CS 783 VISUAL RECOGNITION

Assignment 3

Object Detection

Kritik Soman-18104052
Darshan Ramesh Shet - 16104024

The problem statement for this assignment requires us to perform object detection on the pascal VOC 2007 dataset. Brief description of each of the steps followed in described below.

1. Building the dataset

We created the function read_content() to read xml files and return the bounding box annotations. To only load chair, bottle and aeroplane class images we read the train and test file lists present in 'VOCdevkit/VOC2007/ImageSets/Main/'. The function build_dataset() reads the desired class xml file from 'VOCdevkit/VOC2007/Annotations/' and corresponding image from 'VOCdevkit/VOC2007/JPEGImages/'. It then crops the region in which the object is present and saves in 'data2/val/' and 'data2/train/'. For each image, the function also randomly selects one of a list of predefined bounding boxes and uses it to crop a background patch which has zero IoU with any object present in the image.

2. Training the network

For training the 2 models (1 layer and 2 layer detection), we defined a common function train_model() to train a given model. For loading the data, we used the pytorch function torch.utils.data.DataLoader() with 'data_transforms' to randomly resize, flip and normalize the image. The first model, ie, for layer 1 detection we simply used resnet18 with pretrained weights and only fine-tuned the final residual block and fully connected layer. The fine tuning completed in 20m 52s with the best validation accuracy as: 0.96. We save this model as 'model/resnetFCnLastmodel'. For the second model we simply removed the final residual block from resnet18 and fine-tuned a fully connected layer. We got a validation accuracy of 0.69 and saved this model to 'model/resnet2model'.

3. Testing

We predefined windows with length of shorter side as 32, 64, 128 and 256 pixel and aspect ratio as 1:1, 1:2 and 2:1. We slide this window from left to right and top to bottom with 50% overlap and when the xmax, ymax of the box exceeded the dimensions of the image, we forced the exceeding location to height or width of the image (whichever was more appropriate for the case). For each box we resized the image region to 224 X 224 and classified the region used the model that we trained earlier.

3.1 One Layer Detection

Only the first model was used for classification of each region that the sliding window slid over. Then non maximal suppression (NMS) was applied to all the bounding boxes which had a

prediction score of >5. We then converted the score to probability by applying softmax and used this probability along with the area of the bounding box for NMS. This method was decided because our bounding boxes were predefined and not regressed like in YOLO where only probability is used in NMS. NMS using only probability is inferior in this case as adjacent partly overlapping bounding boxes with high probability will not get suppressed despite using low IoU threshold. The benefit of using area and probability for NMS in the case of predefined bounding boxes has been shown in the following figures:

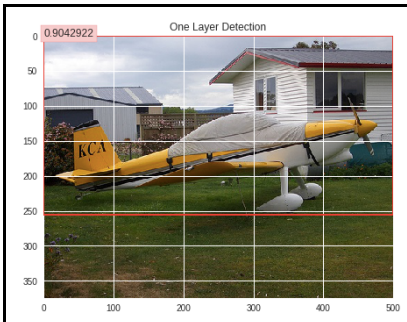

Fig. 1: NMS based on area and probability

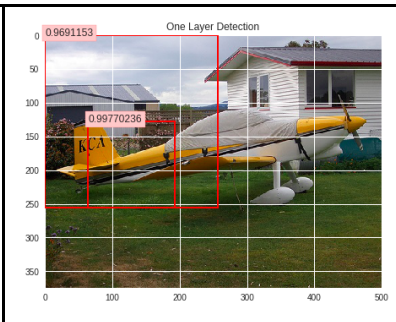

Fig. 2: NMS based on score.

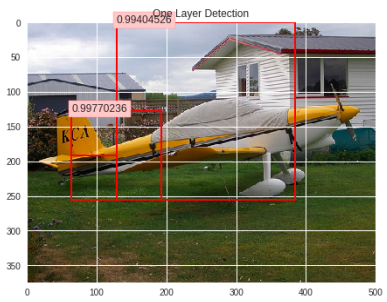

Fig. 3: NMS based on probability

3.2 Two Layer Detection

In this method, for every region of image slid using the sliding window, we performed classification using both model 1 and model 2 and kept those bounding boxes which has high score. Again NMS was applied as described above. We observed that the two layer detection did detect more objects in some cases (Fig. 4), but it also misclassified may bounding boxes which lead to erroneous output (Fig. 5).

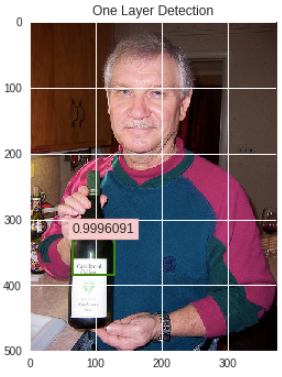

Fig. 4: One Layer

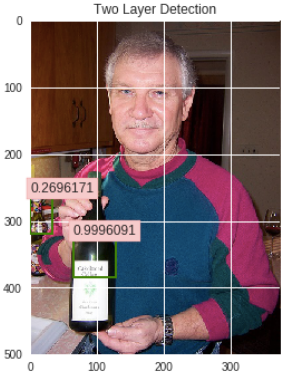

Fig. 5: Two Layer improved detection

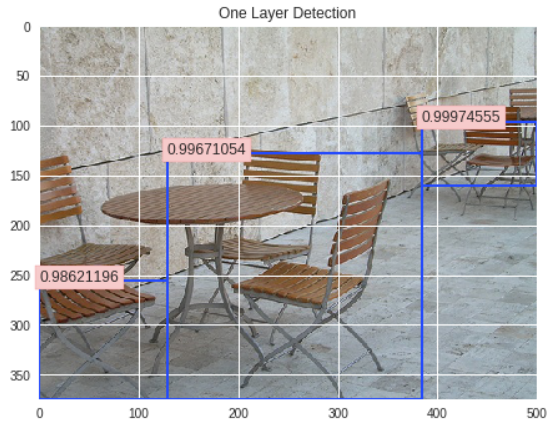

Fig. 6: One Layer

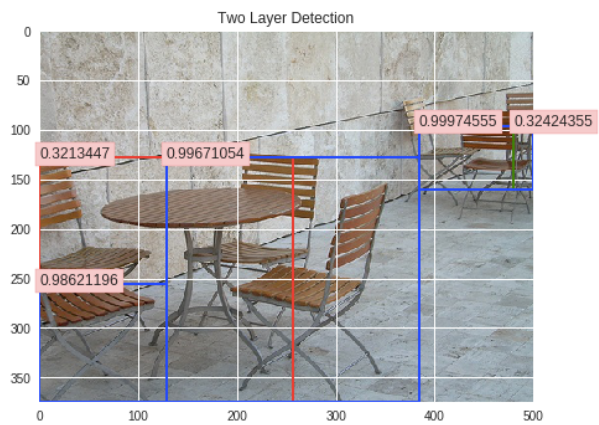

Fig. 7: Two Layer misclassified BB

Legend: Aeroplane (red BB), bottle (green BB), chair (blue BB)

4. Accuracy calculation

For testing the detector on the entire test dataset, we redefined windows with length of shorter side as 64, 128 and 256 pixel and removed the overlap in the sliding window. This was done to reduce the test time of an image as the earlier sliding window configuration was taking around 50 seconds per image. After changing the sliding window configuration, the test time of an image reduced to 3 seconds. Since mAP score has no weightage in the evaluation of this assignment, we used the mAP code available on github [1]. On running the mAP script for both the methods we observed an improvement of 5% for the two layer detection case. Since 5% is considered to be statistically significant, the implementation does show that Two Layer Detection is better than One Layer Detection.

One Layer Detection	Two Layer Detection
0.00% sofa AP	0.00% sofa AP
0.14% chair AP	0.15% chair AP
0.00% pottedplant AP	0.00% pottedplant AP
0.00% diningtable AP	0.00% diningtable AP
5.16% aeroplane AP	5.94% aeroplane AP
0.00% person AP	0.00% person AP
0.12% bottle AP	0.12% bottle AP
0.00% cat AP	0.00% cat AP
0.00% tvmonitor AP	0.00% tvmonitor AP
0.00% dog AP	0.00% dog AP
0.00% horse AP	0.00% horse AP
0.00% bird AP	0.00% bird AP
0.00% bicycle AP	0.00% bicycle AP
0.00% car AP	0.00% car AP
0.00% bus AP	0.00% bus AP
0.00% boat AP	0.00% boat AP
0.00% sheep AP	0.00% sheep AP
0.00% motorbike AP	0.00% motorbike AP
0.30% mAP	0.35% mAP

Table 1

5. Other methods we tried

For the two layer method, we had also tried a resnet18 classifier by concatenating the output of the first, second and third residual blocks and passing to a fully connected layer for classification. This model gave a validation accuracy of 0.82 but when we used this model for detection, it misclassified many bounding boxes. Due to brevity of time, we did not further investigate this model and finalized the model described in 3.2 as it was already showing 5% improvement in mAP. (Code for this method is also present in 'Rough' section present at the end of the ipython notebook)

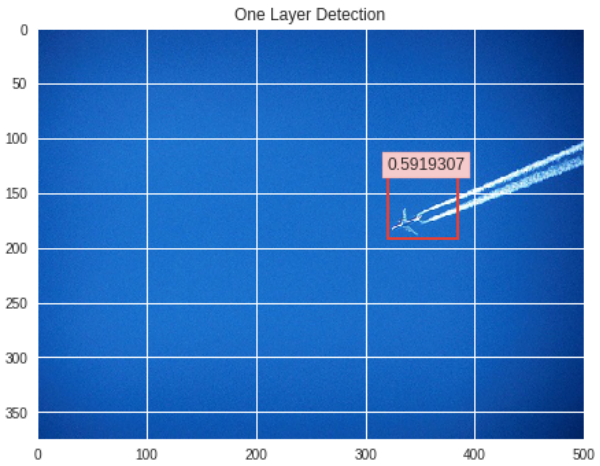

Fig. 8: One Layer

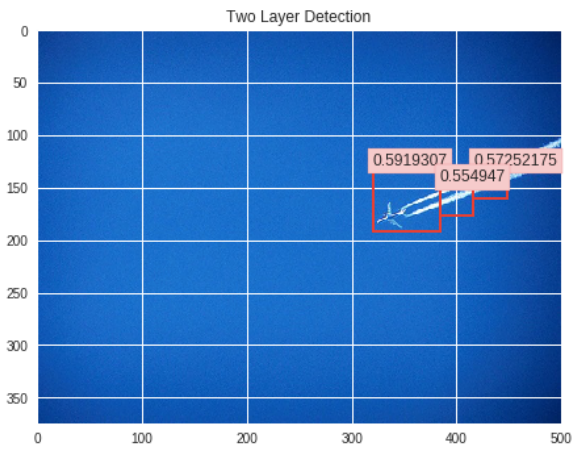

Fig. 9: Two Layer detection

6. References

[1] https://github.com/Pulupu/mAP