CS 783- Visual Recognition (hw 3)

Object Detection

Anurag Pandey-160140
Sudhanshu Bansal-160710

March 28, 2019

Introduction:

The aim of this assignment is to implement object detection using sliding window approach and a classifier on VOC2007 dataset. Given a test image our implementation should effectively identify objects from the set of { 'Aeroplane','Bottle','Chair'} and mark bounding boxed around the same.

Our Approach:

Background

We started with building a dataset such that we can train our classifier on it. The VOC2007 data has the .xml files containing the annotations for the classes of " Aeroplane" , " Bottle " , " Chair " . We went to each of these .xml files, checked if it has an object from one of our useful class, went to the corresponding image and cropped the useful class object and stored in separate folders each.

For creating the background dataset, in each of the image where we found objects of our useful class, we generated random patches of image of random sizes and checked its Intersection over Union (iou) with the objects cropped from that image. If iou of the randomly generated patch is less than a threshold value with all the objects, we say that the random patch is an example of background class. We store all such images in the background class example.

Likewise we creates our test data set for training the classifier.

For the classifier - we choose RESNET18 pre- trained model and added a fully connected layer to it.

Difficulties The dataset was highly unbalanced with less images in the aeroplane and bottle class as compared to the background and chair. We further implemented data augmentation so as to make up for the imbalance but the classifier was still biased towards the examples of the bigger datasets i.e chair. So we added weights to the training images of aeroplane and bottle classes by training on the same images of these classes twice or thrice. This way we brought a level in the dataset.

Again , one of the major difficulties was that their is too much of intra class variations among the chair class and hence it was affecting the accuracy of our classifier.

Models (Single Layer and Two Layer

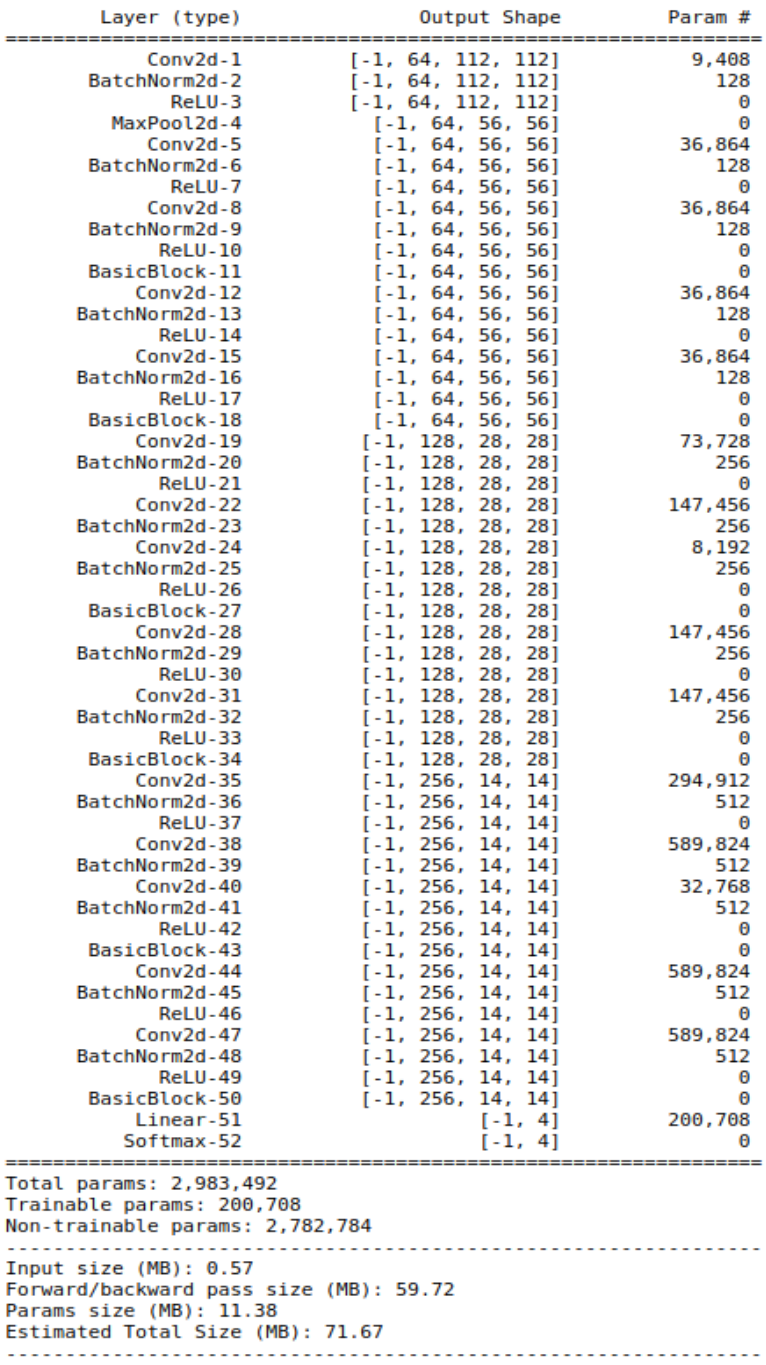
: We implemented both Single Layer Detection and Two Layer Detection.

Single Layer Detection had RESNET18 with an extra fully connected layer at its back classifying objects in each of the sliding window.

Two Layer Detection has one of the model as the earlier RESNET18. The second model is a modification to the pre-trained RESNET18. We removed last 3 convolutional layers and then used the fully connected layer for the classification purpose.

The second model is also printed on the next page.

Difficulties With the classifier, not being that accurate most of the windows were showing wrong predictions. We had to check our sliding window implementation a number of times before concluding the problem lies in the classifier. Also the cv2.imread() function read the image in BGR form, on which the predictions where not that good as the model was trained for images in Gray color space.



```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]          36,864
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]          36,864
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
       BasicBlock-11           [-1, 64, 56, 56]               0
           Conv2d-12           [-1, 64, 56, 56]          36,864
      BatchNorm2d-13           [-1, 64, 56, 56]             128
             ReLU-14           [-1, 64, 56, 56]               0
           Conv2d-15           [-1, 64, 56, 56]          36,864
      BatchNorm2d-16           [-1, 64, 56, 56]             128
             ReLU-17           [-1, 64, 56, 56]               0
       BasicBlock-18           [-1, 64, 56, 56]               0
           Conv2d-19          [-1, 128, 28, 28]          73,728
      BatchNorm2d-20          [-1, 128, 28, 28]             256
             ReLU-21          [-1, 128, 28, 28]               0
           Conv2d-22          [-1, 128, 28, 28]         147,456
      BatchNorm2d-23          [-1, 128, 28, 28]             256
           Conv2d-24          [-1, 128, 28, 28]           8,192
      BatchNorm2d-25          [-1, 128, 28, 28]             256
             ReLU-26          [-1, 128, 28, 28]               0
       BasicBlock-27          [-1, 128, 28, 28]               0
           Conv2d-28          [-1, 128, 28, 28]         147,456
      BatchNorm2d-29          [-1, 128, 28, 28]             256
             ReLU-30          [-1, 128, 28, 28]               0
           Conv2d-31          [-1, 128, 28, 28]         147,456
      BatchNorm2d-32          [-1, 128, 28, 28]             256
             ReLU-33          [-1, 128, 28, 28]               0
       BasicBlock-34          [-1, 128, 28, 28]               0
           Conv2d-35          [-1, 256, 14, 14]         294,912
      BatchNorm2d-36          [-1, 256, 14, 14]             512
             ReLU-37          [-1, 256, 14, 14]               0
           Conv2d-38          [-1, 256, 14, 14]         589,824
      BatchNorm2d-39          [-1, 256, 14, 14]             512
           Conv2d-40          [-1, 256, 14, 14]          32,768
      BatchNorm2d-41          [-1, 256, 14, 14]             512
             ReLU-42          [-1, 256, 14, 14]               0
       BasicBlock-43          [-1, 256, 14, 14]               0
           Conv2d-44          [-1, 256, 14, 14]         589,824
      BatchNorm2d-45          [-1, 256, 14, 14]             512
             ReLU-46          [-1, 256, 14, 14]               0
           Conv2d-47          [-1, 256, 14, 14]         589,824
      BatchNorm2d-48          [-1, 256, 14, 14]             512
             ReLU-49          [-1, 256, 14, 14]               0
       BasicBlock-50          [-1, 256, 14, 14]               0
           Linear-51                   [-1, 4]         200,708
          Softmax-52                   [-1, 4]               0
================================================================
Total params: 2,983,492
Trainable params: 200,708
Non-trainable params: 2,782,784
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 59.72
Params size (MB): 11.38
Estimated Total Size (MB): 71.67
----------------------------------------------------------------
```

FIG : Second Model used in the TWO LAYER detection

Non Maxima Suppression

Our Single Layer and Two Layer detection algorithms identified many boxes around the same single object in the image. The NMS reduces the count of these boxes significantly.

Sliding Window

One of the major **difficulties in Sliding Window** was choosing the aspect ratios and the window sizes suitably so that the computations are not too heavy and the objects are also captured in the windows. Since we were not having enough computational resources, we had to compromise on the number of windows.

A good set of window sizes would be = [(224,224) , (256,256) , (112 ,112), (112, 200), (200, 112), (64,64),(96,96),(192,96),(96,192),(64,128),(128,64)]].

Performance Metric

:

SINGLE LAYER detection

1. AP for Aeroplane =0.187
2. AP for Bottle = 0.084
3. AP for Chair = 0.068

mAP for SINGLE LAYER = 0.113

TWO LAYER detection

1. AP for Aeroplane =0.165136
2. AP for Bottle = 0.10493
3. AP for Chair = 0.09306

mAP for TWO LAYER = 0.121

NOTE : The 20 plots asked for in the notebook are attached in the zip folder in a different folder.

'aeroplane': 0.21605758368968964, 'bottle': 0.08274473994970322, 'chair': 0.07856828719377518

Mean Average Precision (mAP): 0.126

REFERENCES

- **mAP** = https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection?fbclid=IwAR151jH3LRtqb1cxrUosTyOm-ClJ7e1qpIYN5QLEFMxXj2iNSUJu7QTopzU
- **Sliding Window** = https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/
- **traning RESNET18 pytorch** - Our Assignment 2
- **NMS** = https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/
- **IoU** : on lines of https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/