Data Question

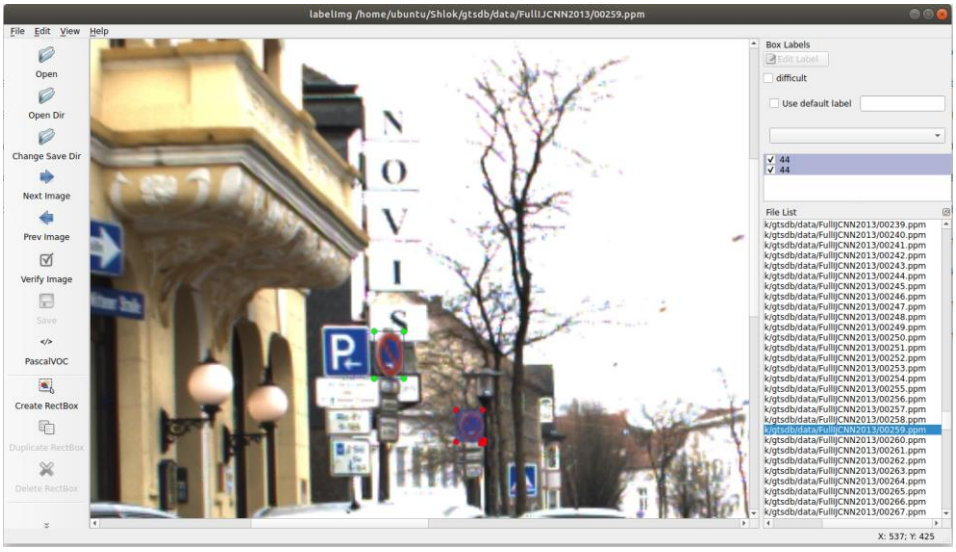
Dharmashloka Debashis (ddebashis3)

Question 1:

I first parsed the GTSDB.json file to identify the images for which the model predicts “pn”. The following are the predictions.

```
00259.ppm [533, 405, 14, 16]
00259.ppm [485, 358, 15, 28]
00261.ppm [1162, 310, 34, 39]
00274.ppm [378, 361, 22, 30]
00304.ppm [971, 432, 16, 18]
00544.ppm [920, 453, 12, 15]
00545.ppm [363, 439, 21, 22]
00555.ppm [812, 492, 14, 14]
00575.ppm [649, 412, 16, 18]
00673.ppm [311, 428, 22, 24]
00697.ppm [789, 423, 16, 16]
00777.ppm [935, 475, 25, 34]
00813.ppm [1025, 231, 47, 57]
00841.ppm [890, 392, 17, 19]
```

I used the open-source labelImg tool [1] to annotate these images and save the annotations in .xml format. A screenshot of the tool usage is shown below.



I annotated signs in the images that are detected by the yolov3 model. I tried to annotate all the signs in these images, rather than just the ones detected by the model. For example, in image 00544.ppm, I annotated two signs whereas there was

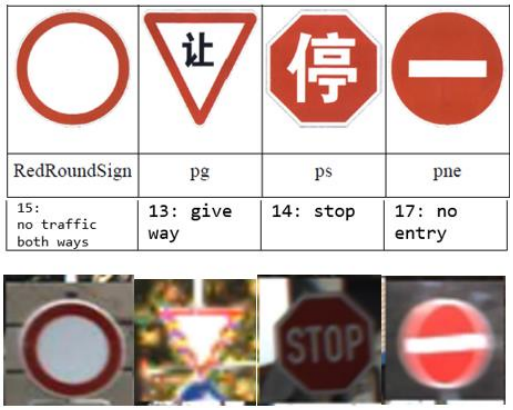
only one detection. To further improve the quality of annotations, the following can be done:

1. Decrease the confidence threshold of the model to improve the recall so that there would be more detections, albeit at the cost of precision. Then manually look at all images/detections and discard the ones which are false positives.
2. Retrain the model using the new data with manual annotations, then do another round of detection followed by manual annotations using detected images.

The figure below is taken from the GTSDB paper [2]. The images are in the order of the class names.

Fig. 3. Traffic sign classes

Using the images above, we can map the detections as follows, to start with. The newly added 43 and 44 class are mapped to “pn”. We can change the mappings with further analysis in part 2.



Question 2:

For analyzing the performance of the model, I computed precision and recall values. mAP cannot be computed since we only have predictions already thresholded at 0.5. So, cannot get the full precision-recall curve.

To calculate the precision/recall values, I consider a hit when the overlap (IoU) between the gt and prediction bbox is more than 0.5 (IoU threshold can be adjusted in a config file). The results are dumped in a csv file.

The script also visualizes the false positives and false negatives per class by cropping the detections/gt from the original image and presenting all of them together per class.

Mapping-1:

I first start with the mapping described in Question-1. That is, the following mapping:

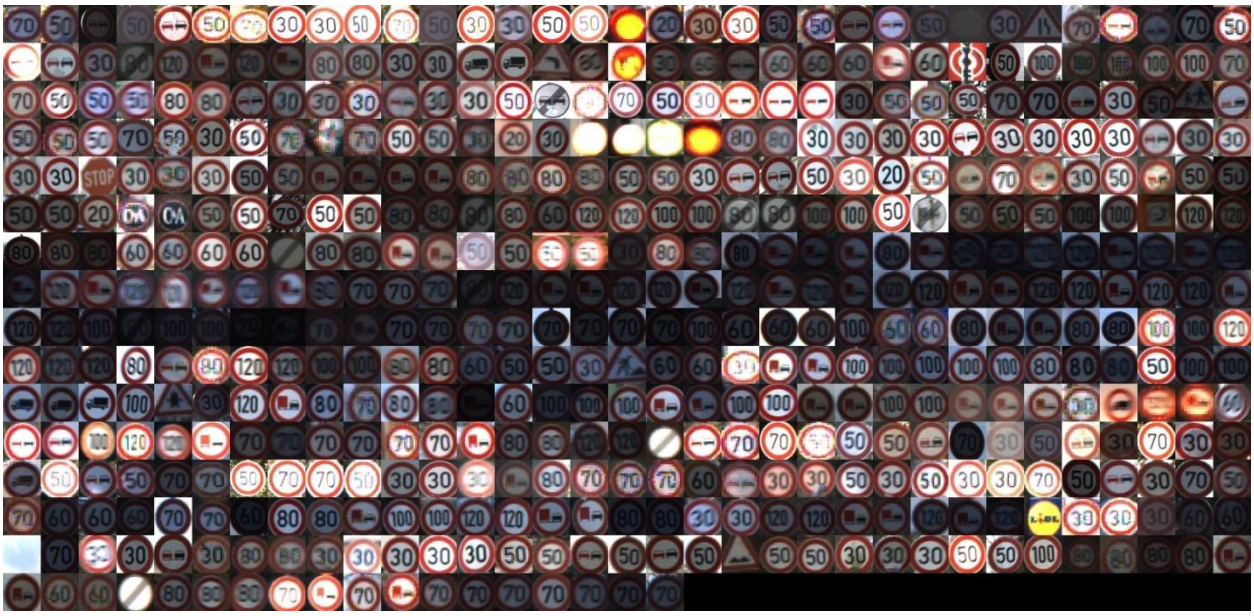
```
CLASS_MAPPING: {
    15: "RedRoundSign",
    13: "pg",
    14: "ps",
    17: "pne",
    43: "pn",
    44: "pn",
}
```

Results:

Class	Precision	Recall
RedRoundSign	2.471483	86.667
pg	98.46154	77.108
pn	100	93.333
pne	75.75758	86.207
ps	85.71429	37.5

For the newly added class “pn”, we see that the precision is 100%, which is expected since we looked at the images to annotate which were predicted by the model. However the recall is ~93%. This is because on one of the images, I had annotated an extra bbox, which was not predicted. So, out of 15 total gt annotations, 14 were already there. 14/15~0.93. To calculations are consistent.

Next, we see that the precision for the class “RedRoundSign” is very low. To investigate the low precision, we need to look at the **false positives**. The figure below shows the false positives from the model. We observe that the speed signs are identified by the model as “RedRoundSign”. This makes sense as they are red and round, and the model must not have seen them as negative examples. So, for the next round, we map all the speed signs to “RedRoundSign “. We also map the “no trucks” sign to “RedRoundSign”. Let’s see if the precision improves without sacrificing recall.



Mapping-2:

For the next round, we map all the speed signs to “RedRoundSign “. We also map the “no trucks” sign to “RedRoundSign”. Let’s see if the precision improves without sacrificing recall.

Class	Precision	Recall
RedRoundSign	95.05703	86.80556
pg	98.46154	77.10843
pn	100	93.33333
pne	75.75758	86.2069
ps	85.71429	37.5

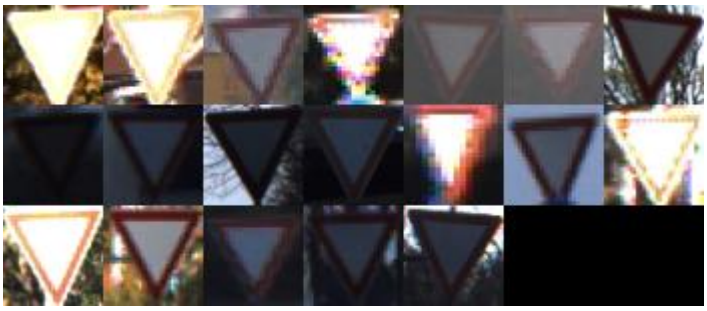
We see that the precision very high now, and the recall is almost the same. So, this is a good measure of the performance of the model. If we want to distinguish between simple RedRoundSign and speed signs, we will need to train the model with speed signs. The figure below shows the **false positives** in the new mapping.

We also see that the recall for “ps” is quite low at 37.5%. To analyze the low recall, we need to look at the **false negatives**. The figure below shows this.

Since the “ps” that the model is trained on is in Chinese, it is difficult to generalize it to “STOP” in English. So, finetuning the model using more stop signs in English will improve the recall.

The figure below shows the false positives for “pne” (precision = ~75.8%). The false positives are understandable since the stop sign and the “pne” sign look very similar with solid reds. Adding these data to finetune the model should improve results.

The figure below shows false negatives for “pg” (recall = ~77.1%). The “pg” sign has something written in Chinese inside the sign. The yield sign does not have anything written inside it. So, adding these data in the training should improve the model.



In general, to improve the model, we can add the images with false positives to the dataset to finetune the model. We need to reserve some data for validation (not use for training). Adding false negatives might not help with a naïve implementation since these portions of the image might not be picked by

the model while training. So, we will need to do hard-negative mining and make sure to include them while training. For the “pn” category, if sufficiently many signs are missed while annotating, they might appear as negative examples while training, confusing the model and it will not learn the “pn” class properly.

To better annotate “pn” in the new dataset to evaluate the performance of the model, we can do the following to make sure we add as much of the annotations as possible. (reiterating Question-1).

1. Decrease the confidence threshold of the model to improve the recall so that there would be more detections, albeit at the cost of precision. Then manually look at all images/detections and discard the ones which are false positives.
2. Retrain the model using the new data with manual annotations, then do another round of detection followed by manual annotations using detected images.

References

1. https://github.com/tzutalin/labelImg
2. Stallkamp, Johannes & Schlipsing, Marc & Salmen, Jan & Igel, Christian. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. Proceedings of the International Joint Conference on Neural Networks. 1453 - 1460. 10.1109/IJCNN.2011.6033395.