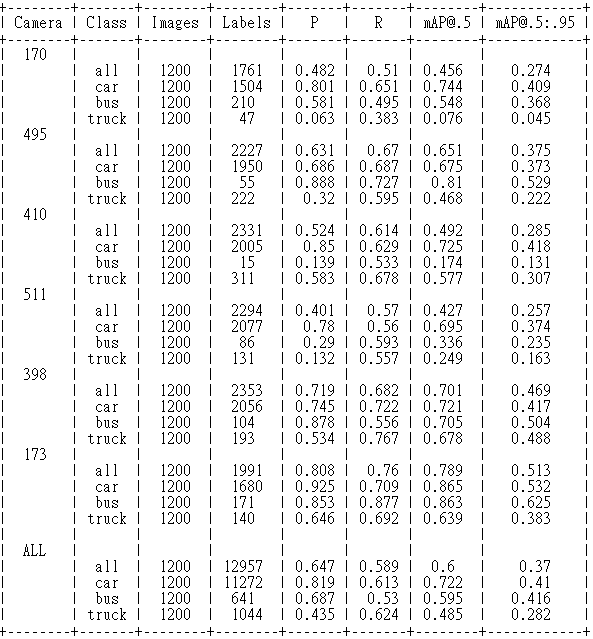
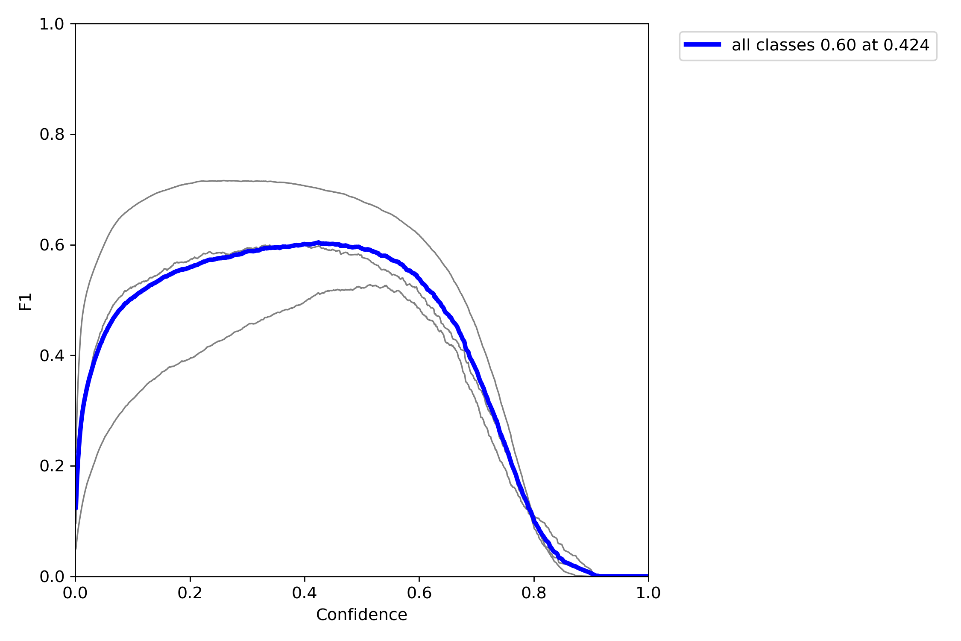
CV\_HW4 Handling Domain Shift 107081028 常安彥

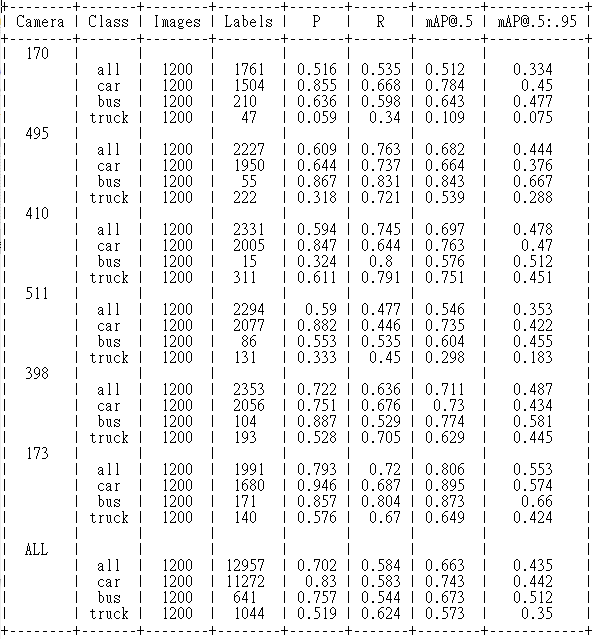
**Q1. Train Valid Split**

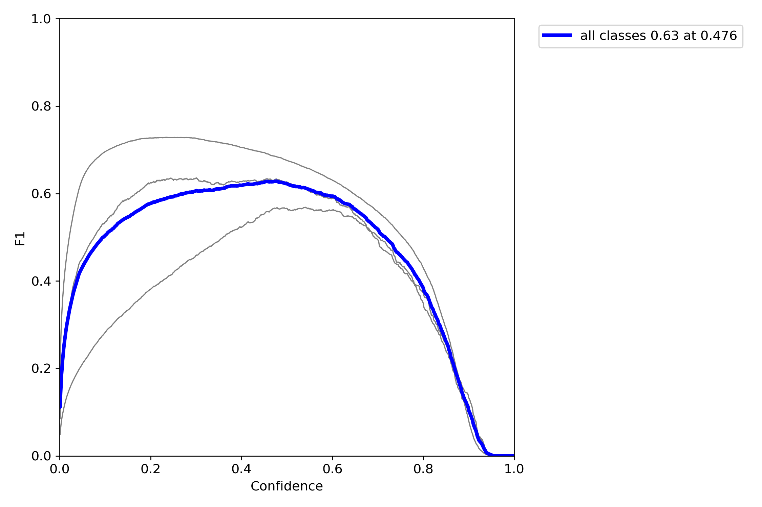
1. Original implement: mAP = 0.6, F1 = 0.6

Simply split data according to the index, which means training set contains 40 images of camera-173 and 140 images of camera-398, and valid set is composed of 20 images of camera-398. **Validation was based on camera-398.**

1. Separate Respectively: mAP = 0.66, F1 = 0.63

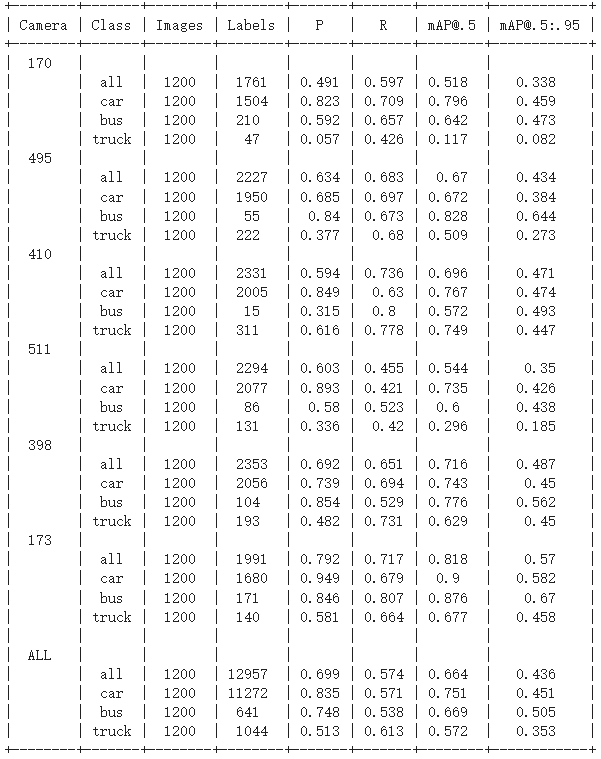
First, I separate into training and valid data on both camera-173 and 398. Therefore, training set contains 36 images of c-173 and 144 images of c-398, while valid set contains 4 images of c-173 and 16 images of c-398. In this case, validation was based on both cameras.

According to the result below, the mAP of camera-173 increased. Furthermore, the performance of the whole dataset significantly increased. Although I assumed the recall of camera-173 would have been better, it didn’t. Based on the high performance of the ‘car’ class with other classes performing badly, I thought that this issue could be dealt as label imbalance in following questions.

 F1 score: 0.63

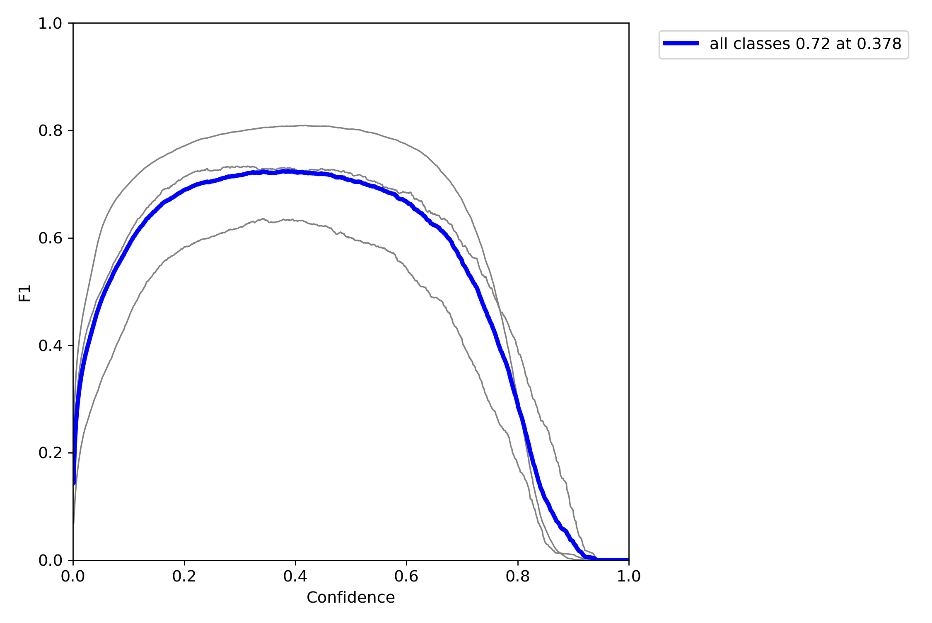
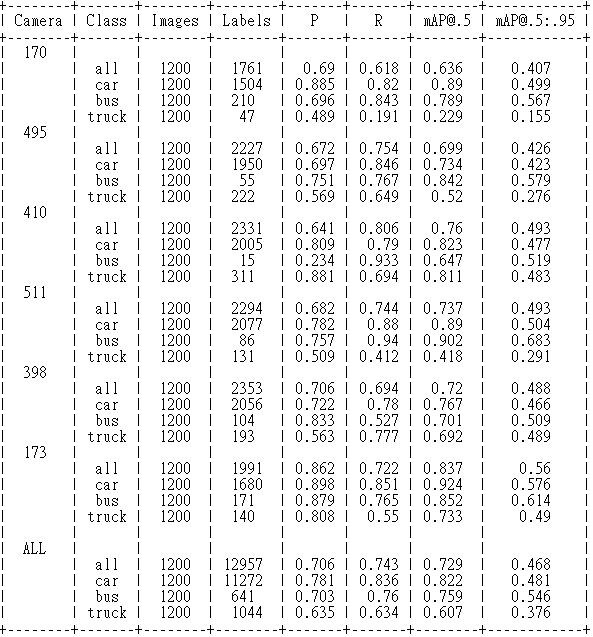
1. Camera Balance: mAP = 0.66, F1 = 0.61

Next, I tried to balance the data size of each camera in order to balance the weights and importance of images of each camera. Therefore, I took the same amount of data from camera-173 and camera –398, both 36 as training and 4 as validation.

The result of camera-173 increased a little, but the overall performance did not go well. Since the size of the data shrinks, the performance went bad.

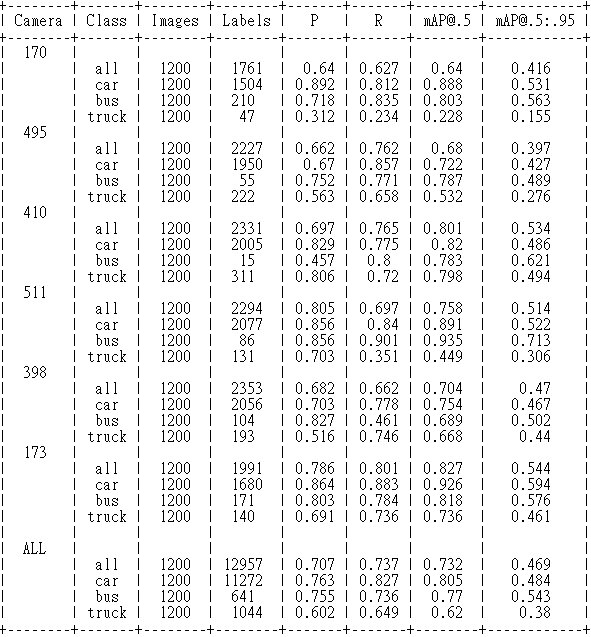
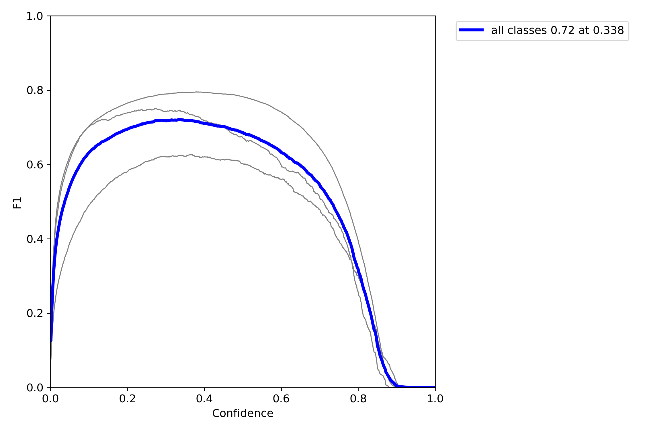
**Q2. Select Images**

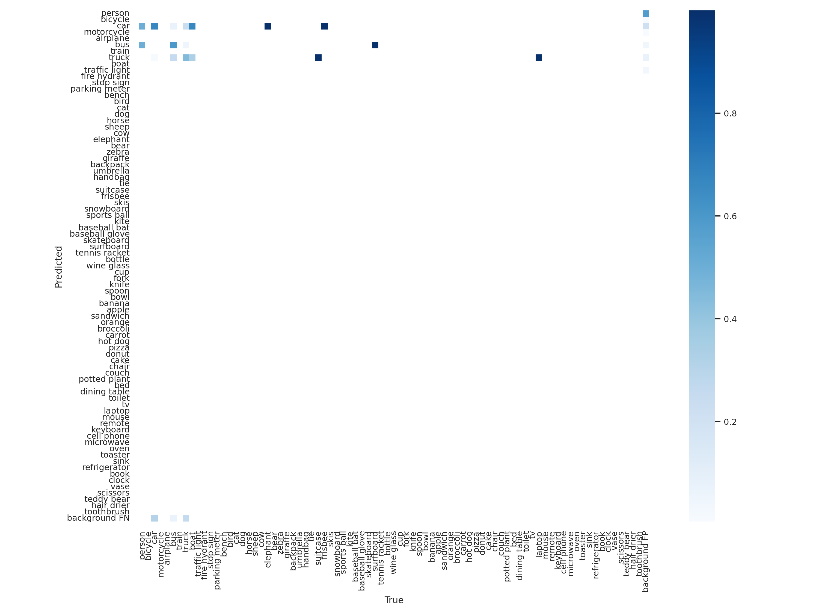
1. Random Sampling: mAP = 0.729, F1 = 0.72

Random Sample from the whole dataset which is composed of images from 6 different cameras. The result is much better than Q1, because the training data contains images from all cameras, which provide images for testing.

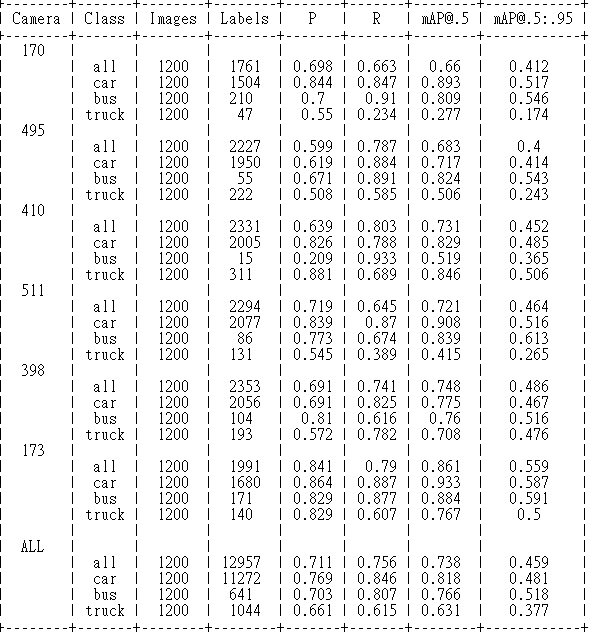
1. Elimination of small bounding boxes & little information: mAP = 0.732, F1= 0.72

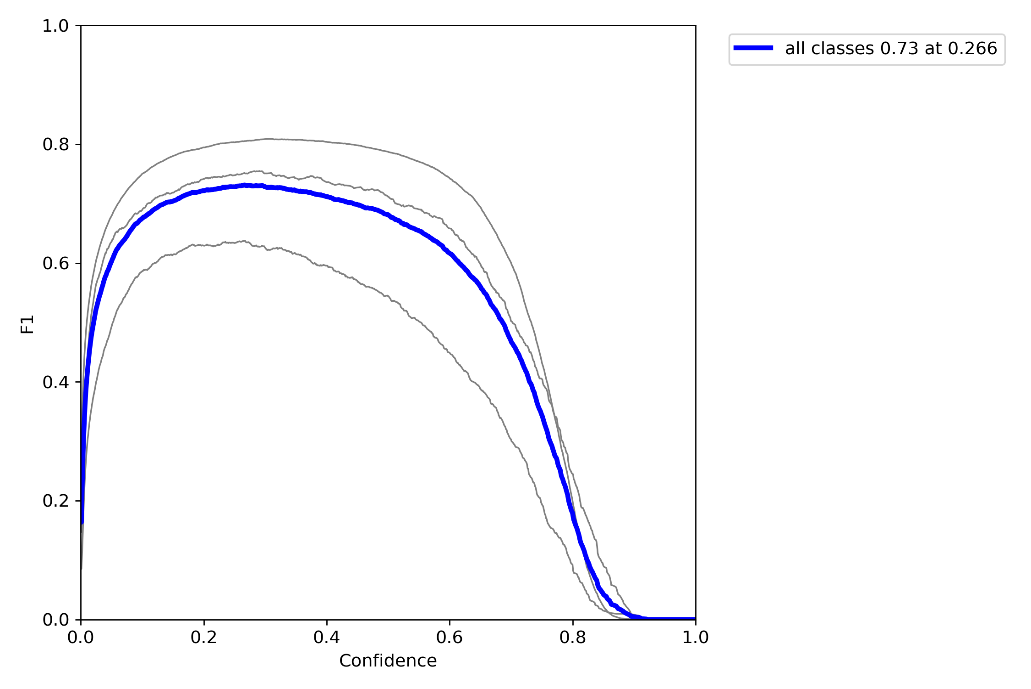
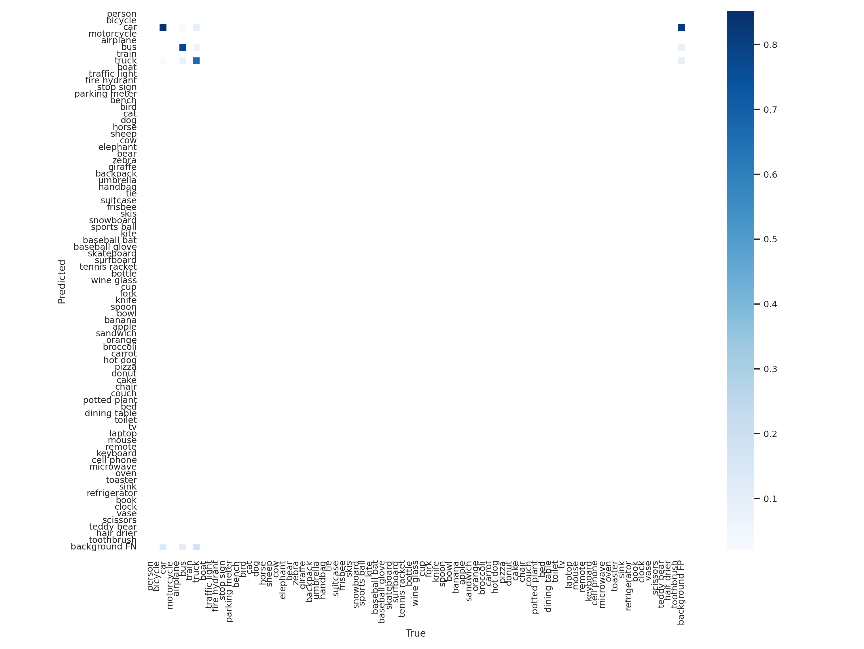
First of all, I figured out that there were some small bounding boxes in several images, which might be too small to provide enough information for the model to learn. So, I analyzed the area of all bounding boxes and calculated the standard deviation. Based on the analyzing, I set a threshold, 0.004, and accumulated the number of labels of each image that had a bounding box with area larger than the threshold.

Images with more big enough boxes had the higher priority to be selected, which I assumed as images providing more useful information. Then, based on the sorted image paths, split into training and valid set with the method implemented on Q1.

According to the overall result above, the performance only increased a little compared with random sampling. Then, based on the confusion matrix below, a huge proportion of other labels were mis-predicted into the three major labels, car, bus and truck, especially car. So, I assumed the issue as data imbalance.

1. Emphasize ‘Bus”: mAP = 0.738, F1 = 0.73

After implementing emphasis on different labels, I figured out that emphasizing the class, ‘bus,’ may help a lot on classifying. Therefore, I modified the sorting method, and made images with ‘bus’ labels higher priority. More images with bus on them would be included to the training and valid data.

According to the result, the performance was the best so far, and according to the confusion matrix, the misprediction issue turned well.

1. Conclusion

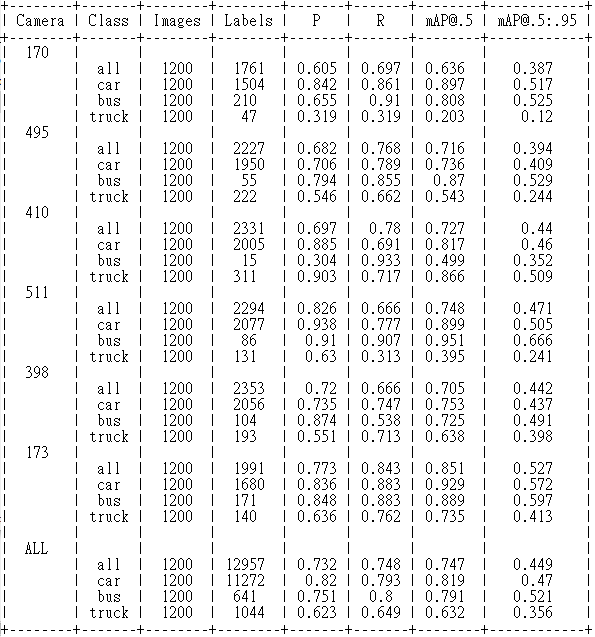
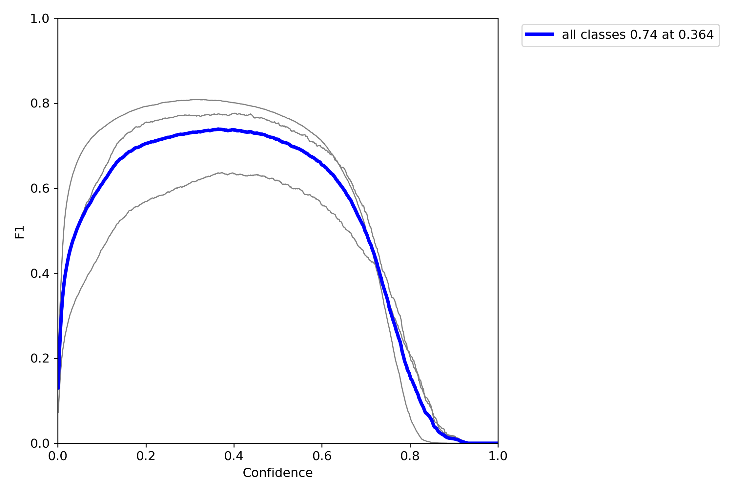
|  |  |  |  |
| --- | --- | --- | --- |
|  | Random | Elimination | Bus |
| mAP | 0.729 | 0.732 | 0.738 |
| F1 | 0.72 | 0.72 | 0.73 |
| Precision | 0.706 | 0.707 | 0.711 |
| Recall | 0.743 | 0.737 | 0.756 |

The precision of simply elimination method was the highest because it was the dataset composed of the most labels. On the other hand, the bus emphasizing method outperformed on the other scores.

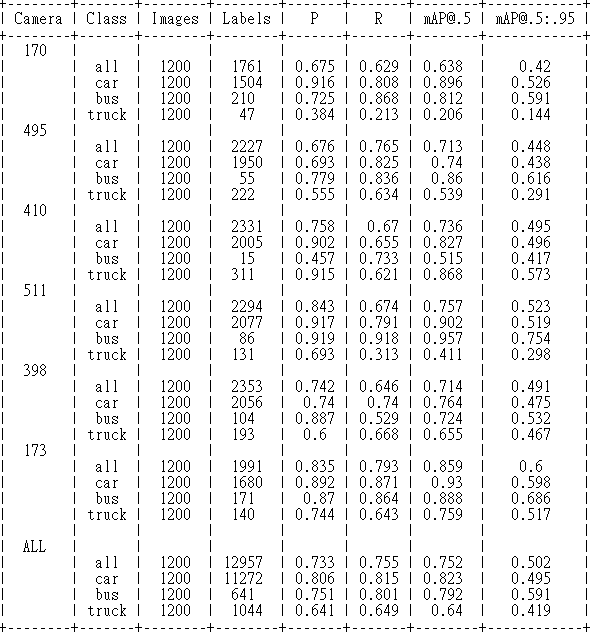
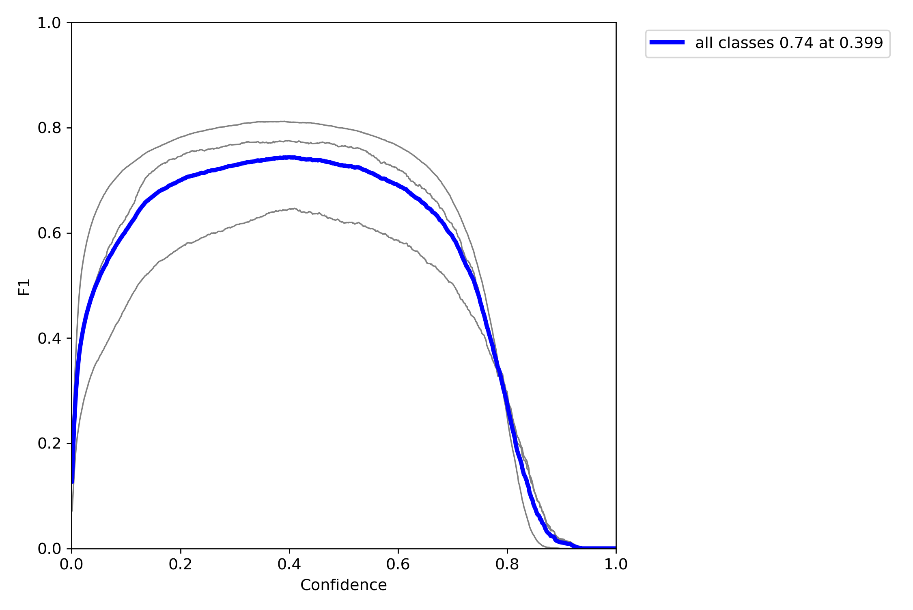
**Q3. Unlabeled Images**

1. Pseudo label: mAP = 0.747, F1 = 0.73

First, I predicted the labels of the 1200 unlabeled images of Q3. With the ‘detect’ python file and the best model trained on Q2, I created the txt labels and boxes of the whole Q3 dataset. Then, I trained the Q3 model based on the 200 selected images and these 1200 Q3 images.

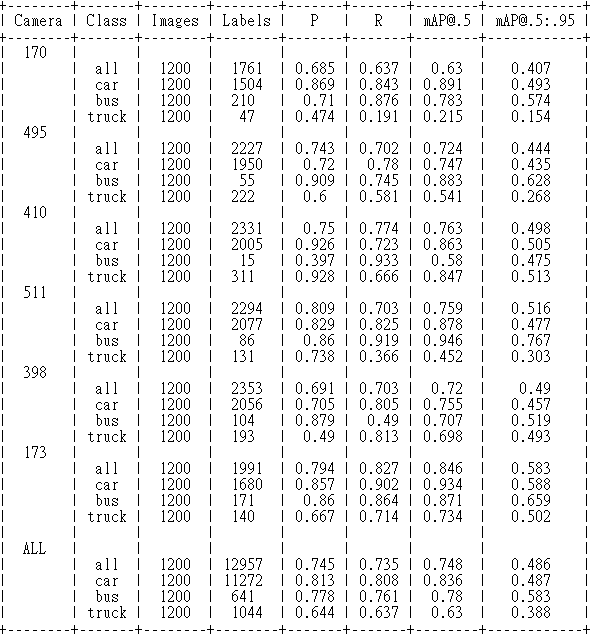
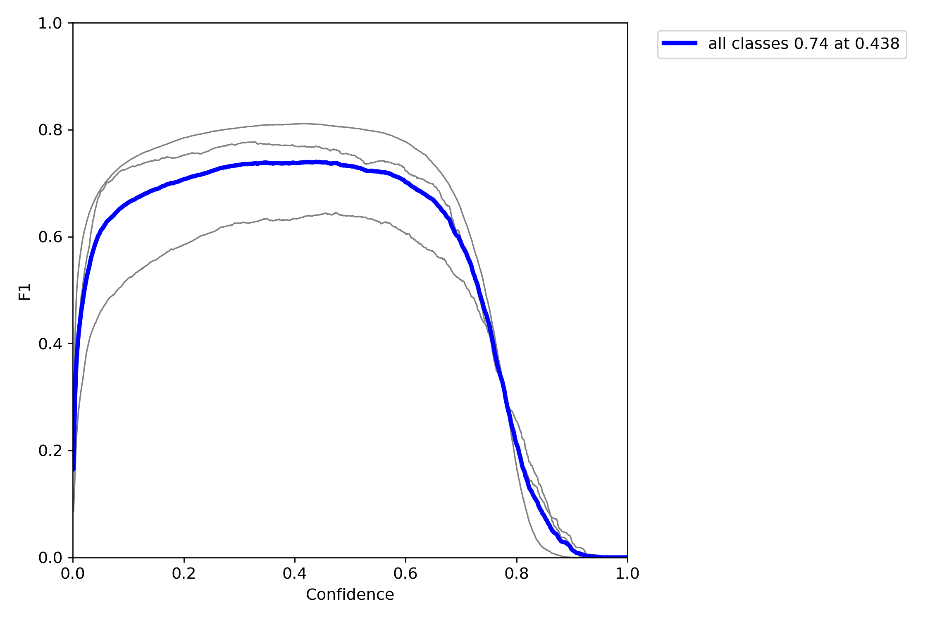
According to the result below, the performance significantly increased, because the dataset augmented a lot. Semi-supervise learning can gather dataset with less cost and enhance the robustness of the model.

1. Transfer Learning: mAP = 0.752, F1=0.74

Next, I implemented transfer learning based on the weight trained on Q2. In this case, I could straightly enhance the robustness of the Q2 best model with these Q3 images, and more quickly train a better model.

1. Freeze Backbone: mAP = 0.748, F1=0.74

Finally, I implemented backbone freezing when transfer learning, and tried freezing 1, 2 and 3 layers. When implementing transfer learning, backbone freezing could decrease the amount of computation and time consumption.

According to the result, although the training time did decrease, the model couldn’t outperform the previous one.