Computer Vision

HW4 Handling Domain Shift

學號:112061619

姓名:王證皓

1. Q1

(1) Example

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.474	0.545	0.499	0.302
	car	1200	1504	0.854	0.684	0.772	0.406
	bus	1200	210	0.517	0.484	0.542	0.37
	truck	1200	47	0.052	0.468	0.185	0.13
495							
	all	1200	2227	0.634	0.649	0.64	0.356
	car	1200	1950	0.71	0.654	0.682	0.352
	bus	1200	55	0.849	0.614	0.777	0.508
	truck	1200	222	0.344	0.68	0.46	0.207
410							
	all	1200	2331	0.496	0.658	0.54	0.313
	car	1200	2005	0.853	0.609	0.735	0.426
	bus	1200	15	0.197	0.591	0.236	0.181
	truck	1200	311	0.438	0.775	0.648	0.332
511							
	all	1200	2294	0.456	0.587	0.498	0.31
	car	1200	2077	0.875	0.509	0.707	0.387
	bus	1200	86	0.358	0.709	0.466	0.346
	truck	1200	131	0.134	0.542	0.32	0.196
398							
	all	1200	2353	0.712	0.688	0.71	0.451
	car	1200	2056	0.751	0.713	0.732	0.404
	bus	1200	104	0.86	0.558	0.736	0.498
	truck	1200	193	0.524	0.793	0.662	0.452
173							
	all	1200	1991	0.781	0.734	0.806	0.516
	car	1200	1680	0.942	0.702	0.888	0.549
	bus	1200	171	0.814	0.83	0.867	0.633
	truck	1200	140	0.587	0.671	0.663	0.366
ALL							
	all	1200	12957	0.647	0.585	0.619	0.372
	car	1200	11272	0.834	0.585	0.734	0.409
	bus	1200	641	0.721	0.504	0.603	0.423
	truck	1200	1044	0.384	0.665	0.52	0.285

此處使用的 split_train_val_path 為範例程式，train 佔 train+validation 的比例為 0.9，最後的 mAP@.5 為 0.619，mAP@.5:.95 為 0.372，train 只使用 173 與 398 這兩個 dataset，所以從圖中可看到這兩組的數據明顯較其他組好。

(2) Random Shuffle

Code

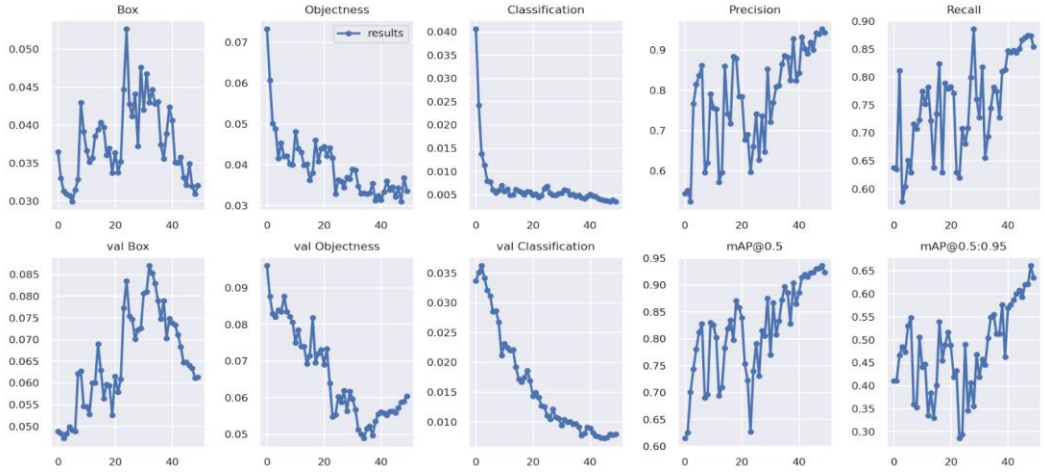
```python
# TODO : split train and val images
def split_train_val_path(all_image_paths, train_val_ratio=0.9):
    """
    :param all_image_paths: all image paths for question in the data folder
    :param train_val_ratio: ratio of image paths used to split training and validation
    :return :
        train_image_paths = ['your_folder/images1.jpg', 'your_folder/images2.jpg', ...]
        val_image_paths = ['your_folder/images3.jpg', 'your_folder/images4.jpg', ...]
    """
    # 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
```

前面的範例程式並無對 dataset 進行優化，所以在這邊我使用 random shuffle 的方法將 dataset 進行隨機排列，將 dataset 內各個 camera 的數量盡量達到平均，來提升訓練效能。

Performance

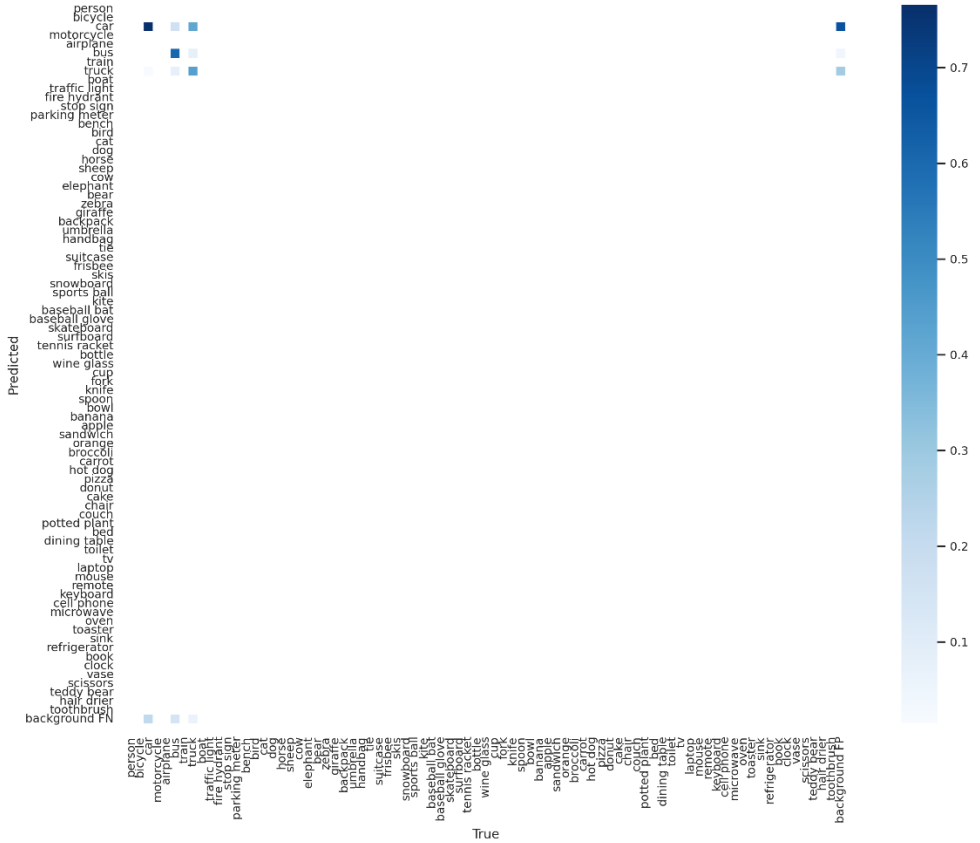
Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.512	0.493	0.476	0.293
	car	1200	1504	0.892	0.622	0.779	0.439
	bus	1200	210	0.601	0.517	0.588	0.399
	truck	1200	47	0.044	0.34	0.061	0.041
495							
	all	1200	2227	0.658	0.649	0.649	0.382
	car	1200	1950	0.742	0.633	0.69	0.387
	bus	1200	55	0.84	0.673	0.752	0.507
	truck	1200	222	0.393	0.64	0.505	0.251
410							
	all	1200	2331	0.554	0.611	0.554	0.345
	car	1200	2005	0.889	0.566	0.754	0.458
	bus	1200	15	0.196	0.533	0.252	0.201
	truck	1200	311	0.577	0.733	0.655	0.375
511							
	all	1200	2294	0.514	0.519	0.513	0.327
	car	1200	2077	0.901	0.475	0.732	0.406
	bus	1200	86	0.494	0.58	0.511	0.367
	truck	1200	131	0.147	0.504	0.296	0.208
398							
	all	1200	2353	0.758	0.688	0.741	0.499
	car	1200	2056	0.743	0.715	0.746	0.452
	bus	1200	104	0.936	0.562	0.765	0.55
	truck	1200	193	0.596	0.787	0.713	0.496
173							
	all	1200	1991	0.779	0.74	0.802	0.529
	car	1200	1680	0.953	0.684	0.896	0.56
	bus	1200	171	0.846	0.877	0.872	0.642
	truck	1200	140	0.538	0.657	0.638	0.385
ALL							
	all	1200	12957	0.652	0.595	0.638	0.402
	car	1200	11272	0.829	0.593	0.74	0.435
	bus	1200	641	0.728	0.53	0.631	0.45
	truck	1200	1044	0.398	0.661	0.543	0.322

Result



從以上兩張圖可觀察到 mAP@.5:.95 從原本的 0.372 提升到了 0.402，代表著 Random Shuffle 可提升訓練效能。

Confusion Matrix



從 Confusion Matrix 中，可以看到 True 為 Car,Bus,Train 大部分都有成功的辨別，尤其以 Car 的顏色為最深，代表其表現得最好，從 Performance 也可觀察到確實如此，另外，從右上角也可看到當 True 為 background FP，錯誤辨別為 car 的機率最高，左下角則可看出 FN 的大小，也是有少許出現錯誤。

2. Q2

(1) Random Sample

Code

```python
# Q2 and Q3 TODO : select "better" images from Q2 folder
def select_imaegs(image_paths, images_num=200):
    """
    :param image_paths: --> ['your_folder/images1.jpg', 'your_folder/images2.jpg', ...]
    :param images_num: choose the number of images
    :return :
        selected_image_paths = ['your_folder/images10.jpg', 'your_folder/images12.jpg', ...]
    """
    # TODO : select images
    random.shuffle(image_paths)
    #selected_image_paths=image_paths[:images_num]
    return selected_image_paths
```

Performance

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.596	0.642	0.609	0.371
	car	1200	1504	0.838	0.833	0.878	0.487
	bus	1200	210	0.696	0.794	0.754	0.487
	truck	1200	47	0.254	0.298	0.196	0.14
495							
	all	1200	2227	0.756	0.695	0.719	0.44
	car	1200	1950	0.705	0.862	0.797	0.451
	bus	1200	55	0.939	0.835	0.889	0.575
	truck	1200	222	0.623	0.387	0.469	0.294
410							
	all	1200	2331	0.747	0.676	0.756	0.484
	car	1200	2005	0.897	0.692	0.823	0.471
	bus	1200	15	0.457	0.733	0.624	0.493
	truck	1200	311	0.886	0.601	0.82	0.489
511							
	all	1200	2294	0.765	0.659	0.716	0.462
	car	1200	2077	0.885	0.784	0.868	0.471
	bus	1200	86	0.846	0.895	0.917	0.683
	truck	1200	131	0.564	0.298	0.362	0.232
398							
	all	1200	2353	0.779	0.708	0.756	0.485
	car	1200	2056	0.742	0.783	0.778	0.448
	bus	1200	104	0.89	0.625	0.747	0.527
	truck	1200	193	0.706	0.715	0.743	0.48
173							
	all	1200	1991	0.787	0.813	0.844	0.548
	car	1200	1680	0.853	0.872	0.914	0.559
	bus	1200	171	0.835	0.801	0.846	0.556
	truck	1200	140	0.672	0.764	0.77	0.529
ALL							
	all	1200	12957	0.742	0.72	0.744	0.461
	car	1200	11272	0.793	0.813	0.826	0.47
	bus	1200	641	0.763	0.757	0.777	0.521
	truck	1200	1044	0.671	0.591	0.629	0.394

(2) Proposed Method

Code

```python
# Q2 and Q3 TODO : select "better" images from Q2 folder
def select_imaegs(image_paths, images_num=200):
    """
    :param image_paths: --> ['your_folder/images1.jpg', 'your_folder/images2.jpg', ...]
    :param images_num: choose the number of images
    :return :
        selected_image_paths = ['your_folder/images10.jpg', 'your_folder/images12.jpg', ...]
    """
    # TODO : select images
    image170 = [path for path in image_paths if os.path.basename(path).startswith('170')]
    image173 = [path for path in image_paths if os.path.basename(path).startswith('173')]
    image398 = [path for path in image_paths if os.path.basename(path).startswith('398')]
    image410 = [path for path in image_paths if os.path.basename(path).startswith('410')]
    image495 = [path for path in image_paths if os.path.basename(path).startswith('495')]
    image511 = [path for path in image_paths if os.path.basename(path).startswith('511')]
    random.shuffle(image170)
    random.shuffle(image173)
    random.shuffle(image398)
    random.shuffle(image410)
    random.shuffle(image495)
    random.shuffle(image511)
    selected_image_paths=image170[:28]+image173[:38]+image398[:34]+image410[:34]+image495[:33]+image511[:33]
    #random.shuffle(image_paths)
    #selected_image_paths=image_paths[:images_num]
    return selected_image_paths
```

Select_images 的函數我使用的改良版本如上圖，方法為先將 camera 做分類，再用 ramdom 的方式將每個 camera 做隨機排列，最後依照 Random sample 訓練出來的 mAP@.5 的比例分配每組 camera 的訓練張數，越大的 mAP@.5 數值能得到越多的張數，反之亦然，分配的公式為先將六組 mAP@.5 加總等於 4.4，取平均等於 0.733，接著因 4.4 放大 45.45 倍為 200 所以再將個別 mAP@.5 乘以 45.45 得到最後張數。

Performance

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.793	0.658	0.665	0.443
	car	1200	1504	0.873	0.829	0.885	0.524
	bus	1200	210	0.707	0.89	0.786	0.576
	truck	1200	47	0.799	0.255	0.326	0.229
495							
	all	1200	2227	0.71	0.778	0.762	0.491
	car	1200	1950	0.672	0.887	0.792	0.454
	bus	1200	55	0.857	0.871	0.924	0.696
	truck	1200	222	0.6	0.577	0.569	0.322
410							
	all	1200	2331	0.705	0.709	0.803	0.542
	car	1200	2005	0.877	0.674	0.817	0.494
	bus	1200	15	0.385	0.8	0.777	0.645
	truck	1200	311	0.853	0.653	0.817	0.487
511							
	all	1200	2294	0.802	0.674	0.741	0.508
	car	1200	2077	0.874	0.828	0.894	0.517
	bus	1200	86	0.814	0.918	0.952	0.77
	truck	1200	131	0.717	0.275	0.377	0.236
398							
	all	1200	2353	0.765	0.74	0.792	0.531
	car	1200	2056	0.73	0.802	0.781	0.459
	bus	1200	104	0.945	0.673	0.851	0.614
	truck	1200	193	0.62	0.746	0.744	0.52
173							
	all	1200	1991	0.798	0.808	0.867	0.583
	car	1200	1680	0.882	0.866	0.928	0.603
	bus	1200	171	0.871	0.831	0.902	0.659
	truck	1200	140	0.641	0.729	0.772	0.487
ALL							
	all	1200	12957	0.759	0.739	0.774	0.508
	car	1200	11272	0.808	0.808	0.834	0.497
	bus	1200	641	0.785	0.808	0.838	0.623
	truck	1200	1044	0.685	0.601	0.651	0.404

從 random 與我使用的改良方法的 performance 皆能看到 mAP@.5:.95 都大於 Q1 的效能，各別的 camera 中的數值皆較平均且更好，而我所使用的方法也較 random 表現更好，從原本的 0.461 上升到 0.508，各別的 camera 也表現得更好。

3. Q3

(1) Pseudo labeling(Conf=0.9)

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.566	0.518	0.532	0.322
	car	1200	1504	0.783	0.554	0.665	0.34
	bus	1200	210	0.785	0.659	0.731	0.485
	truck	1200	47	0.129	0.34	0.201	0.141
495							
	all	1200	2227	0.707	0.527	0.605	0.39
	car	1200	1950	0.771	0.481	0.624	0.342
	bus	1200	55	0.765	0.764	0.8	0.614
	truck	1200	222	0.584	0.338	0.39	0.214
410							
	all	1200	2331	0.496	0.72	0.638	0.436
	car	1200	2005	0.779	0.635	0.692	0.406
	bus	1200	15	0.09	0.867	0.561	0.494
	truck	1200	311	0.618	0.659	0.662	0.409
511							
	all	1200	2294	0.535	0.634	0.606	0.375
	car	1200	2077	0.769	0.607	0.711	0.364
	bus	1200	86	0.546	0.814	0.751	0.547
	truck	1200	131	0.291	0.48	0.355	0.214
398							
	all	1200	2353	0.625	0.514	0.561	0.353
	car	1200	2056	0.763	0.51	0.604	0.345
	bus	1200	104	0.669	0.721	0.725	0.485
	truck	1200	193	0.444	0.311	0.353	0.23
173							
	all	1200	1991	0.658	0.645	0.672	0.417
	car	1200	1680	0.822	0.711	0.821	0.471
	bus	1200	171	0.642	0.754	0.728	0.487
	truck	1200	140	0.511	0.471	0.467	0.292
ALL							
	all	1200	12957	0.602	0.58	0.584	0.362
	car	1200	11272	0.786	0.582	0.677	0.375
	bus	1200	641	0.553	0.685	0.635	0.443
	truck	1200	1044	0.467	0.474	0.44	0.268

(2) Pseudo labeling(Conf=0.5)

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.617	0.498	0.555	0.348
	car	1200	1504	0.808	0.583	0.701	0.384
	bus	1200	210	0.801	0.633	0.755	0.51
	truck	1200	47	0.243	0.277	0.209	0.149
495							
	all	1200	2227	0.661	0.598	0.619	0.409
	car	1200	1950	0.751	0.581	0.646	0.369
	bus	1200	55	0.691	0.853	0.824	0.643
	truck	1200	222	0.543	0.36	0.388	0.215
410							
	all	1200	2331	0.469	0.778	0.661	0.457
	car	1200	2005	0.711	0.691	0.704	0.426
	bus	1200	15	0.093	0.933	0.596	0.519
	truck	1200	311	0.604	0.711	0.683	0.426
511							
	all	1200	2294	0.579	0.582	0.628	0.4
	car	1200	2077	0.789	0.61	0.732	0.392
	bus	1200	86	0.556	0.686	0.761	0.56
	truck	1200	131	0.392	0.45	0.392	0.248
398							
	all	1200	2353	0.595	0.553	0.589	0.381
	car	1200	2056	0.715	0.593	0.632	0.378
	bus	1200	104	0.603	0.75	0.76	0.519
	truck	1200	193	0.465	0.316	0.374	0.245
173							
	all	1200	1991	0.674	0.684	0.701	0.453
	car	1200	1680	0.802	0.777	0.845	0.508
	bus	1200	171	0.638	0.784	0.767	0.535
	truck	1200	140	0.582	0.493	0.491	0.317
ALL							
	all	1200	12957	0.594	0.62	0.614	0.391
	car	1200	11272	0.743	0.648	0.702	0.405
	bus	1200	641	0.549	0.733	0.683	0.484
	truck	1200	1044	0.49	0.479	0.457	0.283

上面兩張圖中的效能為使用 Q2 所訓練出來的 weights 將 Q3 做 pseudo labeling，再透過 Q3 來訓練，第一張圖 confidence 為 0.9，第二張圖 confidence 為 0.5，從此可觀察出 0.5 的表現較好，0.9 對於 label 來說有點太高，因此降為 0.5 能達到更好的效能，mAP@.5:.95 從 0.362 上升到 0.391。

(3) Pseudo labeling(Conf=0.5)+Focal loss

Camera	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
170							
	all	1200	1761	0.664	0.608	0.613	0.375
	car	1200	1504	0.873	0.808	0.866	0.496
	bus	1200	210	0.687	0.805	0.753	0.469
	truck	1200	47	0.433	0.213	0.221	0.16
495							
	all	1200	2227	0.646	0.669	0.637	0.379
	car	1200	1950	0.643	0.859	0.724	0.402
	bus	1200	55	0.631	0.782	0.755	0.466
	truck	1200	222	0.664	0.365	0.433	0.268
410							
	all	1200	2331	0.63	0.76	0.729	0.479
	car	1200	2005	0.812	0.729	0.767	0.458
	bus	1200	15	0.158	0.933	0.619	0.479
	truck	1200	311	0.919	0.617	0.801	0.501
511							
	all	1200	2294	0.729	0.526	0.656	0.417
	car	1200	2077	0.889	0.649	0.825	0.471
	bus	1200	86	0.645	0.592	0.712	0.492
	truck	1200	131	0.653	0.336	0.43	0.287
398							
	all	1200	2353	0.654	0.661	0.648	0.397
	car	1200	2056	0.615	0.835	0.706	0.394
	bus	1200	104	0.666	0.731	0.745	0.493
	truck	1200	193	0.68	0.418	0.493	0.306
173							
	all	1200	1991	0.822	0.65	0.743	0.46
	car	1200	1680	0.871	0.852	0.904	0.57
	bus	1200	171	0.713	0.684	0.72	0.45
	truck	1200	140	0.882	0.414	0.605	0.36
ALL							
	all	1200	12957	0.685	0.663	0.667	0.408
	car	1200	11272	0.738	0.806	0.776	0.45
	bus	1200	641	0.584	0.722	0.662	0.424
	truck	1200	1044	0.733	0.461	0.563	0.349

前面比較了不同的 confidence 對於效能的影響，0.5 明顯較好所以在此處也使用，並另外使用了 Focal loss 來做配合，mAP@.5:.95 也上升到了 0.408，而在物件偵測中背景的比例較物體的比例高，所以採用 Focal loss 來處理處理不平衡數據，也驗證了 Focal loss 能提高效能的表現。