Report

111061590林學謙

1. Build up split-train-val-function and question1

```python
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
```

直接用 random 所有的 image_paths，並且根據輸入的比例進行 train_val 控制，那我主要在第一題測試的部分，為測試不同比例的 train_val 的比例，並且輸出 test 的結果。

Train_val_ratio = 9:1

ALL							
	all	1200	12957	0.61	0.556	0.592	0.355
	car	1200	11272	0.766	0.563	0.704	0.384
	bus	1200	641	0.73	0.493	0.603	0.401
	truck	1200	1044	0.335	0.612	0.47	0.28

map@.5 = 0.592

Train_val_ratio = 8:2

ALL							
	all	1200	12957	0.556	0.656	0.609	0.374
	car	1200	11272	0.762	0.675	0.726	0.425
	bus	1200	641	0.693	0.531	0.618	0.449
	truck	1200	1044	0.214	0.761	0.483	0.248

map@.5 = 0.609

Train_val_ratio = 7:3

ALL							
	all	1200	12957	0.61	0.556	0.592	0.355
	car	1200	11272	0.766	0.563	0.704	0.384
	bus	1200	641	0.73	0.493	0.603	0.401
	truck	1200	1044	0.335	0.612	0.47	0.28

map@.5 = 0.592

可以得出 8:2 的結果比較好。

2. Select_image_function and Q2

方法 2:平均取，並且將結果當作 train and val 資料

```
elif m == 2:
    random.shuffle(image_paths)
    for path in image_paths:
        if "170" in path:
            if(count[0] < 33):
                selected_image_paths.append(path);
                count[0]+=1;
        elif "511" in path:
            if(count[1] < 33):
                selected_image_paths.append(path);
                count[1]+=1;
        elif "495" in path:
            if(count[2] < 33):
                selected_image_paths.append(path);
                count[2]+=1;
        elif "410" in path:
            if(count[3] < 33):
                selected_image_paths.append(path);
                count[3]+=1;
        elif "398" in path:
            if(count[4] < 34):
                selected_image_paths.append(path);
                count[4]+=1;
        elif "173" in path:
            if(count[5] < 34):
                selected_image_paths.append(path);
                count[5]+=1;
```

方法 3: 選擇圖片擁有較多的 class 作為訓練資料

```
elif m == 3:
    path_with_label_count = defaultdict(set);
    random.shuffle(image_paths);

    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]);

            path_with_label_count[path_jpg].add(class_index);


    sorted_paths_counts = sorted(path_with_label_count.items(), key=get_set_length, reverse=True);
    sorted_paths = [paths for paths, _ in sorted_paths_counts];

    selected_image_paths = sorted_paths[0:images_num];
```

這邊分別做了只有 random，方法 2 與方法 3

只有 random;比例 8:2

ALL							
	all	1200	12957	0.642	0.718	0.698	0.414
	car	1200	11272	0.779	0.792	0.811	0.475
	bus	1200	641	0.627	0.719	0.706	0.441
	truck	1200	1044	0.521	0.642	0.577	0.325

因為可以看出 random 比較差，所以再來使用另外兩個方法，並且透過不同比例做比較。

方法 2;比例 9:1

ALL							
	all	1200	12957	0.741	0.715	0.752	0.467
	car	1200	11272	0.811	0.805	0.832	0.461
	bus	1200	641	0.826	0.721	0.796	0.577
	truck	1200	1044	0.585	0.619	0.627	0.362

方法 2;比例 8:2

ALL							
	all	1200	12957	0.701	0.714	0.737	0.482
	car	1200	11272	0.772	0.8	0.813	0.492
	bus	1200	641	0.81	0.64	0.772	0.551
	truck	1200	1044	0.521	0.701	0.627	0.403

方法 3;比例 9:1

ALL							
	all	1200	12957	0.707	0.727	0.738	0.475
	car	1200	11272	0.786	0.754	0.797	0.446
	bus	1200	641	0.722	0.733	0.762	0.55
	truck	1200	1044	0.613	0.695	0.656	0.428

方法 3;比例 8:2

ALL							
	all	1200	12957	0.706	0.75	0.754	0.485
	car	1200	11272	0.811	0.791	0.826	0.514
	bus	1200	641	0.645	0.764	0.752	0.497
	truck	1200	1044	0.662	0.693	0.684	0.444

可以得出方法三比較好，所以接下來的問題 3，都會使用方法三進行。

3. Select_image_function and Q2

第一個方法，我想先測試 positive weight，所以分別用方法 3 與不同比例的 train_ratio 進行比較，但是只取用 Q2 的 200 資料。

方法 3;比例 8:2

ALL							
	all	1200	12957	0.736	0.683	0.732	0.453
	car	1200	11272	0.865	0.723	0.829	0.505
	bus	1200	641	0.71	0.688	0.74	0.51
	truck	1200	1044	0.632	0.638	0.627	0.342

方法 3;比例 9:1

ALL							
	all	1200	12957	0.741	0.719	0.766	0.483
	car	1200	11272	0.849	0.699	0.81	0.469
	bus	1200	641	0.745	0.782	0.803	0.583
	truck	1200	1044	0.629	0.676	0.686	0.398

可以看出，在 positive weight 的情況下，比例 9:1 擁有較好的結果(0.766)，接下來使用 freeze。

Train_val 比例為 9:1，並且只使用 Q2 兩百筆資料

```
1 !python data/pre_process.py --data_folder './data/CityCam' --ques 'Q2' --method 3 --train-val-ratio 0.9
Namespace(data_folder='./data/CityCam', ques='Q2', method=3, train_val_ratio=0.9)
1 !python train.py --project runs/train/Q3 --freeze 5 --name method_3_result_9_1_freeze
```

ALL							
	all	1200	12957	0.716	0.737	0.733	0.478
	car	1200	11272	0.836	0.748	0.82	0.503
	bus	1200	641	0.734	0.764	0.765	0.545
	truck	1200	1044	0.578	0.698	0.615	0.387

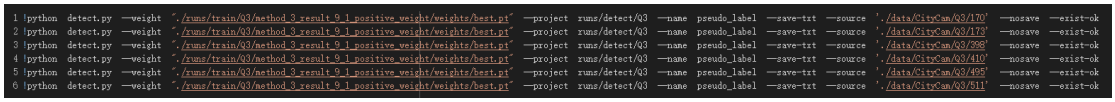
可以看出其結果 0.733，並沒有比 positive weight 來的更好，所以測試其他部分。

使用 focal loss，並且跟前面的方法一樣，得出以下結果

ALL							
	all	1200	12957	0.716	0.685	0.733	0.492
	car	1200	11272	0.792	0.718	0.78	0.48
	bus	1200	641	0.783	0.652	0.759	0.565
	truck	1200	1044	0.572	0.685	0.66	0.43

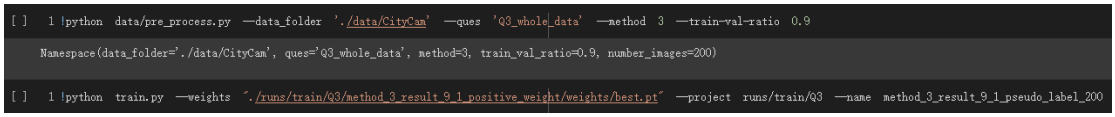
似乎沒有什麼變化，為了讓其可以增加準確度，所以加入 pseudo label，增加訓練資料。

我先使用 detect.py 將資料使用最好的 positvie weight 的結果進行 pseudo label



並且將其下載，並將所有的 Q3 圖片與 pseudo labels 整理成一包，放在 data/CityCam/Q3_whole_data

並且直接將舊的訓練資料，進行訓練，等同於再訓練 1200 張與 200 張(兩個 Case)，並使用 test 看其結果。



使用 1200 筆 Q3 的資料，並且搭配已經訓練過的權重(使用 postitive weight and method3 in 200 data of Q2 dataset)。

all	1200	12957	0.72	0.655	0.727	0.477
car	1200	11272	0.83	0.679	0.801	0.481
bus	1200	641	0.765	0.721	0.778	0.585
truck	1200	1044	0.564	0.565	0.602	0.367

相對於前面的結果，只有 0.727，所以就在思考，因為本身是使用自己訓練的模型，還是會有準確度上的差異，可能因為太多資料餵入，導致其結果開始變差，所以減少資料量，或許可以提升準確度，所以將 1200 筆資料改成 200 筆資料，並且與前面一樣，使用已經訓練過的權重，並且搭配 positive weight

ALL							
	all	1200	12957	0.734	0.738	0.774	0.505
	car	1200	11272	0.833	0.739	0.817	0.476
	bus	1200	641	0.752	0.8	0.808	0.608
	truck	1200	1044	0.617	0.675	0.697	0.43

可以看到其上升的 0.774，比前面任何一種方法來的更好，所以可以得出，pseudo label 是有用的，但是無法使用大量的資料，因為模型並不完美，可能導致其學習到錯誤的特徵，所以這就是必須做 trade-off 的問題，資料越多，只要 label 正確，通常都可以提高準確度，但是因為實際應用上，無法得到相當準確的 label，所以新的資料無法完全進行訓練。

檔案只會留下每一個問題最好的答案，因為發現太多，壓不進去 100MB，就算將 weight 和.pt file, data 刪除也無濟於事，如果助教對於內容有任何疑問，都可以寄信到 chicco881204@gmail.com