Assignment 3: Making an Object Detector

Priyanshu Gupta [170511]   Shivam Kumar [170668]

1 Objective

The objective of the assignment was to make a simple object detector for three lasses. We were required to use resnet-18 model for classification. The base model is used in two different ways, one which uses only the final layer of the model for classification and the other that uses another layer alongside the final layer for classification. After implementing the classifiers, we had to implement the sliding window method for finding bounding boxes, which were then fed to the network for making predictions. At last, we used Non-Maximum Suppression to filter out the bounding boxes predicted to be of a foreground object and print out final output for 20 images. Additionally, a MaP score was generated for each image of the test class.

2 Approach

2.1 Preprocessing

The first hurdle we faced was in the creation of a good training dataset. The first of our challenges were to make a good random sampler for background class generation. The major task was to find a good function to estimate the overlap of a patch with the object classes.

We initially started with using patches with low IoU as the criterion for random sampling from the test images. However, on inspection we found that in cases where class objects have a small size as compared to the patch, even in cases with high overlap, the IoU remained low due the intersection area being dominating.A possible fix was to reduce the IoU threshold, but this faired poorly on images where they were largely occupied by the objects. This made it a challenge to chose a good threshold. The problem is highlighted in the images below. To tackle this we used an

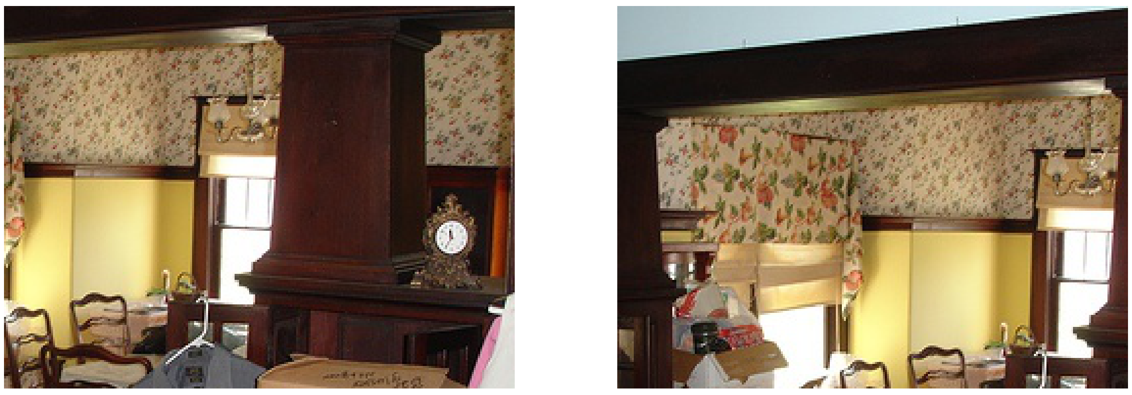

(a) using IOU (b) using overlap

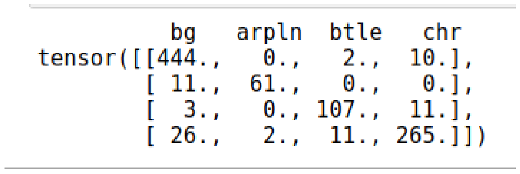
Figure 1

overlap function, which checks for the percentage region of the object class being covered by the patch. This increased our test set accurracy by 5% and 2% for one layer and two layer classifiers respectively. For the test set we considered regions of low IoUs as the parameter to declare a patch as being from the background class.

We also tried augmenting our data with a few transforms like color jitter, random rotation and random noise etc.The models trained well and give good accurracy on validation set. However, the models generalized poorly on the test set. On inspection of class-wise precision we came to the conclusion that the model was biased to classify to a chair or a background. We then tried balancing the dataset. On doing so, the model still was not able to generalize well and gave poor test accurracy. Finally we augmented the data using rotations and flipping only.

2.2 Classification

One Layer Classifier: We fine tuned a resnet 18 model, pre-trained on Image net dataset. We got the following confusion matrix for the test set(The test set was biased to have for background elements):



```
             bg   arpln  btle   chr
tensor([[444.,    0.,    2.,   10.],
        [ 11.,   61.,    0.,    0.],
        [  3.,    0., 107.,   11.],
        [ 26.,    2.,   11., 265.]])
```

Two Layer Classifier: We tried multiple approaches for this. Our first model was as follows: we took the an auxillary output from layer 3 of resnet and trained it by taking a loss function given by,

$$loss = loss_p + k * loss_{aux}$$

, where $loss_p$ is the loss of final layer prediction and $loss_{aux}$ the loss in auxillary prediction ($i.e$, 3^{rd} layer prediction). We tried changing k as a hyperparameter. This was inspired by the loss function used while training inception.[3] We then planned to generate a feature map by pooling the 2 layers. However, we found the model very difficult to train and consistently got below par results. In fact, the accurracy got below 50% on the validation set.There was a high bias to predict the an image as background. On tuning the value of k and other training parameters, we were able to achieve an accurracy of 60% but still were not able to get over the high bias.

Owing to the difficulties in trainig the above model, we decided to change our model. In the new model, we avg pooled the output of the inner layer and that of the final layer. We then concatanated it into a one dimensional tensor and passed it to a fully connected layer. This model gave us a validation accuracy of 84% on first training. On further tuning, we were able to extend it to 91%.

2.3 Sliding Window

We took windows of varying size and aspect ratios and slid it through the image (considering some stride of pixels) from left to right, and top to bottom while detecting the class scores for each of the window, and keeping only those which are were above a certain threshold value. Our implementation lets nine various sizes of windows to slide making a total of nine windows per pixel to slide. We were carrying all these operations for predictions on gpu.

Sliding window implementation was little bit time consuming as it was sliding the windows over each pixels nine times. We optimised it by using numpy as tool to carry all the operations .Also we took larger value for stride but by compromising with our bounding box accuracy.

Another problem which we were facing constantly was error :CUDA out of memory . We overcame this problem by deleting all the caches after each operations

2.4 Non-Maximum Suppression

[4] The main challenge was to implement NMS for filtering most of the sliding windows and keeping those which were more precise and accurate. First we implemented general python code for NMS for each of the three classes but it turned out to be very inefficient considering it's speed.Then we optimised it by using numpy for various operations. But the major code got highly optimised

2.5 Testing

We have provided the required 20 images in the jupyter notebook. We struggled a lot with the mAp script, due to both of us being ill alternatively, we were not able to properly test our model by generating the mAP. However on visual analysis of the images, we got decent results.The very same is reflected on the notebook too. We are including the mAp script we wrote and the code used for testing, however it is buggy.

3 Problems and possible modifications

On testing our models, we found the following problems:

- The model made some interesting confusions in predicting:
  - It mistook sky (to be predicted as background) in certain cases as an aeroplane. This mostly occurred with low probability but sometimes, it did cause problems in final bounding box estimation.
  - It mistook boxes with aeroplane wings as chairs (We found out that their silhouttes did look same in certain angles, especially to a beach chair)
  - People were confused by chairs in a few cases. The possible explanantiion to this can be that in the train set, many images have people sitting on the chairs, and the model was confusing people with chairs.
  - Objects with a glare in the background were many a times identified as 'bottles'.

  To overcome the above shortcomings, we can possibly use hard negative mining. To a certain extent, we did it manually and forsaw significant improvements.

- The model fails quite a lot in images with multiple objects.This shortcoming can easily be overcome by decreasing the stride size. However, doing so will increase the time elapsed by the program manifolds and hence is not preferable.

- The bounding boxes predicted didn't fit quite well on the object. While, we did try to fine-tune the sizes of the boxes, we found it very difficult to find a good set that fits in all the cases. A possible fix can be to train a regression model on the achieved bounding boxes(after NMS) and then, use it to predict the final boxes.

4 Conclusions

We learnt a lot about proper image mining and training a good classifier for a proper image detection task. We concluded that sliding windows is a very slow method for object detection as many of the computations felt redundant. A convolutional implementation[1] may have been better but given the demand of the question we didn't implement it. Further, a SSD like multi box detector may have helped.An alternative implementation of NMS [2] may have also been proven useful.

References

[1] Henry G. R. Gouk and Anthony M. Blake. "Fast Sliding Window Classification with Convolutional Neural Networks". In: *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand.* IVCNZ '14. Hamilton, New Zealand: ACM, 2014, pp. 114–118. ISBN: 978-1-4503-3184-5. DOI: 10.1145/2683405.2683429. URL: http://doi.acm.org/10.1145/2683405.2683429.

[2] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. "Learning non-maximum suppression". In: *CoRR* abs/1705.02950 (2017). arXiv: 1705.02950. URL: http://arxiv.org/abs/1705.02950.

[3] Rethinking the Inception Architecture for Computer Vision. *Object Detction 1: NMS.* URL: https://paperswithcode.com/paper/rethinking-the-inception-architecture-for.

[4] yy. *Object Detction 1: NMS.* URL: https://medium.com/@yusuken/object-detction-1-nms-ed00d16fdcf9.