Image Object Detection Using R-CNN

Tanmay Kirtania
ID: 2016-1-60-071
Department of CSE
East West University
Dhaka, Bangladesh
tanmays.ewu@outlook.com

Ashik Ahmmed
ID: 2016-1-60-098
Department of CSE
East West University
Dhaka, Bangladesh
ashikshuvo1996@gmail.com

Enamul Hassan Bhuiyan
ID: 2016-1-60-131
Department of CSE
East West University
Dhaka, Bangladesh
enamhassan@gmail.com

Abstract— ***The ever increasing power of machine learning to detect all the objects of an image as well as their classes is, now-a-days, becoming more and more powerful and effective. The faster R-CNN algorithm is really doing a very great job in this area. Its affect computational power and accuracy in a comparatively shorter time is its best side to be used in implementing the system to detect image object. Our implementation of this algorithm to detect object of images has also shown the effectiveness of this algorithm.***

1. INTRODUCTION

At present, image object detection has become a very popular and useful task to deal with. In this paper we will introduce about our implementation of RCNN algorithm to detect objects of images. We have used faster R-CNN algorithm which belongs to the R-CNN family to perform image object detection.

In this new era of machine learning we are now capable of making a machine or computer to understand about its environment and taking decision to solve any problem according to its nature. So computer vision is a very important part to understand what is going on all around the environment. So there is a very confusing question, how a computer can see? Ok the answer is computer cannot see but what it can do is it can take a picture as input and by doing some very complicated calculation on the value of pixels of given picture it can come to a decision based on the previous knowledge. So, when the keyword "previous knowledge" came into our discussion with this another topic "machine learning or pattern learning" we have to consider because based on previous knowledge or data trying to classify any object or understand any pattern of a problem is machine learning. So our objective is by using the computational power of computer and processing the data through complex machine learning algorithm we have to make our machine capable of classifying any object in a given picture.

1.1. Study Background

We will know about the problem and the solution we are dealing with in problem demonstration part, but first we will learn Faster R-CNN first. All the important things to learn about this algorithm are described below.

1.1.1. VGG-16 Network

The input to base layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3

conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

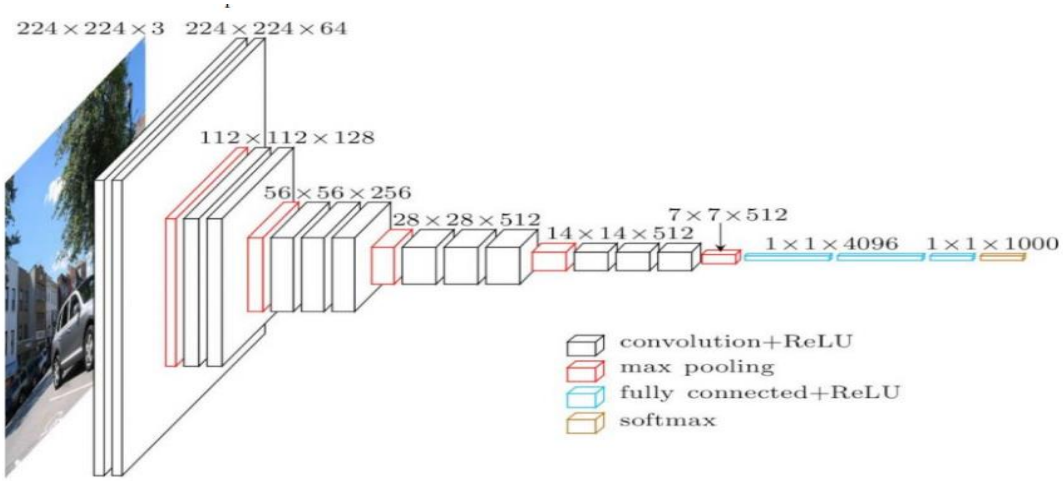


Figure 1: VGG-16 Network Architecture

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalization (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

1.1.2. Anchors

Anchors play an important role in Faster R-CNN. An anchor is a box. In the default configuration of Faster R-CNN, there are 9 anchors at a position of an image. The following graph shows 9 anchors at the position (320, 320) of an image with size (600, 800).

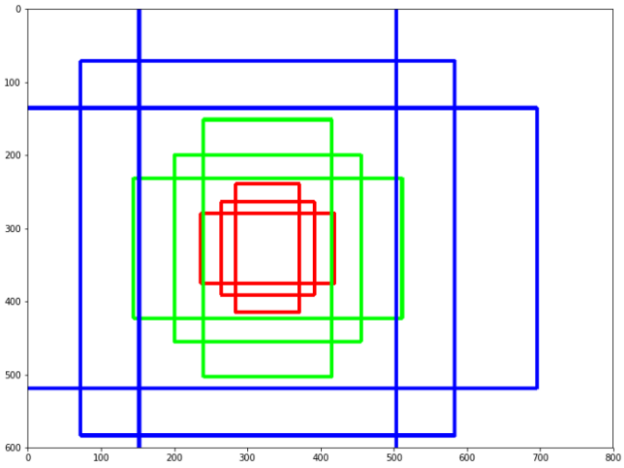


Figure 2: Anchors

Three colors represent three scales or sizes: 128 x 128, 256 x 256, 512 x 512. Let's single out the red boxes/anchors. The three boxes have height width ratios 1:1, 1:2 and 2:1 respectively.
If we choose one position at every stride of 16, there will be 1989 (39x51) positions. Or you can reason this is why it has a coverage as good as other state of the art methods. The bright side here is that we can use region proposal network, the method in Fast RCNN, to significantly reduce number. These anchors work well for Pascal VOC dataset as well as the COCO dataset. However you have the freedom to design different kinds of anchors/boxes. For example, you are designing a network to count passengers/pedestrians, you may not need to consider the very short, very big, or square boxes. A neat set of anchors may increase the speed as well as the accuracy.

1.1.3. Region Proposal Networks

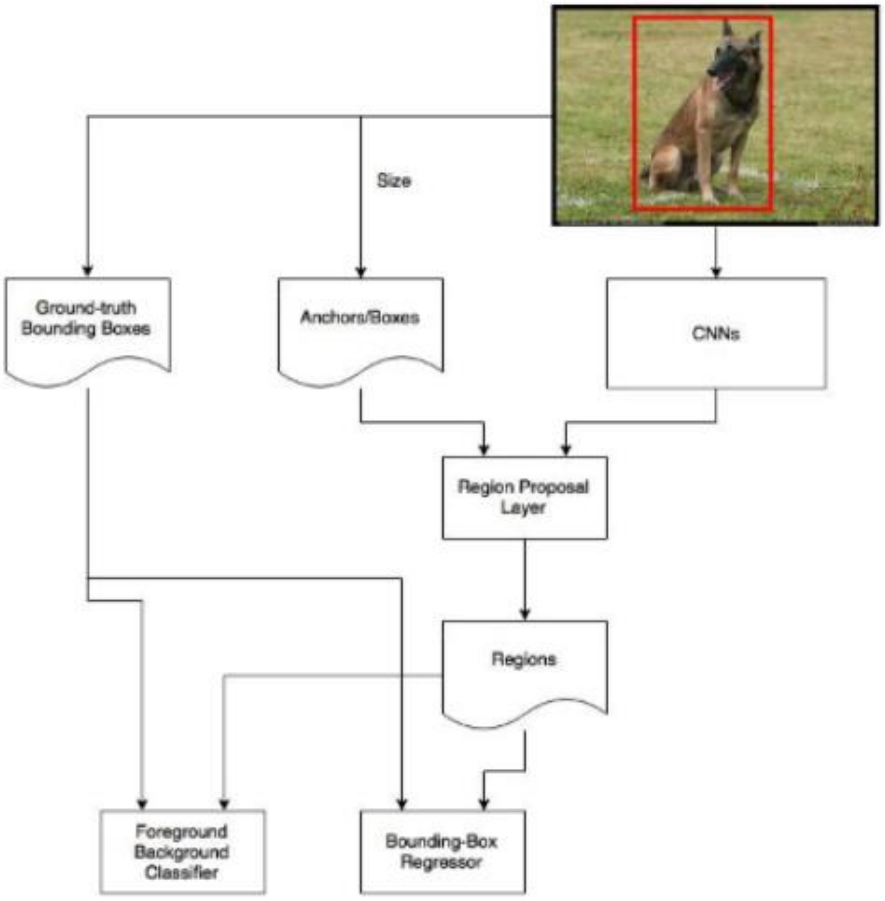


Figure 3: Regional Proposal Network in Training

A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.1 We model this process with a fully convolutional network computation with a Fast R-CNN object detection network , we assume that both nets share a common set of conv layers. In our experiments, we investigate the Zeiler and Fergus model (ZF), which has 5 shareable conv layers and the Simonyan and Zisserman model(VGG -16), which has 13 shareable conv layers. To generate region proposals, we slide a small network over the conv feature map output by the last shared conv layer. This network is fully connected to an n _ n spatial window of the input

conv. feature map. Each sliding window is mapped to a lower-dimensional vector (256-d for ZF and 512-d for VGG). This vector is fed into two sibling fully-connected layers—a box-regression layer (reg) and a box-classification layer (cls). We use n = 3 in this paper, noting that the effective receptive field on the input image is large (171 and 228 pixels for ZF and VGG, respectively). This mininetwork is illustrated at a single position in. Note that because the mini-network operates in a sliding-window fashion, the fully-connected layers are shared across all spatial locations. This architecture is naturally implemented with an n _ n conv layer followed by two sibling 1 _ 1 conv layers (for reg and cls, respectively). ReLUs are applied to the output of the n _ n conv layer. A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.

1.1.4. The Classifier of Background and Foreground

The first step of training a classifier is make a training dataset. The training data is the anchors we get from the above process and the ground-truth boxes. The problem we need to solve here is how we use the ground-truth boxes to label the anchors. The basic idea here is that we want to label the anchors having the higher overlaps with ground-truth boxes as foreground, the ones with lower overlaps as background. Apparently, it needs some tweaks and compromise to separate foreground and background. Now we have labels for the anchors. The second question here is what features of the anchors are.

Let's say the 600x800 image shinks 16 times to a 39x51 feature map after applying CNNs. Every position in the feature map has 9 anchors, and every anchor has two possible labels (background, foreground). If we make the depth of the feature map as 18 (9 anchors x 2 labels), we will make every anchor have a vector with two values (normal called logit) representing foreground and background. If we feed the logit into a softmax/logistic regression activation function, it will predict the labels. Now the the training data is complete with features and labels. Another thing you may pay attention to is receptive field if you want to re-use a trained network as the CNNs in the process. Make sure the receptive fields of every position on the feature map cover all the anchors it represents. Otherwise the feature vectors of anchors won't have enough information to make predictions. In the architecture of Overfeat, it only uses non-overlapping convolutional and pooling filters to make sure every position in the feature map cover its own receptive field without overlapping others. In Faster R-CNN, receptive fields of different anchors often overlap each other, as you can from the above graph. It leaves the RPN to be position-aware.

1.1.5. The Regressor of Bounding Box

As we follow the process of labelling anchors, you can also pick out the anchors based on the similar criteria for the regressor to refine. One point here is that anchors labelled as background shouldn't be included in the regression, as we don't have ground-truth boxes for them. The depth of feature map is 32 (9 anchors x 4 positions).The paper uses smooth-L1 loss on the position (x ,y) of top-left the box, and the logarithm of the heights and widths, which is as the same as in Fast R-CNN.

1.1.6. RPN Loss Function

$$L(\{p_i\},\{t_i\}) = (1/N_{cls})\times\Sigma L_{cls}(p_i,p_i^*) + (\lambda/Nreg)\times\Sigma p_i^* Lreg(t_i,t_i^*)$$

1.1.7. Region of interest pooling

Our ultimate goal is to share computation with a Fast R-CNN object detection network, we assume that both nets share a common set of conv layers. In our experiments, we investigate the Zeiler and Fergus model, which has 5 shareable conv layers and the Simonyan and Zisserman model (VGG), which has 13 shareable conv layers. To generate region proposals, we slide a small network over the conv feature map output by the last shared conv layer. This network is fully connected to an n*n spatial window of the input conv.

What does the RoI pooling actually do? For every region of interest from the input list, it takes a section of the input feature map that corresponds to it and scales it to some pre-defined size (e.g., 7×7). The scaling is done by:

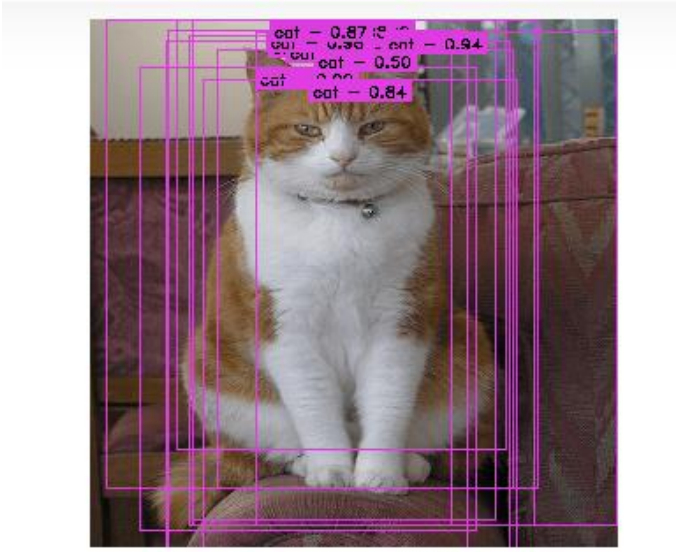


Figure 4: Region of Interest Pooling

a) Dividing the region proposal into equal-sized sections (the number of which is the same as the dimension of the output).
b) Finding the largest value in each section.
c) Copying these max values to the output buffer.

The result is that from a list of rectangles with different sizes we can quickly get a list of corresponding feature maps with a fixed size. Note that the dimension of the RoI pooling output doesn't actually depend on the size of the input feature map nor on the size of the region proposals. It's determined solely by the number of sections we divide the proposal into. What's the benefit of RoI pooling? One of them is processing speed. If there are multiple object proposals on the frame (and usually there'll be a lot of them), we can still use the same input feature map for all of them. Since computing the convolutions at early stages of processing is very expensive, this approach can save us a lot of time.

1.2. Related Work

There are many related work on image object detection. Many algorithms like CNN, RCNN have been used to do the task. In paper [1], the researchers have used the faster RCNN method to implement the image object detection system.

On the other hand, in the papers [2], [3], and [4], the researchers have analyzed different efficient ways to improve the RCNN algorithm and make it fast and faster. Thus the RCNN family has three members, namely, RCNN, Fast RCNN, and Faster RCNN.

2. IMPLEMENTATION

2.1. Dataset

For training we took the "Google Open Image Dataset with Bounding Boxes V4". In the dataset there are bounding box annotations, image urls for the source of the images, and class descriptions. We selected 8 classes for our training and filtered randomly 1000 images per class. Then we stored the corresponding images by crawling the given urls on the web. We divided the images in 80:20 ratio for training and testing considering each classes and the extracted those images using their annotation information. After all the training dataset is containing 26253 data for 6395 sample images, and the test data is containing 6532 data for 1598 sample images. Thus the preprocessing of the dataset is done.

2.2. Training

The training has been done using the "Jupyter Notebook" tool which uses Python as its native programming language. The training classes and there corresponding number of images are as follows:

```
Training images per class:
{'Boat': 2617,
 'Car': 2731,
 'Flower': 5048,
 'Food': 2762,
 'Fruit': 4183,
 'Mobile phone': 1266,
 'Motorcycle': 1684,
 'Person': 6062,
 'bg': 0}
Num classes (including bg) = 9
{'Mobile phone': 0, 'Fruit': 1, 'Person': 2, 'Motorcycle': 3, 'Flower': 4, 'Car': 5, 'Food': 6, 'Boat': 7, 'bg': 8}
Config has been written to N:\Study\Object Detection\model_config.pickle
```

Figure 5: Classes with Number of Images

While training, the system takes all the bounding boxes in an images as input, and determine the negative and positive boxes so that it can determine the classes properly. Even there can be multiple objects of multiple classes in a single images.

We trained the system using 47 epoches where each contains 1000 epoch length. The last 5 epoch result is as follows:

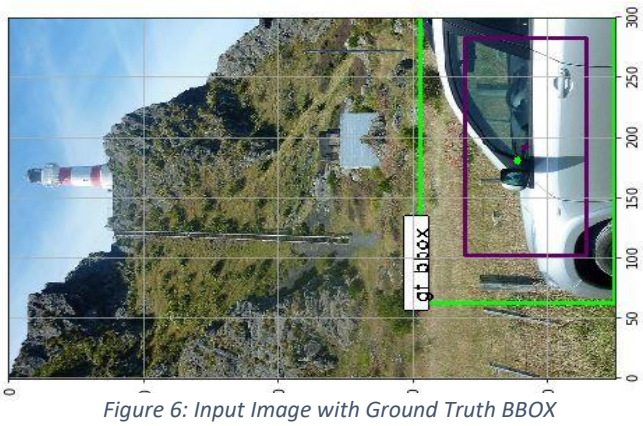
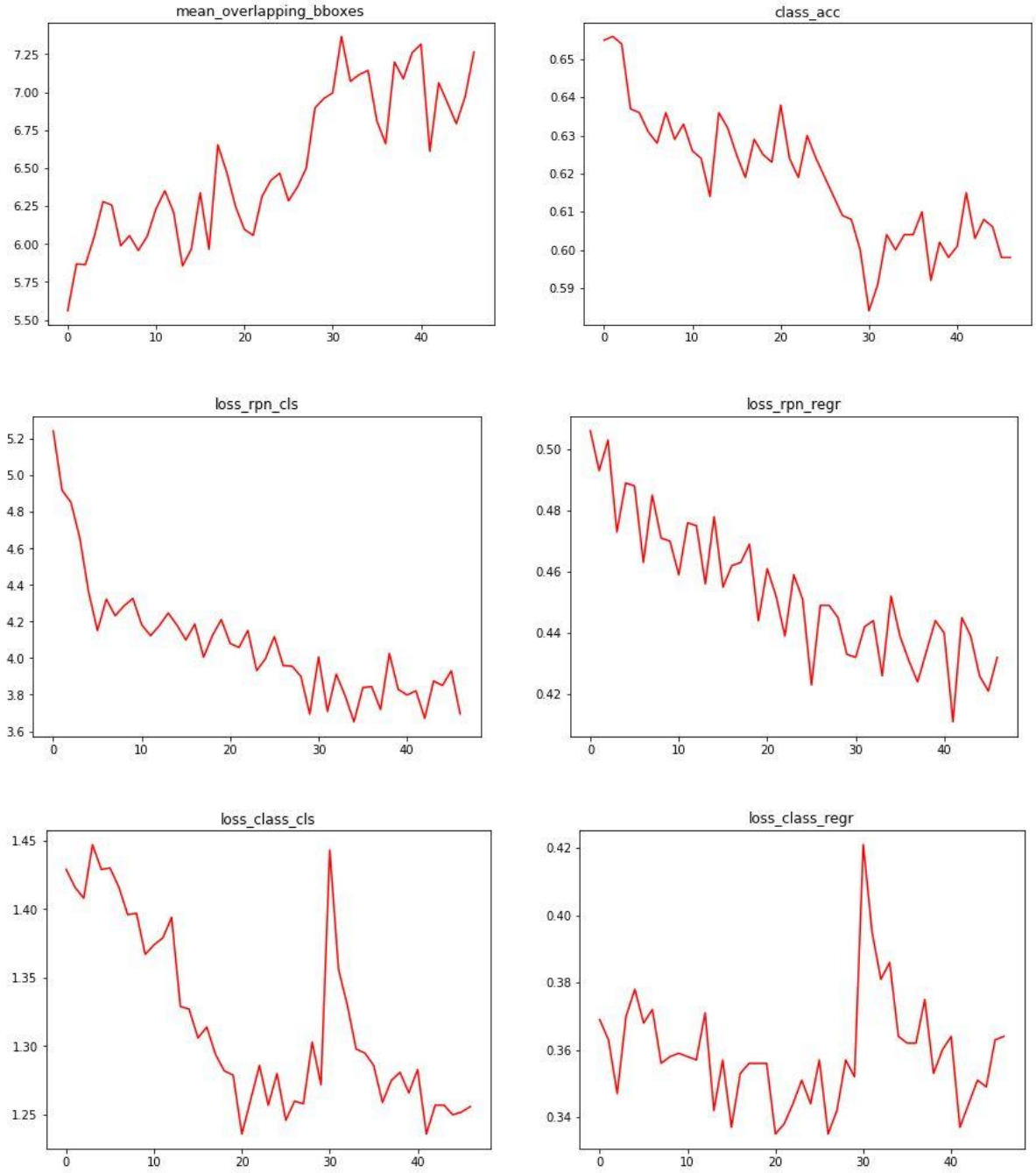
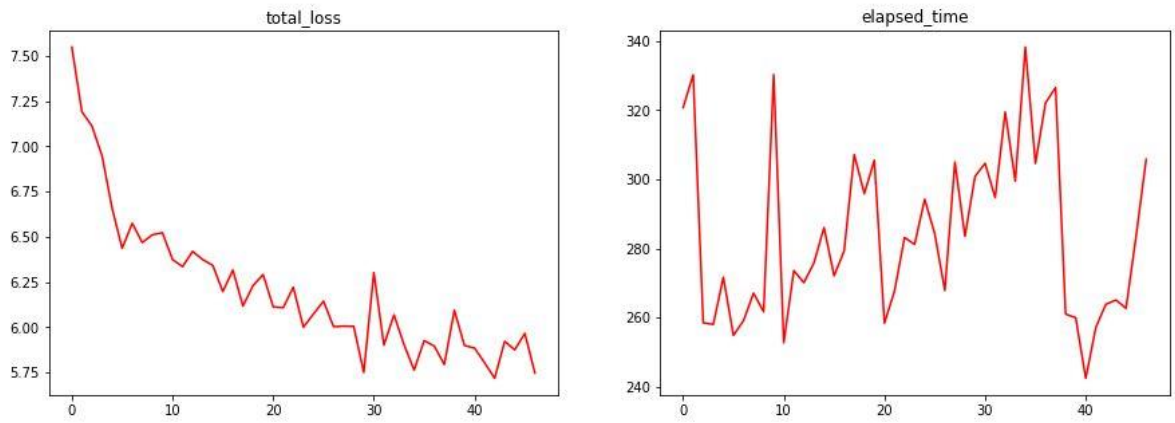


Figure 6: Input Image with Ground Truth BBOX

epoch	mean_overlapping_bboxes	class_acc	loss_rpn_cls	loss_rpn_regr	loss_class_cls	loss_class_regr	curr_loss	elapsed_time	mAP
42	6.61	0.615	3.821	0.411	1.236	0.337	5.804	257.139	0
43	7.063	0.603	3.671	0.445	1.257	0.344	5.718	263.825	0
44	6.93	0.608	3.875	0.439	1.257	0.351	5.922	265.055	0
45	6.792	0.606	3.85	0.426	1.25	0.349	5.875	262.642	0
46	6.972	0.598	3.931	0.421	1.252	0.363	5.966	283.554	0
47	7.266	0.598	3.695	0.432	1.256	0.364	5.747	305.785	0

Here we can see the cost (curr_loss) after the 43rd iteration is 5.71 which is the lowest cost and so it generates the ultimate weights for the model. The simulation for all the epochs can be explained by the plotting below:





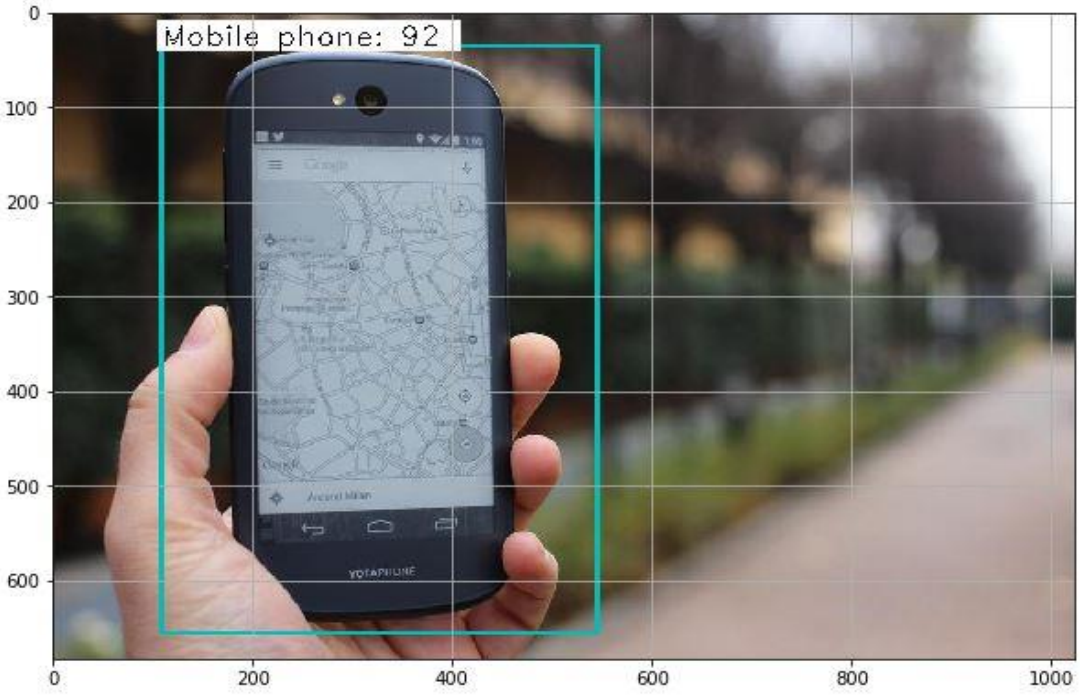
3. PERFORMANCE ANALYSIS

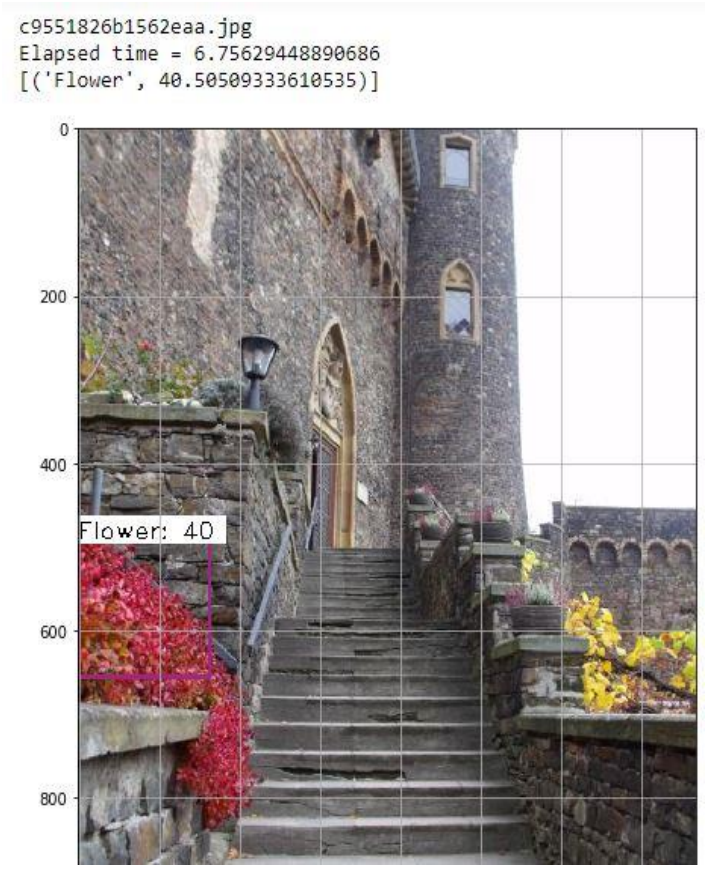
The performance could be better if we could train the model more and more. We trained the model in only 47 thousand batches, which should be at least 140 thousand to get the expected result. This is the drawback of our system which has forcefully decreased the performance of the system and also the lack of ability of our training machine is also a drawback to the building of the system. So solving these issues we can certainly increase the performance of our system and can make a better image object detection system undoubtedly.

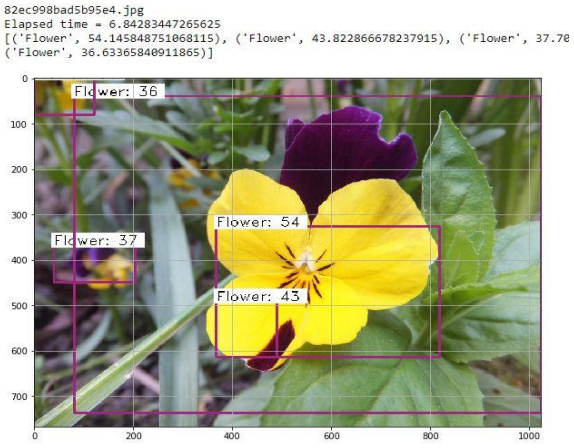
3.1. Output

Here are some sample outputs:

```
4d31d34fb44bfbb4.jpg
Elapsed time = 7.539783477783203
[('Mobile phone', 92.14403033256531)]
```







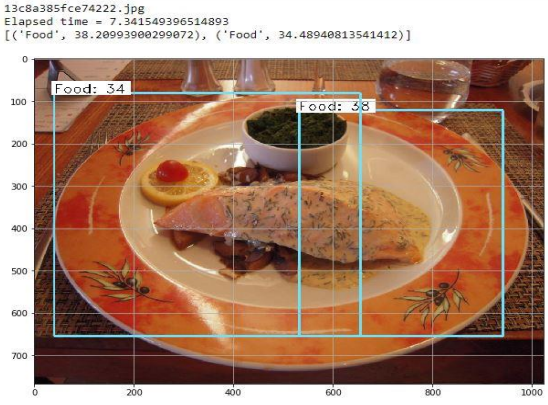


Figure 7-11: Sample Outputs for Object Detection

3.2. Statistical Measurement

The Mean Average Precision (mAP) of the system is 0.457 or 45.7%

Due to the lack of training, the score is too low. By training the system more, the score can be increased to a better position.

4. CONCLUSION

Image object detection is now-a-days a very important thing according to our need in the era of this Machine Learning. And the faster R-CNN algorithm does a very good job and better job than any other algorithms do. More works on image object detection using this algorithm can further be done to make the system more efficient.

5. REFERENCES

[1] Chen, X. and Gupta, A. *An Implementation of Faster RCNN with Study for Region Sampling*. [online] arXiv.org. Available at: https://arxiv.org/abs/1702.02138.

[2] Ren, S., He, K., Girshick, R. and Sun, J., *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv.org. Available at: https://arxiv.org/abs/1506.01497.

[3] Simonyan, K. and Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [online] arXiv.org. Available at: https://arxiv.org/abs/1409.1565

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition.* [online] arXiv.org. Available at: https://arxiv.org/abs/1406.4729