Object Detection:

A deep learning approach

SE-801

Md. Nazmul Haque
Bachelor of Science in Software Engineering
Institute of Information Technology, University of Dhaka
Roll: 0635
Registration Number: 2013-312-017
Session: 2013-14

Institute of Information technology

University of Dhaka

Date of submission: 12/20/2017

Contents

Chapter 1

Introduction

Object detection is one of the basic application domains in the field of image processing and computer vision. It has been and even now is an extensive research area for many decades. Nowadays, with the increase of image related sites like Facebook, Pinterest etc., fast and accurate object detection systems are rising in demand. The common proceeding of the schemes is that, given some knowledge about the appearance of certain objects, one or more images are examined in order to detect which objects are present and localizing those objects with bounding boxes or some manners. As every application has some specific requirements and constraints, it has led to a rich diversity of algorithms.

In order to give an introduction into the topic, I have segmented into some sub sections like overview of object detection, several areas of applications, different types of requirements and constraints.

1.1. Overview

Detecting objects in images is one of the primary areas of application of image processing and computer vision. While the term “object detection” is widely used, it is worthwhile to take a closer look what is meant by this term. Essentially, most of the schemes related to object detection have in common that one or more images are examined in order to evaluate which objects are present and where. To this end they usually have some knowledge about the appearance of the objects to be searched (the model, which has been created in advance). As a special case appearing quite often, the model database contains only one object class and therefore the task is simplified to decide whether an instance of this specific object class is present and, if so, where. On the other hand, every application has its specific characteristics. In order to meet these specific requirements, a rich diversity of algorithms has been proposed over the years.

1.2. Application areas

One way of demonstrating the diversity of the subject is to outline the spectrum of applications of object recognition. This spectrum includes industrial applications (here often the term “machine vision” issued), security/tracking applications as well as searching and detection applications. Some of the areas are described below:

1.2.1. Classification

Image characterization alludes to the undertaking of extricating data classes from pictures. It is an errand of allocating input image into one name from a settled arrangement of classifications. It is one of the essential issues in image preparing and computer vision that, notwithstanding its effortlessness, has an expansive assortment of handy applications.

1.2.2. Position measurement

Mostly in industrial applications, it is necessary to accurately locate the position of objects. This position information is, e.g., necessary for gripping, processing, transporting or placing parts in production environments. As an example, it is necessary to accurately locate electrical components such as ICs before placing them on a PCB (printed circuit board) in placement machines for the production of electronic devices (e.g., mobile phones, laptops, etc.) in order to ensure stable soldering for all connections. The position of the object together with its rotation and scale is often referred to as the object pose.

1.2.3. Counting

Some applications demand the determination of the number of occurrences of a specific object in an image, e.g., a researcher in molecular biology might be interested in the number of erythrocytes depicted in a microscope image.

1.2.4. Sorting

To give an example, parcels are sorted depending on their size in postal automation applications. This implies a previous identification and localization of the individual parcels.

1.3. Requirements and Constraints

Requirements and constraints are one of the two fundamental components in object detection. Each application imposes several requirements and constraints. Some are described in bellow-

1.3.1. Evaluation time

Data has to be processed in real time in real life industrial applications. For example, Bangladesh has completed biometric registration. In this scenario, evaluation time is a key feature. And Of course, evaluation time strongly depends on the number of pixels covered by the object as well as the size of the image area to be examined.

1.3.2. Accuracy

In some applications, the object position has to be determined very accurately: error bounds must not exceed a fraction of a pixel. If the object to be detected has sufficient structural information sub-pixel accuracy is possible, e.g., the vision system of SMD placement machines is capable of locating the object position with absolute errors down to the order of 1/10th of a pixel. Again, the number of pixels is an influence factor: evidently, the object covers the more pixels the more information is available and thus the more accurate the component can be located. During the design phase of the vision system, a trade-off between fast and accurate recognition has to be found when specifying the pixel resolution of the camera system.

In contrast to that, industrial applications usually offer some degrees of freedom, which often can be used to eliminate or at least reduce many variances, e.g., it can often be ensured that the scene image contains at most one object to be detected, that the viewpoint and the illumination are well designed and stable, and so on. On the other hand, industrial applications usually demand real-time processing and very low error rates.

1.4. Objective and Scope

Object detection is one of the most challenging sectors in today's world. If we want to detect objects accurately, it will take lot of times. On the other hand, if we want to detect objects quickly, it has a great chance to misclassify the objects. It is one kind of speed-accuracy tradeoff. Therefore, my objective is basically finding the answers of two question-

i. What is the object?
ii. Where is it located?

I have used MSCOCO 2014 and PASCAL VOC-2007 datasets for training purpose. Therefore, my scope detecting objects is MSCOCO and PASCAL VOC-2007 datasets that considers 'most salient features' in an image. 'Most salient features' means MSCOCO's and PASCAL VOC-2007's ground truth boxes that has objects. Therefore, I only consider those ground truth boxes.

1.5. Research Questions

Research question is one of the fundamental part of research study. Research questions are given bellow-

i. How a system automatically detects different objects in an image?
ii. How a system automatically localizes different objects in an image?
iii. What features are important for containing an object?
iv. How we differentiate foreground and background images?
v. What type of models are appropriate for MSCOCO and PASCAL VOC-2007 datasets?

1.6. Conclusion

In this chapter, I have discussed the intuition and basic overview of object detection, so may application areas, constraints as well as requirements, accuracy-speed tradeoff. I have also discussed objective and scope, and lastly research questions.

Chapter 2

Background Study

2.1. Introduction

Object detection systems that can detect objects speedily and precisely are rising in demand because of the rising of enormous images containing many objects and various people containing applications. It is a common task in computer vision and refers to the determination of the presence or absence of features in image data and that can be further classified as belonging to one of a pre-defined set of classes. It includes not just identifying and classifying each object in a picture yet additionally confining every one by drawing the fitting bounding box around it. For this reason, object detection is going to harder than image classification. The main objective of detection is to differentiate objects from the background. Normally, objects must be identified against jumbled, uproarious background and different objects under various light and difference conditions. Appropriate feature representation is a critical advance as it enhances execution by separating the object from the background or different objects in various lightings and situations.

The different nature of each application, its specific requirements, and constraints are some reasons why there exist so many distinct approaches to object detection. In the modern age, machine learning have become normal requirement for every application. There is no "general-purpose-scheme" applicable in all situations, simply because of the great variety of requirements. Instead, there are many different approaches, each of them accounting for the specific demands of the application context it is designed for. Nevertheless, a categorization of the methods and their mode of operation can be done by means of some criteria. Some of these criteria refer to the properties of the model data representing the object, others to the mode of operation of the detection scheme and that is why there have so many research in these fields.

In last few years, there have been many types of research conducted based on object detection. For this reason, I have studied different research methodology related to this research. Rest of this chapter will describe some important classical approach along with deep learning technique.

2.2. Classical Approach:

There have heaps of understood established strategies. For example, Local Binary Pattern (LBP) [1], which has picked up ubiquity because of their computational simplicities and better correct nesses though it is exceptionally delicate to uniform and close uniform locale. This problem can be handled by local ternary pattern (LTP) [7] and Completed local binary pattern (CLBP) [8] though CLPBP performs better than LTP because of rotation invariant. CENsus Transform hiSTogram (CENTRIST) [2] has become popular by joining Spatial Pyramid (SP) structure. Gabor [3], Scale Invariant Feature Transform (SIFT) [4] which use local region by detecting a key point, Histogram of Oriented Gradient (HOG) [5], GIST [6] etc. which are used for detecting objects. But in recent, Completed CENTRIST (cCENTRIST)[9] and Ternary CENTRIST (tCENTRIST) [10] have achieved high accuracies and precision for object detection. In the following part, I will describe in detail about some of the object detection approaches.

2.2.1. SCALE INVARIANT FEATURE TRANSFORM (SIFT)

When an image has different scales and rotations, SIFT is more useful on that case. The algorithm is given bellow-

i. By generating scale space, we need to represent of the original image because of scale invariance.
ii. Using DoG(Difference of Gaussians) in Lowe's SIFT paper[1], find interest point (key point).
iii. By eliminating edges and low contrast regions, we get rid of bad key points.
iv. For each key point, calculate orientation that remove the effect of orientation. That is called rotation invariant
v. With the help of scale and rotation invariance, we calculate SIFT feature that can uniquely identify features.

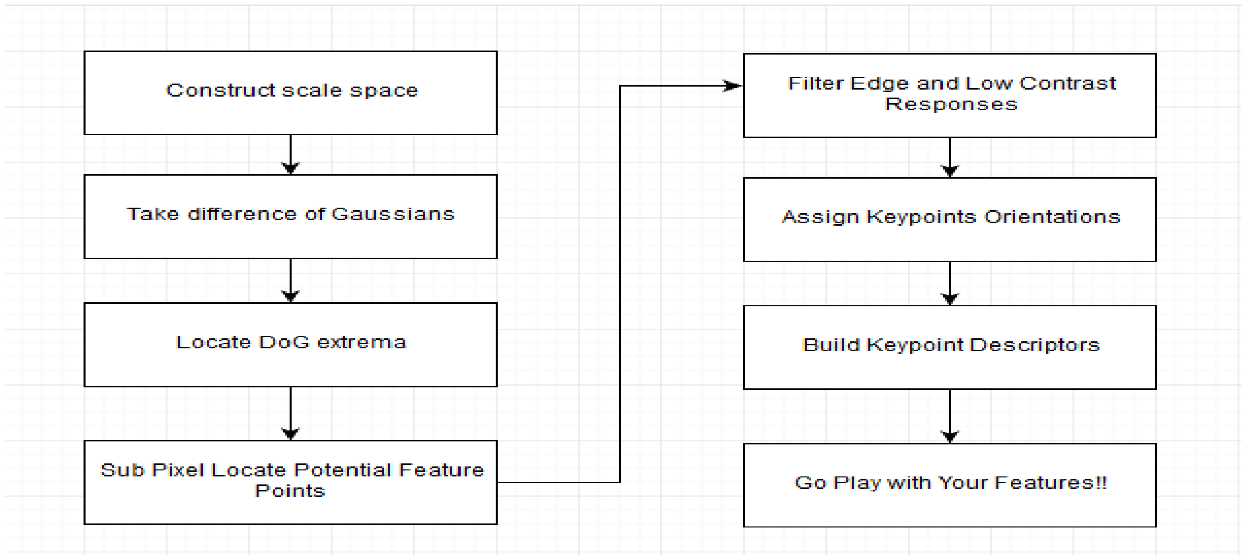

Construct scale space
Take difference of Gaussians
Locate DoG extrema
Sub Pixel Locate Potential Feature Points
Filter Edge and Low Contrast Responses
Assign Keypoints Orientations
Build Keypoint Descriptors
Go Play with Your Features!!

Figure 1: SIFT

2.2.2. Histograms of Oriented Gradient (HOG)

Histograms of Oriented Gradient (HOG) is well known as a feature descriptor. The basic intuition of this algorithm is local object appearance and shape. The calculation procedure of HOG feature descriptor is given bellow-

i. In image may be any size. But 64x128 patch is used for HOG feature descriptor. So first, crop the image and resized to 64x128.
ii. Then we need to calculate vertical and horizontal gradients by using kernels [-1, 0, 1] and calculate the histogram of gradients.
    - Magnitude of gradient, $g = \sqrt{g_x^2 + g_y^2}$
    - Direction of gradient, $\theta = \arctan \frac{g_y}{g_x}$
iii. For calculating histogram of gradients, image is divided into 8x8 cells. Histogram is a vector of 9 bins that represents angles (0,20,40,60,80,100,120,140,160). The appearance of all pixel in the 8x8 cells are represented to the 9 bin histogram.
iv. Normalize the histogram into 16x16 block so that they are not affected by lighting variations.
v. Then calculate HOG feature vector using 36x1 feature vector
    - For 16×16 block
        i. Half of block size, 7 x 15 = 105 positions.
        ii. Obtain a 36×105 = 3780 dimensional vector that is used for object detection pattern.

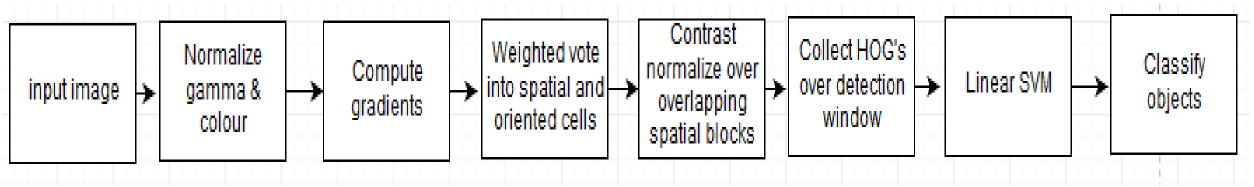

input image
Normalize gamma & colour
Compute gradients
Weighted vote into spatial and oriented cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Linear SVM
Classify objects

Figure 2: HOG

2.2.3. Global Image Search Tree (GIST))

GIST is a low dimensional feature representation of an image, which does not need any form of segmentation. Given an input image, the algorithm is given bellow-

i. Producing 32 feature maps with the convolution of 32 Gabor filters (4 scales, 8 orientation).
ii. Splitting each feature map into 16 regions with square grid.
iii. Averaging feature values within each region.
iv. Merging 16 averaged values of feature maps that brings 16x32=512 GIST descriptor that is used for object detection as a pattern.

2.2.4. Point detectors

Point indicators are utilized as a part of discovering some helpful focuses in pictures which have an expressive surface in their particular territories [11]. A helpful interest point is one which is invariant to changes in enlightenment and camera perspective. Some normally utilized interest point indicators incorporate Moravecs finder, Harris identifier, KLT locator, SIFT finder [4].

2.2.5. Background Subtraction

One reliable strategy for object detection includes building a portrayal of the scene known as the background model and discovering deviations from the model for every approaching edge in the video pictures. Any huge change in a picture district from the foundation show is noted down as a moving article. The pixels in the areas of the experiencing change are set apart as moving items and saved for additionally handling. This procedure is alluded to as the foundation subtraction. There are different techniques for foundation sub-footing as examined in the overview [12] are Frame Differencing Region-based (or) spatial data, Hidden Markov models (HMM) and Eigen space disintegration [11]. There have two methodologies:

- Recursive Algorithm:

Recursive methods for foundation sub-footing [13] [14] do not keep up a cradle for background estimation. This strategy re-cursively refreshes a solitary foundation show in view of each information outline. In this circumstance, input outlines from far off past could affect the present foundation show being dissected. Recursive procedures require a littler measure of capacity as contrasted and non-recursive strategies, yet any blunder out of sight model can have a huge impact for an any longer timeframe. This system incorporates different techniques, for example, rough middle, versatile background, Gaussian of mixture [1].

- Non Recursive Algorithm:

For evaluating changes in the background, a sliding window approach is used for non-recursive technique [11] [12]. It collects a buffer of the previous video frames and then evaluating the background technique based on the temporal variation of each pixel within the buffer. For simple background subtraction, we take the distance between present and background image for the purpose of calculate motion detection mask. Then a threshold is taken that identifies a pixel is foreground (difference of pixel value>=0) or background (difference of pixel value<0).

2.2.6. Local Binary Pattern (LBP)

LBP has been a standout amongst the most helpful visual descriptors that is utilized for grouping in the field of PC vision. It is viewed as the most intense descriptor for texture classification. LBP for the most part concentrates on the neighborhood structure of a picture. Therefore, it is utilized for some applications, for example, outward appearance investigation [11], confront identification and acknowledgment [11], and so on. LBP is an exceptionally well known descriptor due to computational straightforwardness, which examines pictures continuously, heartiness and gives better grouping and identification execution in numerous areas. To enhance its execution, heaps of work have been done in light of LBP based technique, which is fit for various applications. For example, dominant LBP [13], use most frequently occurred patterns to capture descriptive textural information, subsidiary based LBP [14], focus symmetric LBP [15] use image pixels as descriptors and then using random forest algorithm, and so on. LBP discovers the nearby structure from a picture by contrasting focus pixel and its neighbor pixel. On the off chance that the force of the middle pixel is more prominent than its neighbor or equivalent, it indicates 1, generally 0. For every pixel, you will wind up with some twofold numbers, for example, 11001100. For 8 neighbor pixels you will discover 28 conceivable mixes. Figure 3 speaks to 3 × 3 neighborhood, subsequently; LBP really works along these lines.

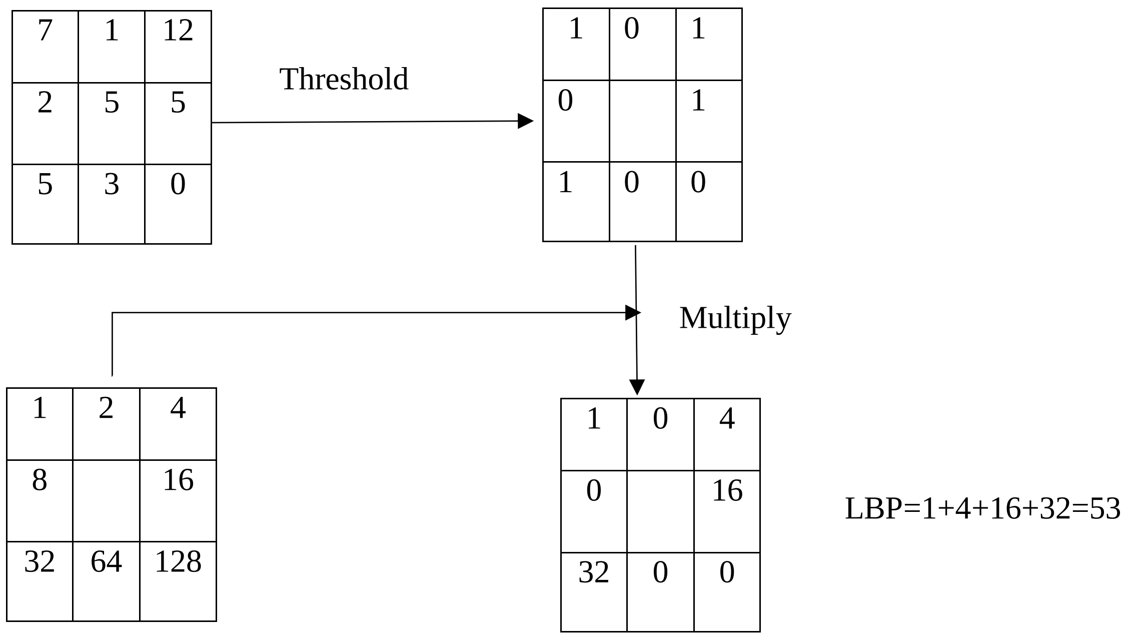


Figure 3: LBP, available at *http://what-when-how.com/face-recognition/local-representation-of-facial-features-face-image-modeling-and-representation-face-recognition-part-1/*

2.2.7. Completed Local Binary Pattern (CLBP)

CLBP is likewise fundamentally the same as LBP. Primary contrast between these two strategies is CLBP considers both the signs (CLBP S) and the extent (CLBP M) data, which originate from the contrasts between each neighboring pixel to the focal pixel. In CLBP parallel code is additionally created (CLBP C). Figure 4 demonstrates the fundamental advances and the estimation of CLBP S and CLBP M segments. For this, I consider 3 × 3 picture area. Here, d=

contrast, s= signs. On the off chance that d>= 0 then s=1 generally s=0. At last, - 1 is supplanted by 0 in CLBP S and CLBP M changed over as a double number organization.

19	94	26
87	48	35
104	29	135

a) 3x3 neighborhood

-29	46	-22
39		-13
56	-19	87

b) the differences

-1	1	-1
1		-1
1	-1	1

c) Sign component

29	46	22
39		13
56	19	87

d) Magnitude component

Figure 4: CLBP, available at *http://ieeexplore.ieee.org/document/5427137/?part=1*

2.2.8. Census Transform Histogram (CENTRIST)

Based on the concept of Census Transform (CT), CENTRIST is a visual descriptor for detecting scene and place categories. It compares pixel values like LBP as well as intensity value with it's 8-neighbor and then produce 8-bit CT values. CENTRIST is fundamentally the same as LBP, yet the principle contrasts amongst LBP and CENTRIST are: LBP performs insertion while considering the corner pixels yet CENTRIST considers the corner pixels as the corner pixels seem to be. Both LBP and CENTRIST depend on the complexity of pixel force in a 8 neighborhoods. From Figure 5, we can see that the way toward processing the CT value is comparative with LBP. CENTRIST uses Spatial Pyramid Matching (SPM) plan to catch the worldwide structure of a picture.

To maintain a strategic distance from the ancient rarities which are made by the non-covering blocks in the conventional SP, CENTREST utilizes add up to 31 pieces. Among them, 1 block from level 0, 5 squares from level 1 and 25 pieces originate from level 2. CENTRIST calculates CT values from every 25 pieces.

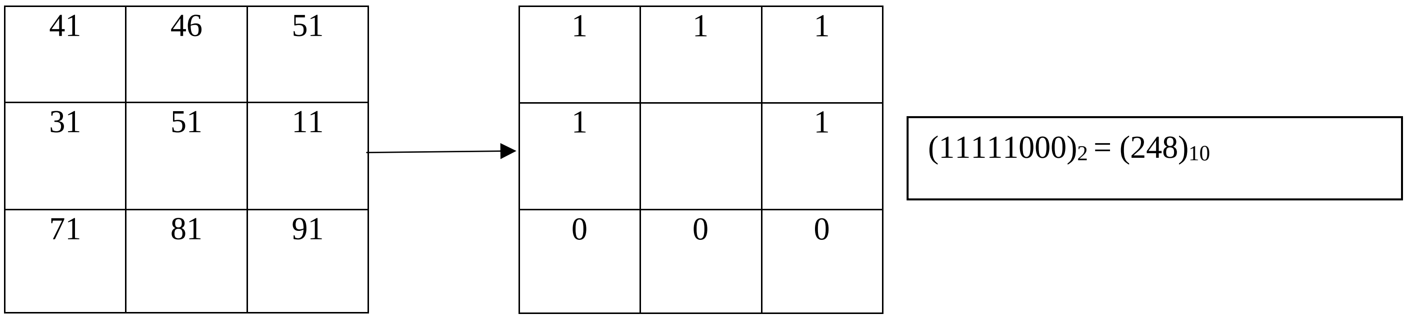


Figure 5: Census Transform (CT), available at *https://software.intel.com/en-us/sample-census-transform-census-transform-algorithm-overview*

2.2.9. Local Ternary Pattern (LTP)

LTP is introduced to manage the variances of intensity and similar to LBP. LTP considers a threshold value like t= -5 and +5. To lessen the extent of feature vector, LTP utilizes two twofold codes, upper and lower patterns. Two histograms are connected to represent the feature vector. Figure 6 represents basic steps of LTP.

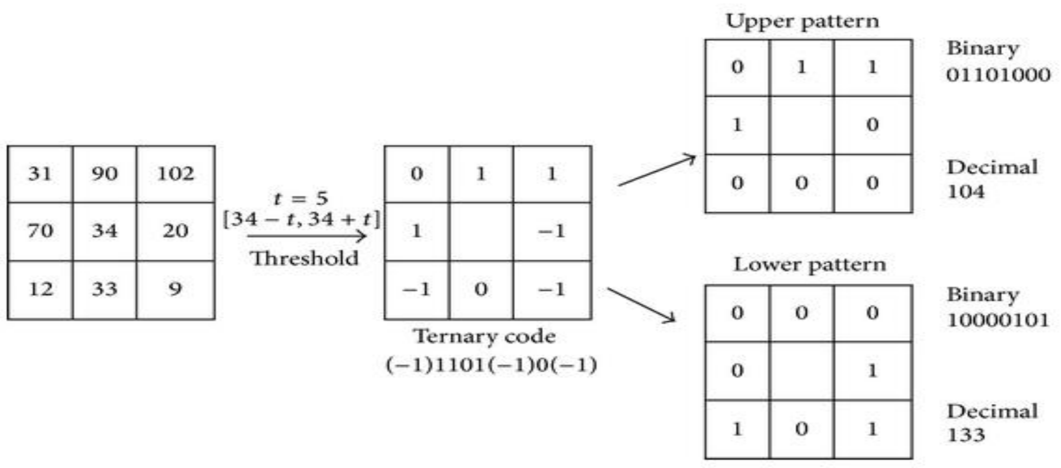


Figure 6: Local Ternary Pattern (LTP), designed by rayryeng

2.2.10 Completed CENTRIST (cCENTRIST)

Completed CENTRIST is introduced based on the idea of CLBP and CENTRIST. It is a combination of CLBP and CENTRIST. It considers corner pixel for generating CT values and CLBP properties like sign, magnitude and center pixel information. It also considers each block spatial pyramid (SP) and for each block, a 3D histogram is generated. After completing all steps, Histograms are concatenated. For constructing final feature vector, Principle Component Analysis (PCA) and it reduces noise. After executing PCA, objects are classified. The whole procedure of cCENTRIST algorithm is given bellow-

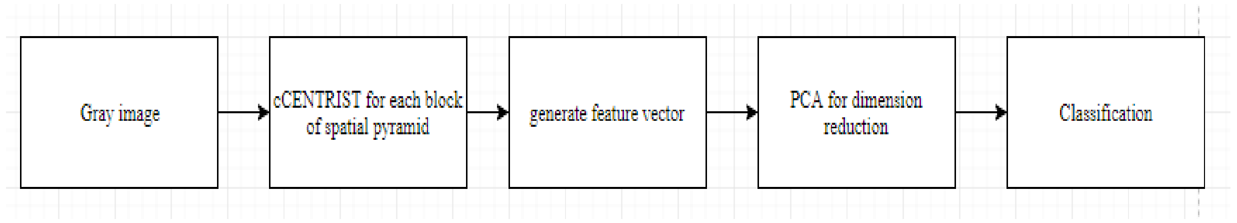

Gray image
cCENTRIST for each block of spatial pyramid
generate feature vector
PCA for dimension reduction
Classification

Figure 7: cCENTRIST algorithm

2.2.11. Ternary CENTRIST (tCENTRIST)

Ternary CENTRIST (tCENTRIST) is introduced based on the idea of Local Ternary pattern (LTP) and CENTRIST. It is the aggregation of both LTP and CENTRST. It also considers spatial pyramid (SP). LTP is then calculated -

- Based on spatial pyramid (SP) and
- Generate two types of histograms for upper and lower codes of LTP.

After generating histograms, two histograms are concatenated for building single histogram. Lastly final feature vector is generated by PCA.

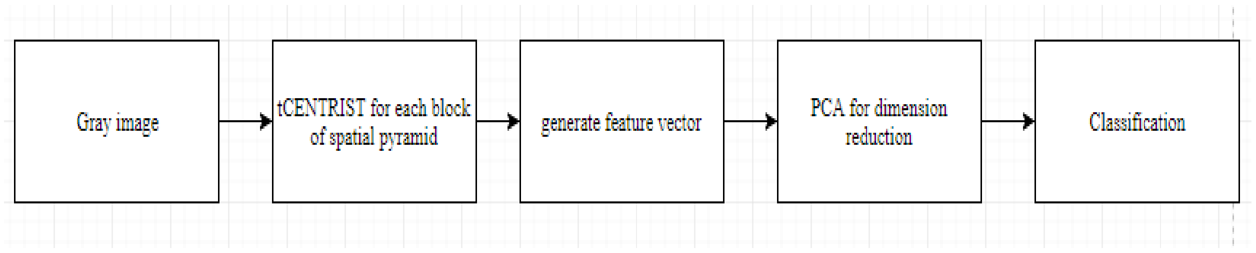

Gray image
tCENTRIST for each block of spatial pyramid
generate feature vector
PCA for dimension reduction
Classification

Figure 8: tCENTRIST algorithm

Although several classical approaches exist for object detection, but recently, deep learning is used in this field as, Deep Convolutional Neural Networks (CNNs) have shown better performances for different object detection. Classical feature extraction methods are quite old and its performance is not satisfactory in modern age on big data. Therefore, deep learning works well on both supervised and unsupervised data to build such an application. Therefore, in this research I have focused on deep learning to acquire better performance in this sector. In this paper, my goal is to apply an appropriate deep learning model for object detection that give better performance.

2.3. Deep Learning Approach

Object detection is an exceptionally difficult and challenging area on the field of deep learning. It is not just about answering the question of 'what?' it's also about answering the question of 'where?'. These two questions bring in several challenges. The recognition that we called 'what?' part needs to be invariant to image transformations whereas the localization that we called 'where' part needs to recover those transformations.

Deep ConvNets [17] recently have significantly improved object detection and image classification accuracy. In this section, I review various well-known Deep ConvNets, namely, Region based Convolutional Neural Networks methods including R-CNNs [18], Fast R-CNNs [19], and Faster R-CNNs [20].

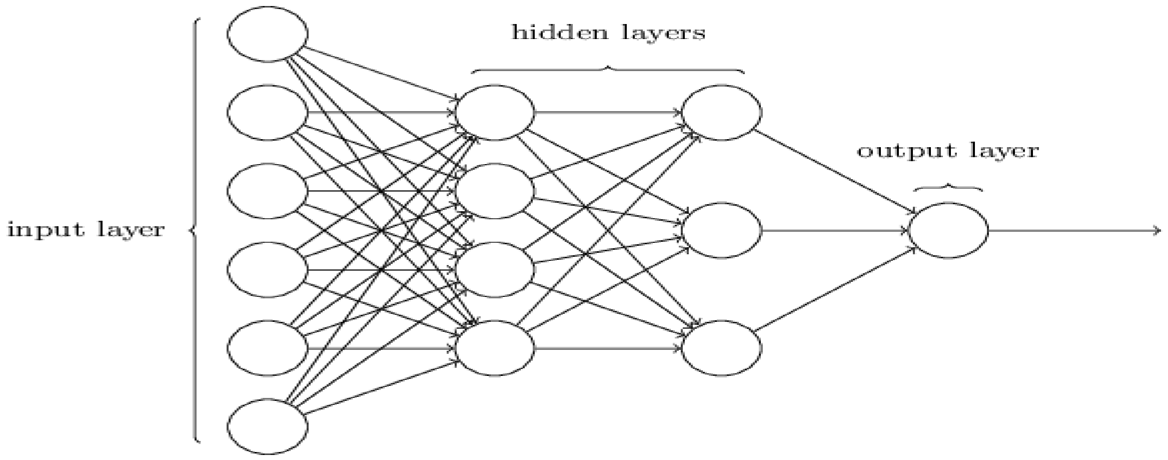


Figure 9: A deep learning architecture, available at *https://www.ibm.com/developerworks/library/cc-machine-learning-deep-learning-architectures/index.html*

2.3.1. Different steps of Deep learning

The Conv layer is the center building piece of a Convolutional Network that does a large portion of the computational heavy lifting. In this section, I will discuss Convolutional layer, Pooling layer and Fully connected layer along with some methods.

2.3.1.1. Convolutional layer

Convolutional layer is the center piece of a CNN display. It comprises of an arrangement of learnable filters or portion and it reaches out through profundity of the information volume. Normally, pictures have same sorts of property that implies one part in the picture is the same as some other part. As an outcome, in the event that we learn one part in the picture, we can likewise be connected to different parts of the picture. For this, one can utilize similar highlights at all areas. Assume, we have a 96×96 image and we need to extract features from this picture to convolve with the scholarly 8x8 features to get a different activation value from image. For this, we can begin from (1, 1), (1, 2),.. (89, 89) to get features from each 8×8 region and each time we can remove 8×8 patches to get the feature activation values. This would bring about 100 sets 89×89 convolved features after convolutional layer. Figure 8 demonstrates how convolution is finished.

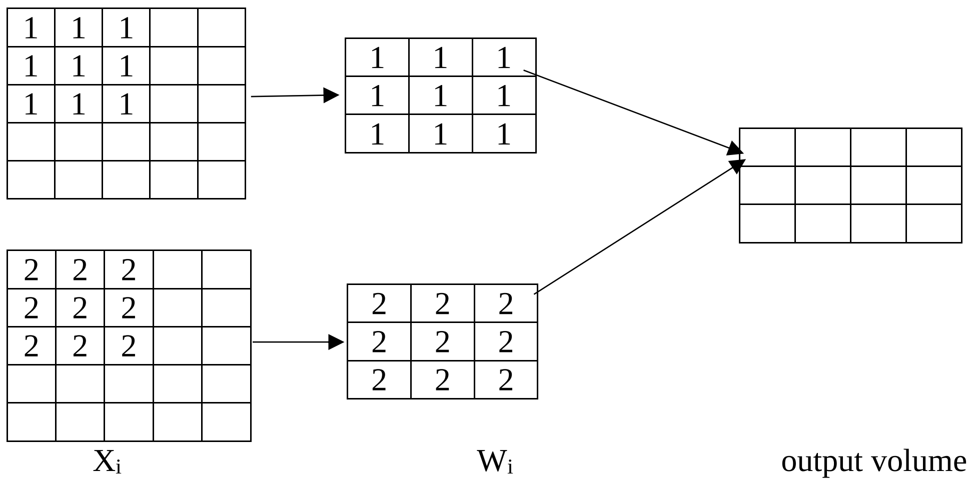


Figure 10: Convolutional layer, Source: *http://cs231n.github.io/convolutional-networks/*

2.3.1.2. Max Pooling layer

It is normal to intermittently embed a Pooling layer in the middle of progressive Conv layers in a ConvNet design. Its capacity is to logically lessen the spatial size of the portrayal to decrease the measure of parameters and calculation in the system, and henceforth to likewise control overfitting. The Pooling Layer works autonomously on each profundity cut of the information and resizes it spatially, utilizing the MAX operation. The most widely recognized shape is a pooling layer with channels of size 2x2 connected with a walk of 2 downsamples each profundity cut in the contribution by 2 along both width and tallness, disposing of 75% of the actuations. Each MAX operation would for this situation be taking a maximum more than 4 numbers (minimal 2x2 area in some profundity cut). The profundity measurement stays unaltered. All the more for the most part, the pooling layer:

Suppose, we have a volume of size $W1 \times H1 \times D1$

Requires two hyper parameters: spatial degree F, The walk S,

Produces a volume of size $W2 \times H2 \times D2$ where:

- $W2=(W1-F)/S+1$
- $H2=(H1-F)/S+1$
- $D2=D1$

It is significant that there are just two regularly observed varieties of the maximum pooling layer found by and by: A pooling layer with F=3, S=2F=3, S=2 (additionally called covering pooling), and all the more generally F=2, S=2F=2, S=2. Pooling sizes with bigger responsive fields are excessively ruinous.

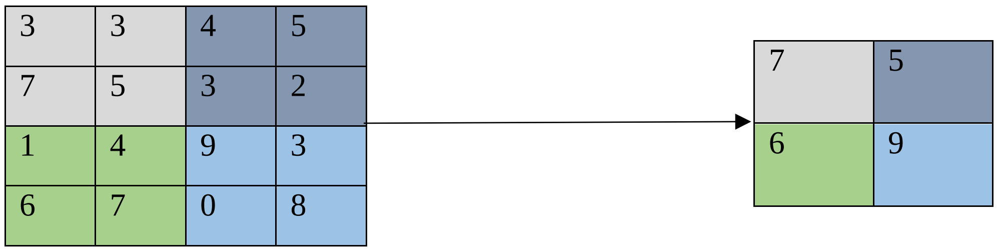


Figure 11: Pooling layer Source: *http://cs231n.github.io/convolutional-networks/*

2.3.1.3. Fully connected layer (FC layer)

In fully connected layer, all previous activation layers are fully connected with fully connected layer. Figure 10 show fully connected (FC) layer.

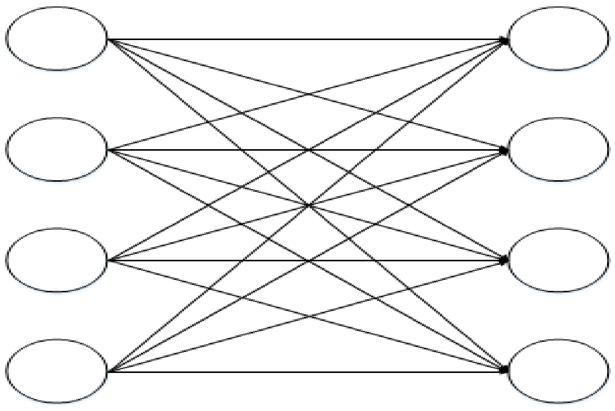


Figure 12: Fully connected layer, Source: *http://cs231n.github.io/convolutional-networks/*

And there have some concepts when we want to train or test.

2.3.1.4. Reducing Overfitting

In machine taking in, the most imperative and normal undertaking is to fit a model for preparing set of information with the goal that we can ready to make dependable expectations on general untrained information. Overfitting mostly happens when factual model depicts commotion rather than the fundamental relationship or model is excessively mind boggling, for example, numerous parameters with respect to the quantity of perceptions. While overfitting happens, model will for the most part have poor prescient execution. Accordingly, it can improve minor change in the information. When you work with vast datasets, in some cases it is very difficult to learn such a significant number of parameters without extensive overfitting in light of the fact that it can hamper your model prescient execution. Beneath, we will depict two essential courses in which we can keep from overfitting.

2.3.1.5. Data Augmentation

To lessen overfitting issue from image data, the most widely recognized technique or practice is to artificially develop the dataset utilizing label-preserving transformations. To decrease this issue, we utilize two different types of data augmentation, for this, we have to change pictures from original pictures with next to very little calculation, so changed images don't should be put away on disk.

To start with, methods for data augmentation comprise of generating image translations and horizontal reflections. From the 256×256 images we can extricate arbitrary 224×224 patches and preparing our system on these removed patches. Along these lines, our training set is expanded by a factor of 2048 and coming about preparing cases are associated. Without this a network suffer from overfitting issue and a network execution is hampered. Accordingly, you need to power to utilize substantially littler networks. At test time, the network makes an expectation by removing complete ten patches those are five 224 × 224 patches (the four corner patches and the inside fix) and additionally their even reflections. At last, by utilizing these ten patches, softmax layer makes a normal prediction. Second, the method for information expansion is changing the forces of the RGB directs in preparing pictures and all through the ImageNet preparing set, we perform PCA on the arrangement of RGB pixel esteems. We likewise utilize products rule segments to each preparation images with magnitudes.

2.3.1.6. Dropout

In machine learning frameworks deep neural nets with numerous parameters are all the more capable. Overfitting is one of the principal issues in these networks yet we can decrease test errors by joining the forecast of numerous different models. Now and then it is excessively costly in view of huge system. Notwithstanding, there is an extremely effective model blend amid preparing and it costs just factor of two. Dropout techniques diminish overfitting issues by setting to zero each shrouded layer yield with likelihood 0.5. Moreover, dropped out neurons don't add to the forward and back proliferation pass. Here, each time inputs show neural network tests as a different design and every one of those engineering share weights. This additionally lessens complex co-adjustments of neurons in light of the fact that a neuron can't depend on alternate neurons. This significantly lessens overfitting and gives real upgrades over other regularization strategies.

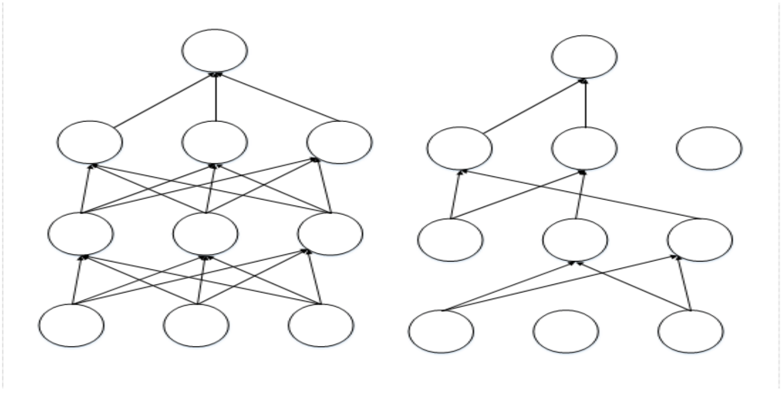


Figure 13: Dropout(Left: original, Right: dropout), source *http://cs231n.github.io/convolutional-networks/*

There have many deep learning models that provide state of the art performance. In the next section, I will describe those models.

2.3.2. R-CNN

The Region-based Convolutional Neural Network(R-CNN) [17] uses a deep ConvNet to detect given object proposals. It achieves great accuracy in both training and testing time but is time-consuming. It is first trained on object proposals and fine-tunes a ConvNet with softmax regression layer at last. Then by replacing the last layer with support vector machine(SVM) and using the features from fine-tuned ConvNet, the system is further trained for object detection. Finally, it performs bounding-box regression using four values. The system takes a long time to extract features from each input image and store the features in a hard disk, which also takes up a large amount of space. At test-time, the detection process takes 47s for one image (with VGG16, on a GPU) because of the slowness of feature extraction.

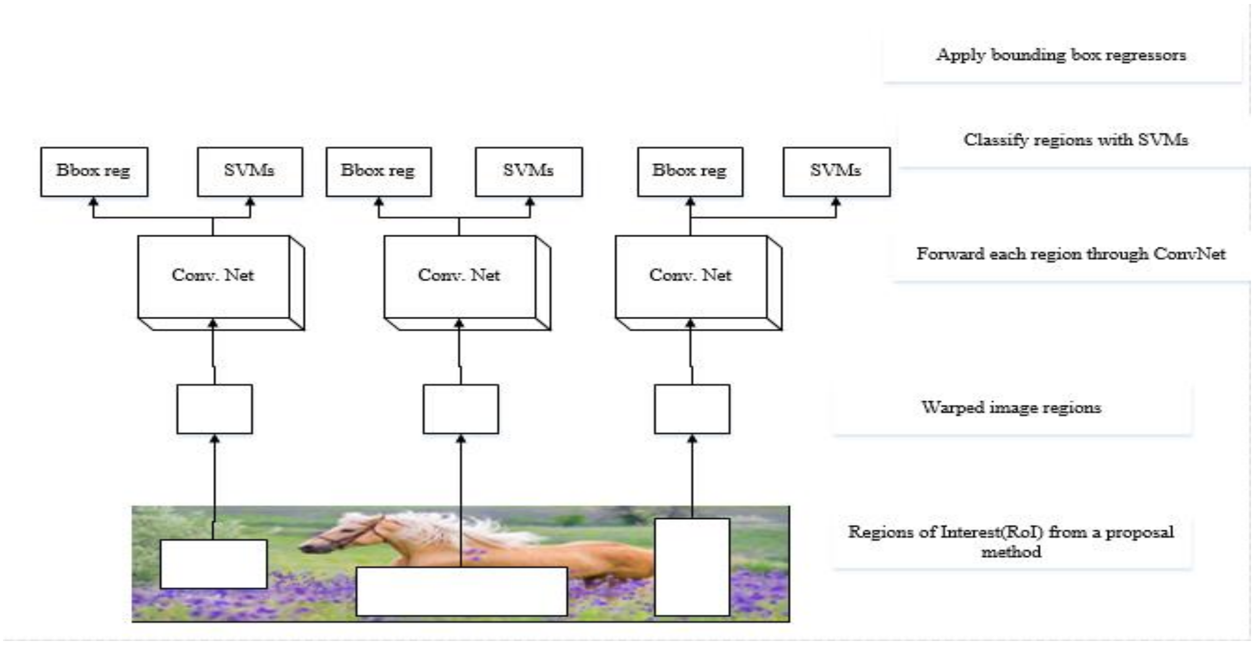

Apply bounding box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Forward each region through ConvNet
Conv. Net
Conv. Net
Conv. Net
Warped image regions
Regions of Interest(RoI) from a proposal method

Figure 14: R-CNN architecture

2.3.2. Fast R-CNN

The main reason for R-CNN being slow is that it processes each object proposal independently without sharing computation. Fast R-CNN [19] tries to share the features between proposals. It uses ConvNet just only once on the whole input image. At test-time, it only extracts features once per image and uses region of interest(ROI)-pooling to extract features from the convolution feature map for each object proposal. It also uses two multi-task losses, i.e. classification loss and bounding box regression loss. Based on the two improvements, the framework is trained end-to-end. The processing time for each image significantly reduced to 0.3s compared to R-CNN(47s).

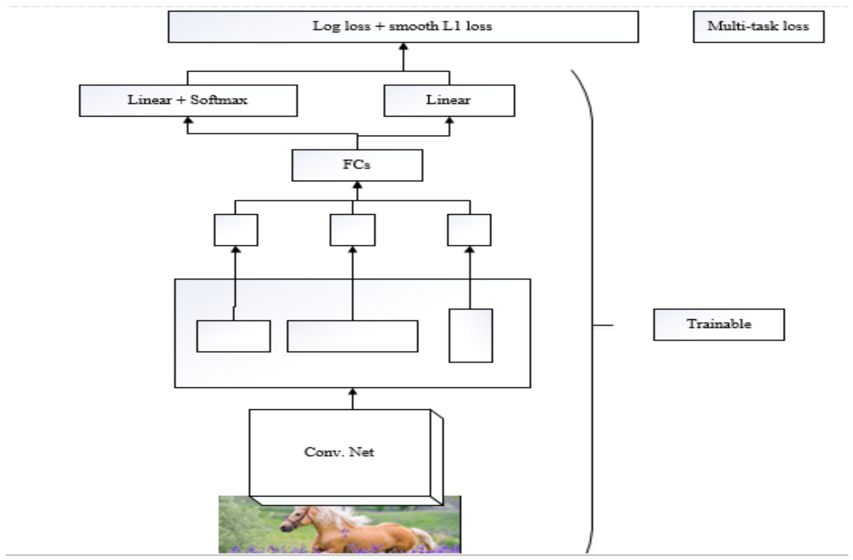
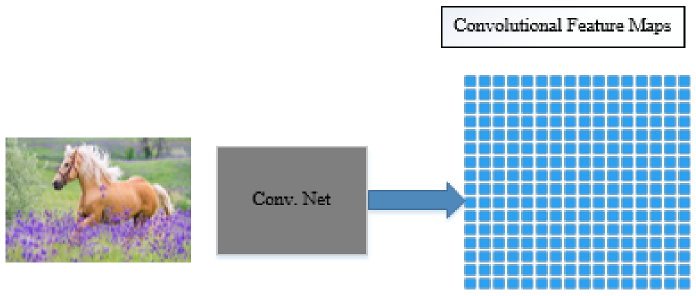


Figure 15: Fast R-CNN architecture(training)

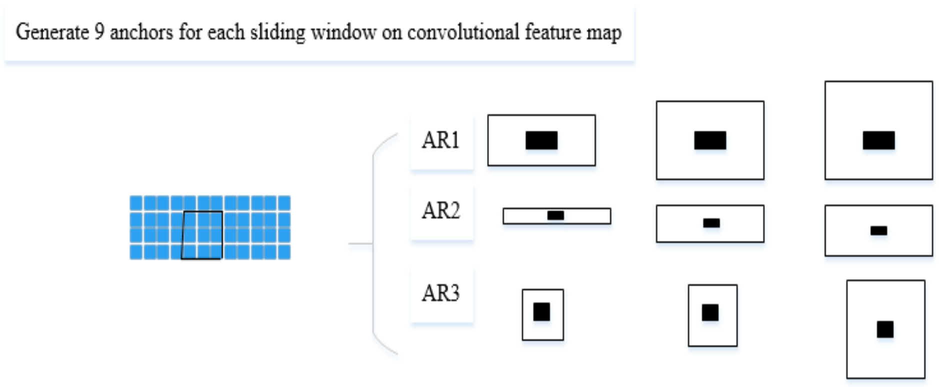
2.3.3. Faster R-CNN

ROI-pooling layer is used in Fast R-CNN that accelerates the detection network. However the region proposal step is out of the network hence remains a bottleneck, resulting in suboptimal solution and dependence on the external region proposal methods. Faster R-CNN [20] addresses this problem with the region proposal network (RPN). I have summarized the RPN in three steps.

i. Input image goes through a convolution network. After completing ConvNet on the last convolutional layer, generates a set of convolutional feature maps:



ii. Then a sliding window is selected and run spatially on these feature maps that we calculated from (i). Generally, the size of sliding window is n×n (here 3×3). For each sliding window, a set of 9 anchors are generated which all have the same center (x_a, y_a) but with 3 different aspect ratios and 3 different scales as shown below. Note that all these coordinates are computed with respect to the original image.


Generate 9 anchors for each sliding window on convolutional feature map
AR1
AR2
AR3

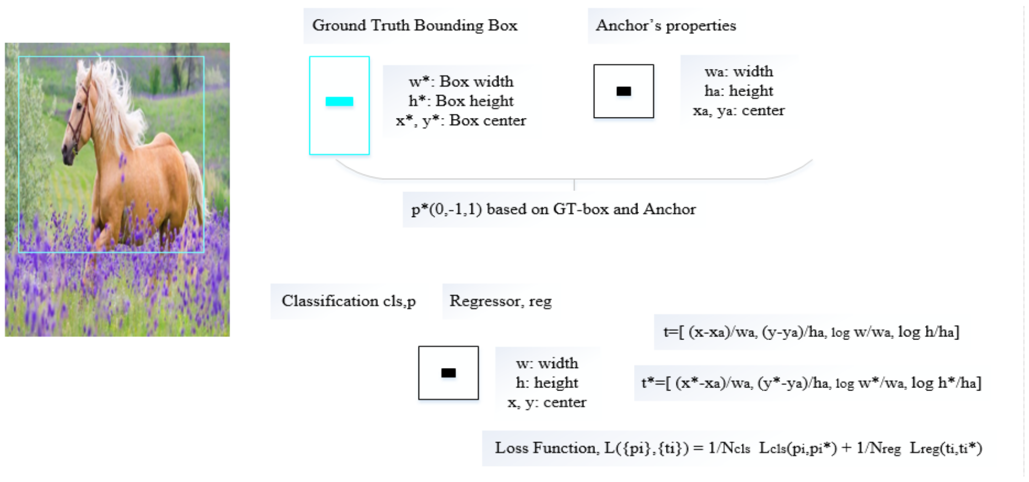
Furthermore, for each of these anchors, a value p* is computed which indicated how much these anchors overlap with the ground-truth bounding boxes.

$$P^* = \begin{cases} -1, & if\ IoU < 0.3 \\ 1, & x \geq 0.7 \\ 0 & otherwise \end{cases} \tag{2.1}$$

Where IoU is intersection over union and is defined below:

$$\text{IoU} = \frac{\text{Predicted anchor} \cap \text{GTBox}}{\text{Predicted anchor} \cup \text{GTBox}} \tag{2.2}$$

iii. Finally, the 3×3 spatial features extracted from those convolution feature maps (shown above within red box) are bolstered to a smaller network, which has two tasks: classification (cls) and regression (reg). The yield of regressor decides a anticipated bounding-box (x,y,w,h), The yield of classification sub-network is a probability p indicating whether the the predicted box contains an object (1) or it is from background (0 for no object).


Ground Truth Bounding Box
w*: Box width
h*: Box height
x*, y*: Box center
Anchor's properties
wa: width
ha: height
xa, ya: center
p*(0,-1,1) based on GT-box and Anchor
Classification cls,p
Regressor, reg
t=[(x-xa)/wa, (y-ya)/ha, log w/wa, log h/ha]
w: width
h: height
x, y: center
t*=[(x*-xa)/wa, (y*-ya)/ha, log w*/wa, log h*/ha]
Loss Function, L({pi},{ti}) = 1/Ncls Lcls(pi,pi*) + 1/Nreg Lreg(ti,ti*)

The loss function is defined over output of both sub-networks, with 2 terms and a balancing factor λ. Here is the total faster R-CNN architecture.

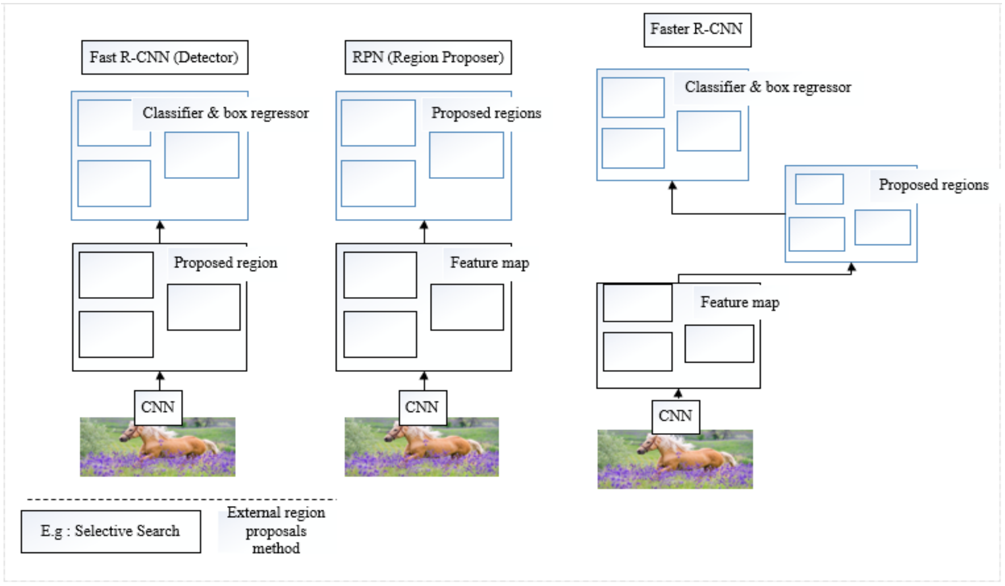


Figure 16: Faster R-CNN architecture

2.3.4. ResNet

Not at all like customary successive system designs, for example, AlexNet, OverFeat, and VGG, ResNet is rather a type of "fascinating engineering" that depends on smaller scale engineering modules (likewise called "arrange in-organize models").

The term smaller scale engineering alludes to the arrangement of "building squares" used to develop the system. A gathering of small scale engineering building hinders (alongside your standard CONV, POOL, and so on layers) prompts the large scale design (i.e., the end organize itself).

To begin with presented by He et al. in their 2015 paper[21], the ResNet engineering has turned into an original work, exhibiting that greatly profound systems can be prepared utilizing standard SGD (and a sensible introduction work) using leftover modules:

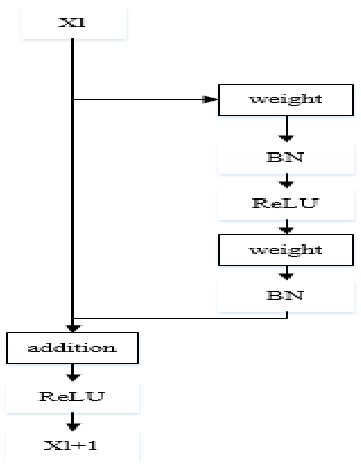


Figure 17: The residual module in ResNet as originally proposed by He et al. in 2015

To use identity mappings, accuracy can be obtained by updating the residual module, as demonstrated in their 2016 follow-up publication [22]:

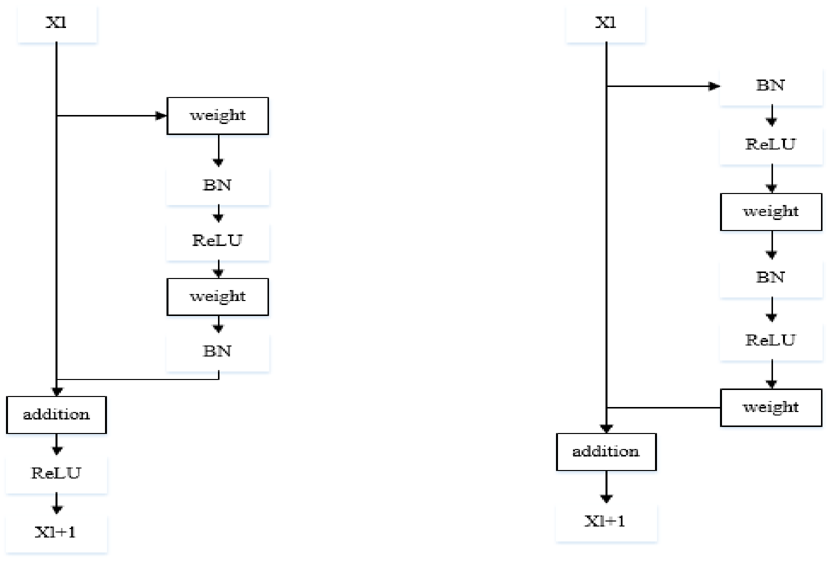


Figure 18: (Left) The original residual module. (Right) The updated residual module using pre-activation proposed by He et al. in 2015.

ResNet has more hidden layer than VGG16 or VGG19.

2.4. Conclusion

In this chapter, I have discussed several classical and deep learning approach. Here, I have also discuss a deep learning ResNet models.

Chapter 3

Literature Review

3.1. Introduction

In this chapter, I discuss about some existing object detection strategies with existing descriptors. We also describe some deep learning based work; that are used for several applications in computer vision.

3.2. Classical Approach Related Research

There is a large literature on object proposal methods based on classical approach. Comprehensive surveys and comparisons of object proposal methods can be found in [23], [24], [25]. Widely used object proposal methods include those based on grouping super-pixels (e.g., Selective Search [29], CPMC [26], MCG [27]) and those based on sliding windows (e.g., objectness in windows [35], EdgeBoxes [30]). Object proposal methods were adopted as external modules independent of the detectors (e.g., Selective Search [29] object detectors). Selective search performs better when selecting only region proposal in an image. How efficiently compute region proposal is out of scope. However, it is very efficient for fast R-CNN and faster R-CNN for detecting region proposals. EdgeBoxes method uses object boundaries estimates (obtained via structured decision forests) as feature for the scoring. Local Binary Pattern (LBP) [8] is basically used for pattern matching problems like face recognition. But in real life scenarios, object would be different shapes, poses, expressions etc. So better accuracy cannot be achieved by LBP.

3.3. Deep Learning Related Research

The R-CNN method [16] trains CNNs end-to-end to classify the proposal regions into object categories or background. For every single region proposal for every single image (that's around 2000 forward passes per image!) it requires a forward pass of the CNN (AlexNet). R-CNN has to train three unique models independently –

- to generate image features uses CNN
- the classifier that predicts the class
- the regression model to tighten the bounding boxes.

This makes the pipeline extremely hard to train because every time for every region need to use convNet separately. R-CNN mainly plays as a classifier, and it does not predict object bounds (except for refining by bounding box regression). Its accuracy depends on the performance of the region proposal module (see comparisons in [24]). There have many articles for predicting object bounding boxes [31], [34], [32], [33]. In the OverFeat method [34], a fully-connected layer is trained to predict the box coordinates for the localization task that assumes a single object. The fully-connected layer is then turned into a convolutional layer for detecting multiple class specific objects. The MultiBox methods [32], [33] generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes, generalizing the "single-box" fashion of OverFeat. These class-agnostic boxes are used as proposals for R-CNN [16]. The MultiBox proposal network is applied on a single image crop or multiple large image crops (e.g., 224×224), in contrast to our fully convolutional scheme. MultiBox does not share features between the proposal and detection networks. We discuss OverFeat and MultiBox in more depth later in context with our method. Concurrent with our work, the DeepMask method [35] is developed for learning segmentation proposals. Shared computation of convolutions [34], [36], [37], [19] has been attracting increasing attention for efficient, yet accurate, visual recognition. The OverFeat paper [34] computes convolutional features from an image pyramid for classification, localization, and detection. Adaptively-sized pooling (SPP) [36] on shared convolutional feature maps is developed for efficient region-based object detection [36], [30] and semantic segmentation [37]. Fast R-CNN [19] enables end-to-end detector training on shared convolutional features and shows compelling accuracy and speed. Indeed, even with every one of these headways, there was as yet one residual bottleneck in the Fast R-CNN process—the region proposer. As we saw, the very initial step to detecting the locations of objects is generating a bunch of potential bounding boxes(bBox) or regions of interest(RoI) to test. In Fast R-CNN, these proposals were created using Selective Search [29], a genuinely moderate process that was observed to be the bottleneck of the general procedure.

3.4. Conclusion

A lot of progress has been made in recent years on object detection due to the use of convolutional neural networks (CNNs) like R-CNN, Fast R-CNN that give state of the art result on test time but are time consuming on training phase. In this paper, I will propose Faster R-CNN model with ResNet101 that gives state of the art performance both train and test phase.

Chapter 4

Proposed Model for Object Detection

4.1. Introduction

With the success of region proposal methods and R-CNN, object detection system has gained about real time detection. However, proposals are the test time computational congestion as it relies on inexpensive features and economical inference schemes.

In this section, I described the assumptions to detect objects and the systematic procedure what I used for detecting objects in detail.

4.2. Assumptions

To detect objects, I assume the following-

- To detect an object, I consider most salient features. Most salient feature means foreground objects.
- Foreground and background objects are classified based on MSCOCO and PASCAL VOC-2007 dataset's bounding boxes. The ground truth boxes of their dataset's are considered as foreground objects.
- This model perform well when trained and tested using single scale images. Single item scales are those with which only one item is measured.
- When anchors are generated on multiscale, it produces wrong output. In addition, give best result when anchors are generated on single scale.

4.3. Proposed Faster R-CNN

In this section, I will describe an elegant and effective solution, Faster R-CNN where proposal computation is cost free. Faster R-CNN is composed of two modules-

- Deep fully convolutional network that proposes regions.
- Fast R-CNN detector that uses the proposed regions.

It is a single and unified network for object detection. The region proposal network (RPN) modules tells second modules (Fast R-CNN) where to look. To look in detail about the procedures first, we will see the full architecture of Faster R-CNN.

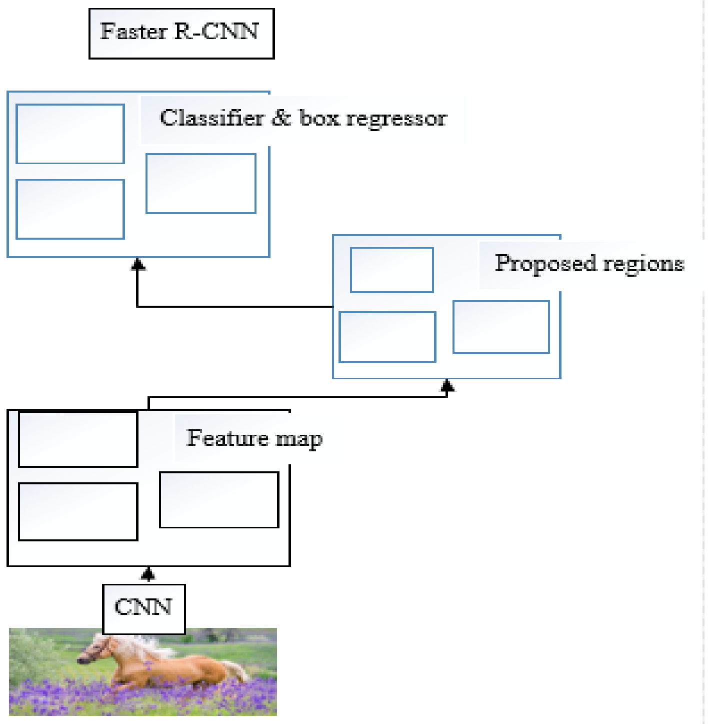


Figure 19: Faster R-CNN

4.3.1 ResNet-101 (CNN)

To train MSCOCO dataset, I choose Resnet-101. It is like VGG-16 but have some different structure. This model was done at Microsoft research issue. They propose a framework of deep residual learning that maps any two-stacked layer in a plane network. After completing all layer, it produces 2048-d feature vector. In addition, this feature vector goes to region proposal network (RPN).

4.3.2. Region Proposal Network (RPN)

RPN takes an input of any size as input and resize it 600x1000 and produces number of object proposals, which is rectangular. Every object proposal has an objectness score. I will fit this model with fully Conv. Network, as the ultimate goal is to share computation with Fast R-CNN.

To generate region proposal, we slide a small network especially 3x3 sliding window over the feature map which is mapped to a lower dimensional feature for example in Resnet-101, 2048-d dimensional feature. The output of each sliding window is fed into two fully connected layers-

- A box regression (reg) layer
- A box classification (cls) layer

I will describe in detail about these two layer in the following subsections. So in summarize, RPN is implemented with 3x3 conv. Layer with two siblings 1x1 conv. Layer one for regression and another classification layer.

4.3.2.1. Anchors

After completing conv. Network, we get 40x60x2048 (ResNet-101) and 40x60x512 (VGG-16) feature map. For every position, we use sliding window that is in general 3x3 and depth will be the same (for VGG-16: 512 and for ResNet-101:2048) and each sliding window location, predict multiple region proposals called Anchors, where the number of maximum possible proposals for each position is denoted as k. For every proposal,k-

- The reg layer has 4k outputs stands for coordinates of k boxes.
- The cls layer has 2k outputs stands for estimating probability of object or not.

An anchor is associated with a scale and aspect ratio. In my thesis, I use 3 scales and 3 aspect ratios that means 9 anchors are generated for every location. So after completing all computation, it produces WHk anchors that means 40x60x9=21,600.

In this approach, there has two important property-

4.3.2.1.1. Translation-invariant Anchors

When we can recognize an object as an object, even the appearance of that object varies in some way (position change) called invariance. Translation means each point/pixel in the image has been moved the same amount in the same direction. This approach has such property in terms of both the anchors and functions that compute proposals. If one translates an object, the proposal also can translate and predict that proposal in either location. It also reduces the model size. For fully connected layer, it has only (4+2)x9 dimensional convolutional output layer where 4 stands for coordinates (reg), 2 stands for estimating probability of being object (cls) and 9 stands for anchors (k). As a result, our model has the following parameters-

- For VGG-16: 512 x (4 + 2) x 9
- For Resnet-101: 2048 x (4 + 2) x 9

Because of less parameters, there has little chance of overfitting on small datasets.

4.3.2.1.2 Multi-Scale Anchors

The design of anchors presents a novel scheme for addressing multiple scales (128, 256, and 512) and ratios (1:1, 1:2, 2:1). Anchor based method is built on a pyramid of anchors that is cost-efficient. For several anchor boxes based on different scales as well as aspect ratios, RPN classifies and regresses bounding boxes that is relies on only images and feature map. RPN uses filter on single scale. As multi-scale RPN is designed based on anchors, we can easily apply conv. features computed on a single scale image as is also done by Fast R-CNN. It is a key component for sharing features without extra cost.

4.3.2.2. Calculation of Intersection over union (IoU)

After completing ResNet-101, it produces 2048-d feature vector. Then I use 3x3 sliding window (kernel) over the feature vector and it generates 40*60*9 = 21,600 anchors. Then I calculate Intersection over union (IoU) for every anchor with ground truth box. The algorithm is given below-

```
Parameters: BoxA, BoxB

Function IoU:

    xA = max (BoxA[0], BoxB[0])

    yA = max (BoxA[1], BoxB[1])

    xB = min (BoxA[2], BoxB[2])

    yB = min (BoxA[3], BoxB[3])

    Calculate intersection area:

        Interarea = (xB-xA+1)*(yB-yA+1)

    boxAreaA = (BoxA[2] – BoxA[0] + 1)* (BoxA[3] – BoxA[1] + 1)

    boxAreaB = (BoxB[2] – BoxB[0] + 1)* (BoxB[3] – BoxB[1] + 1)

    IoU = Interarea / float ( boxAreaA + boxAreaB – InterArea )
```

After calculating IoU, we use cross-boundary for omitting anchors. By apply cross-boundary; about 6000 anchors have probability having objects. Then using non-maximum suppression, we get about 2000 anchors that has higher probability of having object on that anchors.

4.3.2.3. Calculation of Loss Function

As we generate anchors that has the highest probability of having object, we can dole out a twofold class name of being object or not. We set a positive class label to two different kinds of anchors:

- Highest Intersection-over-Union (IoU) overlap with a ground truth box or
- Has IoU>0.7 with ground truth box.

Then for non-positive, if IoU<0.3 for all ground truth box anchor, we set a negative label. Another option would be if an anchor is neither positive nor negative, it will not contribute anything to the training time. Our loss is-

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N}\sum_i L_{cls}\ (P_i, P_i^*) + \lambda\,\frac{1}{N_{cls}}\sum_i P_i^*\ L_{reg}\left(t_i, t_i^*\right) \qquad (4.1)$$

p_i: predicted probability of an anchor to be an object

p_i*: ground truth label which has two values. 1 for positive anchors and 0 for negative anchors

N_{cls} : number of anchors in mini-batch (512)

λ: Constant value

N_{reg} : Number of total anchors (about 2000)

L_{reg} : smooth (ti – ti*) function where ti is predicted box and ti* is ground truth box.

Now I will describe two parts of equations:

Object/ not object classifier:

In the first part, we calculate the probability an anchor of having an object using softmax function then calculate loss. The output of cls layer consist of pi respectively.

Box regressor:

By the second part, we will try to fix predicted box with ground truth box. The output of reg layer consist of {ti} respectively.

In this manner, we can predict boxes of different sizes and scales though the features that we consider are of a fixed size or scales.

4.3.2.4 Training RPN

The RPN can be trained end-to-end by backpropagation and stochastic gradient descent (SGD). Each mini-batch arises from a single image that contains many positive and negative example anchors. It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominating. Instead, we arbitrarily sample two hundred and fifty-six (256) anchors in a training image to compute the loss function. We choose half of them (128) for positive anchors and half of them (128) for negative anchors.

We arbitrarily initialize all new layers using zero-mean Gaussian distribution which has standard deviation 0.01. All other layers (i.e., the shared conv.) are initialized by a pre-trained model for ImageNet classification, as is standard practice. We use learning rate .001 for first 60k and .0001 for the remaining.

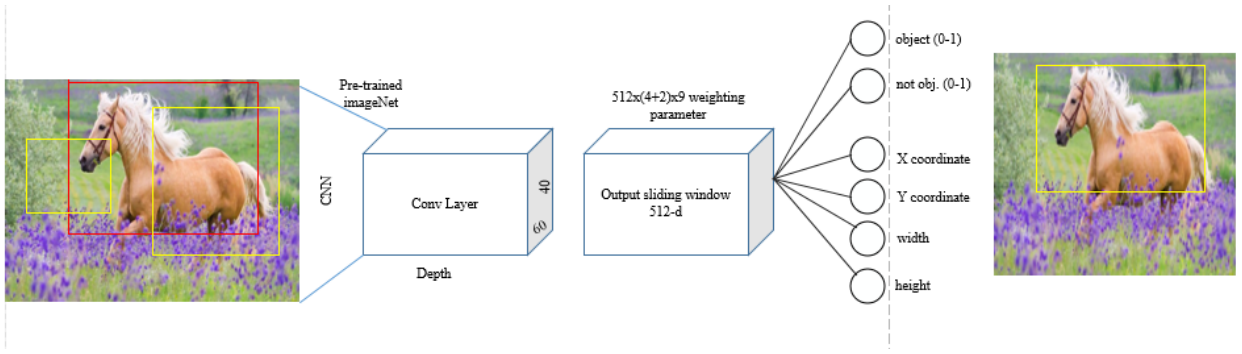
The RPN can be trained in many ways like stochastic gradient descent or back-propagation. I used ‘image centric’ strategy to train my network.

4.3.3 Sharing features for RPN and Fast R-CNN

In this section, I describe a pragmatic four step training algorithm that shares features.

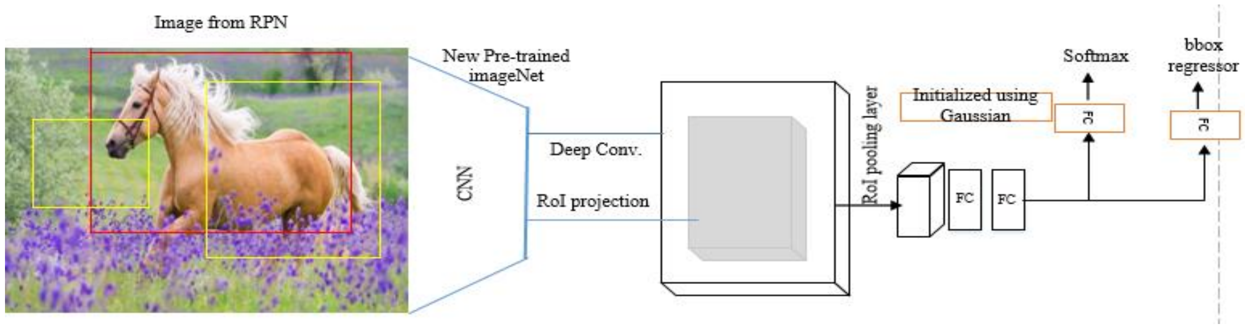
4.3.3.1. Proposer

I train region proposal network (RPN) with ImageNet pre-trained model. I have training model, CNN architecture, fully connected layer and output. For CNN, I used pre-trained ImageNet because it is already good for feature extraction and used 512 x (4 + 2) x 9 weighting parameters initialized by Gaussian Function N (0, 0.01) with mean 0 and standard deviation 0.01. I will try to train this network so that this becomes capable of producing proposals.



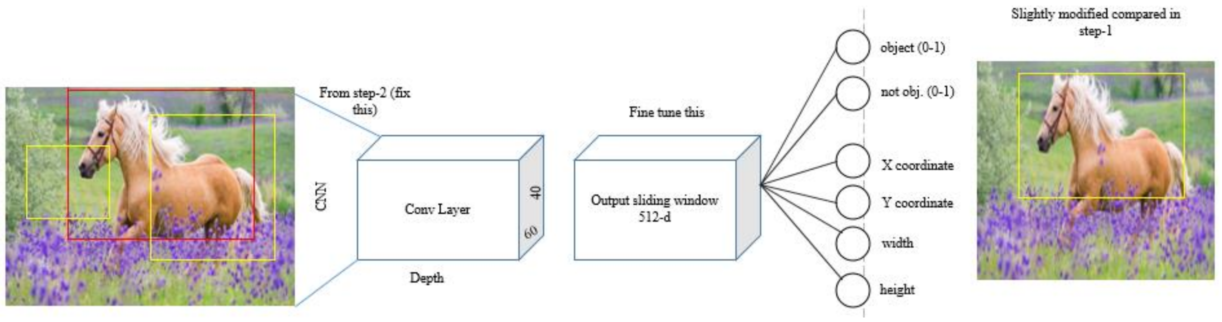
4.3.3.2. Detector

Using proposals generated by step-1, I trained a separable detection network by Fast R-CNN. RPN is initialized by ImageNet pre-trained model.



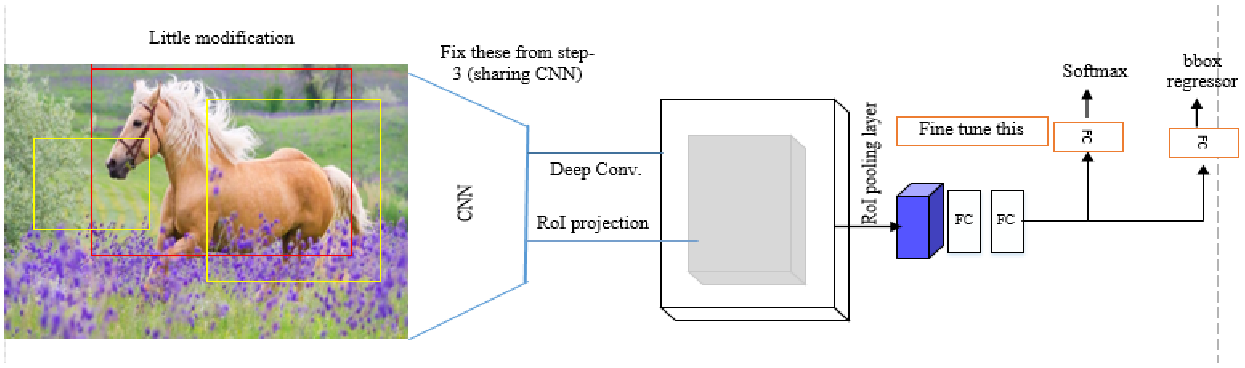
4.3.3.3. Re-train proposer

In this step, I fixed conv. Layer using the result in step-2. I fine-tune unique layers to RPN that produces slightly modified result compared in step -1.



4.3.3.4. Re-train proposer

This is the last step. I fixed conv. Layer and fine-tune fully connected layers of Fast R-CNN. I again use detector because from step-3 the region proposals have a little modification so I need to fine-tune fully connected layer for detection. And in this step, CNN is shared with step-3.



In this way, features are shared between Fast R-CNN and region proposal network (RPN).

4.4. Conclusion

In this chapter, I have discussed Faster R-CNN alongside ResNet-101 model. ResNet-101 generate 2048-d feature map after completing 101-layer computation. Faster R-CNN uses Region Proposal Network (RPN) that produces about 2000 object proposals. This model gains more significant results than any other deep learning models.

Chapter 5

Experiments

5.1. Introduction

In this chapter, I will discuss experiment results. The aim of this experiment is to find out more accurate result as well as speed up the process in object detection.

5.2 Environment setup

Environment setup is one of the biggest challenges in deep learning approach. For training and testing purpose, I set my experimentation environment by the following process. I choose Ubuntu 16.04 LTS operating system (64 bit). My PC's CPU configuration is Intel® Core™ i7-6700 CPU @ 3.40GHz, 8 GB RAM.

In first phase, I completed setup with GPU. GPU configuration is NVIDIA GeForce 930M, 8 GB RAM for faster training and testing as GPU is about 10 times faster than CPU and I have successfully executed code in small dataset like CIFAR-10. But when I tried to execute my MSCOCO dataset, all memory was exhausted. For this reason, I could not train my dataset using GPU. Because GPU only takes 1.90 GB from 8 GB.

5.3. Description of Dataset

MSCOCO is a large dataset of images designed for object detection, image captioning, classification etc. It has 90 categories of objects. Every image has more than one object and it is the only one dataset that has more objects in an image. In total, the dataset has 2,500,000-labeled instances in 328,000 images. Images are segmented into two sections i) 80k images are used for training purpose and ii) 20k images are used for validation purpose.

Because of resource limitation, I used 50k images for training purpose and 20k images for validation purpose.

I have worked on another dataset, PASCAL VOC 2007. It consists of a set of images and 20 categories in the following manners-

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: aero plane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

Each image has an annotation file giving a bounding box and class label for each object in one of the 20 categories present. Note that multiple objects from multiple classes may be present in the same image. Data has been separated into 50% for training/validation and 50% for testing. The distributions of images and objects by class are approximately equal across the training/validation and test sets. In total, there are 9,963 images, containing 24,640 annotated objects.

5.4. Results

In this section, I present detail results on the MSCOCO-2014 and PASCAL VOC-2007 object detection dataset.

MSCOCO-2014:

This dataset has 90 object categories but 10 object are missed. I analysis 53k images on the training dataset and 20k images on the validation set. I have completed 18008 iterations for training images (53k) in RPN and Fast R-CNN step. I use first 15k iterations with learning rate 0.003 and remaining iterations with learning rate 0.0003. I take 300 proposals using RPN algorithm. Testing time of MSCOCO dataset is 200ms. After completing 18008 iterations, fine-tuning faster R-CNN pre-trained model, loss decreases 0.003. In the following table, I will show mAP@0.5 results for every category. mAP@0.5 means mean average precision with threshold 0.5. I choose 300 proposals and MSCOCO 2014 val dataset using region proposal network.

Airplane	Apple	Backpack	Baseball bat	Baseball	Bed	Bench	Bicycle	Boat
1.0000	0.5000	0.0020	0.5000	0.01200	0.8333	0.1786	0.00027	0.4620

Book	Bottle	Bowl	Car	Carrot	Cat	Cell Phone	chair	Clock
0.2500	0.7000	0.7274	0.5448	0.1006	1.0000	0.5000	0.3850	0.6667

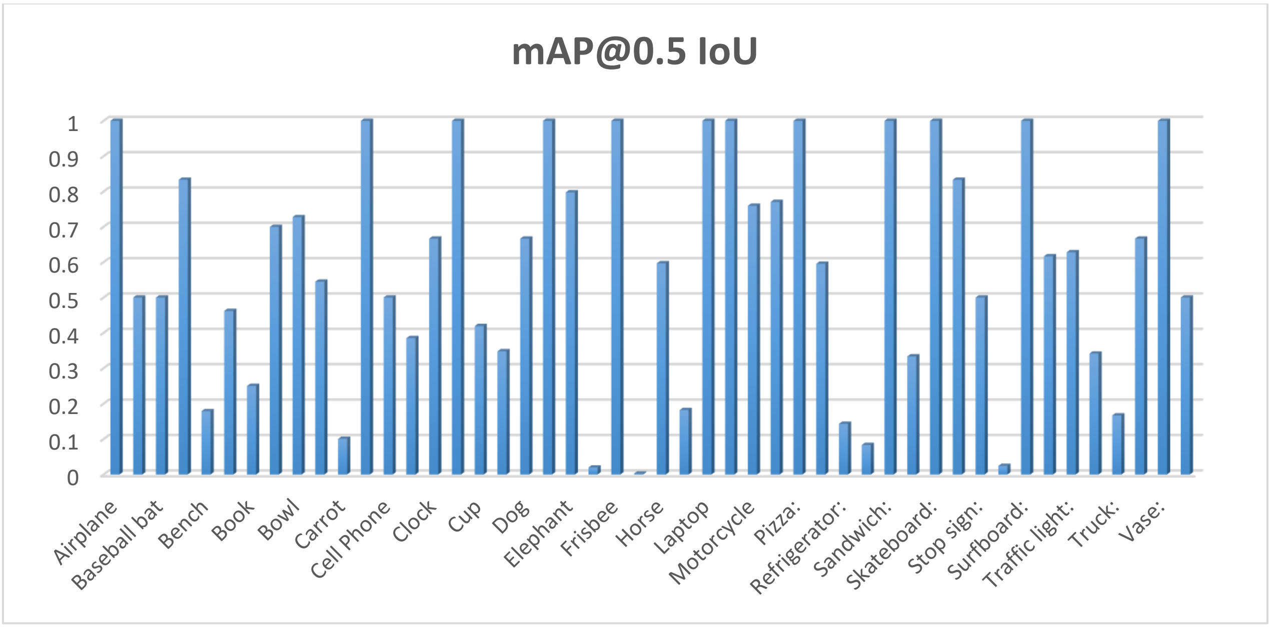
Couch	Cup	Dining table	Dog	Donut	Fork	Frisbee	Handbag	Horse
1.0000	0.4193	0.3478	0.6667	1.0000	0.01961	1.0000	0.0024155	0.5969

Elephant	Knife	Laptop	Microwave	Motorcycle	Orange	Oven	Person	Pizza
0.7976	0.1818	1.0000	1.0000	0.7597	0.0041	0.0050	0.7706	1.0000

Potted plant	Refrigerator	Remote	Sandwich	Sink	Skateboard	Spoon	Sports ball	Stop sign
0.5955	0.1429	0.08333	1.0000	0.3333	1.0000	0.00064	0.8333	0.5000

Suitcase	Surfboard	Toilet	Traffic light	Train	Truck	TV	Vase	Wine glass
0.02410	1.0000	0.6167	0.6280	0.3413	0.1667	0.6667	1.0000	0.5000

In the following graph, I show every categories mAP@0.5 IoU.



Overall precision is 0.5249 after completing 18008 iteration. Now I will show relationship IoU with Recall for 300 proposals.

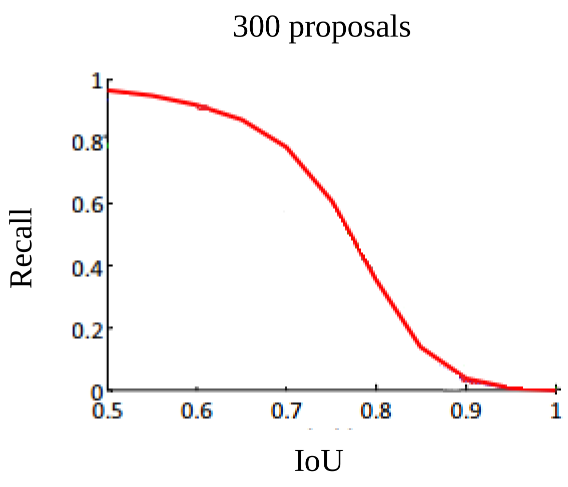
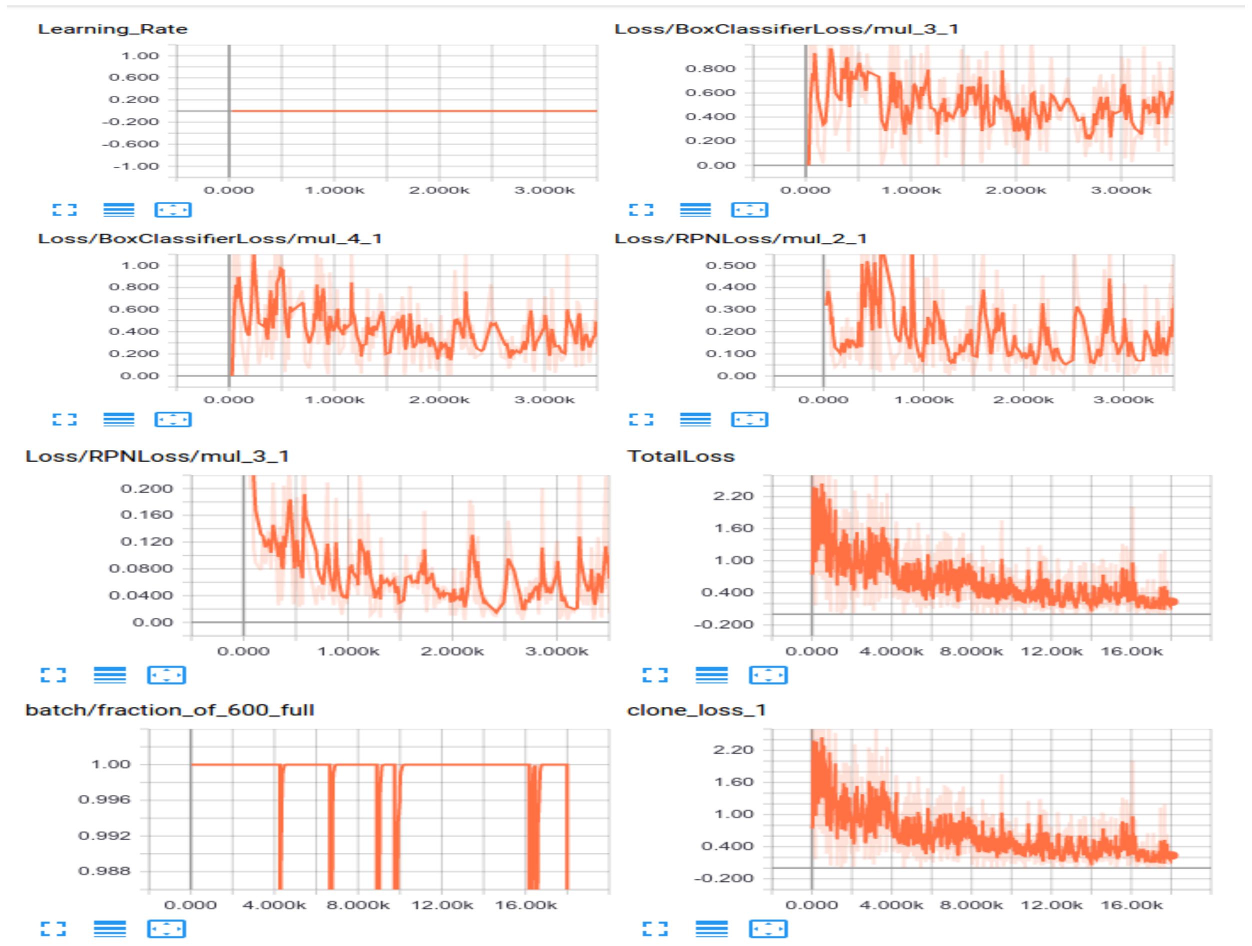
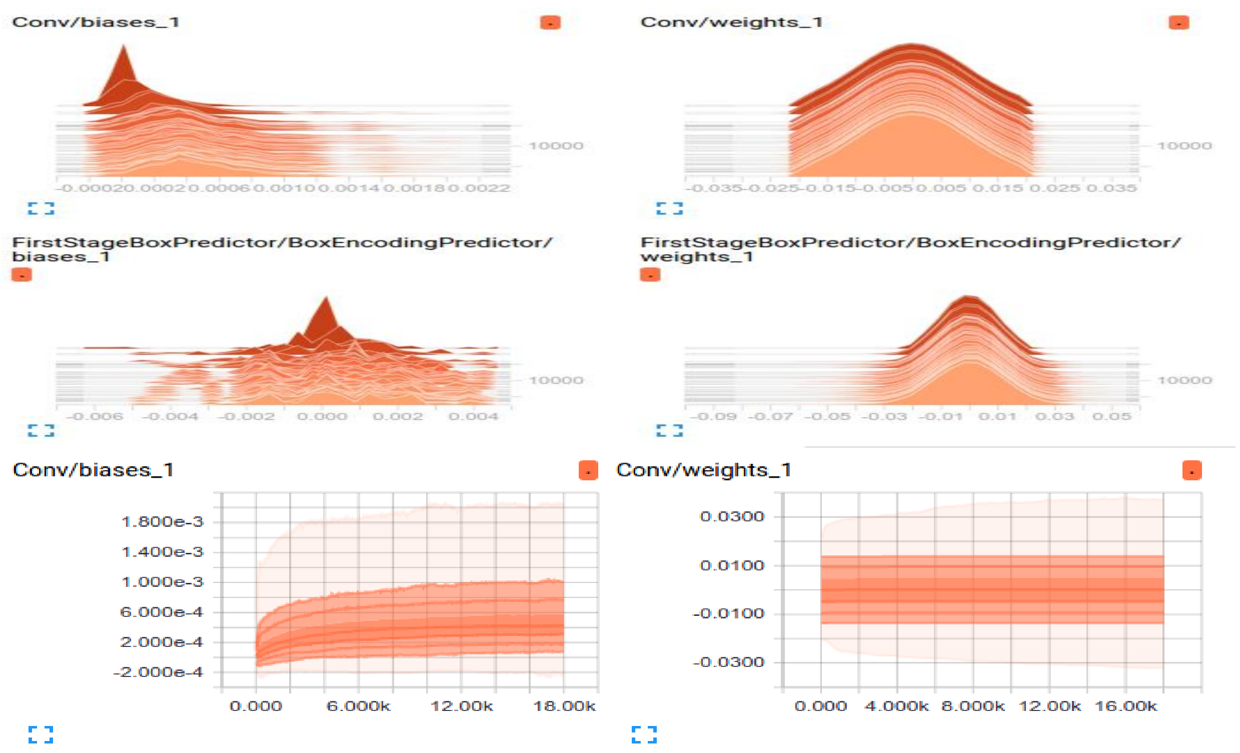


Figure 20: Recall vs IoU

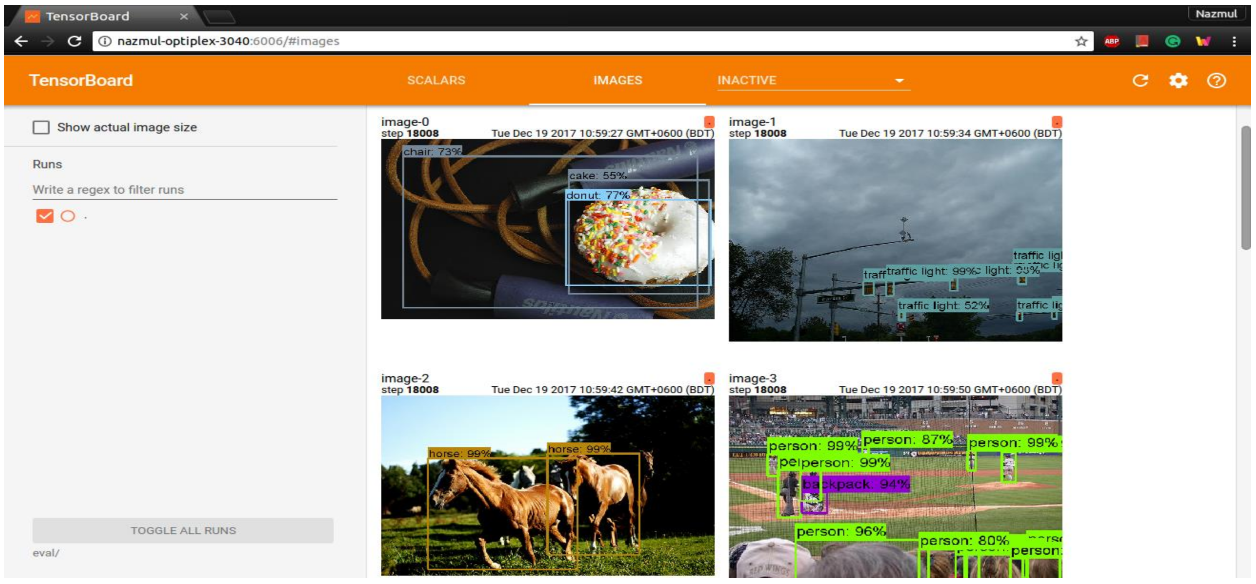
And the RPN some step and total loss is given bellow-



After executing every step, it calculates loss and updates weights of every layer. How model's weights are updated and biases are given bellow.

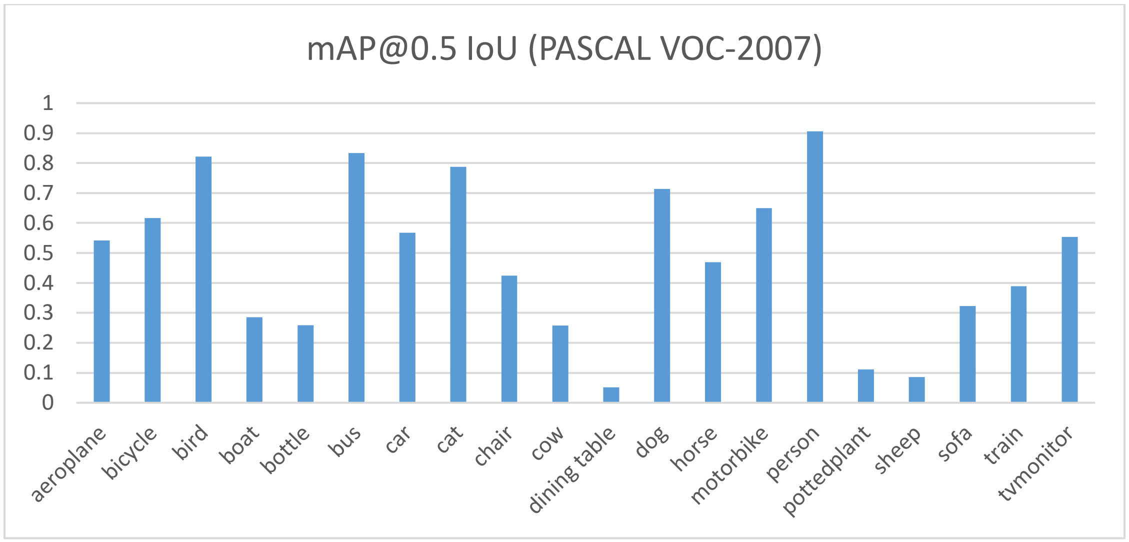


After 18008 steps of iterations, what, where and how my model detects objects are given bellow.

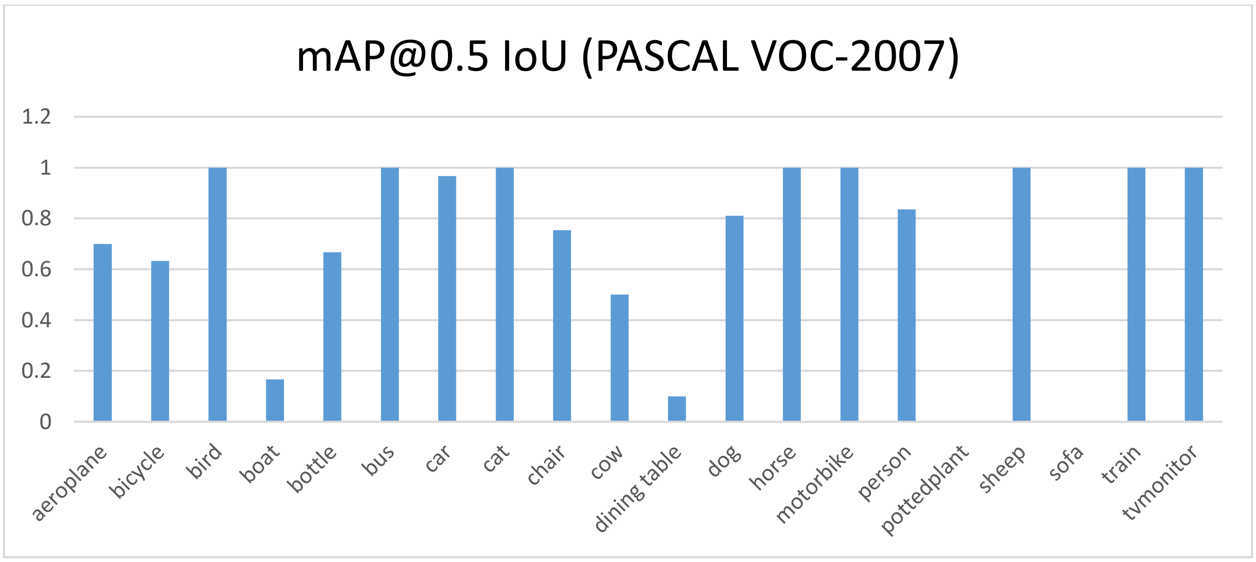


PASCAL VOC-2007:

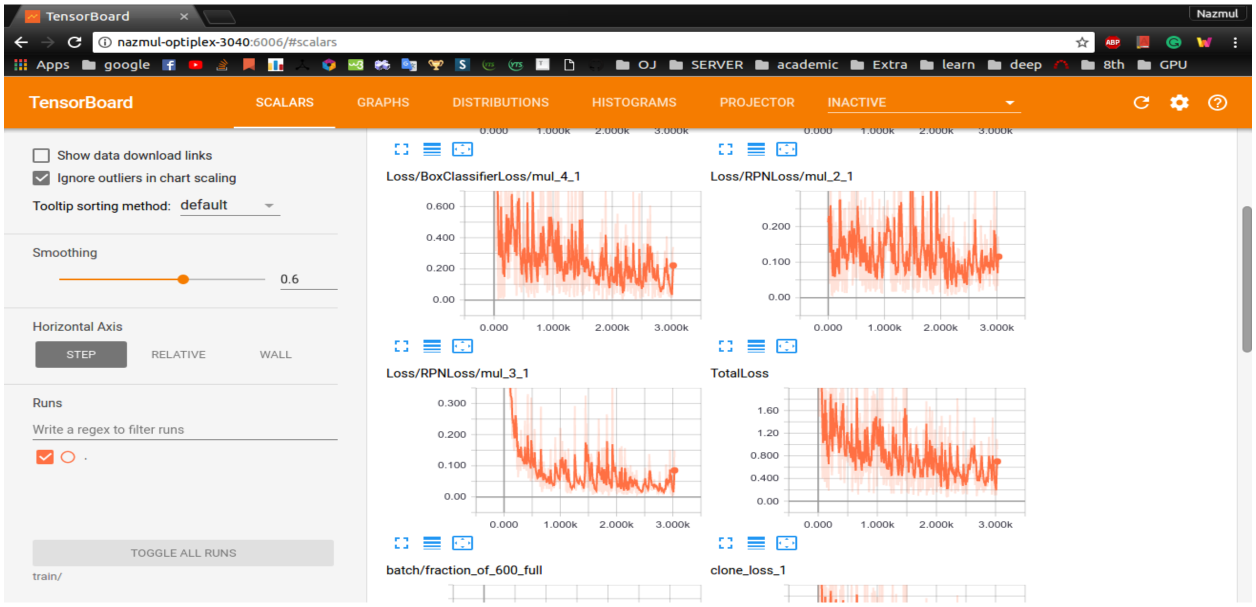
This dataset has 20 categories objects. There are 9,963 images, containing 24,640 annotated objects. The distribution of this dataset is 50% for training and 50% for testing purpose. After completing 1017 iterations, the overall mAP@0.5 IoU is 0.**4800** and every categories mAP@0.5 IoU is given bellow.



And after completing 3031 steps, the overall mAP@0.5 IoU is 0.**7851** and every categories mAP@0.5 IoU is given bellow

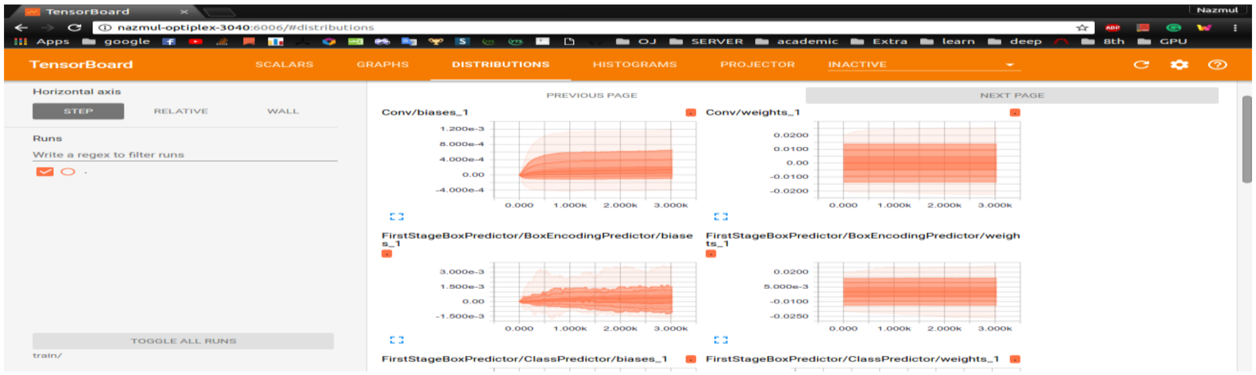


And the RPN some step and total loss is given bellow-

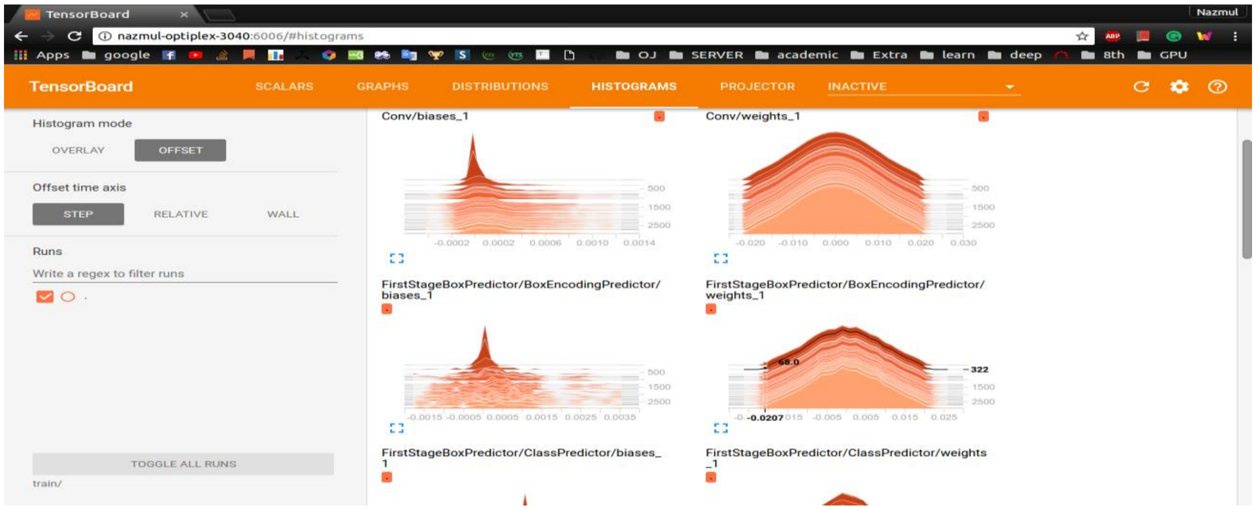


Show data download links
Ignore outliers in chart scaling [✓]
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: [✓ STEP] RELATIVE WALL

After executing every step, it calculates loss and updates weights of every layer. How model's weights are updated and biases are given bellow.

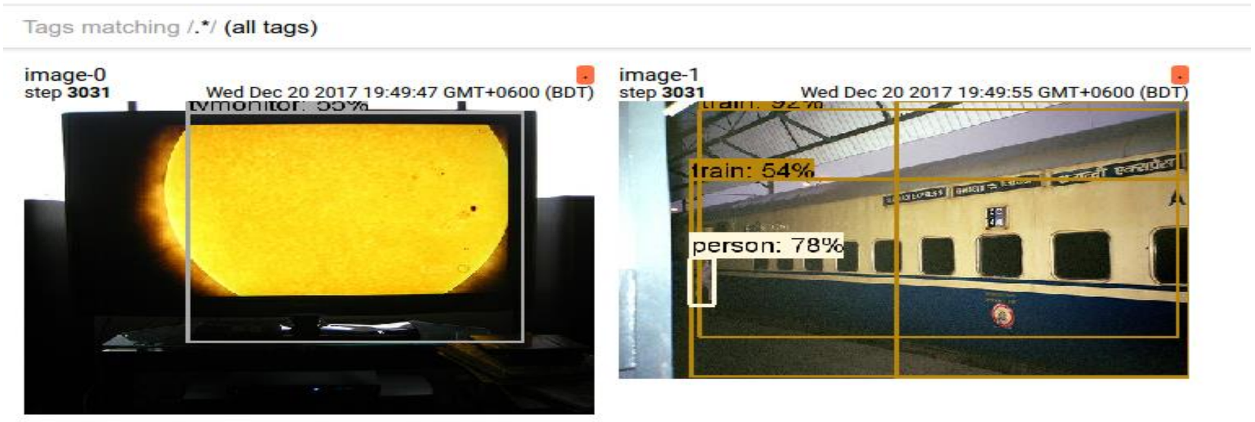

TensorBoard
nazmul-optiplex-3040:6006/#distributions
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
PROJECTOR
INACTIVE
PREVIOUS PAGE
NEXT PAGE
Conv/biases_1
Conv/weights_1
FirstStageBoxPredictor/BoxEncodingPredictor/biases_1
FirstStageBoxPredictor/BoxEncodingPredictor/weights_1
FirstStageBoxPredictor/ClassPredictor/biases_1
FirstStageBoxPredictor/ClassPredictor/weights_1
TOGGLE ALL RUNS
train/

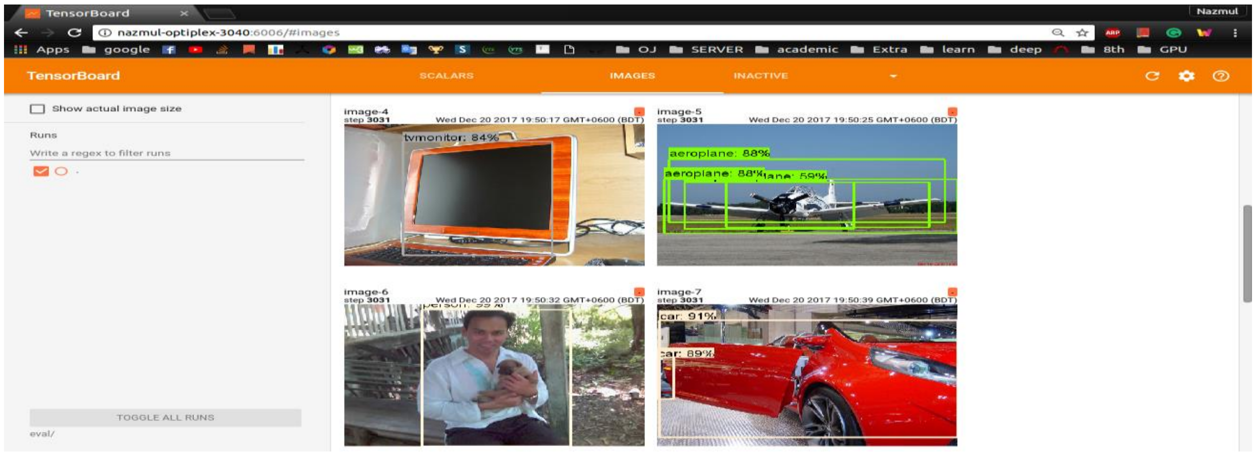
Horizontal axis: [✓ STEP] RELATIVE WALL

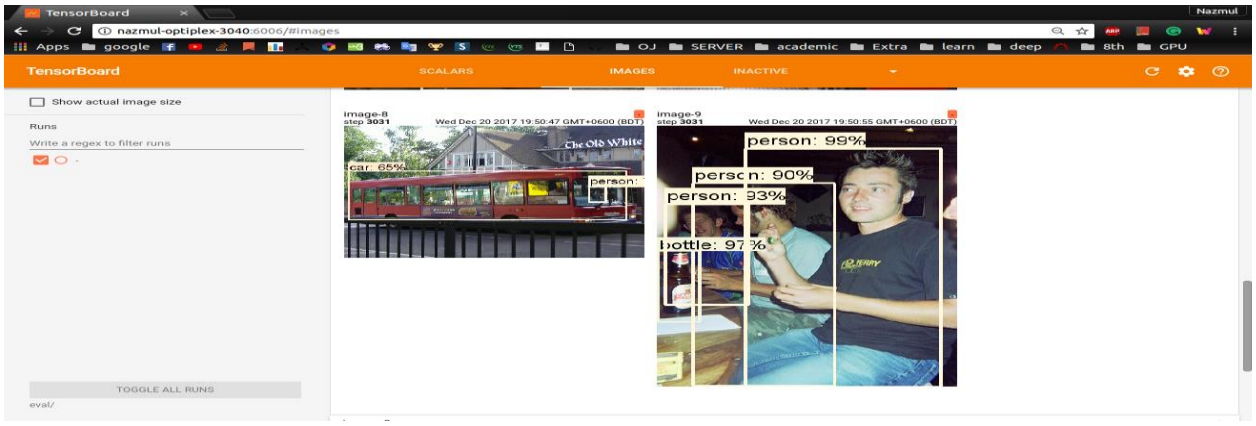

TensorBoard
nazmul-optiplex-3040:6006/#histograms
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
PROJECTOR
INACTIVE
Conv/biases_1
Conv/weights_1
FirstStageBoxPredictor/BoxEncodingPredictor/biases_1
FirstStageBoxPredictor/BoxEncodingPredictor/weights_1
FirstStageBoxPredictor/ClassPredictor/biases_1
FirstStageBoxPredictor/ClassPredictor/weights_1
TOGGLE ALL RUNS
train/

Histogram mode: OVERLAY [✓ OFFSET]
Offset time axis: [✓ STEP] RELATIVE WALL

After 3031 steps of iterations, what, where and how my model detects objects are given bellow.


Tags matching /.*/ (all tags)
image-0
step 3031
Wed Dec 20 2017 19:49:47 GMT+0600 (BDT)
tvmonitor: 55%
image-1
step 3031
Wed Dec 20 2017 19:49:55 GMT+0600 (BDT)
train: 92%
train: 54%
person: 78%


TensorBoard
nazmul-optiplex-3040:6006/#images
SCALARS
IMAGES
INACTIVE
Show actual image size
Runs
Write a regex to filter runs
TOGGLE ALL RUNS
eval/
image-4
step 3031
Wed Dec 20 2017 19:50:17 GMT+0600 (BDT)
tvmonitor: 84%
image-5
step 3031
Wed Dec 20 2017 19:50:25 GMT+0600 (BDT)
aeroplane: 88%
aeroplane: 88%
lane: 59%
image-6
step 3031
Wed Dec 20 2017 19:50:32 GMT+0600 (BDT)
image-7
step 3031
Wed Dec 20 2017 19:50:39 GMT+0600 (BDT)
car: 91%
car: 89%



5.5. Conclusion

In this chapter, I have covered discussing my working environment to execute my work, a short overview of dataset, MSCOCO what I used for training purpose and lastly the main part, results of my work. In the next chapter, I will try to draw a summary.

Chapter 6

Conclusion

The goal of this research paper is to find an efficient way of developing an application that can detect objects accurately from an image and for videos. As I am a newbie in deep learning field, at first, I tried to train CIFAR-10 small dataset that has only 10 types of objects. To train CIFAR-10 dataset, I used GPU computation and use simple deep learning architecture. Every image has only one object. Therefore, we may call it classification problem. After successfully completing training on CIFAR-10 dataset, I wanted to train average sized dataset and choose PASCAL VOC-2007 dataset that has 20 categories of objects. After working relentlessly few weeks, I have successfully executed on my training dataset. I used faster R-CNN with resnet-50 model.

Then I used MSCOCO dataset for training purpose. There have so many images and 90 types of objects. Every image has more than one object. I used Faster R-CNN with ResNet-101 that has 101 layer and tensorflow for coding purpose. As I am newbie in deep learning and tensorflow, these types of thesis was so much tough within 6 months. But at the end, I have enjoyed with learning deep learning related works.

Faster R-CNN uses RPN for selecting region that provides state of the art result among all existing approaches.

6.1. Discussion

As we have seen in experiments, Faster-RCNN improved not only speed but also performance compared to any existing object detection approaches. That is, RPN get better proposals with better speed than previous approaches. RPN shares features that is calculated from CNN with the rest of the CNN especially the early convolutional layers. The RPN feeds from the same convolutional feature map as the detection sub-system. This sharing process speeds things up and that is why it is called Faster R-CNN.

Not being able to train on the entire dataset decreases my performance significantly. Further-more, due to machine capability constraints, I could not even train until convergence on the smaller dataset using GPU.

6.2. Future Work

Feature extraction is one the fundamental steps of every image processing applications and computer vision. For feature extraction, Faster R-CNN provides more accurate results with less time with respect other methods. There have so many applications like classification, image captioning, face detection, different types of disease identification etc. that can be used faster R-CNN.

References

[1] T. Ojala, M. Pietika¨inen, and T. Ma¨enpa¨¨a, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 7, pp. 971–987, 2002.

[2] J. Wu and J. M. Rehg, “Centrist: A visual descriptor for scene categorization,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 33, no. 8, pp. 1489–1501, 2011.

[3] S. Arivazhagan, L. Ganesan, and S. P. Priyal, “Texture classification using gabor wavelets based rotation invariant features,” Pattern recognition letters, vol. 27, no. 16, pp. 1976–1982, 2006.

[4] Lindeberg, Tony. "Scale invariant feature transform." Scholarpedia 7.5 (2012): 10491.

[5] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[6] Oliva, Aude, and Antonio Torralba. "Building the gist of a scene: The role of global image features in recognition." Progress in brain research 155 (2006): 23-36.

[7] Wang, Hongbing, et al. "Overexpression of type-1 adenylyl cyclase in mouse forebrain enhances recognition memory and LTP." Nature neuroscience 7.6 (2004): 635-642.

[8] Z. Guo, L. Zhang, and D. Zhang, “A completed modeling of local binary pattern operator for texture classification,” Image Processing, IEEE Transactions on, vol. 19, no. 6, pp. 1657–1663, 2010.

[9] Xiao, Yang, Jianxin Wu, and Junsong Yuan. "mCENTRIST: A multi-channel feature generation mechanism for scene categorization." IEEE Transactions on Image Processing 23.2 (2014): 823-836.

[10] Dey, Emon Kumar, Md Nurul Ahad Tawhid, and Mohammad Shoyaib. "An automated system for garment texture design class identification." Computers 4.3 (2015): 265-282.

[11] S. Liao, W. Fan, A. Chung, and D.-Y. Yeung, “Facial expression recognition using advanced local binary patterns, tsallis entropies and global appearance features,” in Image Processing, 2006 IEEE International Conference on, pp. 665– 668, IEEE, 2006.

[12] N. Sun, W. Zheng, C. Sun, C. Zou, and L. Zhao, “Gender classification based on boosting local binary pattern,” in Advances in Neural Networks-ISNN 2006, pp. 194–201, Springer, 2006.

[13] S. Liao, M. W. Law, and A. Chung, “Dominant local binary patterns for texture classification,” Image Processing, IEEE Transactions on, vol. 18, no. 5, pp. 1107– 1118, 2009.

[14] X. Huang, S. Z. Li, and Y. Wang, “Shape localization based on statistical method using extended local binary pattern,” in Image and Graphics (ICIG’04), Third International Conference on, pp. 184–187, IEEE, 2004.

[15] M. Heikkila¨, M. Pietik¨ainen, and C. Schmid, "Description of interest regions with local binary patterns," Pattern recognition, vol. 42, no. 3, pp. 425–436, 2009.

[16] 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 (CVPR), 2014 IEEE Conference on, pp. 580–587, IEEE, 2014.

[17] I. S. A. Krizhevsky and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, pages 1106–1114, 201.

[18] T. D. J. M. R. Girshick, J. Donahue. Region-based convolutional networks for accurate object detection and semantic segmentation. IEEE Transactions on PAMI, Accepted may 2015.

[19] R. Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 1440–1448, 2015.

[20] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

[21] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[22] He, Kaiming, et al. "Identity mappings in deep residual networks." European Conference on Computer Vision. Springer International Publishing, 2016.

[23] J. Hosang, R. Benenson, and B. Schiele, "How good are detection proposals, really?" in British Machine Vision Conference (BMVC), 2014.

[24] J. Hosang, R. Benenson, P. Doll´ar, and B. Schiele, "What makes for effective detection proposals?" IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015.

[25] N. Chavali, H. Agrawal, A. Mahendru, and D. Batra, "Object-Proposal Evaluation Protocol is 'Gameable'," arXiv: 1505.05836, 2015.

[26] J. Carreira and C. Sminchisescu, "CPMC: Automatic object segmentation using constrained parametric min-cuts," IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2012.

[27] P.Arbel´aez,J.Pont-Tuset,J.T.Barron,F.Marques,andJ.Malik, "Multiscale combinatorial grouping," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[28] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2012.

[29] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," International Journal of Computer Vision (IJCV), 2013.

[30] C. L. Zitnick and P. Doll´ar, "Edge boxes: Locating object proposals from edges," in European Conference on Computer Vision (ECCV), 2014.

[31] C. Szegedy, A. Toshev, and D. Erhan, “Deep neural networks for object detection,” in Neural Information Processing Systems (NIPS), 2013.

[32] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, “Scalable object detection using deep neural networks,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[33] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov, “Scalable, high-quality object detection,” arXiv:1412.1441 (v1), 2015.

[34] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” in International Conference on Learning Representations (ICLR), 2014.

[35] P. O. Pinheiro, R. Collobert, and P. Dollar, “Learning to segment object candidates,” in Neural Information Processing Systems (NIPS), 2015.

[36] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in European Conference on Computer Vision (ECCV), 2014.

[37] J. Dai, K. He, and J. Sun, “Convolutional feature masking for joint object and stuff segmentation,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.