# Multi Object Detection using Convolutional Neural Networks

Swetanjal Murati Dutta
International Institute of Information Technology, Hyderabad
Gachibowli, Telangana
swetanjal.dutta@research.iiit.ac.in

## Abstract

*This is a report for Assignment 4 of the Computer Vision Course taught by Dr. Avinash Sharma during Spring Semester of 2020 at IIIT-H. The goal of this assignment is to familiarise myself with how a CNN works and get hands on experience implementing a Convolutional Neural Network from scratch using PyTorch.*

## 1. Introduction

The assignment is divided into two parts. First part is to train a Convolutional Neural Network for image classification task on the PASCAL VOC2007[1] dataset. The second part is to perform multi object detection on the same PASCAL VOC2007[1] dataset using the trained CNN. This report is divided into two major sections, one on training CNN for classification and the other one on detecting multiple objects in an image. Each of these sections present relevant results according to the requirements of this assignment.

## 2. Image Classification using CNN

Previous work has shown the success achieved in the task of image classification by employing Convolutional Neural Networks[3]. In this section we describe how we trained and tested a simple CNN to achieve a classification accuracy of 60% on the PASCAL VOC2007[1] test set.

### 2.1. Training Data

We tried two sets of training data prepared as described in this subsection.

The first set we tried was prepared by cropping the ground truth boxes of the Training Data in the PASCAL VOC2007[1] dataset. We had access to the ground truth labels of the cropped regions as this is how the data is prepared.

The second set we tried was prepared by generating approximately 2K proposals(using the Fast Selective Search Algorithm[6]) from an image and then assigning each proposal the label of ground truth bounding box with which it had an IoU overlap of 0.5 or more. Proposals which did not have IoU overlap of 0.5 with any ground truth bounding boxes were not included in the training set.

It was observed that using the second dataset gave us an increase of 10% in classification accuracy on the test set.

### 2.2. Testing Data

The test data set was prepared by cropping the ground truth boxes of the Testing Data in the PASCAL VOC2007 dataset[1].

### 2.3. Architecture

We experimented with two architectures, one light weight and the other is a heavy architecture motivated by [5]. The light weight architecture is shown in Fig 1 and the heavier architecture is shown in Fig 2. In both the architectures, each convolution layer is followed by a ReLU non linearity. We learn only $3 * 3$ kernels in each convolution layer.

The light-weight architecture1 takes in an image of size $64 * 64 * 3$ where as the heavier architecture2 takes in an image of size $128 * 128 * 3$.

### 2.4. Training the networks

Since, either network takes a fixed size image as input, the first step involves resizing the images in training dataset. To resize images, standard bilinear interpolation is used. The training is done in batches where the batch size is fixed to 20 samples. The initial learning rate for both the networks was fixed to 0.07. We use Stochastic Gradient Descent as the optimizer. The loss criterion used was Cross Entropy as this is a classification task. Both the architecture required training for no more than 10 epochs to get the desired models. The heavy architecture took approximately 20 minutes for a single epoch where as the light weight architecture took a couple of minutes for training on a single epoch.

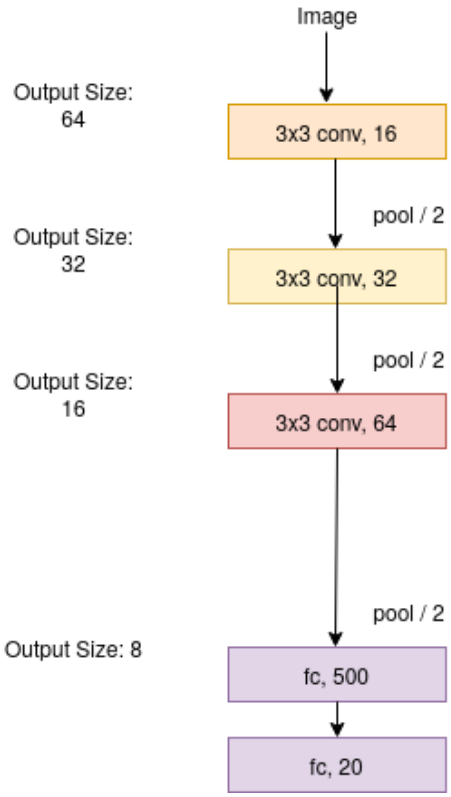



Figure 1. The light weight architecture CNN model

### 2.5. Results

We evaluate our classification results on the PACAL VOC2007[1] training set which contains close to 15K samples across 20 different classes. The classification accuracy on the test set in the light weight and heavy architectures are 53% and 60% respectively. The minute difference in these numbers indicate the sparsity of our training data, which was intentionally done for this assignment keeping in mind students don't have access to GPUs.

### 2.6. Analysis of classification results

Here I present my analysis on the performance of my architectures:

- Making the architectures substantially deep does not have a very strong impact on the performance. This is due to the absence of sufficient training data.
- Classes with more amount of training data has a better classification accuracy than the classes with less training data. In general, training samples are not equally distributed over all the 20 classes. The best performing class is the person class, owing to the largest number of training samples in that class.
- The networks end up learning colors as no standard color augmentation is used as a data pre-processing step. This is highly evident from the large number of false positives in the car class. Since, car class has too many red cars in training data, many red images of other classes are getting classified as 'car'.

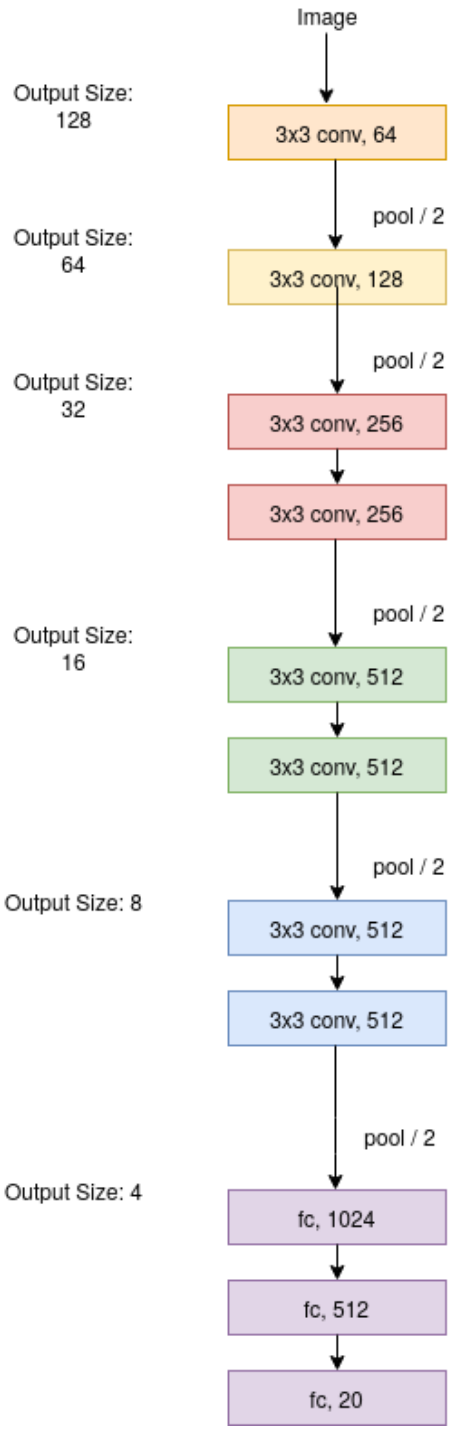



Figure 2. The heavy architecture CNN model

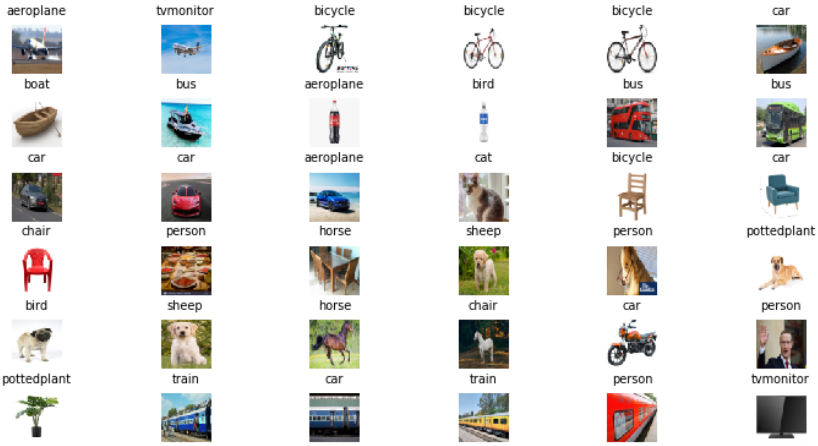



Figure 3. Some classification results(my own images) of the light weight architecture CNN model. Each images has it's label on top of it

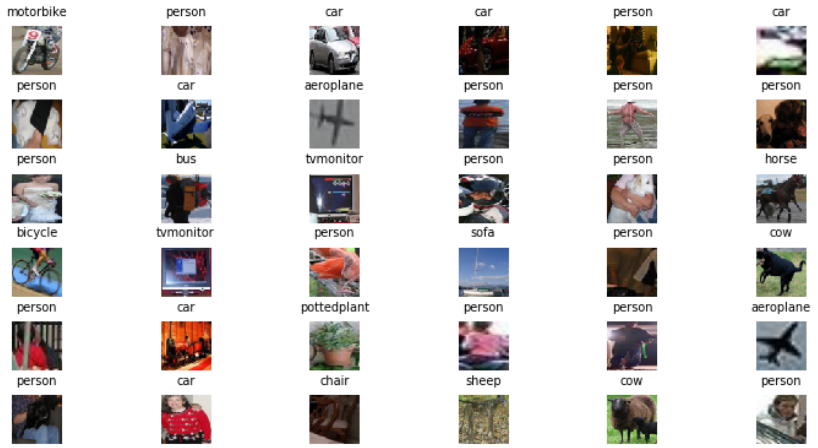



Figure 4. Some classification results(from test set) of the light weight architecture CNN model. Each images has it's label on top of it

- Classes which show less intra-class variation(like car) perform better compared to classes which show high intra-class variation(like cats and dogs)

## 3. Object Detection

The next part of the assignment required us to use the CNN in the previous part to perform multi-object detection in an image. The pipeline was motivated by that of RCNN[2].

### 3.1. Object Detection Pipeline

The object detection system overview is given in Fig. 7. Here, we discuss each step in detail:

1. The first step takes an image as input.
2. The second step extracts approximately 2K region proposals using the Fast Selective Search Algorithm[6]. The proposal is also resized to dimensions of $128 * 128 * 3$ as our convolutional neural network takes as input fixed dimensional images of $128 * 128 * 3$.
3. The third step involves extracting features from our trained CNN network.

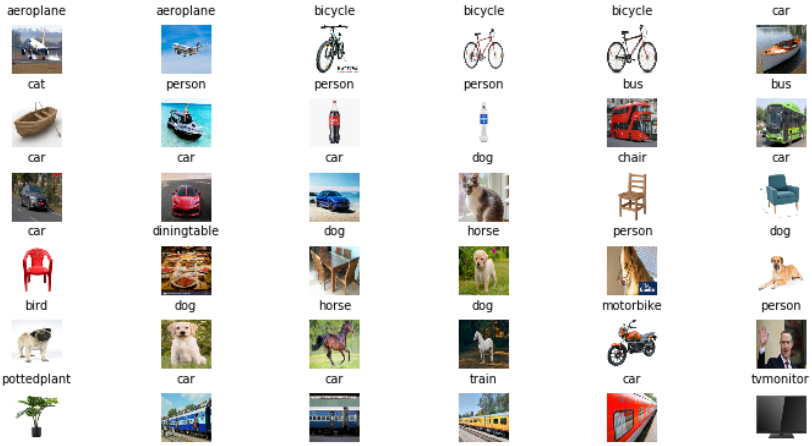



Figure 5. Some classification results(my own images) of the heavy weight architecture CNN model. Each images has it's label on top of it

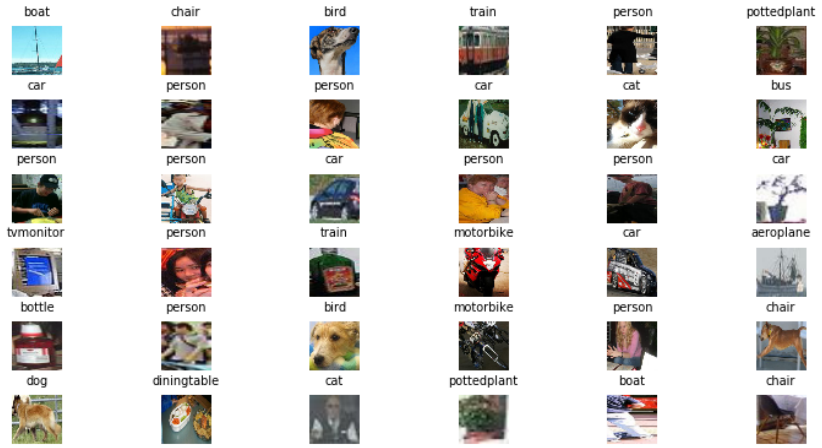



Figure 6. Some classification results(from test set) of the heavy weight architecture CNN model. Each images has it's label on top of it

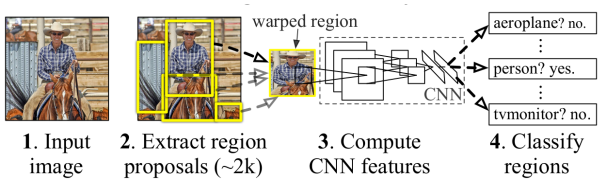



Figure 7. Object detection system overview.

4. The fourth step involves classification of regions.

5. The fifth step though not shown in figure involves non maximal suppression. This step is primarily to reject redundant detections.

### 3.2. Test Time Detection

At test time, we run selective search on the test image to extract around 2000 region proposals (we use selective search's "fast mode" in all experiments). We warp each proposal and forward propagate it through the CNN in order to compute features. Then, we classify each proposal and depending on class specific confidence of classification and size of proposal we reject or accept the proposal. Finally, we apply a greedy non-maximum suppression (for each class independently) that rejects a region if it has an intersection-over-union (IoU) overlap with a higher scoring selected region larger than an empirically decided

| Class | Light-weight | Heavy-weight |
|---|---|---|
| Aeroplane | 53% | 56% |
| Bicycle | 44% | 45% |
| Bird | 26% | 28% |
| Boat | 40% | 35% |
| Bottle | 12% | 42% |
| Bus | 61% | 37% |
| Car | 67% | 72% |
| Cat | 28% | 34% |
| Chair | 32% | 50% |
| Cow | 42% | 29% |
| Dining Table | 30% | 35% |
| Dog | 18% | 23% |
| Horse | 44% | 58% |
| Motorbike | 30% | 65% |
| Person | 78% | 82% |
| Potted Plant | 49% | 59% |
| Sheep | 43% | 46% |
| Sofa | 16% | 11% |
| Train | 46% | 70% |
| TV/Monitor | 69% | 75% |
| Overall accuracy | 53% | 60% |

Table 1. Top-1 Classification accuracy for each class in both the architectures

threshold(0.1 in our case).

### 3.3. Evaluation Criteria

For detection, a common way to determine if one object proposal was right is Intersection over Union (IoU). This takes the set A of proposed object pixels and the set of true object pixels B and calculates:

$IoU(A, B) = \frac{A \cap B}{A \cup B}$

Commonly, IoU $\geq 0.5$ means that it was a hit, otherwise it was a fail. For each class, one can calculate the true positive and false positive.

A true positive(TP) in the context of object detection for a class means a proposal was made for the class and in ground truth there exists an object of that class. A false positive(FP) for a class means a proposal was made for the class but there does not exist an object of that class in ground truth.

Average Precision for a class $c$ can be calculated as follows:

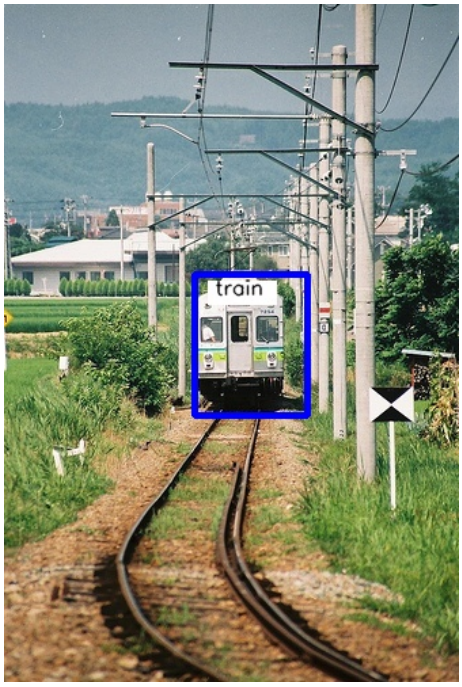
$AP = \frac{\#TP(c)}{\#TP(c)+\#FP(c)}$

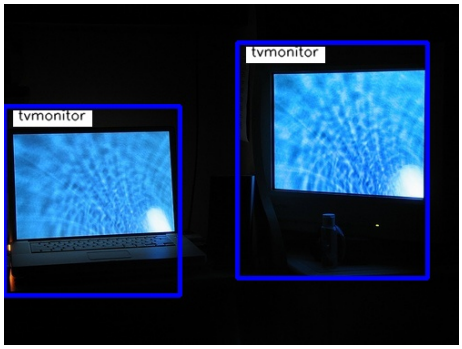
We use the metric, Mean Average Precision(mAP) to determine the performance of object detection systems. mAP is defined as follows:

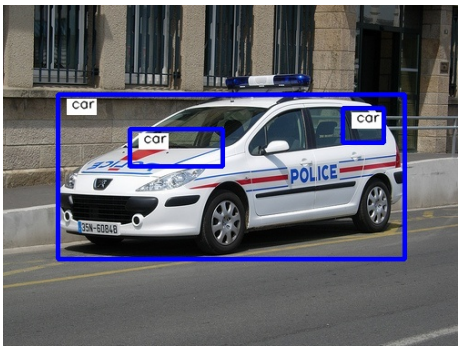
$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c)+\#FP(c)}$

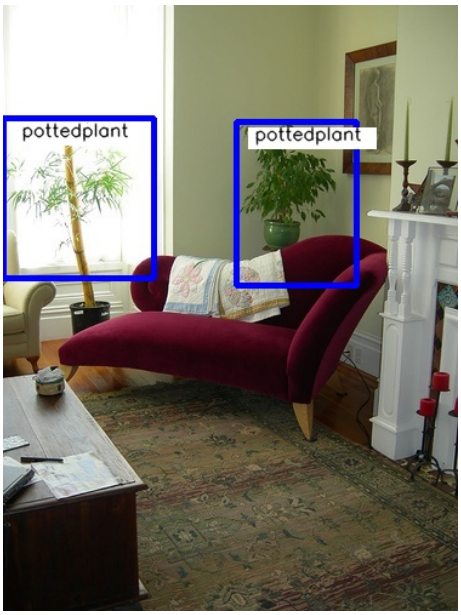
Multiple detections of the same object in an image were considered false detections e.g. 5 detections of a single object counted as 1 correct detection and 4 false detections.

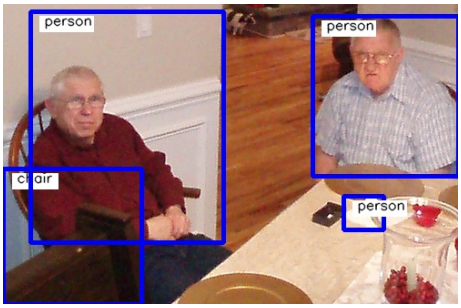
We use [4] to compute the above metrics in this assignment.

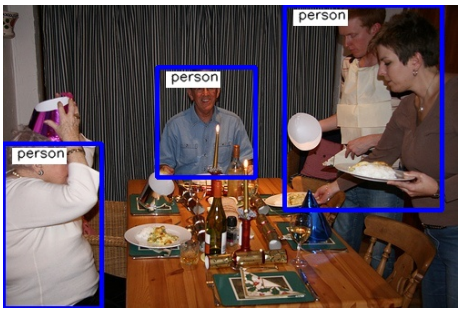


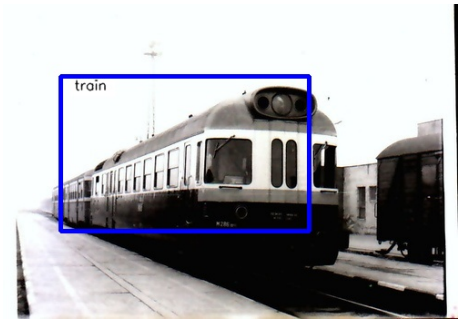


car
car
car
POLICE

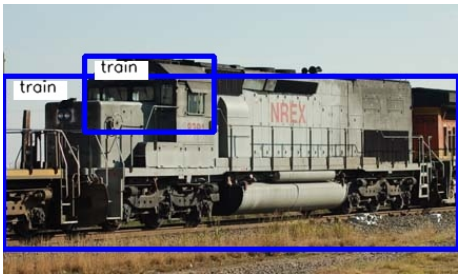

pottedplant
pottedplant


person
person
person

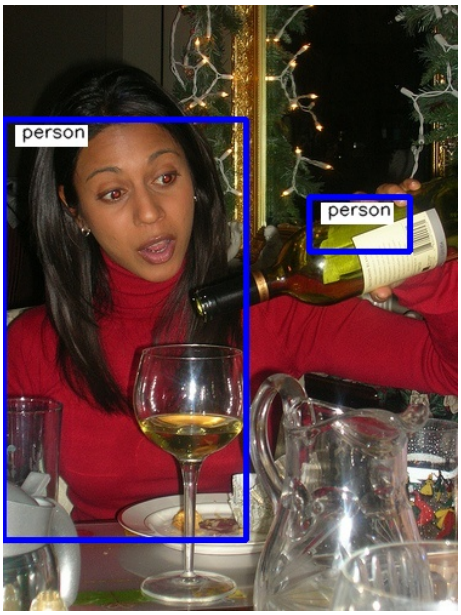

person
person
person

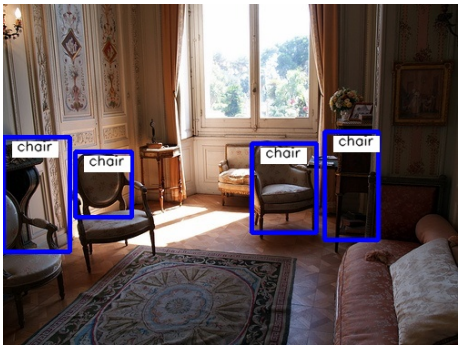

Scotiabank Place
person
person


train


train
train
NREX

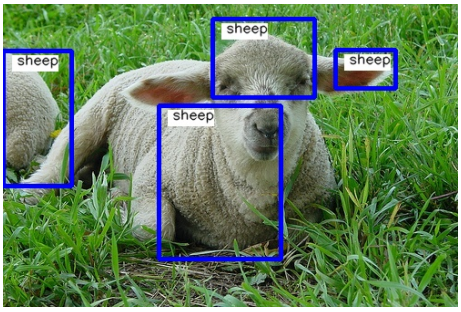

pottedplant

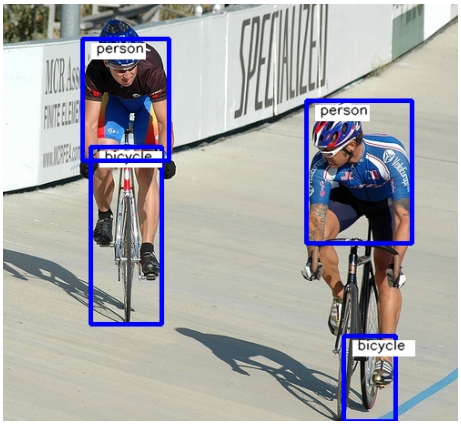

person
person

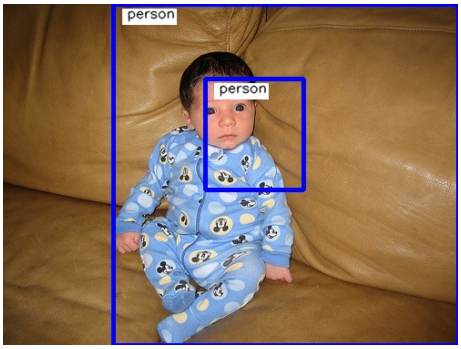

chair
chair
chair
chair

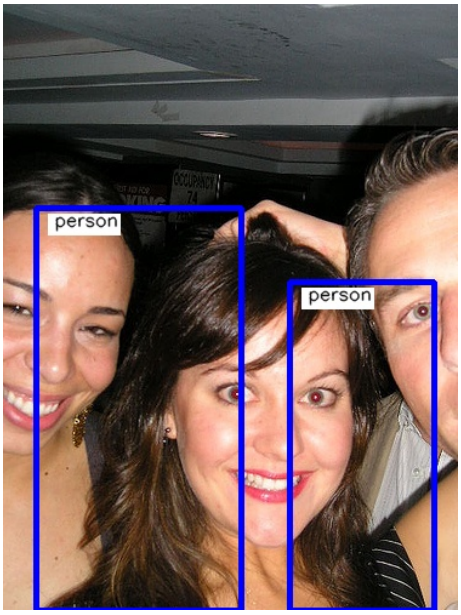

car
car
car

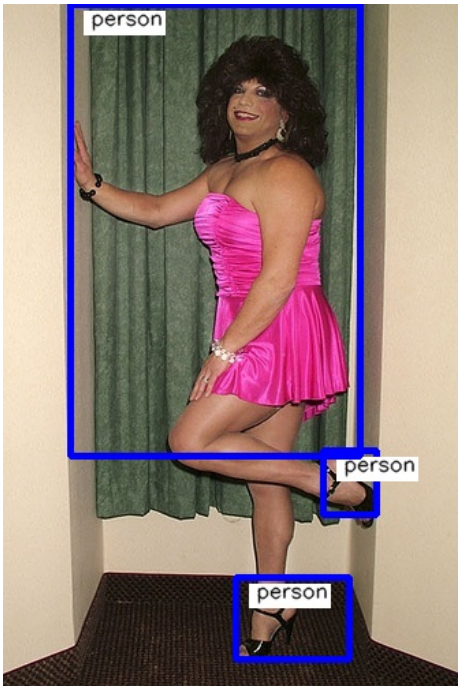

person


sheep
sheep
sheep
sheep

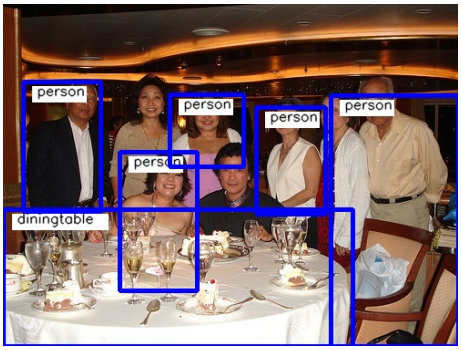

person
person
bicycle


person
person

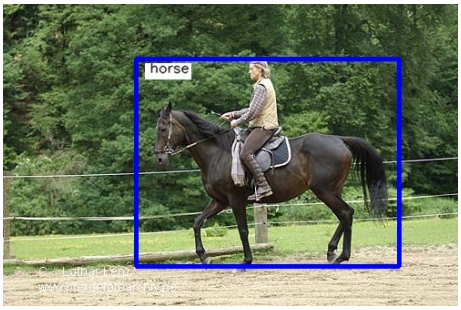

person
person

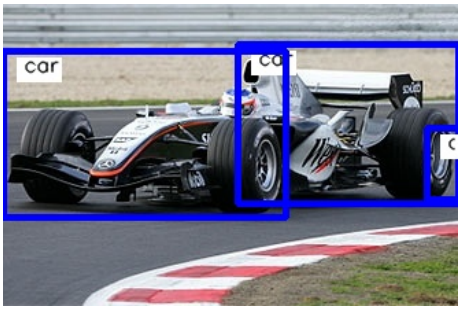

person
person
person


diningtable

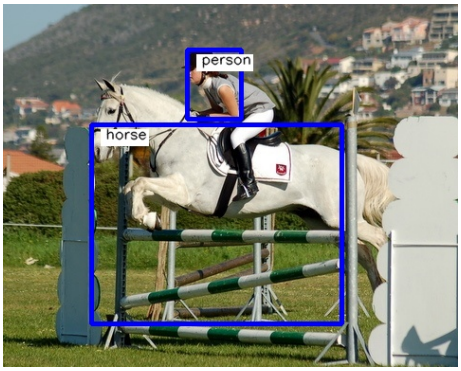

person
person
person
person
diningtable

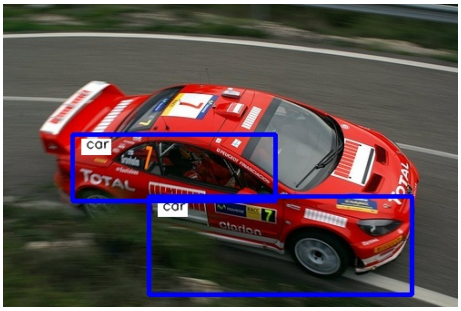

person

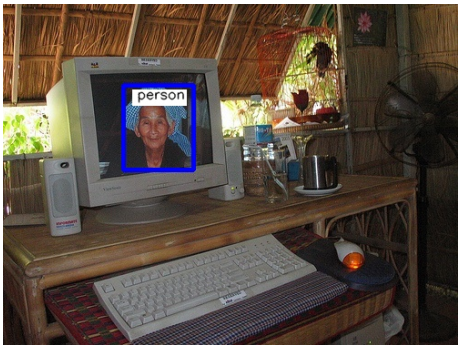

horse

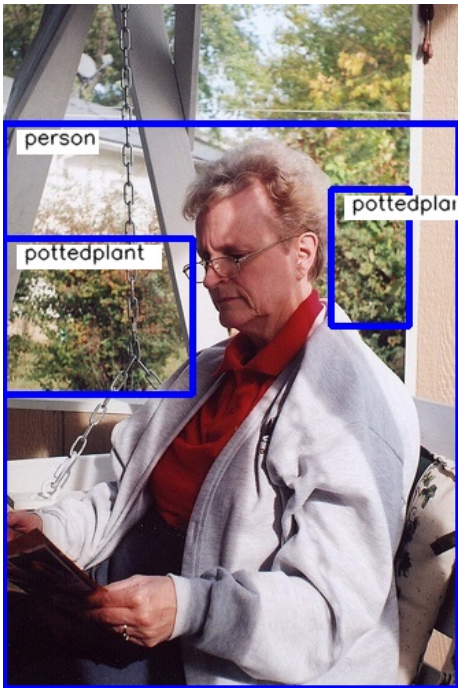

car

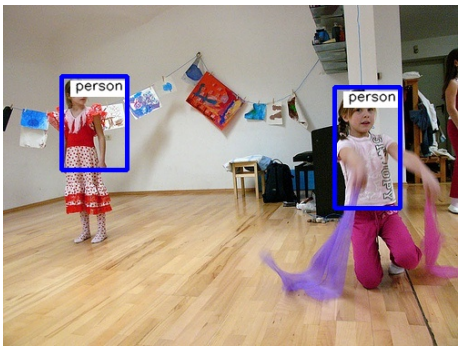

person


person
horse

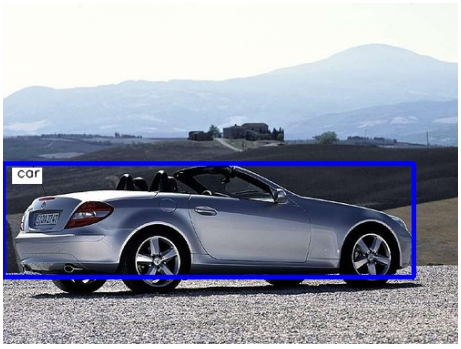

car
car

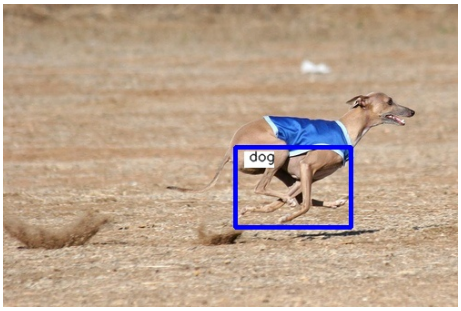

person

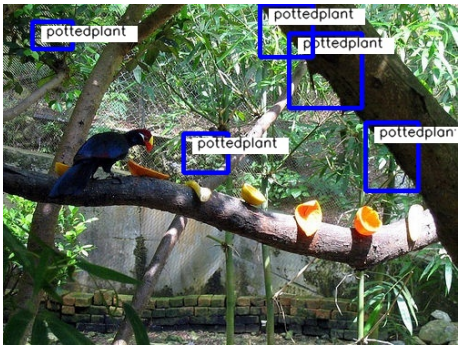

person
pottedplant

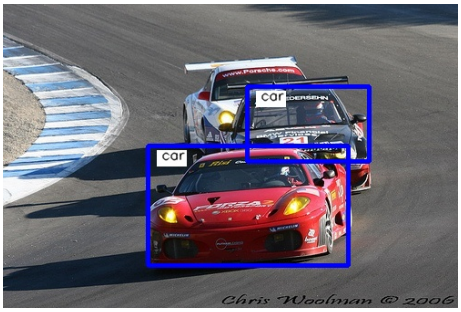

person
person


BMW 645Ci
Nouvelle BMW Série 6
car

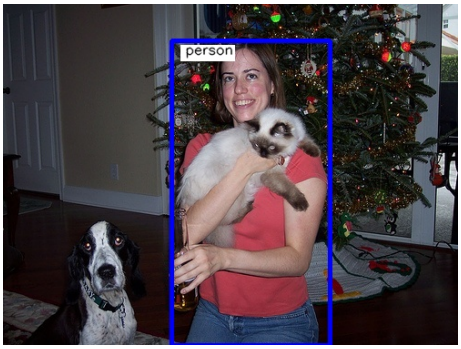

car

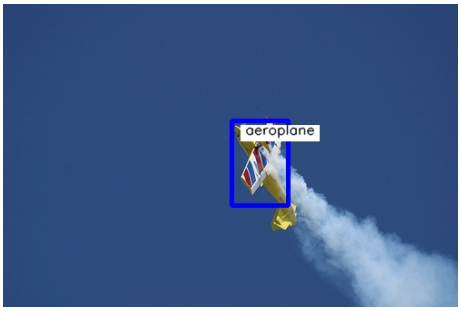

dog


pottedplant
pottedplant
pottedplant
pottedplant
pottedplan

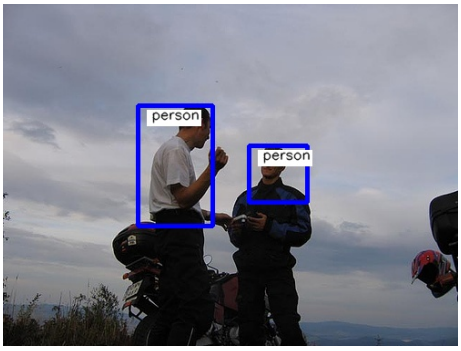

car
car
Chris Woolman © 2006

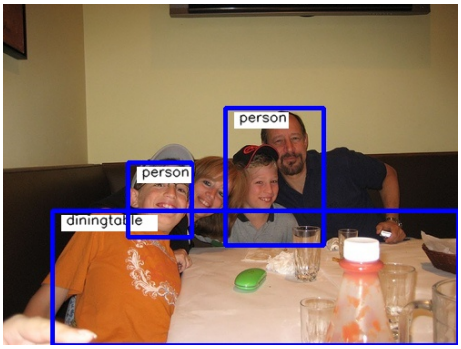

pottedplant
potted

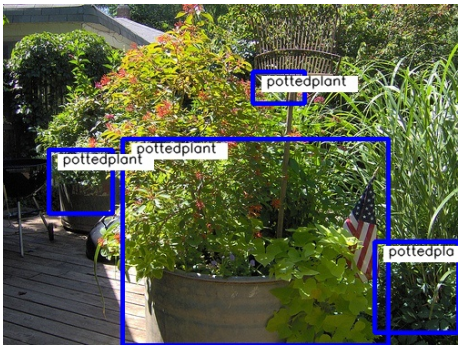

person

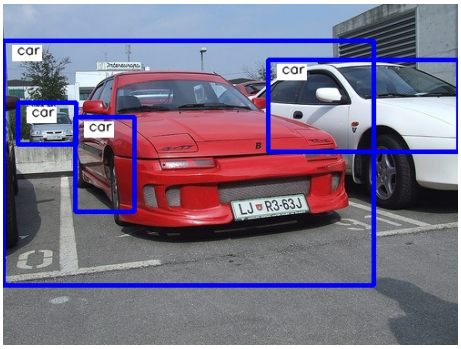

aeroplane


car

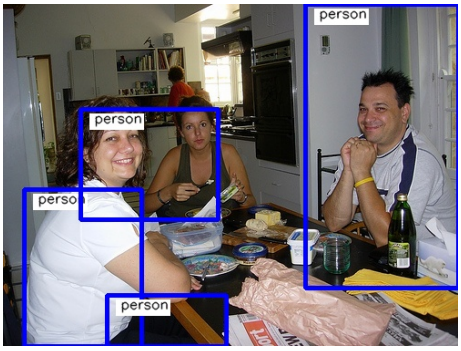

person
person

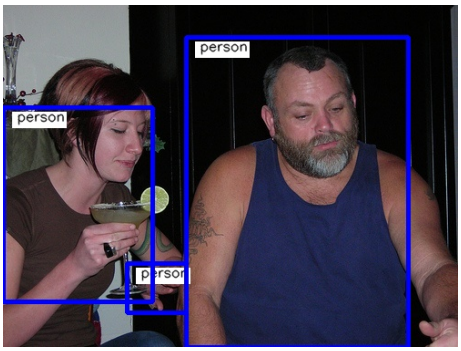

person
person
diningtable

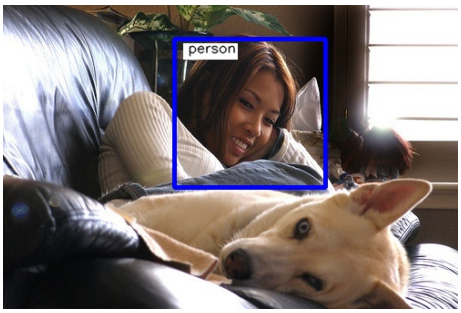

pottedplant
pottedplant
pottedplant
pottedpla


car
car
car
car
LJ R3-63J

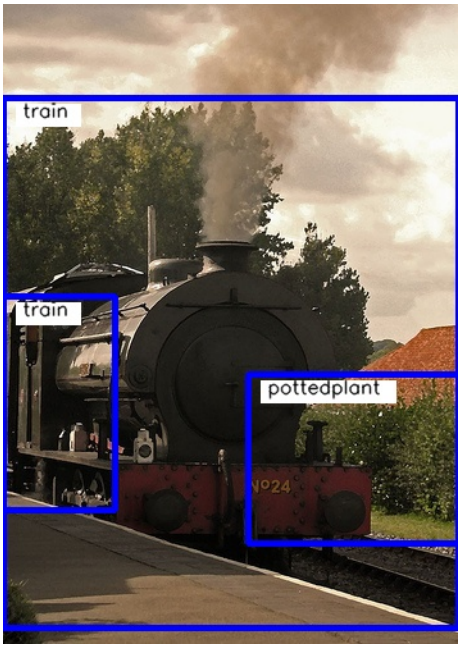

aeroplane

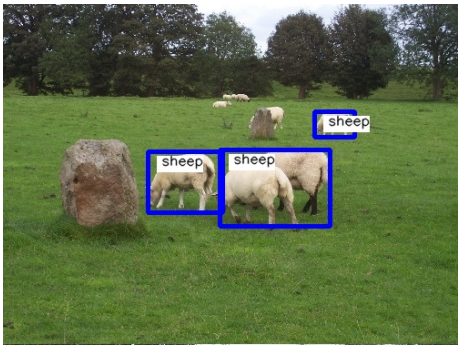

person
person
person
person


person
person
person

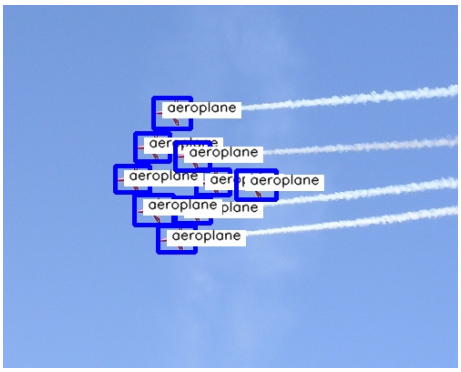

person

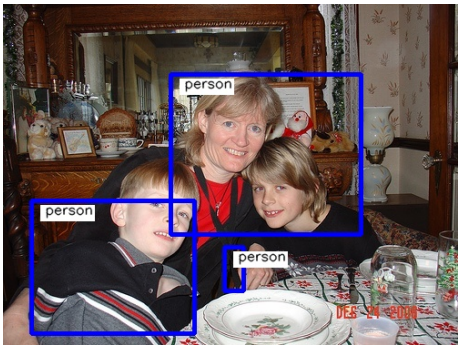

pottedplan
pottedplant
pottedplant


train
train
pottedplant

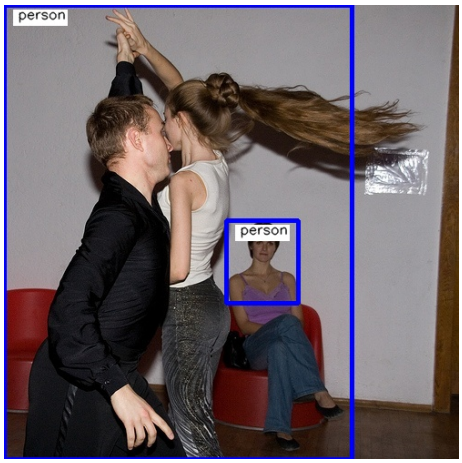

sheep
sheep
sheep

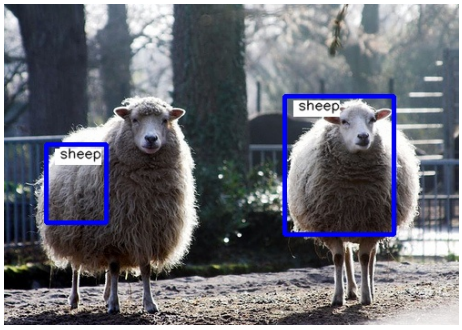

diningtable
pottedplant



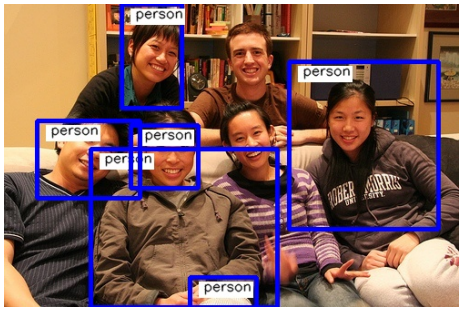









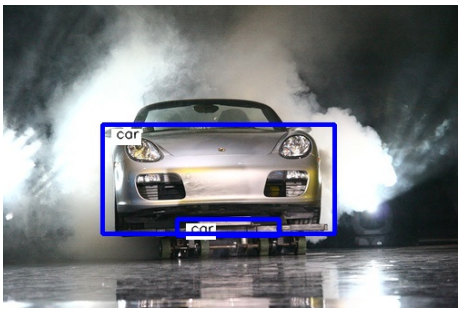


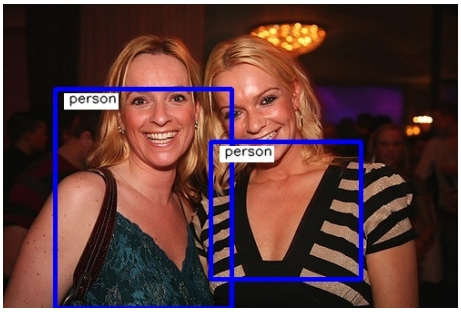

## 4. Results

We achieved an mAP score of approximately 15% on the PASCAL VOC2007 testing dataset, using IoU threshold of 0.2 to distinguish between correct and incorrect detections.

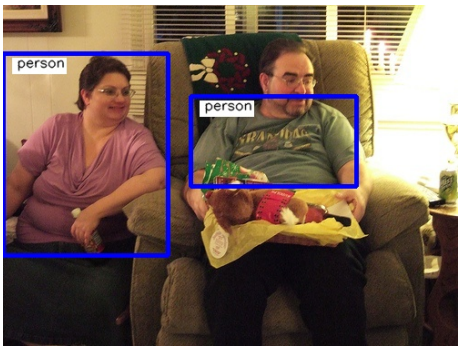
## 5. Source Code

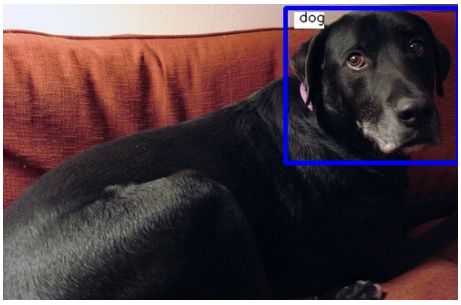
The source code has been submitted with this report. This is the repository which contains source code.



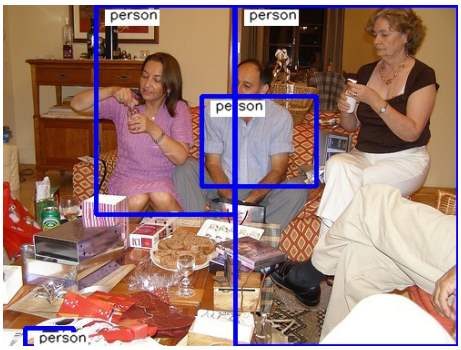


### 5.1. Running a demo on a sample image

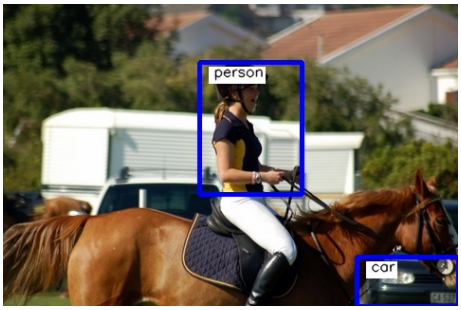
- python3 demo.py <image_path>

### 5.2. Testing Object Detection

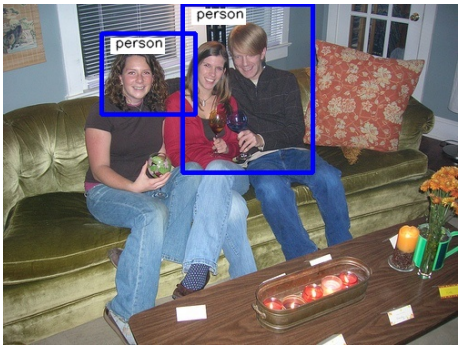
1. Download and extract the test data from here and place it in the same directory as the report.
2. Download the pretrained model from here (or train your own, refer Section 5.4 or 5.5) and place it in the same directory as the report.
3. Run test_rcnn.py
4. Check the results in 'Results' directory. The 'detections' folder contains the bounding boxes(with labels) for each test image in appropriate text files. The bounding boxes are stored as top left and bottom right coordinates format.



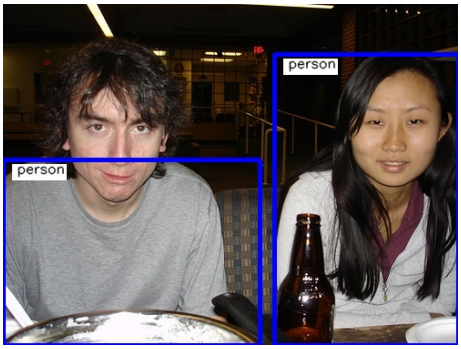


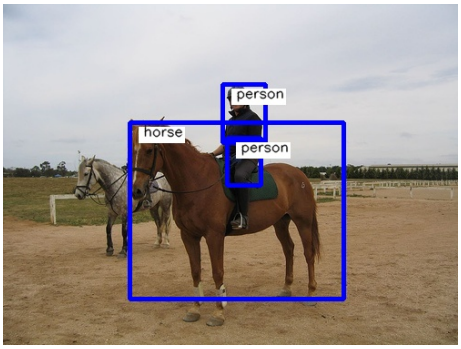











### 5.3. Testing Object Classification

1. Download and extract the test data from here and place it in the same directory as the report.

2. Download the pretrained model from here (or train your own, refer Section 5.4 or 5.5) and place it in the same directory as the report.

3. Run test_cnn.py

4. The statistics will be displayed on the terminal as output to code.





### 5.4. Training the CNN model on your own dataset

- Put the training images in appropriate sub-directories according to label of image inside directory 'training/cnn_data' and/or 'training/JPEGImages'. Follow format mentioned in [1].
- Open the file train_cnn.py and put the path to validation set in the string named test_data_path
- Run train_cnn.py
- Best model will be stored in this same directory as report named 'model_voc.pt'
- Make sure to rename the model as 'model_voc_60.pt' before using it for testing as this is the naming convention used in all parts of the code.

| Class | AP score |
|---|---|
| Aeroplane | 19.28% |
| Bicycle | 15.65% |
| Bird | 0.00% |
| Boat | 0.00% |
| Bottle | 0.15% |
| Bus | 17.97% |
| Car | 39.55% |
| Cat | 0.00% |
| Chair | 2.38% |
| Cow | 4.74% |
| Dining Table | 5.93% |
| Dog | 1.37% |
| Horse | 33.33% |
| Motorbike | 22.22% |
| Person | 27.25% |
| Potted Plant | 9.06% |
| Sheep | 8.27% |
| Sofa | 0.13% |
| Train | 34.15% |
| TV/Monitor | 6.78% |
| mAP Score | 12.41% |

Table 2. Average Precision scores for each class. We used the Heavy weight CNN model as the backbone.

### 5.5. Training the CNN model on our dataset to replicate results

- Download and extract the train data from here and place it in the same directory as this report.
- Open the file train_cnn.py and put the path to validation set in the string named test_data_path.
- Run train_cnn.py
- Best model will be stored in this same directory as report named 'model_voc.pt'
- Make sure to rename the model as 'model_voc_60.pt' before using it for testing as this is the naming convention used in all parts of the code.

## 6. Study on trying various parameters on the CNN model

- A learning rate of $0.07$ was used. Using too large a learning rate resulted in oscillation where as too small a learning rate resulted in very slow decrease of loss with every epoch.
- Instead of using the entire training set for weight updation in every epoch, if we use a random small subset of samples we get a decreasing loss plot which is jittery, showing that loss does not decrease in every epoch continuously rather there is increase/decrease in consecutive epochs, but the net result is a decreasing loss.
- When using cross entropy loss, using softmax or not in the architecture does not affect the results.
- Using SGD optimizer with Cross Entropy Loss(and without softmax) and setting the initial learning rate to $0.07$, gave a testing loss of $1.48$ in a couple of epochs. The corresponding testing loss with Softmax was $2.53$.
- By increasing our pooling filter size, we decrease the resolution of image further and lose more information. Hence, we get lower accuracy by training for same number of epochs.
- Using tanh and sigmoid activation functions did not give us desirable performance on the heavy network especially. This is due to the problem of vanishing gradient due to saturation in these activations.
- ReLU and Leaky ReLU gave us nearly similar performance.
- Adding momentum did not help us in getting better performance as loss function either converged or diverged substantially in every epoch in our case.

## References


[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 1, 2, 16

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014. 3

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1

[4] S. L. N. Rafael Padilla and E. A. B. da Silva. Survey on performance metrics for object-detection algorithms. 2020. 5

[5] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 1

[6] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *Int. J. Comput. Vision*, 104(2):154–171, Sept. 2013. 1, 3