Report

Q1 討論:

首先關於 **split train val path function**

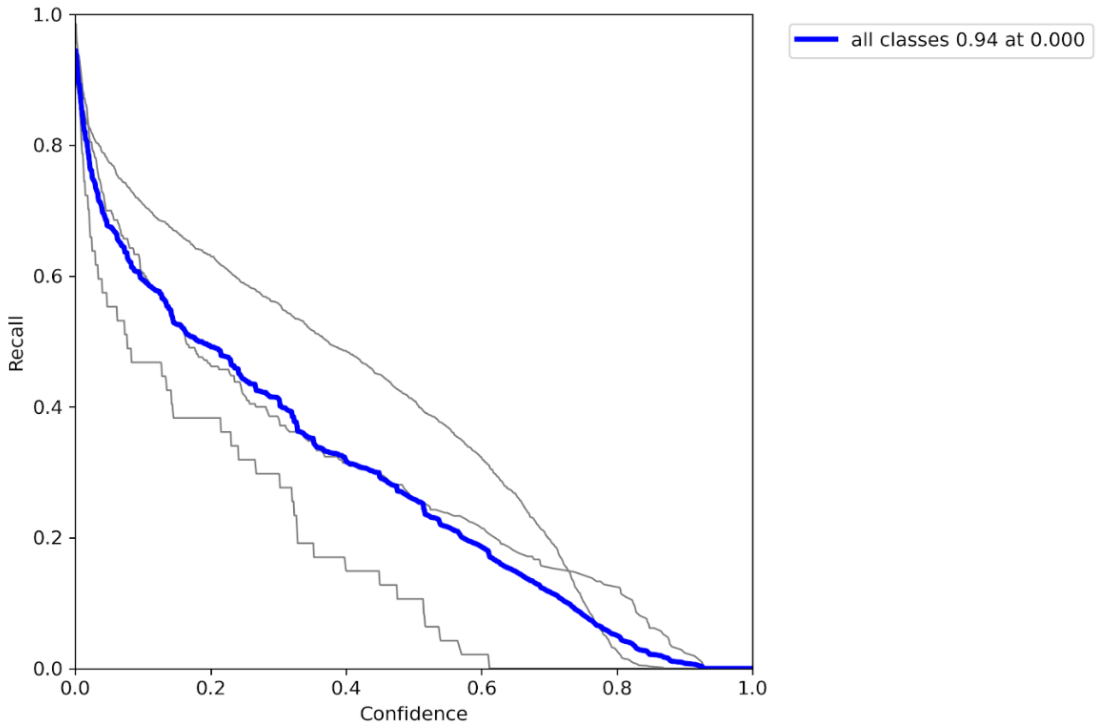
```
def split_train_val_path(all_image_paths, train_val_ratio=0.9):
    # TODO : split train and val
    random.shuffle(all_image_paths)
    train_image_paths = all_image_paths[: int(len(all_image_paths) *
train_val_ratio)]  # just an example
    val_image_paths = all_image_paths[int(len(all_image_paths) *
train_val_ratio):]  # just an example

    return train_image_paths, val_image_paths
```

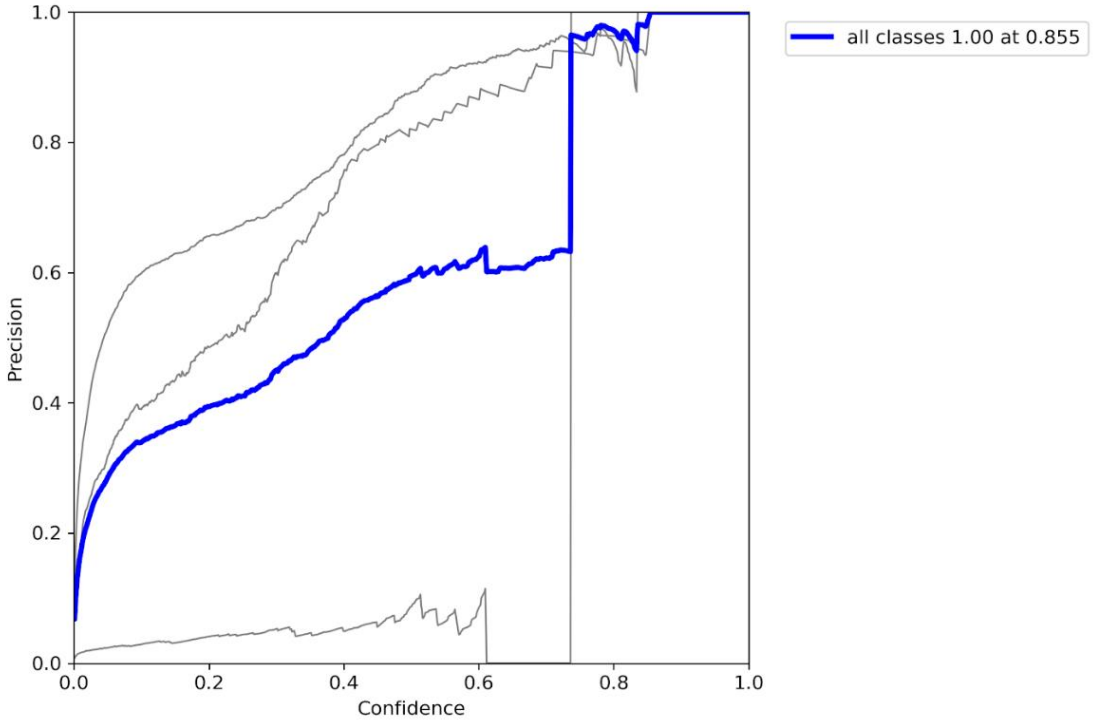
使用 random.shuffle 打亂圖片順序，在分割訓練集和驗證集，預設為 0.9（90%用於訓練，10%用於驗證）。

170 相機:

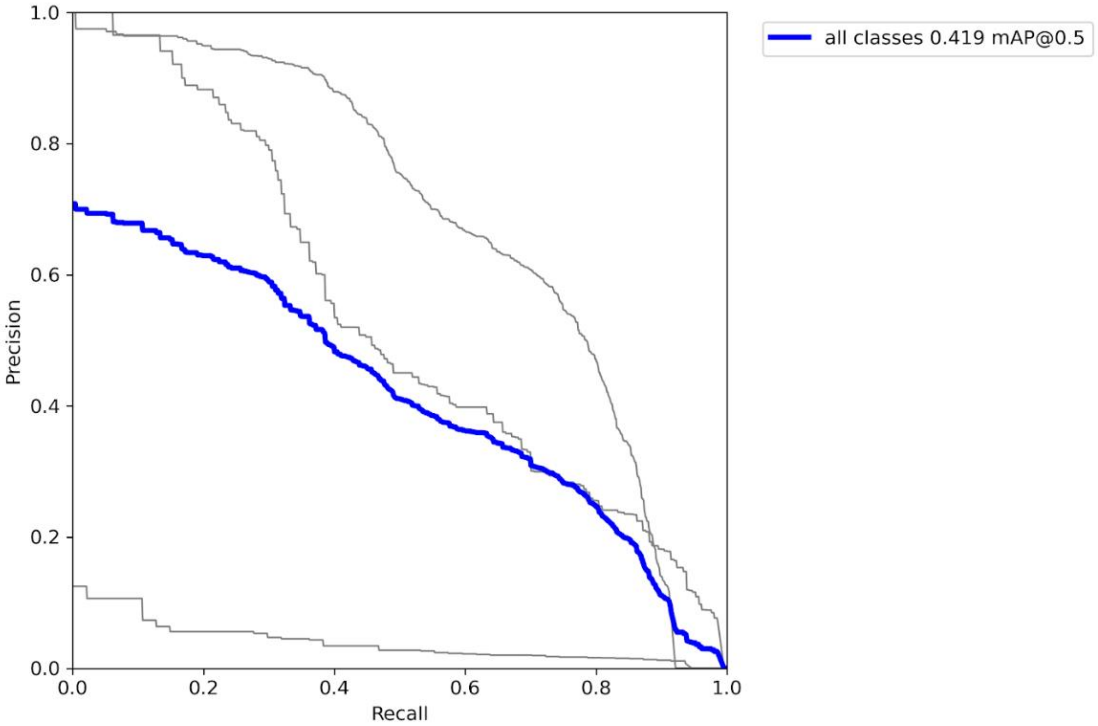
R_curve:



P_curve:

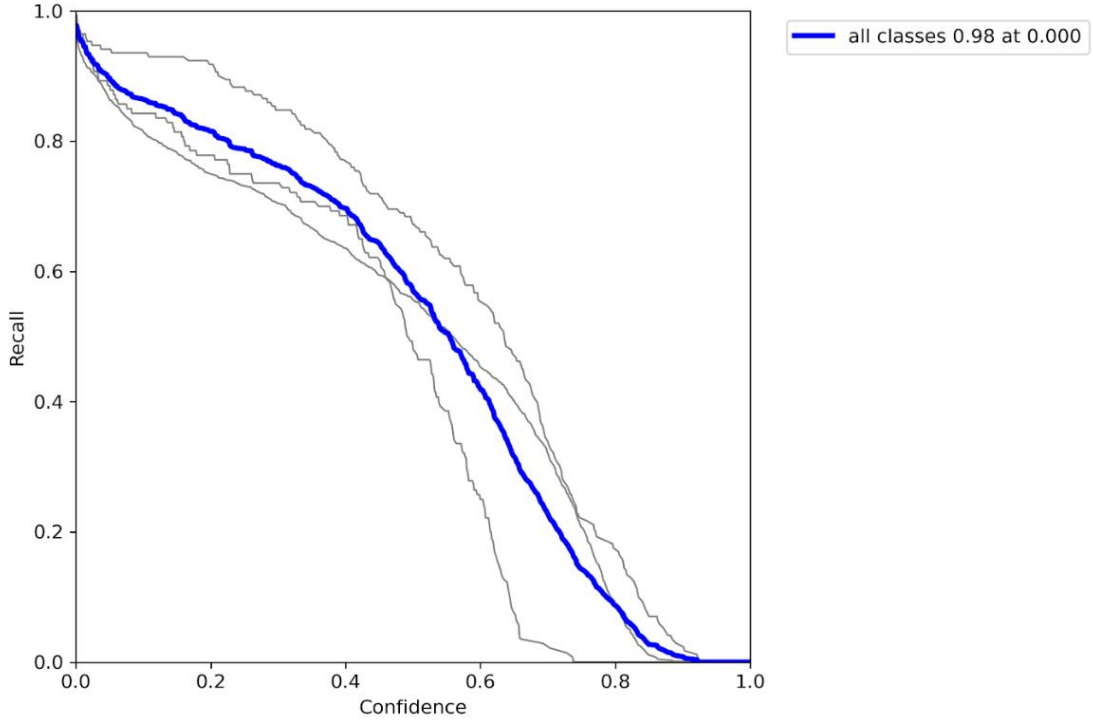


PR_curve:

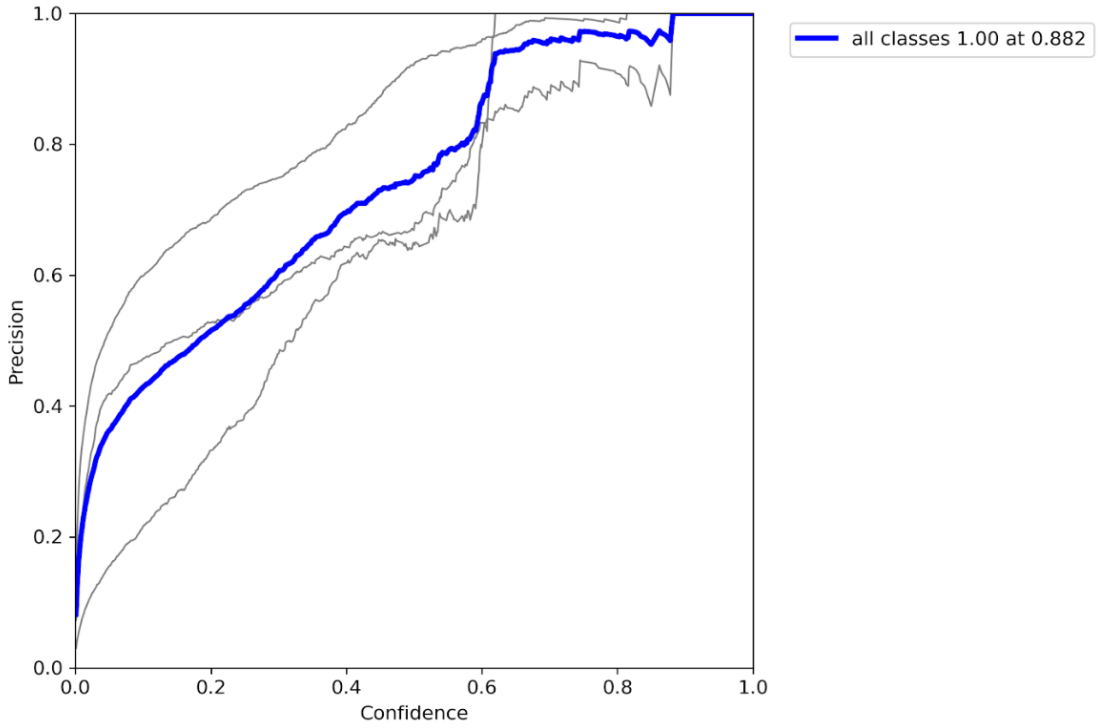


173 相機:

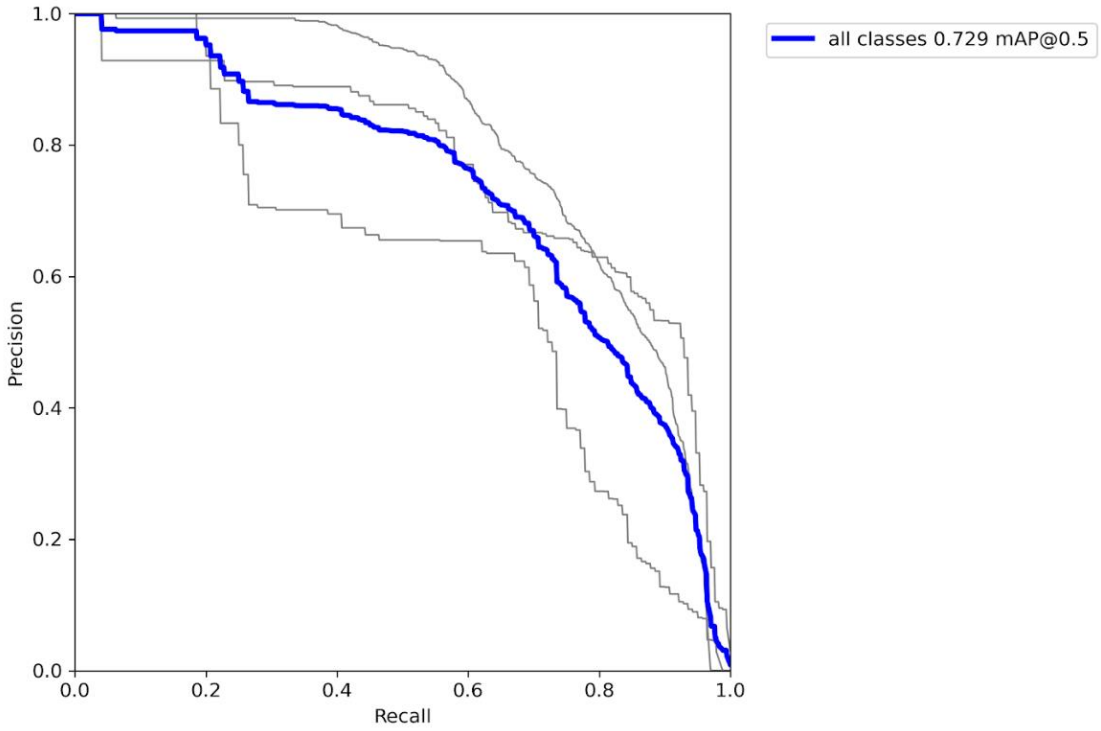
R_curve:



P_curve:

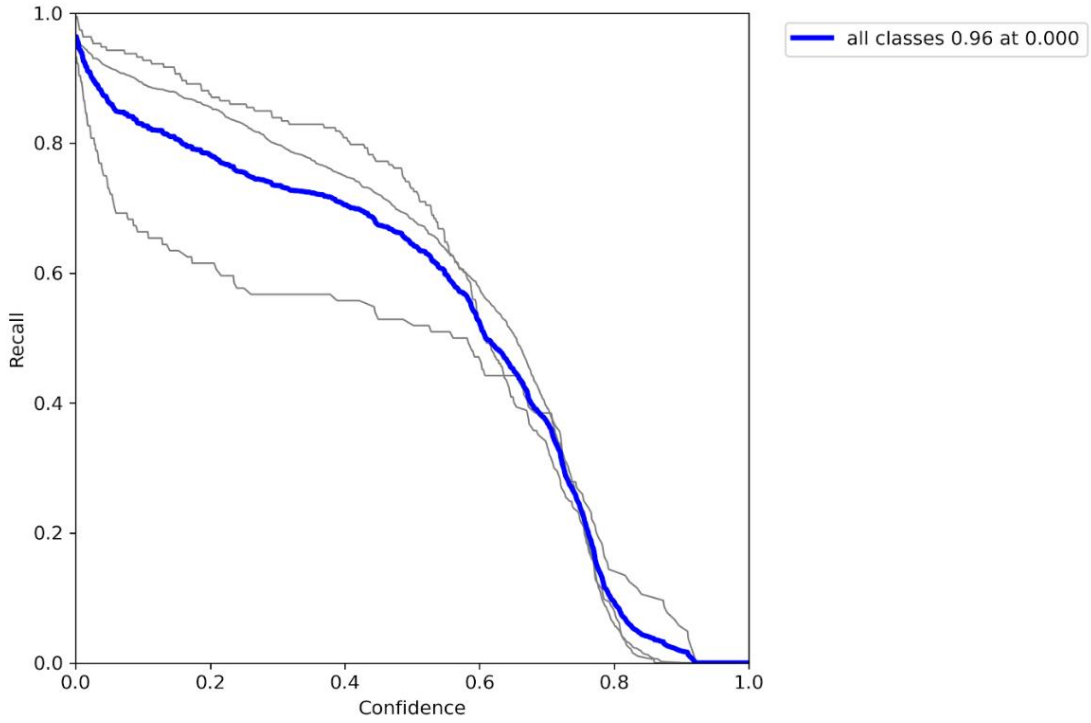


PR_curve:

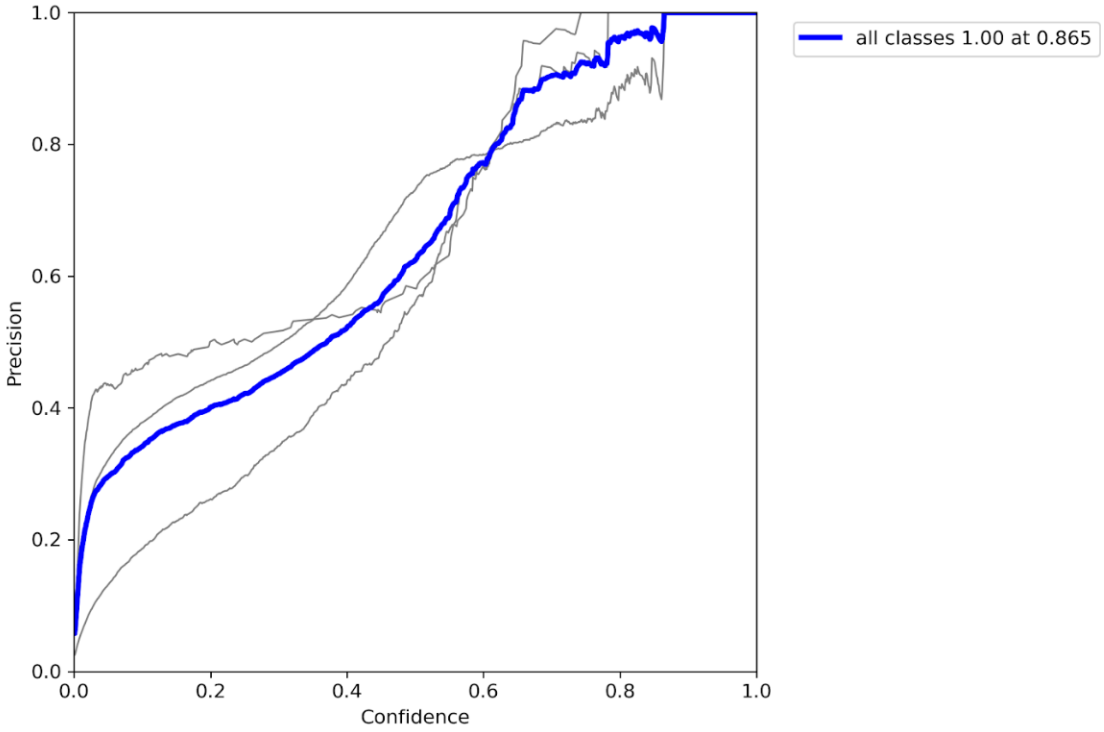


398 相機：

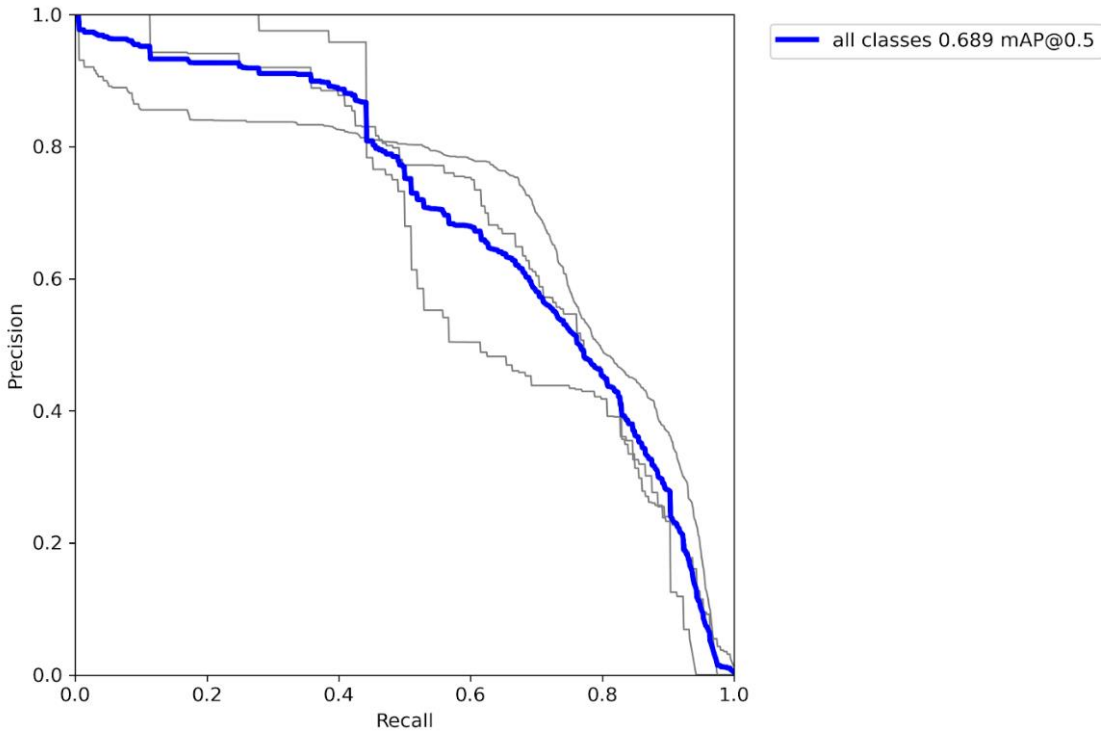
R_curve：



P_curve：

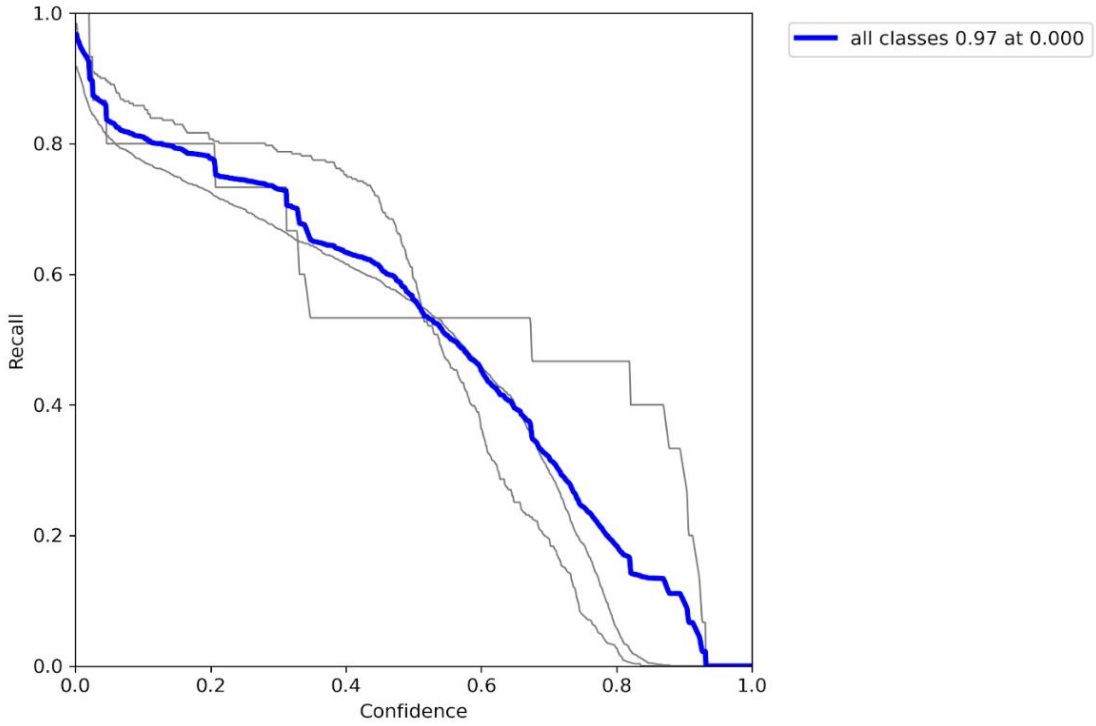


PR_curve:

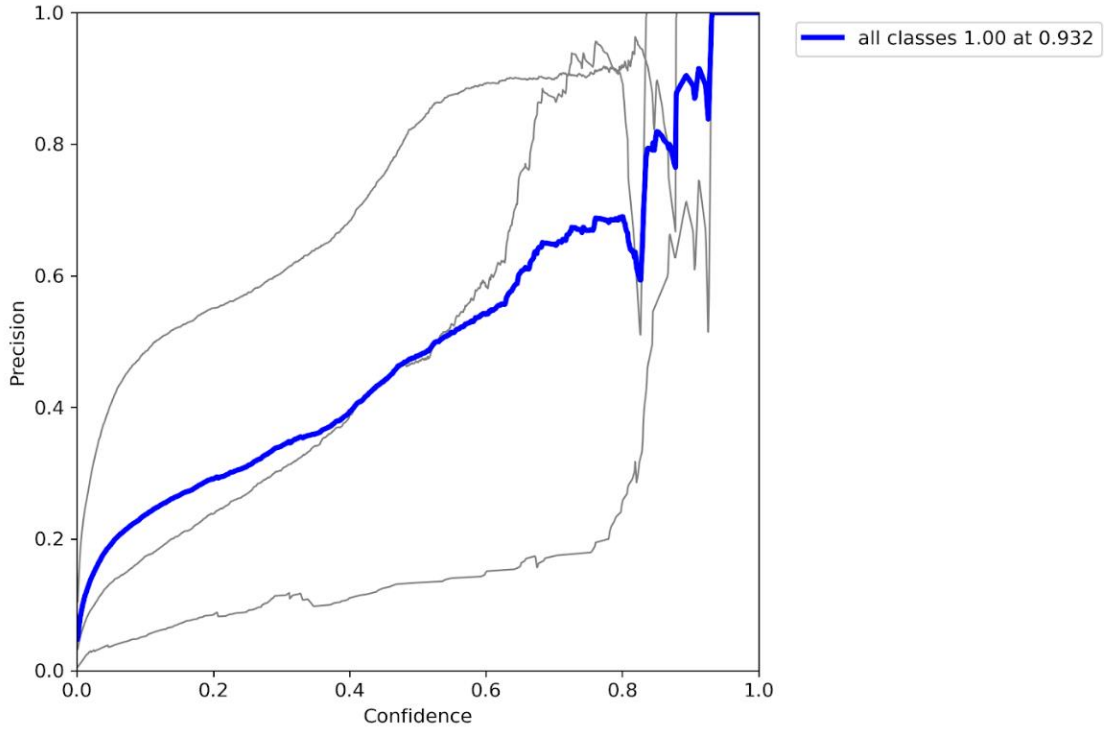


410 相機:

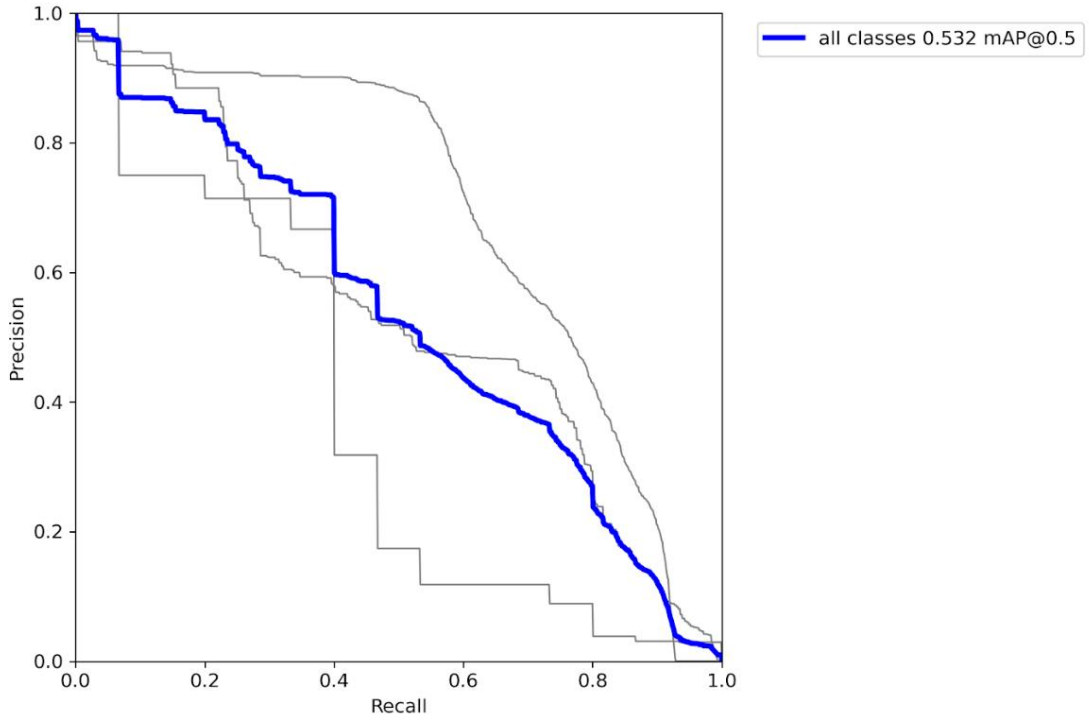
R_curve:



P_curve:

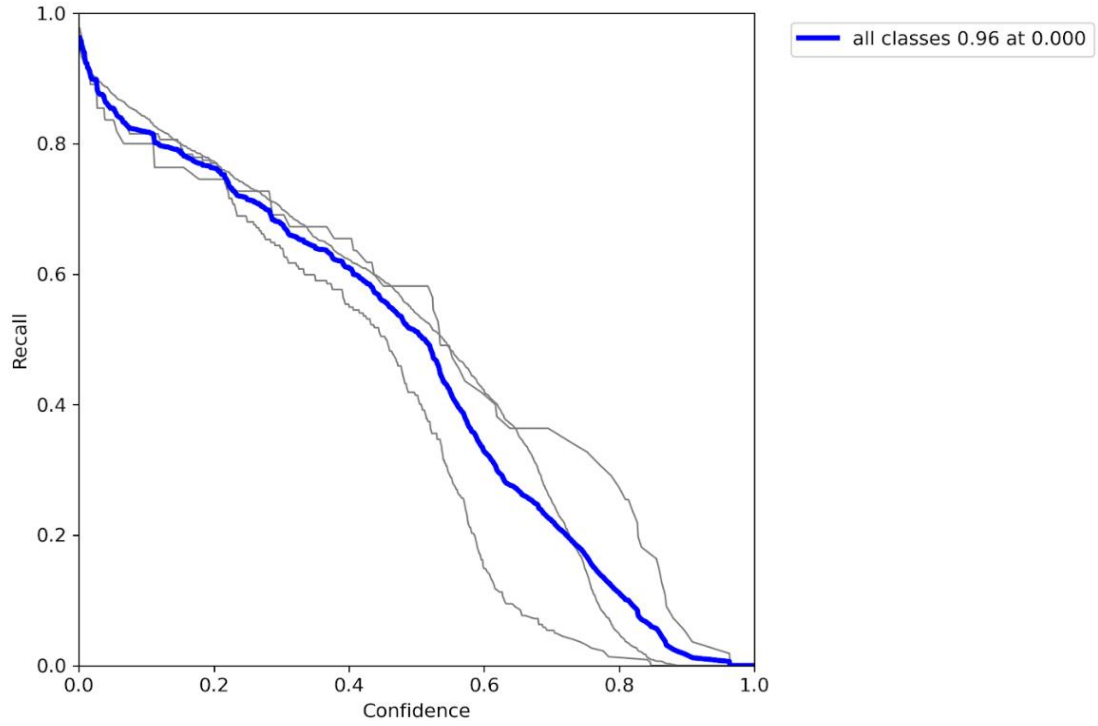


PR_curve:

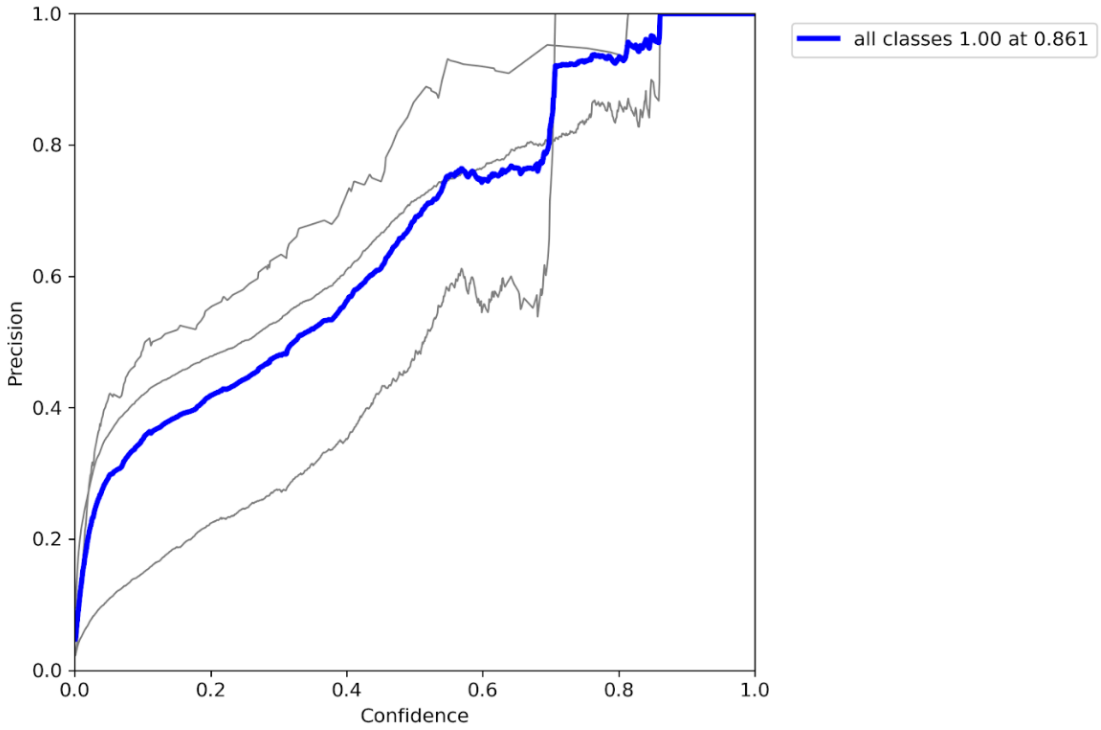


495 相機：

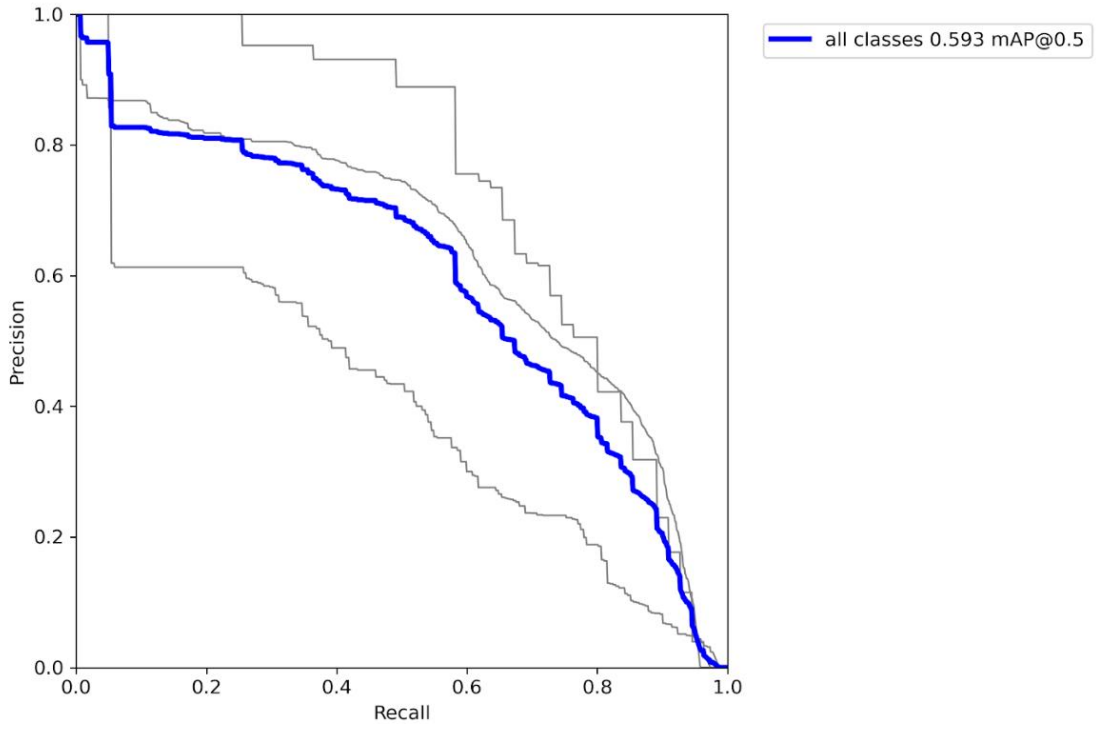
R_curve：



P_curve：

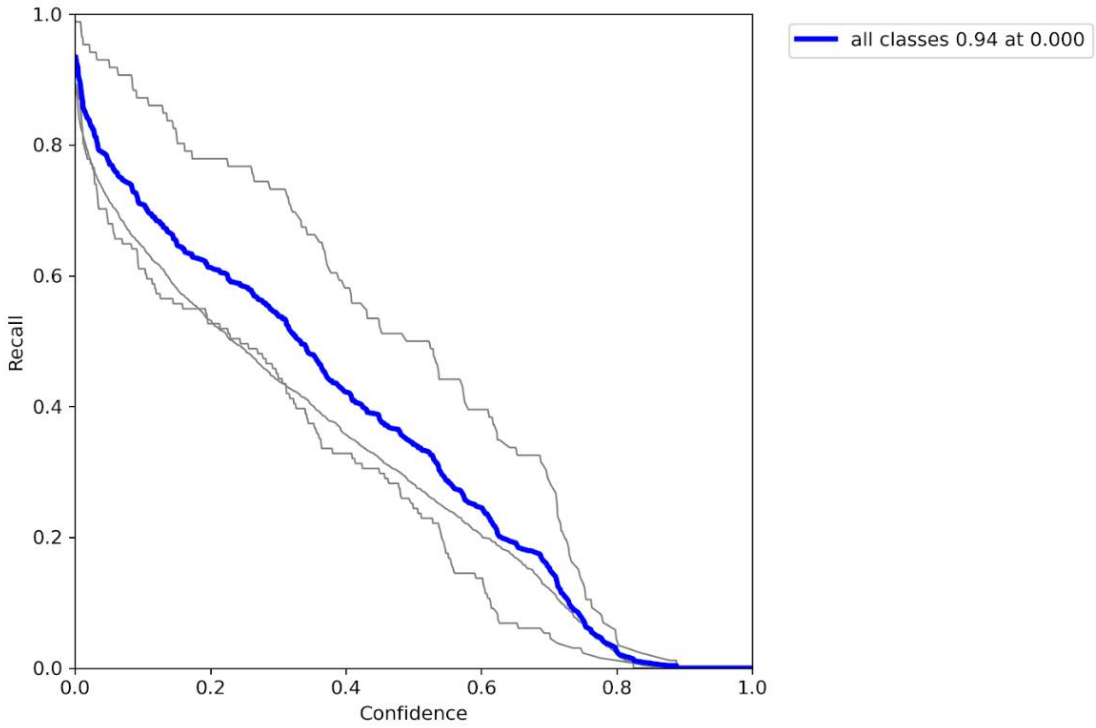


PR_curve:

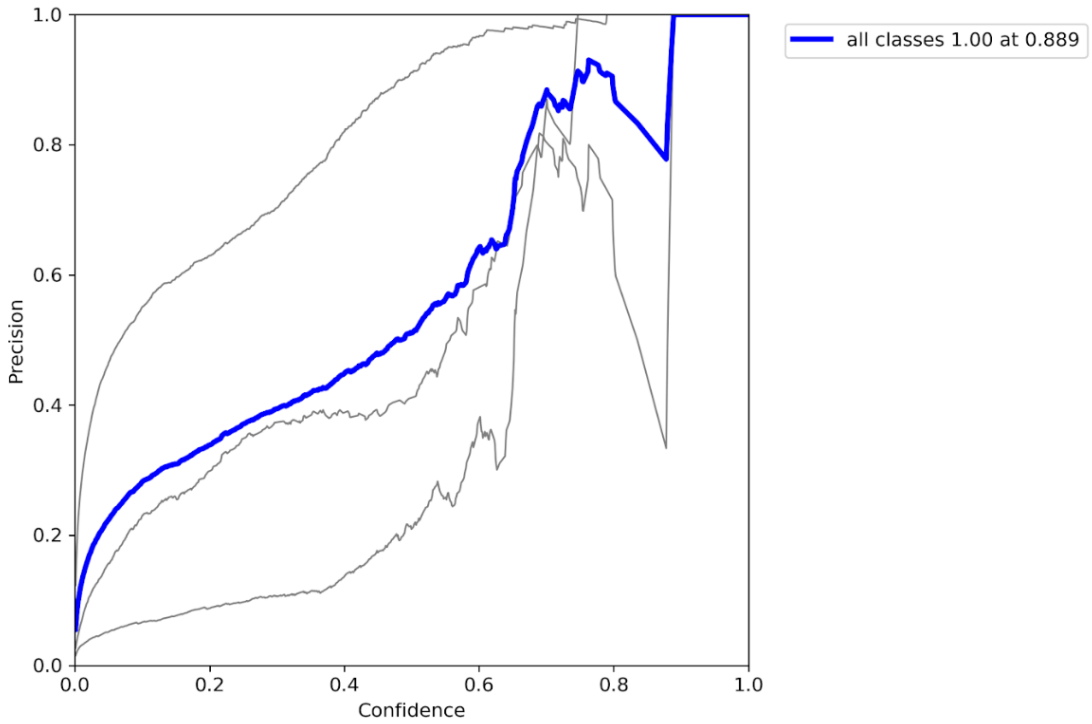


511 相機:

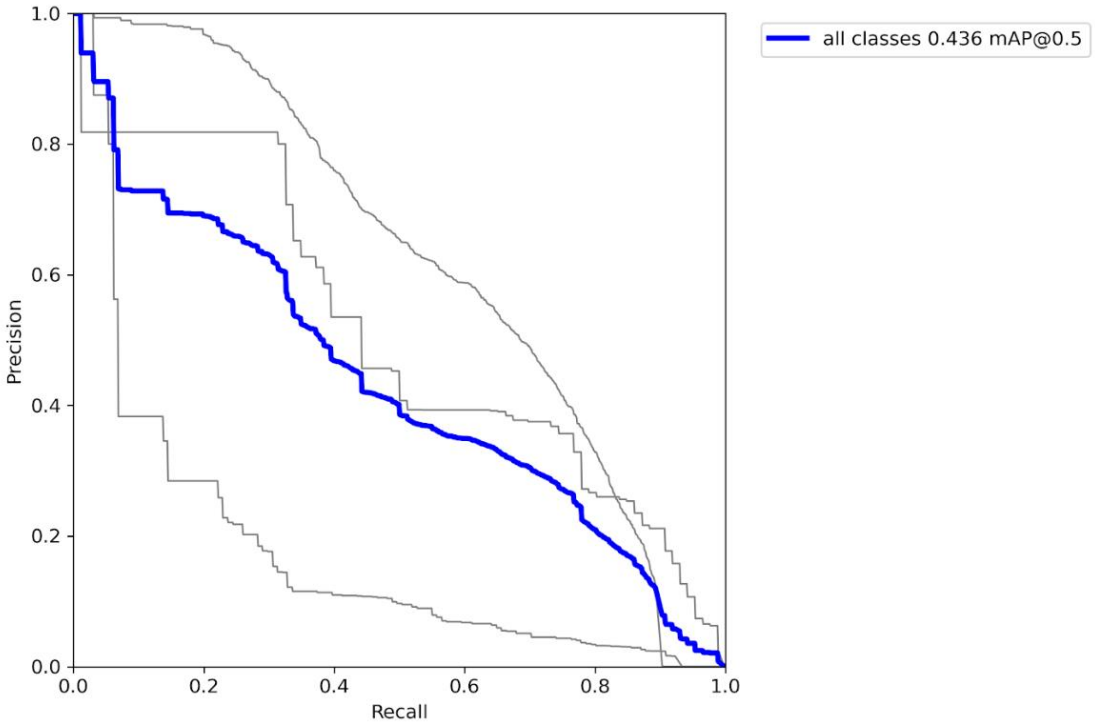
R_curve:



P_curve:



PR_curve:



Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.394	0.494	0.419	0.253
	car	1200	1504	0.656	0.633	0.685	0.384
	bus	1200	210	0.487	0.467	0.534	0.352
	truck	1200	47	0.041	0.383	0.039	0.022
495							
	all	1200	2227	0.658	0.535	0.593	0.354
	car	1200	1950	0.695	0.565	0.636	0.356
	bus	1200	55	0.825	0.582	0.741	0.509
	truck	1200	222	0.454	0.457	0.403	0.198
410							
	all	1200	2331	0.462	0.597	0.532	0.33
	car	1200	2005	0.79	0.576	0.688	0.404
	bus	1200	15	0.131	0.533	0.373	0.295
	truck	1200	311	0.466	0.682	0.535	0.291
511							
	all	1200	2294	0.376	0.578	0.436	0.275
	car	1200	2077	0.673	0.479	0.624	0.331
	bus	1200	86	0.356	0.766	0.512	0.378
	truck	1200	131	0.1	0.489	0.174	0.116
398							
	all	1200	2353	0.753	0.568	0.689	0.449
	car	1200	2056	0.78	0.602	0.699	0.413
	bus	1200	104	0.726	0.5	0.665	0.459
	truck	1200	193	0.753	0.601	0.704	0.473
173							
	all	1200	1991	0.698	0.696	0.729	0.466
	car	1200	1680	0.83	0.634	0.81	0.499
	bus	1200	171	0.643	0.768	0.764	0.533
	truck	1200	140	0.623	0.686	0.612	0.367
ALL							
	all	1200	12957	0.589	0.529	0.558	0.343
	car	1200	11272	0.771	0.531	0.669	0.386
	bus	1200	641	0.57	0.495	0.545	0.377
	truck	1200	1044	0.425	0.561	0.459	0.266

分析討論:

我們可以藉由分析 R_curve 圖，觀察到 173 相機擁有最好的 True Positive Rate 為 0.98，而 170 相機和 511 相機相對 True Positive Rate 較差為 0.94，表示 173 相機在檢測實際正例方面稍微更為優越。

另外分析 P_curve 圖，我們可以觀察到 410 相機擁有最好的精確度 (Precision)為 0.932 而 170 相機是最差的精確度(Precision)為 0.855，表示當這個 410 相機預測為正例時，正例的確認機率更高。這可能意味著在這個閾值下，410 相機的正例預測相對更可信賴。

分析 PR_curve 圖，我們可以觀察到 173 相機 mAP@0.5 最高而 170 相機 mAP@0.5 最低，這可能表示在 IoU 閾值為 0.5 時，173 相機的平均精確度最高。

Q2 討論:

首先關於 select images function，我使用了 3 種方法抽樣測試

第一種是直接使用 random_shuffle 後直接抽樣 200 張

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

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.654	0.63	0.644	0.414
	car	1200	1504	0.927	0.765	0.863	0.515
	bus	1200	210	0.753	0.826	0.84	0.555
	truck	1200	47	0.283	0.298	0.229	0.174
495							
	all	1200	2227	0.678	0.777	0.742	0.482
	car	1200	1950	0.679	0.882	0.757	0.444
	bus	1200	55	0.841	0.873	0.894	0.637
	truck	1200	222	0.514	0.576	0.574	0.364
410							
	all	1200	2331	0.726	0.663	0.764	0.535
	car	1200	2005	0.896	0.581	0.799	0.493
	bus	1200	15	0.478	0.733	0.681	0.592
	truck	1200	311	0.803	0.675	0.812	0.52
511							
	all	1200	2294	0.819	0.664	0.768	0.508
	car	1200	2077	0.951	0.705	0.888	0.513
	bus	1200	86	0.906	0.884	0.934	0.695
	truck	1200	131	0.601	0.405	0.482	0.318
398							
	all	1200	2353	0.738	0.663	0.734	0.483
	car	1200	2056	0.745	0.714	0.743	0.449
	bus	1200	104	0.915	0.529	0.804	0.555
	truck	1200	193	0.553	0.746	0.657	0.445
173							
	all	1200	1991	0.735	0.744	0.803	0.548
	car	1200	1680	0.886	0.845	0.918	0.592
	bus	1200	171	0.813	0.737	0.832	0.6
	truck	1200	140	0.505	0.65	0.66	0.451
ALL							
	all	1200	12957	0.735	0.678	0.743	0.484
	car	1200	11272	0.823	0.725	0.798	0.482
	bus	1200	641	0.818	0.694	0.802	0.561
	truck	1200	1044	0.566	0.613	0.627	0.409

第二種方法是對於每個相機類別都先 random_shuffle 後，再每個相機類別分別抽樣平均數量到達 200 張。

```
categories = ['Q2/170', 'Q2/173', 'Q2/398', 'Q2/410', 'Q2/495',
'Q2/511']
# 創建一個字典來存儲每個類別的圖像路徑
category_paths = {category: [i for i in image_paths if category in i]
for category in categories}
```

```
# 對每個類別的圖像路徑進行洗牌
for category in category_paths:
    random.shuffle(category_paths[category])
    # 計算每個類別的平均圖像數量
avg_images_num = round(images_num / len(categories))
# 從每個類別選擇圖像
selected_image_paths = [path[:avg_images_num] for path in
category_paths.values()]
# 展平所選擇的圖像路徑列表
selected_image_paths = [item for sublist in selected_image_paths for
item in sublist]
```

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.604	0.672	0.616	0.391
	car	1200	1504	0.843	0.822	0.868	0.5
	bus	1200	210	0.704	0.852	0.777	0.541
	truck	1200	47	0.265	0.34	0.204	0.132
495							
	all	1200	2227	0.672	0.671	0.694	0.432
	car	1200	1950	0.703	0.863	0.778	0.449
	bus	1200	55	0.791	0.564	0.758	0.542
	truck	1200	222	0.522	0.585	0.546	0.305
410							
	all	1200	2331	0.658	0.72	0.665	0.423
	car	1200	2005	0.862	0.718	0.811	0.482
	bus	1200	15	0.235	0.8	0.342	0.253
	truck	1200	311	0.877	0.641	0.841	0.533
511							
	all	1200	2294	0.743	0.686	0.742	0.493
	car	1200	2077	0.83	0.822	0.885	0.501
	bus	1200	86	0.821	0.93	0.925	0.711
	truck	1200	131	0.577	0.305	0.415	0.269
398							
	all	1200	2353	0.729	0.643	0.707	0.454
	car	1200	2056	0.746	0.736	0.745	0.429
	bus	1200	104	0.92	0.442	0.704	0.519
	truck	1200	193	0.523	0.751	0.673	0.413
173							
	all	1200	1991	0.802	0.788	0.839	0.543
	car	1200	1680	0.897	0.835	0.914	0.579
	bus	1200	171	0.875	0.842	0.878	0.626
	truck	1200	140	0.635	0.686	0.727	0.424
ALL							
	all	1200	12957	0.693	0.736	0.723	0.459
	car	1200	11272	0.79	0.808	0.815	0.477
	bus	1200	641	0.727	0.766	0.739	0.532
	truck	1200	1044	0.563	0.633	0.614	0.369

第三種方法是先讀取所有圖像的 txt 檔後，計算每張圖的 label 數量，選取最多 label 數量的前 200 張圖後，在進行最後 random_shuffle。

```
# 收集每個圖像的標籤計數
label_counts = defaultdict(set)
for path_jpg in image_paths:
    path_txt = path_jpg[:-3] + 'txt'
    with open(path_txt, 'r') as file:
         lines = file.readlines()
    for line in lines:
        class_index = int(line.split()[0])
        label_counts[path_jpg].add(class_index)
# 根據每個圖像擁有的獨特類別數量對路徑進行排序
sorted_paths = sorted(image_paths, key=lambda path:
len(label_counts[path]), reverse=True)
# 根據類別進行分組
categories = ['Q2/170', 'Q2/173', 'Q2/398', 'Q2/410', 'Q2/495',
'Q2/511']
category_paths = {category: [path for path in sorted_paths if category
in path] for category in categories}
# 計算每個類別的平均圖像數量
avg_images_num = round(images_num / len(categories))
# 從每個類別中選擇圖像
selected_image_paths = [path[:avg_images_num] for path in
category_paths.values()]
# 展平所選擇的圖像路徑列表
selected_image_paths = [item for sublist in selected_image_paths for
item in sublist]
# 確保最終選擇包含至少 200 條路徑
selected_image_paths = selected_image_paths[:200] if
len(selected_image_paths) >= 200 else selected_image_paths
# 對所選擇的圖像路徑進行洗牌
random.shuffle(selected_image_paths)
```

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.652	0.707	0.68	0.436
	car	1200	1504	0.904	0.809	0.884	0.516
	bus	1200	210	0.658	0.867	0.801	0.552
	truck	1200	47	0.395	0.445	0.354	0.242
495							
	all	1200	2227	0.736	0.747	0.767	0.516
	car	1200	1950	0.71	0.877	0.76	0.442
	bus	1200	55	0.868	0.836	0.902	0.688
	truck	1200	222	0.63	0.527	0.639	0.419
410							
	all	1200	2331	0.625	0.796	0.694	0.461
	car	1200	2005	0.855	0.725	0.802	0.469
	bus	1200	15	0.279	0.933	0.487	0.392
	truck	1200	311	0.742	0.729	0.793	0.521
511							
	all	1200	2294	0.799	0.766	0.788	0.512
	car	1200	2077	0.878	0.821	0.889	0.507
	bus	1200	86	0.885	0.807	0.907	0.654
	truck	1200	131	0.632	0.672	0.567	0.374
398							
	all	1200	2353	0.761	0.709	0.752	0.486
	car	1200	2056	0.748	0.745	0.761	0.449
	bus	1200	104	0.895	0.683	0.797	0.537
	truck	1200	193	0.641	0.699	0.699	0.472
173							
	all	1200	1991	0.838	0.787	0.843	0.564
	car	1200	1680	0.936	0.812	0.924	0.586
	bus	1200	171	0.857	0.876	0.865	0.623
	truck	1200	140	0.72	0.671	0.74	0.482
ALL							
	all	1200	12957	0.732	0.767	0.76	0.493
	car	1200	11272	0.814	0.803	0.82	0.483
	bus	1200	641	0.749	0.81	0.804	0.566
	truck	1200	1044	0.634	0.687	0.657	0.431

分析討論:

因為 Q2 資料集比起 Q1 資料集擁有更多的數據，因此可以看到第一種抽樣方法與測試 Q1 時的一樣，但是我們可以從 P_curve、R_curve 和 PR_curve 觀察到，Q2 時每個相機的準確率基本上都比

Q1 高，並且 mAP@0.5 準確率來到 0.743，相比 Q1 mAP@0.5 準確率 0.558 好上許多。

另外從 3 種抽樣方法數據對比可以看到，第三種方法的 mAP@0.5 準確率是最高的來到 0.76，可能原因是抽樣訓練的資料擁有最多的 label 資訊，而第一種抽樣方法理論上來說應該最差，但是剛好單純 random 後都可以選取到較好數據，因此數據上準確率第二種方法略低於第一種方法。不過最後測試完第三種抽樣方法是最好的方式。

Q3 討論:

Q2+Q3 pseudo label 資料測試:

首先我使用 detect.py 對 Q3 資料進行 label 輸出，後使用第三種抽樣方法對於 Q2 資料選取 200 張，也對於 Q3 資料集抽取 200 張進行訓練。

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.669	0.69	0.692	0.468
	car	1200	1504	0.918	0.808	0.894	0.544
	bus	1200	210	0.711	0.881	0.789	0.568
	truck	1200	47	0.377	0.383	0.393	0.29
495							
	all	1200	2227	0.635	0.887	0.77	0.531
	car	1200	1950	0.641	0.939	0.789	0.481
	bus	1200	55	0.753	0.982	0.899	0.684
	truck	1200	222	0.511	0.739	0.623	0.427
410							
	all	1200	2331	0.693	0.757	0.759	0.526
	car	1200	2005	0.848	0.715	0.804	0.501
	bus	1200	15	0.362	0.8	0.628	0.508
	truck	1200	311	0.869	0.756	0.846	0.569
511							
	all	1200	2294	0.714	0.791	0.759	0.498
	car	1200	2077	0.865	0.864	0.893	0.529
	bus	1200	86	0.722	0.814	0.838	0.611
	truck	1200	131	0.555	0.694	0.546	0.355
398							
	all	1200	2353	0.712	0.787	0.715	0.479
	car	1200	2056	0.712	0.823	0.754	0.463
	bus	1200	104	0.912	0.798	0.81	0.6
	truck	1200	193	0.511	0.741	0.58	0.373
173							
	all	1200	1991	0.831	0.831	0.877	0.609
	car	1200	1680	0.932	0.829	0.923	0.61
	bus	1200	171	0.865	0.901	0.936	0.687
	truck	1200	140	0.695	0.763	0.773	0.529
ALL							
	all	1200	12957	0.719	0.793	0.771	0.518
	car	1200	11272	0.803	0.834	0.831	0.513
	bus	1200	641	0.757	0.821	0.815	0.594
	truck	1200	1044	0.598	0.724	0.667	0.445

另外我也進行 Q3 數量 1200 測試，因此總共利用 1400 個數據進行訓練，想觀察是否改善準確率。

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.678	0.702	0.661	0.45
	car	1200	1504	0.929	0.817	0.897	0.542
	bus	1200	210	0.696	0.862	0.72	0.525
	truck	1200	47	0.408	0.426	0.365	0.281
495							
	all	1200	2227	0.73	0.797	0.768	0.511
	car	1200	1950	0.696	0.912	0.787	0.468
	bus	1200	55	0.907	0.89	0.911	0.65
	truck	1200	222	0.586	0.59	0.607	0.414
410							
	all	1200	2331	0.608	0.781	0.722	0.496
	car	1200	2005	0.819	0.736	0.807	0.489
	bus	1200	15	0.263	0.8	0.6	0.5
	truck	1200	311	0.742	0.807	0.759	0.498
511							
	all	1200	2294	0.815	0.72	0.803	0.544
	car	1200	2077	0.911	0.785	0.89	0.523
	bus	1200	86	0.947	0.779	0.915	0.71
	truck	1200	131	0.586	0.595	0.604	0.399
398							
	all	1200	2353	0.699	0.772	0.674	0.449
	car	1200	2056	0.7	0.834	0.755	0.46
	bus	1200	104	0.891	0.788	0.786	0.59
	truck	1200	193	0.507	0.694	0.48	0.296
173							
	all	1200	1991	0.853	0.814	0.881	0.604
	car	1200	1680	0.919	0.843	0.923	0.604
	bus	1200	171	0.885	0.848	0.911	0.684
	truck	1200	140	0.755	0.75	0.809	0.525
ALL							
	all	1200	12957	0.727	0.79	0.749	0.499
	car	1200	11272	0.798	0.841	0.832	0.506
	bus	1200	641	0.772	0.834	0.776	0.576
	truck	1200	1044	0.61	0.694	0.638	0.415

分析討論:

可以發現其實 Q3 選取 200 個數據和 Q2 選取 200 個數據時，mAP@0.5 準確率是最好的結果，來到準確率是 0.771，而 Q3 選取 1200 個數據和 Q2 選取 200 個數據時，準確率是 0.749，反而降低了，可能原因是使用了 Q3 所有的數據後，裡面可能有不好的數據影響到訓練品質。

freeze backbone 測試:

首先我使用 freeze backbone 固定了 10 層參數進行測試

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.682	0.646	0.662	0.416
	car	1200	1504	0.9	0.741	0.851	0.48
	bus	1200	210	0.668	0.814	0.792	0.542
	truck	1200	47	0.48	0.383	0.342	0.226
495							
	all	1200	2227	0.73	0.765	0.752	0.486
	car	1200	1950	0.67	0.858	0.743	0.419
	bus	1200	55	0.833	0.909	0.932	0.718
	truck	1200	222	0.689	0.528	0.582	0.319
410							
	all	1200	2331	0.641	0.737	0.713	0.47
	car	1200	2005	0.857	0.702	0.802	0.471
	bus	1200	15	0.171	0.933	0.556	0.463
	truck	1200	311	0.895	0.575	0.78	0.476
511							
	all	1200	2294	0.753	0.651	0.677	0.422
	car	1200	2077	0.898	0.742	0.88	0.484
	bus	1200	86	0.812	0.593	0.629	0.452
	truck	1200	131	0.55	0.618	0.521	0.33
398							
	all	1200	2353	0.726	0.718	0.721	0.465
	car	1200	2056	0.729	0.757	0.74	0.432
	bus	1200	104	0.887	0.692	0.785	0.575
	truck	1200	193	0.562	0.705	0.639	0.389
173							
	all	1200	1991	0.865	0.739	0.854	0.573
	car	1200	1680	0.951	0.706	0.907	0.554
	bus	1200	171	0.858	0.846	0.874	0.632
	truck	1200	140	0.787	0.664	0.781	0.532
ALL							
	all	1200	12957	0.717	0.722	0.725	0.459
	car	1200	11272	0.801	0.765	0.801	0.461
	bus	1200	641	0.685	0.783	0.746	0.532
	truck	1200	1044	0.666	0.619	0.627	0.382

另外我也使用 freeze backbone 固定了 3 層參數進行測試

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.686	0.675	0.657	0.435
	car	1200	1504	0.945	0.749	0.88	0.536
	bus	1200	210	0.718	0.829	0.797	0.582
	truck	1200	47	0.395	0.447	0.293	0.187
495							
	all	1200	2227	0.742	0.773	0.775	0.541
	car	1200	1950	0.731	0.866	0.778	0.476
	bus	1200	55	0.848	0.836	0.888	0.725
	truck	1200	222	0.646	0.616	0.66	0.423
410							
	all	1200	2331	0.713	0.711	0.678	0.455
	car	1200	2005	0.884	0.66	0.814	0.5
	bus	1200	15	0.384	0.8	0.397	0.324
	truck	1200	311	0.871	0.674	0.824	0.542
511							
	all	1200	2294	0.77	0.764	0.784	0.531
	car	1200	2077	0.878	0.835	0.889	0.522
	bus	1200	86	0.928	0.756	0.913	0.719
	truck	1200	131	0.503	0.702	0.55	0.353
398							
	all	1200	2353	0.778	0.705	0.77	0.529
	car	1200	2056	0.768	0.701	0.765	0.47
	bus	1200	104	0.932	0.663	0.825	0.635
	truck	1200	193	0.633	0.75	0.721	0.482
173							
	all	1200	1991	0.823	0.798	0.865	0.608
	car	1200	1680	0.938	0.789	0.925	0.613
	bus	1200	171	0.865	0.863	0.896	0.681
	truck	1200	140	0.666	0.741	0.773	0.532
ALL							
	all	1200	12957	0.726	0.779	0.774	0.524
	car	1200	11272	0.818	0.798	0.826	0.508
	bus	1200	641	0.77	0.798	0.81	0.613
	truck	1200	1044	0.59	0.74	0.686	0.449

分析討論:

可以觀察到如果固定太多層參數後，mAP@0.5 準確率會下降，而固定 3 層參數時的 mAP@0.5 準確率較好，但是跟不使用 freeze backbone 相比效果沒有比較好。

positive weight 測試:

Result:

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.637	0.703	0.665	0.424
	car	1200	1504	0.912	0.749	0.857	0.499
	bus	1200	210	0.688	0.871	0.802	0.548
	truck	1200	47	0.311	0.489	0.337	0.225
495							
	all	1200	2227	0.749	0.791	0.782	0.506
	car	1200	1950	0.757	0.835	0.758	0.45
	bus	1200	55	0.78	0.873	0.901	0.681
	truck	1200	222	0.711	0.667	0.689	0.386
410							
	all	1200	2331	0.595	0.824	0.698	0.454
	car	1200	2005	0.829	0.76	0.797	0.491
	bus	1200	15	0.203	0.932	0.467	0.371
	truck	1200	311	0.752	0.781	0.828	0.499
511							
	all	1200	2294	0.764	0.741	0.781	0.526
	car	1200	2077	0.915	0.747	0.874	0.525
	bus	1200	86	0.844	0.907	0.908	0.691
	truck	1200	131	0.534	0.568	0.562	0.362
398							
	all	1200	2353	0.765	0.715	0.75	0.497
	car	1200	2056	0.754	0.729	0.737	0.444
	bus	1200	104	0.89	0.769	0.83	0.635
	truck	1200	193	0.651	0.648	0.683	0.414
173							
	all	1200	1991	0.835	0.819	0.872	0.571
	car	1200	1680	0.941	0.796	0.92	0.585
	bus	1200	171	0.828	0.883	0.883	0.633
	truck	1200	140	0.736	0.779	0.813	0.495
ALL							
	all	1200	12957	0.726	0.781	0.776	0.499
	car	1200	11272	0.821	0.779	0.807	0.489
	bus	1200	641	0.728	0.85	0.809	0.584
	truck	1200	1044	0.629	0.713	0.711	0.425

分析討論:

原本沒有使用 positive weight 時，從 mAP@0.5 準確率只有 0.76，而使用 positive weight 後，可以看到 mAP@0.5 準確率提升到 0.776，可能原因是使用 positive weight 可以調整正類樣本和負類樣本的權重，使得兩者在訓練過程中受到的重視更加平衡。這有助於防止模型傾向於只學習負類樣本的特徵，從而提升對正類樣本的識

別能力，總的來說，positive weight 抽樣可以幫助模型更好地應對正類樣本不足的問題，提高對少數類的識別性能，進而提升整體的準確率。然而，正確的 positive weight 的選擇仍然取決於具體的任務和數據集。