ratio 下的 PR Curve result，左邊 column 為 train result，右邊 column 為 test result，在 ratio=0.9 時，test result 有最好的表現，mAP@0.5 為 0.617。

總結 WAY1 及 WAY2，使用 WAY1 去 split train validation path，ratio 設 0.8，會有最好的 test result。

## 2. finish the select images function in data/preprocess.py to generate the training YAML file.

在 select images function 中，使用三種方式去對 Q2 datasets 進行挑選，Q2 datasets 共有 6 個資料夾(170, 173, 398, 410, 595, 511)，每個資料夾有 200 張照片，共 1200 張。如何從 1200 張照片中選擇 200 張，以下介紹三種方法及其分別對應的結果。

### (1) WAY1(random shuffle)

```
28 random.shuffle(image_paths)
29 selected_image_paths = image_paths[:images_num]
```

輸入 image\_paths，這裡為 Q2 六個資料夾所有的 paths，並直接打亂 image\_paths 的順序，打亂之後取前 200 個 paths(images\_num=200)，並將每個資料夾提供的 paths 數量 print 出來，如下：

```
Namespace(data_folder='../CityCam', ques='Q2')
NOW IS IN Q2
在 170 資料夾的路徑數量: 27
在 173 資料夾的路徑數量: 40
在 398 資料夾的路徑數量: 36
在 410 資料夾的路徑數量: 28
在 495 資料夾的路徑數量: 34
在 511 資料夾的路徑數量: 35
Number of paths written to train.txt: 160
Number of paths written to val.txt: 40
Number of paths written to test.txt: 1200
```

每個資料夾提供的 paths 數量不相等，為隨機取出，共 200 個 selected\_image\_paths，而 train.txt 數量為 160，val.txt 數量為 40 是因為在 split\_train\_val\_path function 中 train\_val\_ratio 為 0.8。

## (2) WAY2(equally distributed path)

```
32     selected_image_paths = []
33     folder_paths = {}
34     for path in image_paths:
35         folder = os.path.dirname(path)
36         if folder not in folder_paths:
37             folder_paths[folder] = []
38         folder_paths[folder].append(path)
39     print(folder_paths)
40     for key, values in folder_paths.items():
41         values_list = [values] if isinstance(values, str) else values
42         num_to_select = min(33, len(values))
43         selected_values = random.sample(values_list, num_to_select)
44         selected_image_paths.extend(selected_values)
45
46     selected_keys = random.sample(folder_paths.keys(), 2)
47     for key in selected_keys:
48         values = folder_paths[key]
49         selected_value_last = random.choice(values)
50         # print(selected_value_last)
51         selected_image_paths.append(selected_value_last)
```

先將 image\_paths 全部輸入到 folder\_paths，folder\_path 是一個 dict 的形式，有 6 個 key 分別為 Q2 datasets 六個資料夾路徑，每個 key 對應到的 200 個 value 為每個資料夾內中的 image\_paths。

遍歷 folder paths 中的 key 和 value，將每個 key 隨機挑選 33 個 value 出來，意即將六個資料夾中，每個資料夾隨機挑選 33 個 image\_paths 出來，並加入進 selected\_image\_paths，最後第 46 行代表從六個資料夾中隨機選擇兩個並個抓一個 image\_path 加入至 selected\_image\_paths。

總結來說此方法做到在 6 個資料夾中盡量取出平均數量，且是隨機的，不會特別偏袒某個資料夾。Print 出的結果如下：

```
在 170 資料夾的路徑數量: 33
在 173 資料夾的路徑數量: 34
在 398 資料夾的路徑數量: 33
在 410 資料夾的路徑數量: 34
在 495 資料夾的路徑數量: 33
在 511 資料夾的路徑數量: 33
Number of paths written to train.txt: 160
Number of paths written to val.txt: 40
Number of paths written to test.txt: 1200
```

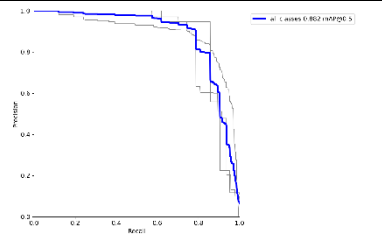
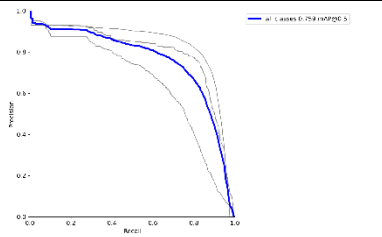
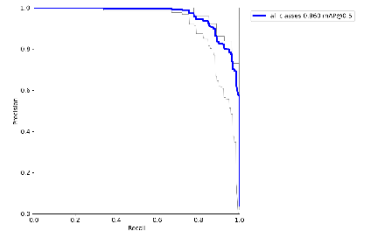
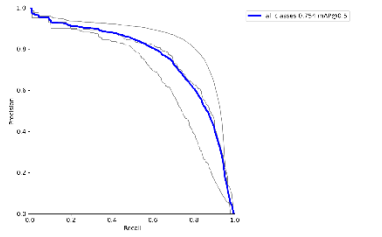
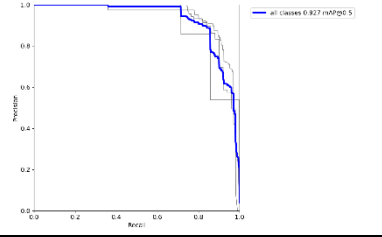
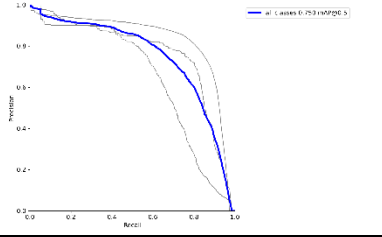
### (3) WAY3(Distribute in proportion)

```
54 selected_image_paths = []
55 selected_paths_170 = [path for path in image_paths if '/CityCam/Q2/170' in path]
56 random.shuffle(selected_paths_170)
57 selected_image_paths_temp = selected_paths_170[:24]
58 selected_image_paths = selected_image_paths + selected_image_paths_temp
59
60 selected_paths_173 = [path for path in image_paths if '/CityCam/Q2/173' in path]
61 random.shuffle(selected_paths_173)
62 selected_image_paths_temp = selected_paths_173[:35]
63 selected_image_paths = selected_image_paths + selected_image_paths_temp
64
65 selected_paths_398 = [path for path in image_paths if '/CityCam/Q2/398' in path]
66 random.shuffle(selected_paths_398)
67 selected_image_paths_temp = selected_paths_398[:39]
68 selected_image_paths = selected_image_paths + selected_image_paths_temp
69
70 selected_paths_410 = [path for path in image_paths if '/CityCam/Q2/410' in path]
71 random.shuffle(selected_paths_410)
72 selected_image_paths_temp = selected_paths_410[:31]
73 selected_image_paths = selected_image_paths + selected_image_paths_temp
74
75 selected_paths_495 = [path for path in image_paths if '/CityCam/Q2/495' in path]
76 random.shuffle(selected_paths_495)
77 selected_image_paths_temp = selected_paths_495[:44]
78 selected_image_paths = selected_image_paths + selected_image_paths_temp
79
80 selected_paths_511 = [path for path in image_paths if '/CityCam/Q2/511' in path]
81 random.shuffle(selected_paths_511)
82 selected_image_paths_temp = selected_paths_511[:27]
83 selected_image_paths = selected_image_paths + selected_image_paths_temp
```

此方法先對 Q2 中六個資料夾進行訓練及測試，並對其結果表現進行排序，實測結果表現由好到壞為:495→398→173→410→511→170，將這六個資料夾按照比例提供不等的 path 數量，我採用比例 0.9 去按照比例分配 path 數量，假設 495 提供 x 個 path，計算公式為: $x + 0.9x + 0.81x + 0.73x + 0.63x + 0.55x = 200$ ，x 算出來約為 44，所以從資料夾 495→398→173→410→511→170，分別提供 44、39、35、31、27、24 筆 image\_paths，print 出的結果如下:

```
在 170 資料夾的路徑數量: 24
在 173 資料夾的路徑數量: 35
在 398 資料夾的路徑數量: 39
在 410 資料夾的路徑數量: 31
在 495 資料夾的路徑數量: 44
在 511 資料夾的路徑數量: 27
Number of paths written to train.txt: 160
Number of paths written to val.txt: 40
Number of paths written to test.txt: 1200
```



Method	Train result	Test result
WAY1(random select)		
WAY2(equally distributed)		
WAY3(distributed in proportion)		

在 WAY1, WAY2, WAY3 三種方法中，我實際各測試了三次，上圖是各取最佳的表現，test result 的 mAP @0.5 由 way1, way2, way3 依序為 0.759、0.754、0.75，way1(random select)表現最好。

造成此結果的原因，我推測是因 random select 方法透過隨機選取 200 張，造成其 test result 最不穩定，變動幅度很大，偶爾會出現比其他方法還要好的結果，但也會出現低於平均值的表現。way2 採平均選取，每個資料夾選取 33 或 34 張，此方法的表現最穩定，估計是因為每個資料夾貢獻的量都差不多所導致。Way3 採比例分配，先獨立選取每個資料夾的 200 張照片作為 selected images 去觀察每個資料夾的表現，讓表現最好的資料夾貢獻最多照片，表現最差的資料夾貢獻最少照片，並按照比例去分配每個資料夾的數量，六個資料夾加起來共 200 張照片，此方法下 test result 的變動幅度介於中間。

雖然 way1 使用 random 方式擁有最高的 mAP @0.5，但在選擇輸入資料時穩定度也是很重要的一部份，故本次作業採用 way2 去作為 Q2 及 Q3 的照片選擇。

**3. In the final question, besides utilizing the dataset selected in Q2, you can incorporate another 1,200 unlabeled images. In the following rules, you can try any method to improve performance.**

在 Q3 中，除了 Q2 的 selected\_images，還加入了 Q3 的照片，與 Q2 依樣有六個資料夾共 1200 張照片，但差別是這些照片皆為 unlabeled，在選擇 Q3 照片時方法與選擇 Q2 時一樣，採用平均數量方法去選擇照片。

除 Q2 selected images 之外，還加入 Q3 的 200 張照片後，train.txt 及 val.txt 如下圖 Fig1、Fig2:

```
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/410/410-20160429-07-000218.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/170/170-20160429-18-000021.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/410/410-20160430-15-000133.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/173/173-20160503-07-000244.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/495/495-20160704-12-000108.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/170/170-20160508-15-000182.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/410/410-20160502-18-000072 (1).jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/495/495-20160704-18-000189.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/511/511-20160429-07-000269.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/495/495-20160704-07-000178.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/495/495-20160505-18-000131.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/410/410-20160704-12-000236.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/398/398-20160429-07-000028.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/511/511-20160508-18-000036.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/495/495-20160429-12-000025.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/173/173-20160704-15-000294.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/511/511-20160704-15-000153.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/170/170-20160430-16-000176.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/410/410-20160429-18-000232.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/173/173-20160508-15-000131.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/170/170-20160430-16-000155.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/495/495-20160508-15-000021.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/410/410-20160429-07-000281.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/398/398-20160508-18-000226.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/511/511-20160502-18-000227.jpg
```

Fig 1 train.txt

```
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/410/410-20160704-18-000149.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/511/511-20160502-18-000045.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/410/410-20160505-18-000149.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/511/511-20160504-18-000034.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/410/410-20160704-12-000285.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/398/398-20160503-15-000139.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/170/170-20160430-18-000187.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/410/410-20160704-15-000217.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/170/170-20160508-15-000158.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/410/410-20160503-07-000176.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/173/173-20160704-07-000224.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/495/495-20160704-12-000080.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/511/511-20160505-18-000232.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/511/511-20160429-18-000219.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/495/495-20160503-18-000286.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/398/398-20160429-07-000065.jpg

/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/173/173-20160429-18-000028.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q3/170/170-20160503-18-000029.jpg
/content/drive/MyDrive/Colab Notebooks/CV_HW4/CityCam/Q2/511/511-20160502-18-000173.jpg
```

Fig 2 val.txt

首先，執行沒有任何 improvement method 的 code，純粹輸入 400 張照片，其中來自於 Q3 的 200 張照片沒有 label。

```
# training command
!python train.py --project runs/train/Q3/report/original --name refine
```

Test result 如下圖：

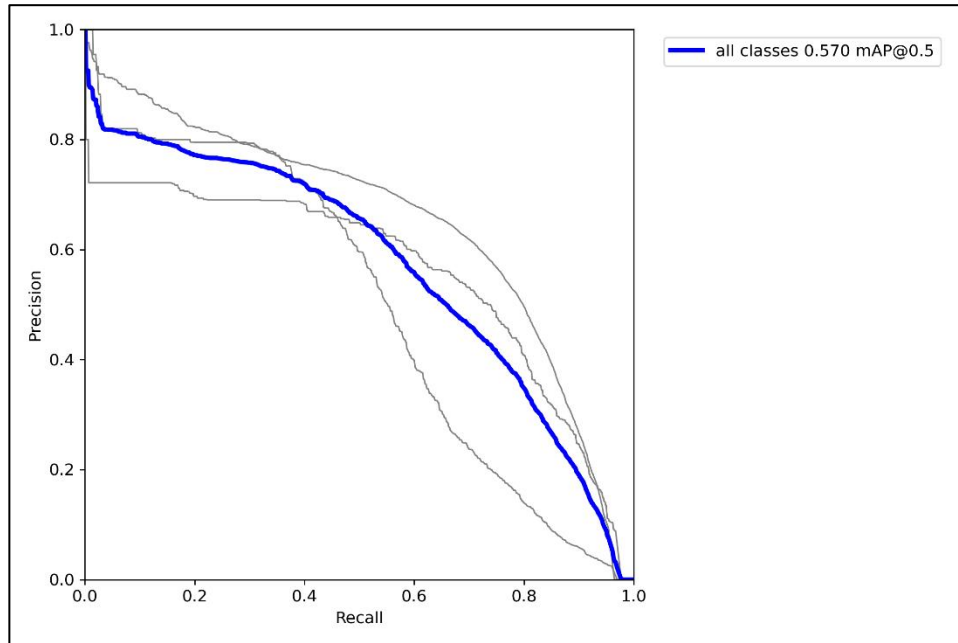


Fig 3 no improvement method result

mAP@0.5 遠低於 Q1 及 Q2 的 test result，推測是因為輸入照片中有一半的照片是沒有 label 的狀態，接下來會嘗試透過設定 freeze 及 weights，freeze 設定為 2，weights 則是沿用 Q2 所產生的 weights 去優化訓練結果，以及調整 train.py 內的 positive weight 部分去優化訓練結果。

加入 Q2 weights 及 freeze 後的 code:

```
# training command
!python train.py --project runs/train/Q3/report/with_Q2_weight_and_freeze --name refine

weights '/content/drive/MyDrive/Colab Notebooks/CV_HW4/yolov7/runs/train/Q2/report/way2/refine/weights/best.pt' --freeze
```

Test result 如下圖:

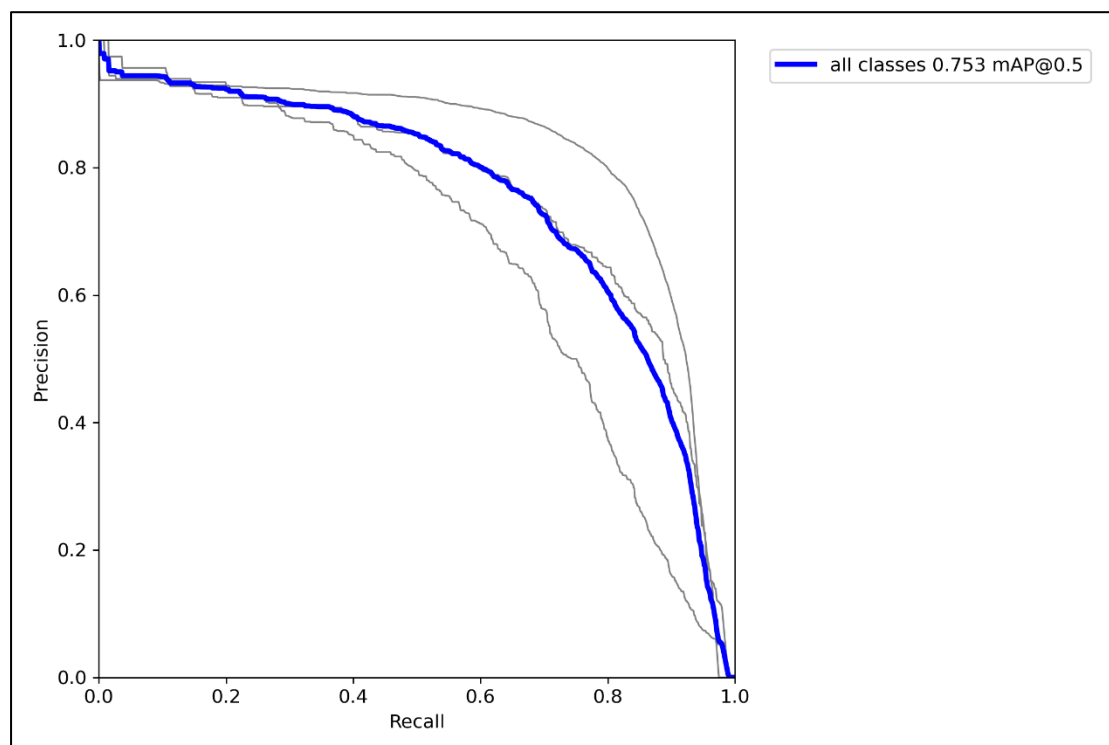


Fig 4 with Q2 weights

mAP@0.5 有顯卓的提升，從原本的 0.57 增加 0.753，甚至比 Q2 的一些方法還要高，由此可見就算沒有每張照片都 label 還是能夠將其準確度、精確度等指標提高。

調整 train.py 內的 positive weight:

```
if hyp['cls_pw'] == -1:
    hyp['cls_pw'] = [
        sum(labels[:, 0]) / sum(labels[:, 0] == n) if sum(labels[:, 0] == n) != 0 else 0 for n in range(nc)
    ]
hyp['box'] *= 3. / nl # scale to layers
hyp['cls'] *= nc / 80. * 3. / nl # scale to classes and layers
hyp['obj'] *= (imgsz / 640) ** 2 * 3. / nl # scale to image size and layers
hyp['label_smoothing'] = opt.label_smoothing

model.nc = nc # attach number of classes to model
model.hyp = hyp # attach hyperparameters to model
model.gr = 1.0 # iou loss ratio (obj_loss = 1.0 or iou)

# Calculate class weights
class_weights = labels_to_class_weights(dataset.labels, nc).to(device) * nc

# Normalize class weights to sum to 1
class_weights /= class_weights.sum()

model.class_weights = class_weights # attach normalized class weights
model.names = names
```

更改部分

Test result 如下圖:

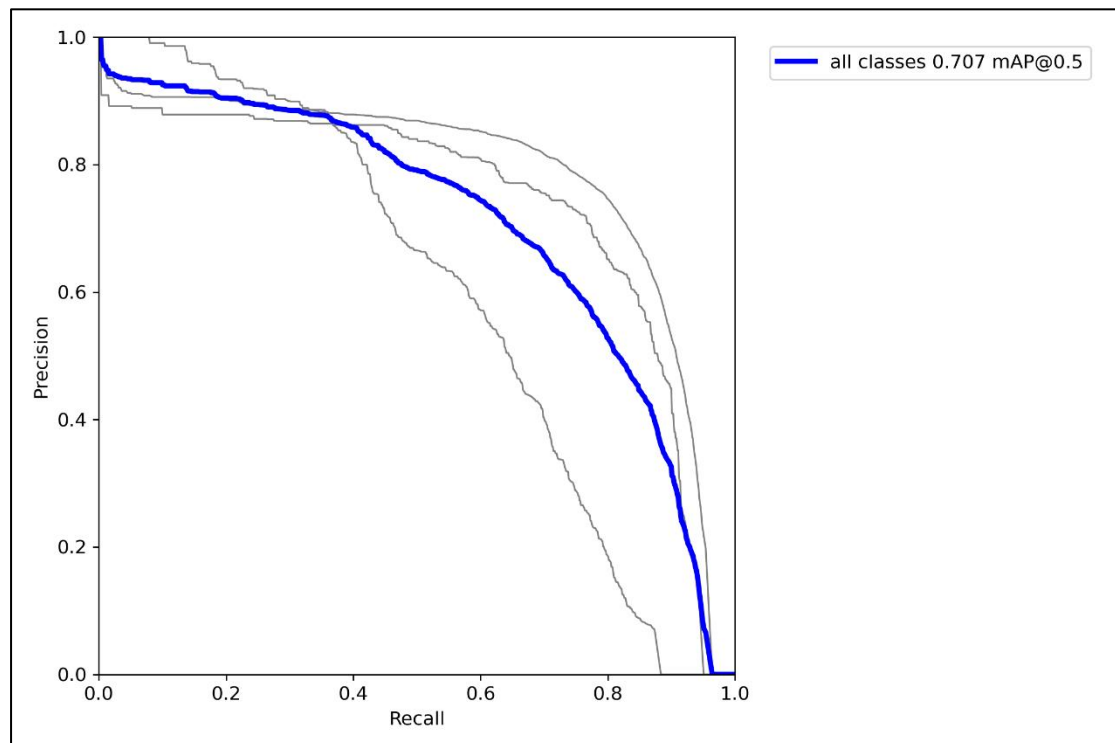


Fig 5 調整 positive weight code

## Discussion

### 1. Precision and Recall

在機器學習中，評估模型好壞的常見指標有 Accuracy、Precision、Recall 與 F1-Measure，每個指標評估的模型能力方向都有各自的意義及使用時機，在介紹這些指標前有幾個名詞(TP、TN、FP、FN)須先了解。

True Positive (TP): 預測為 Positive 且預測準確

True Negative (TN): 預測為 Negative 且預測準確

False Positive (FP): 預測為 Positive 但預測錯誤

False Negative (FN): 預測為 Negative 但預測錯誤

Accuracy 為模型預測正確數量所佔整體的比例，即正確分類的樣本數與總樣本數之比。

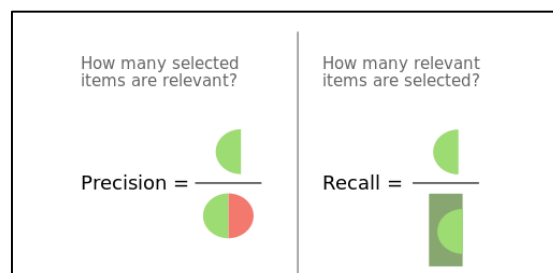
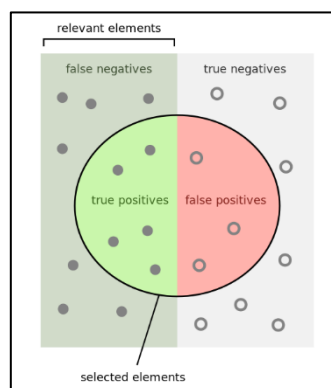
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision 為精確率，這個指標在意的是在預測為 Positive 的結果中，其精確性是多少，即被預測為 Positive 的資料中，有多少是真的 Positive。

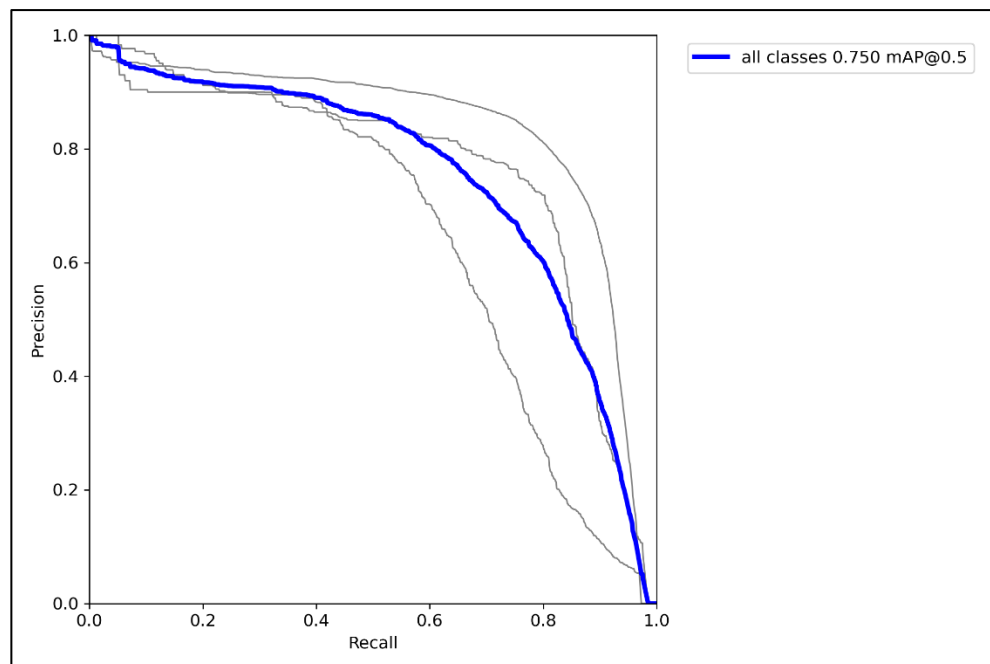
$$Precision = \frac{TP}{TP + FP}$$

Recall 為召回率，這個指標在意的是否觸及了所有的 Positive case，在所有 positive 樣本中(TP、FN)被準確預測的比例是多少。

$$Recall = \frac{TP}{TP + FN}$$



由 precision 和 recall 可組成 PR Curve，將模型評估視覺化，如下圖：



右上角的  $\text{mAP}@0.5$ ，0.5 是 IOU(Intersection Over Union)閾值，閾值越高代表對於準確度或精確度等指標要求更高，條件更嚴苛，示意圖如下圖：



由左至右，閾值越來越高，預測準確的難度更高，只有在非常確信的情況才會被視為預測正確，就會導致 FP 很少，FN 很多，進而造成 precision 很高 recall 很低的狀況。從 PR Curve 大致可推斷 P 與 R 之間的關係，而  $\text{mAP}@0.5$  是閾值為 0.5 下，PR Curve(上方藍線)與 Recall(X 軸)所圍的面積，此數值越高越好。

## 2. Q1

Q1 是實現 train\_val\_split 功能，輸入的 image path 為兩個資料夾 173、398，是由兩個不同視角的鏡頭所拍攝的畫面，兩資料夾共 200 張照片，在 train\_val\_ratio 為 0.8 時，有最佳 test result(mAP@0.5 最高)，推測原因是因為 datasets 輸入本身就來自兩個鏡頭而已，拍攝到車子的特徵不夠多元，若此時又使更多 datasets 當作 training datasets，可能會導致 overfitting，training result 非常優秀，接近 100%，但 test result 不盡理想。

不同 train\_val\_ratio 下的 train/test mAP@0.5:

train_val_ratio	Train mAP@0.5	Test mAP@0.5
Ratio = 0.7	0.958	0.602
Ratio = 0.8	0.952	0.632
Ratio = 0.9	0.960	0.618

ratio = 0.9 的 training result 雖然比 ratio = 0.8 優秀，但其 testing result 卻不是，可推斷是 overfitting 導致。



### 3. Q2

Q2 是實現 selected\_images 功能，Q2 dataset 比起 Q1 多了許多資料夾，亦即擁有多元視角下的車子照片，共六個資料夾(六個視角的鏡頭)，每個資料夾街友 200 張照片，Q2 共計有 1200 張照片，Q2 要實現的便是如何從 1200 張照片挑選 200 張照片作為訓練及驗證集。

我使用三種方法去實現照片選擇功能，分別為隨機選取、平均選取、按照資料夾有效程度進行比例分配，並搭配不同 train\_val\_ratio 去觀察結果，在 train\_val\_ratio 為 0.9 時有較佳的結果，與 Q1 結果不同，造成此差異的原因我認為是 Q2 datasets 的相機視角較多元，不會這麼容易造成 overfitting 的問題，理論上給予更多有效資料去做訓練確實也會提高預測模型的準確度。

Train\_val\_ratio = 0.9 下，不同 selected\_images 的方式之對應結果(各別實測 3 次):

Method	Train mAP@0.5	Test mAP@0.5
WAY1(random select)	0.872~0.882	0.724~0.759
WAY2(equally distributed)	0.87~0.96	0.733~0.754
WAY3(distributed in proportion)	0.805~0.927	0.7~0.75

## 4. Q3

Q3 中的 400 張照片包含 Q2 的 200 張 labeled images 以及 Q3 的 200 張 unlabeled images，這雖然可以節省人力成本(不需要花時間去人工標籤)，但會對訓練模型造成影響，甚至導致訓練結果變差。

本次作業主要介紹幾種方式去優化訓練結果，positive weights、freeze backbone 及 focal loss，下面簡單說明三種方式及其目的。

### (1) Freeze backbone

主幹網絡通常指的是深度學習模型的前幾層，用於提取特徵。在訓練過程中，凍結主幹網絡可以防止其權重被更新，本次作業在 train.py 的 code 選擇沿用 Q2 所訓練的權重(best.pt)，為避免在訓練過程中此權重被更新，可使用 freeze backbone，除了防止權重更新之外，還可增加訓練速度。

### (2) Positive weights

資料集的正樣本和負樣本數量不平衡，有時會影響模型的性能。透過賦予正樣本較大的權重，使模型更加關注對正樣本的準確預測，進而提高正樣本的召回率。

```
# Normalize class weights to sum to 1
class_weights /= class_weights.sum()

model.class_weights = class_weights # attach normalized class weights
model.names = names
```

更改前的 code 計算了各個類別的權重，現在在計算完後進行歸一化，確保類別權重總和為 1，這樣的好處在於平衡不同類別之間的影響，這樣每個類別的權重都在相同的比例範圍內，如果某些類別的權重過大，模型可能會更加關注那些權重大的類別，而對其他類別忽視。這可能導致模型的不穩定性，也可能因模型可能會傾向於過擬合該類別的訓練而造成過擬合的現象。

### (3) Focal loss

解決類別不平衡和困難樣本問題，將模型更集中地學習對難以分類的樣本進行準確預測，此方法主要針對交叉損失函數在處理大量易分類樣本時的問題，降低了對易分類樣本的關注，同時加強了對難以分類樣本的關注，這樣可以使模型更專注於那些難以區分的樣本，提高對這些樣本的預測準確性。