

REGION BASED CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT DETECTION
AND RECOGNITION IN ADAS APPLICATION

by

SACHIT KAUL

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2017

Copyright © by Sachit Kaul 2017

All Rights Reserved



Acknowledgements

First, I would like to thank Dr. Kamesh Subbarao for his guidance, constant support and motivation through entire process of my dissertation development. While discussing about the area of study I wanted to pursue my graduate thesis it was him who introduced me to the area of Object Detection and Recognition using Computer Vision and Deep Learning and generously agreed to be my faculty advisor for the thesis.

Since then he has been instrumental in developing my concepts in the field of study. He helped me built my research in a structured manner while giving me freedom to innovate and pursue my own ideas.

I would also like to thank Dr. Ratan Kumar and Dr. Ashfaq Adnan for being a part of my thesis committee and taking time to give their inputs for making this work possible.

Second, I would also like to thank my family and friends for always being there for me when I needed them and believing in me whatever the circumstances. They have filled me with enthusiasm and have always encouraged me to pursue my dreams. It is their trust in me which has made me the person I am today.

Last, I would like to thank MathWorks, Inc for providing me with all the tools necessary for my thesis. It is their tools which has made my thesis possible.

November 30, 2017

Abstract

REGION BASED CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT DETECTION AND RECOGNITION IN ADAS APPLICATION

Sachit Kaul, MS

The University of Texas at Arlington, 2017

Supervising Professor: Kamesh Subbarao

Object Detection and Recognition using Computer Vision has been a very interesting and a challenging field of study from past three decades. Recent advancements in Deep Learning and as well as increase in computational power has reignited the interest of researchers in this field in last decade.

Implementing Machine Learning and Computer Vision techniques in scene classification and object localization particularly for automated driving purpose has been a topic of discussion in last half decade and we have seen some brilliant advancements in recent times as self-driving cars are becoming a reality. In this thesis we focus on Region based Convolutional Neural Networks (R-CNN) for object recognition and localizing for enabling Automated Driving Assistance Systems (ADAS). R-CNN combines two ideas: (1) one can apply high-capacity Convolutional Networks (CNN) to bottom-up region proposals in order to localize and segment objects and (2) when labelling data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific-fine-tuning, boosts performance significantly.

In this thesis, inspired by the RCNN framework we describe an object detection and segmentation system that uses a multilayer convolutional network which computes

highly discriminative, yet invariant features to classify image regions and outputs those regions as detected bounding boxes for specifically a driving scenario to detect objects which are generally on road such as traffic signs, cars, pedestrians etc.

We also discuss different types of region based convolutional networks such as RCNN, Fast RCNN and Faster RCNN, describe their architecture and perform a time study to determine which of them leads to real-time object detection for a driving scenario when implemented on a regular PC architecture.

Further we discuss how we can use such R-CNN for determining the distance of objects on road such as Cars, Traffic Signs, Pedestrians from a sensor (camera) mounted on the vehicle which shows how Computer Vision and Machine Learning techniques are useful in automated braking systems (ABS) and in perception algorithms such as Simultaneous Localization and Mapping (SLAM).

Table of Contents

Acknowledgements	iii
Abstract	iv
List of Illustrations	viii
List of Tables	ix
Chapter 1: Introduction.....	1
Chapter 2: Literature Review.....	3
Chapter 3: Introduction to Convolutional Neural Networks.....	7
3.1. Architecture.....	7
3.1.1 Convolution Layer.....	8
3.1.2 Pooling Layer.....	10
3.1.3 Normalization Layer.....	11
3.1.4 Fully-Connected Layer.....	12
3.2 Famous Convolutional Network Architectures.....	12
Chapter 4: Transfer Learning.....	14
4.1 AlexNet Architecture.....	14
4.1.1 ReLU Nonlinearity.....	14
4.1.2 Training on Multiple GPU's.....	16
3.1.3 Localization Response Normalization.....	16
4.1.4 Overlapping Pooling.....	17
4.1.5 Overall Architecture.....	17
4.2 Fine-tuning AlexNet for Road Objects Detection.....	18
Chapter 5: Region Based Convolutional Networks for Object Detection.....	20
5.1 R-CNN.....	20

5.1.1 R-CNN Architecture.....	21
5.1.2 Implementation of R-CNN for Road Object Detection.....	22
5.1.3 Results of R-CNN.....	22
5.1.4 Conclusion.....	24
5.2 Fast R-CNN.....	25
5.2.2 Implementation of Fast R-CNN.....	26
5.2.3 Results Fast R-CNN.....	26
5.2.4 Conclusion.....	27
5.3 Faster R-CNN.....	28
5.3.2 Implementation of Faster R-CNN.....	29
5.3.3 Results Faster R-CNN.....	30
5.3.4 Conclusion.....	32
Chapter 6: Implementation of Faster R-CNN for Depth Estimation.....	33
6.1 Introduction.....	33
6.2 Stereo Vision.....	33
6.3 Camera Calibration.....	35
6.4 Disparity Mapping.....	37
6.5 Object Detection.....	38
6.6 3D Reconstruction And Depth Estimation.....	38
6.7 Results of Depth Estimation.....	40
Chapter 7: Conclusion.....	41
Chapter 8: Future Scope.....	42
References.....	43
Appendix.....	48
Biographical Information.....	51

List of Illustrations

Figure.3.1 Regular 3 Layer Neural Network Vs Convolutional Neural Network.....	8
Figure 3.2 Representation of Layers of Convolutional Neural Network.....	12
Figure 4.1 Training error rate vs Epochs.....	15
Figure 4.2. AlexNet Final Architecture.....	18
Figure 5.1. R-CNN Architecture.....	21
Figure 5.2. Procedure for Object detection and Localization in R-CNN.....	22
Figure 5.3. Time study.....	24
Figure 5.4. Fast R-CNN Architecture.....	25
Figure 5.5. Detection results.....	27
Figure 5.6. Faster R-CNN Architecture.....	28
Figure 5.7. Comparison of Detection Time Per Image for R-CNN Architectures.....	30
Figure 5.8. Detection Results Faster R-CNN.....	31
Figure 5.9. Mean Average Precision (mAP) of our Faster RCNN.....	32
Figure 6.1 A Stereo Imaging System.....	33
Figure 6.2. Depth Estimation Process Work Flow.....	35
Figure 6.3 Examples of what you can do after calibrating the camera.....	36
Figure 6.4 Intrinsic and Extrinsic parameters.....	36
Figure 6.5 Camera Calibration Session in MATLAB R2017b.....	37
Figure 6.6 Stereo Camera Geometry.....	39
Figure 6.7. Results of Depth Estimation.....	40

List of Tables

Table 5.1 Time Study of R-CNN	23.
Table 5.2 Time Study of Fast R-CNN	26.

Chapter 1

INTRODUCTION

Implementing Machine Learning and Computer Vision techniques in scene classification and object localization particularly for automated driving purpose has been a topic of discussion for a long time. As the automobile sector is now moving very fast towards developing advanced driver assistance systems (ADAS) and autonomous vehicles there have been certain advances in deep learning architectures and also the hardware required for the same.

Deep Learning algorithms have been in existence from a very long time but object detection time, availability of training data and best training practices are still debated widely. In this thesis we create a Neural Network for its implementation in real-time object detection and localization in a driving scene.

We address above mentioned problems by comparing different Convolutional Neural Network (CNN) architectures fine-tuned for a particular task; In our case a driving scenario. We begin with summarizing some prior work in this direction, and then we focus specifically on Region based Convolutional Neural Network (R-CNN) family

First, we start with elaborating about the CNN architecture which is specifically designed for object detection and classification in an image input. We also discuss certain CNN architectures developed over last decade. We address the problem of detection time, training data availability by applying a high-capacity Convolutional Networks (CNN) to generate region proposals in order to localize and segment objects. As training data is scarce, we use a supervised pre-trained CNN (AlexNet), which is trained for an auxiliary task, followed by a domain-specific-fine-tuning i.e. for object detection and localization in a driving scenario in our problem which helps to boost the performance of the resulting CNN significantly.

Second, we discuss in detail about the R-CNN family, provide their architectural details, fine tune each of them for object detection and localization in a driving scenario and then compare them and discuss their advantages, disadvantages on the basis of detection time, network complexity and training time.

Finally, after concluding which R-CNN framework is suitable for real-time object detection and localization in an image we move further to implement our R-CNN for depth estimation task. We start with discussing in brief, the techniques which are used for perceiving depth information, explain why stereo vision is a reliable and robust way for depth perception. Further, we talk in detail about concepts used in stereo vision and how we combine it with our R-CNN for perception of depth information.

Chapter 2

LITERATURE REVIEW

Deep Convolutional Networks for Object Detection: Convolutional Neural Networks (CNN) saw their usage initially in handwritten character recognition in 1990s but fell out of fashion due to use of support vector machines. In 2012 Krizhevsky et al. [4] reignited the interest in CNN by showing a substantial improvement in image classification accuracy on the ImageNet large scale visual recognition challenge (ILSVRC [5] [6]). Their success resulted from training a large CNN on 1.2 million labeled images, together with a few changes on CNN from the 1990s (e.g., $\max(x,0)$ “ReLU” non-linearities, “dropout” regularization, and a fast GPU implementation).

There are certain other methodologies to use Convolutional networks concurrent to the RCNN's discussed in this thesis such as Szegedy et al. [7] model object detection posed as a regression problem. In an image window, they use a CNN to predict foreground pixels over a coarse grid for the whole object as well as the object's top, bottom, left and right halves. A grouping process then converts the predicted masks into detected bounding boxes. Szegedy et al. train their model from a random initialization on PASCAL visual object classes (VOC) 2012 training and evaluation and get a mean average precision (mAP) of 30.5 percent on VOC 2007 test. In comparison, an R-CNN using the same network architecture gets a mAP of 58.5 percent, but uses supervised ImageNet pre-training.

Scalability and Speed: In recent years Object detection techniques have evolved over the last decade due largely to the use of low level image features, such as scale-invariant feature transform (SIFT) [2] and histogram of oriented gradients (HOG) [3], in sophisticated Machine Learning frameworks [1] which increases the accuracy. These

features are often combined with support vector machine (SVM) algorithm and we have seen their implementation in many object detection tasks such as pedestrian detection.

In addition to being accurate, it's important for object detection systems to scale well with the increase in number of object categories. Methods such as Discriminatively trained part based models (DPM) [8] scale to thousands of object categories. For example, in Dean et al. [9] hashtable lookups are used instead of exact filter convolutions in DPM. Their results show that with this technique it's possible to run 10k DPM detectors in about 5 minutes per image on a desktop workstation. However, there is a tradeoff. when a large number of DPM detectors compete, the approximate hashing approach causes a substantial loss in detection accuracy. R-CNNs, in contrast, scale very well with the number of object classes to detect because nearly all computation is shared between all object categories.

Despite the scaling behavior, an R-CNN can take 10 to 45 seconds per image on a GPU, depending on the network used, since each region is passed through the network independently [1]. There has been recent work to reduce training and detection time while increasing the accuracy and simplifying the training process Fast RCNN [10] is one of them which has higher detection quality (mAP) than R-CNN and other being SPPnet in which training is single-stage, using a multi-task loss. Training in SPPnet can update all network layers and no disk storage is required for feature caching. Another method is Faster RCNN [11] In this work, a Region Proposal Network (RPN) is introduced that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position.

Localization Methods: Sliding-window detectors have been a dominant approach in object detection. This approach traces back to early face detectors [12] and continued

with HOG based pedestrian detection [13] and DPM based generic object detection [8]. An alternative was introduced to compute a pool of likely overlapping regions each serving as a candidate object and then to filter these candidates. The “selective search” algorithm of van de Sande et al. [14] popularized the multiple segmentation approach for object detection by showing strong results on PASCAL object detection. RCNN approach was inspired by the success of selective search. Object proposal generation is now an active research area, for an in-depth survey of region proposal we can refer to Hosang et al. [15] which provides an evaluation of recent methods.

Depth Estimation: Depth information can be perceived using many techniques such as Laser Ranging, Structured Light and Stereo Vision

Perception of depth information from time-of-flight (ToF) or structured light sensor is considered an active sensor, unlike a stereo vision camera which is a passive sensor. Devices of this type such as the Microsoft Kinect are cheap hence seeing usage widely. However, these active sensors suffer from certain characteristic problems [31] They are subject to errors such as noise and ambiguity, which are due to the sensor that is used and also, they are subject to non-systematic errors such as scattering and motion blur.

According to the comparative analyses performed by Foix et al. [32], Kim et al. [33], and Zhang et al. [34], ToF devices perform satisfactorily only up to a maximum distance of approximately 5–7 meters and are too sensitive to be used in outdoor environments, especially in very bright areas. Because of these limitations of ToF sensors, stereo vision sensors are more reliable and robust; they are capable of producing high-resolution disparity maps and are suitable for both indoor and outdoor environments [35].

Stereo vision addresses the problem of reconstruction of the three-dimensional coordinates of points for depth estimation. A stereo vision system consists of a stereo camera, namely, two cameras placed horizontally (i.e., one on the left and the other on the right). The two images captured simultaneously by these cameras are then processed for the recovery of visual depth information [27].

Determination of disparity is one of big challenges faced in stereo vision. Intuitively, a disparity map represents corresponding pixels that are horizontally shifted between the left image and right image. New methods and techniques for solving this problem are developed every year and exhibit a trend toward improvement in accuracy and time consumption.

As the number of pixels in images increase number of calculations required increase for disparity map processing. This phenomenon causes the matching problem to be computationally complex [36]. The improvements in computational complexity achieved with recent advances in hardware technology have led to advancement of research in the stereo vision field. Thus, the main motivation for hardware-based implementation is to achieve real time processing [37]. In real-time stereo vision applications, such as autonomous driving, 3D gaming, and autonomous robotic navigation, fast but accurate depth estimations are required [38]. Additional processing hardware is therefore necessary to improve the processing speed.

Chapter 3

INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNN) are very similar to ordinary Neural Networks which were inspired by biological processes. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. CNN architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture which makes the forward function more efficient to implement and vastly reduce the number of parameters in the network.

In this section, we will discuss about the CNN Architecture, certain famous CNN architectures and their details. We propose a method for object detection using deep convolutional networks, especially for detection of common objects present on Road (Car Pedestrians, Traffic Signs etc.). Our approach is simple, by transferring the rich feature hierarchies of CNN learned by large scale image dataset to a specific task in our case objects on a Road.

3.1 Architecture

Convolutional Neural Networks take advantage of the fact that they are specifically designed for image inputs. In CNN's neurons are arranged in a three-dimensional way i.e. Width, Height and Depth which constrains their architecture in more sensible way.

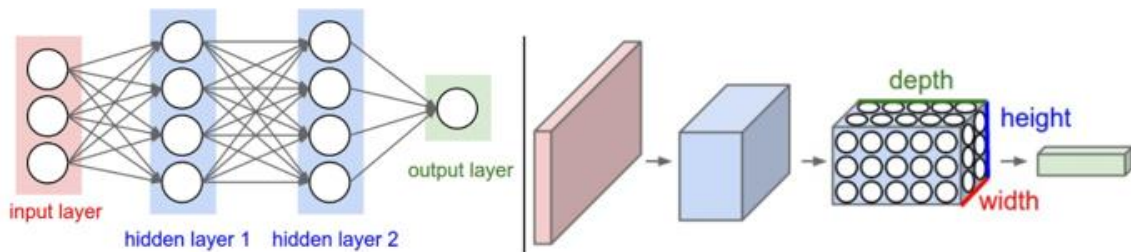


Figure.3.1 Regular 3 Layer Neural Network Vs Convolutional Neural Network [24].

In the above figure, In left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this figure the red input layer holds the image, so its width and height would be the dimensions of the image and the depth would be 3 (Red, Green, Blue channels).

3.1.1 Convolution Layer

The Convolution layer is the core building block of a Convolutional Neural Network that does most of the computational heavy lifting. The CONV layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume.

For example, a typical filter on a first layer of a ConvNet might have size $5 \times 5 \times 3$ (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels). During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the

responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network

Spatial Arrangement of Convolution Layer

Three hyper-parameters control the size of the output volume: the depth, stride and zero-padding.

1. **Depth** of the output volume is a hyper-parameter: it corresponds to the number of filters which are used for learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various orientations, edges, or blobs of color. A set of neurons that are all looking at the same region of the input as a depth column will be referred.

2. **Stride** is specified with which we slide the filter. When the stride is 1 then the filters move one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

3. Sometimes it will be convenient to pad the input volume with zeros around the border. The size of this **zero-padding** is a hyperparameter. Zero padding allows us to control the spatial size of the output volumes.

The convolution layer can be summarized as,

- Accepts a volume of size $W_1 \times H_1 \times D_1$.
- Requires four hyper-parameters:
 - Number of filters K ,
 - their spatial extent F ,

- the stride S ,
- the amount of zero padding P .
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F + 2P)/S + 1$
 - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D_2 = K$
- With parameter sharing, it introduces $F \times F \times D_1$ weights per filter, for a total of $(F \times F \times D_1) \times K$ weights and K biases.
- In the output volume, the d -th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d -th filter over the input volume with a stride of S , and then offset by d -th bias.

A common setting of the hyperparameters is $F = 3$, $S = 1$, $P = 1$. However, there are common conventions and rules of thumb that motivate these hyper-parameters [24].

3.1.2 Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Convolution layers in a CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX

operation would in this case be taking a max over 4 numbers (little 2×2 region in some depth slice). The depth dimension remains unchanged.

Pooling Layer can be summarized as,

- Accepts a volume of size $W_1 \times H_1 \times D_1$.
- Requires two hyper-parameters:
 - their spatial extent F ,
 - the stride S ,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F)/S + 1$
 - $H_2 = (H_1 - F)/S + 1$
 - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input, note that it is not common to use zero-padding for Pooling layers, it is worth noting that there are only two commonly seen variations of the max pooling layer found in practice:
 1. A pooling layer with $F = 3, S = 2$ (also called overlapping pooling).
 2. And more commonly $F = 2, S = 2$.

Pooling sizes with larger receptive fields are too destructive [24].

3.1.3 Normalization Layer

Many types of normalization layers have been proposed for use in Convolutional Net architectures, sometimes with the intentions of implementing inhibition schemes observed in the biological brain. However, these layers have since fallen out of favor because in practice their contribution has been shown to be minimal, if any [24].

3.1.4 Fully-Connected Layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

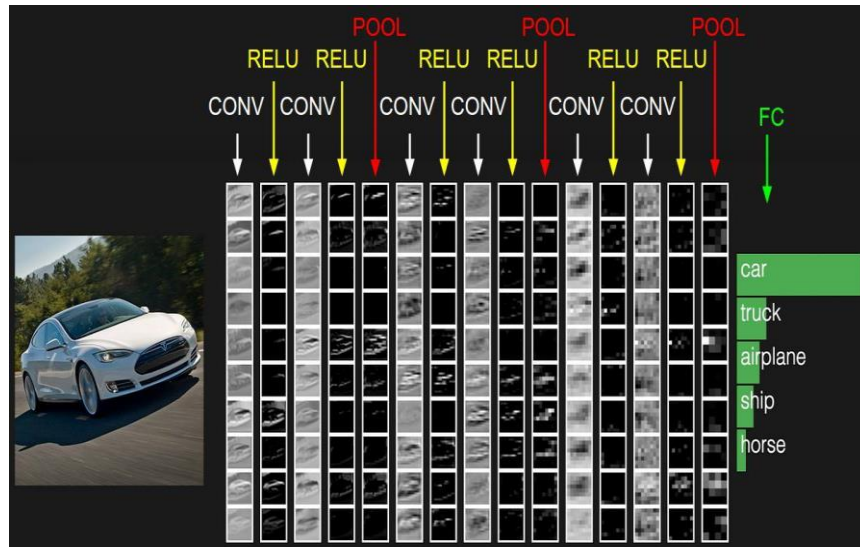


Fig.3.2 Representation of Layers of Convolutional Neural Network [24].

3.2 Famous Convolutional Neural Network Architectures.

There are several architectures in the field of Convolutional Networks. The most common are the following.

1. LeNet. The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.

2. AlexNet. The first work that popularized Convolutional Networks in Computer Vision was the AlexNet, developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton. The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly

outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer).

3. ZF Net. The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus. It became known as the ZFNet (short for Zeiler & Fergus Net). It was an improvement on AlexNet by tweaking the architecture hyperparameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller.

4. GoogLeNet. The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. [16] from Google. Its main contribution was the development of an *Inception Module* that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, this paper uses Average Pooling instead of Fully Connected layers at the top of the ConvNet, eliminating a large amount of parameters that do not seem to matter much.

5. VGGNet. The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end.

6. ResNet. Residual Network developed by Kaiming He et al. was the winner of ILSVRC 2015. It features special *skip connections* and a heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network.

Chapter 4

TRANSFER LEARNING

In our research, we started with AlexNet which was mentioned briefly in previous chapter. AlexNet was developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton and was submitted to the ImageNet ILSVRC challenge in 2012 and it significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). It was trained to classify 1.2 million images for ILSVRC challenge. AlexNet has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax.

In this chapter we discuss about the AlexNet architecture, its features which encouraged us to use as the CNN base for performing transfer learning and then we discuss about transferring the rich feature hierarchies of AlexNet learned by large image dataset to our specific task of road object detection.

4.1 AlexNet Architecture

AlexNet contains eight learned layers five convolutional and three fully-connected. Below, we describe some of the novel or unusual features of AlexNet.

4.1.1 ReLU Nonlinearity

Instead of using the standard way to model a neuron's output i.e. f as a function of its input x with $f(x) = \tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$ they used $f(x) = \max(0, x)$, as in terms of training time with gradient descent saturating nonlinearities are much slower than the non-saturating nonlinearity.

Following Nair and Hinton [17], they refer to neurons with this nonlinearity as Rectified Linear Units (ReLUs). Deep convolutional neural networks with ReLUs train several times faster than their equivalents with *tanh* units [19].

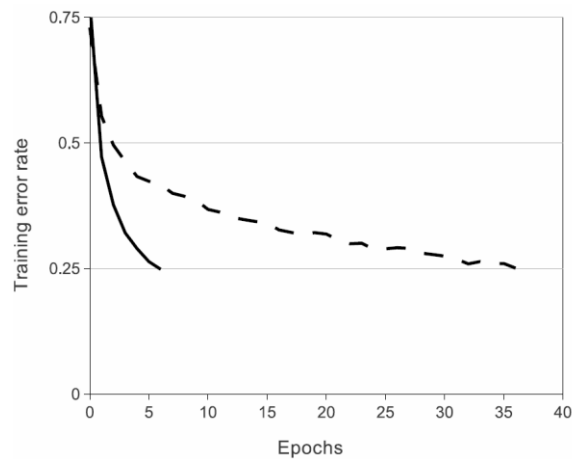


Figure 3.1 Training error rate vs Epochs [19]

This is demonstrated in Figure 3.1, which shows A four-layer convolutional neural network with ReLUs (solid line) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with *tanh* neurons (dashed line). The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture. Networks with ReLUs consistently learn several times faster than equivalents with saturating neurons [19].

4.1.2 Training on Multiple GPU's

A single GPU has memory constraints, which limits the maximum size of the networks that can be trained on it. Therefore, in AlexNet the net is spread across two GPUs. Current GPUs are particularly well-suited to cross-GPU parallelization, as they can read from and write to one another's memory directly, without going through host machine memory.

The employed parallelization scheme essentially puts half of the kernels (or neurons) on each GPU, with one additional trick: the GPUs communicate only in certain layers. For example, the kernels of layer 3 take input from all kernel maps in layer 2. However, kernels in layer 4 take input only from those kernel maps in layer 3 which reside on the same GPU.

The resultant architecture is somewhat like that of the "columnar" CNN employed by Cireşan et al. [18], except that AlexNet columns are not independent. This scheme reduces AlexNet's top-1 and top-5 error rates by 1.7% and 1.2%, respectively, as compared with a net with half as many kernels in each convolutional layer trained on one GPU. The two-GPU net takes slightly less time to train than the one-GPU net [4].

4.1.3 Localization Response Normalization.

ReLUs have the property that they do not require input normalization to prevent them from saturating. If at least some training examples produce a positive input to a ReLU, learning will happen in that neuron.

However, the following local normalization scheme aids generalization. Response normalization reduces AlexNet's top-1 and top-5 error rates by 1.4% and 1.2%

$$b_{x,y}^i = a_{x,y}^i / (k + \alpha \sum_{j=\max(0,i-\frac{n}{2})}^{\min(N-1,i+\frac{n}{2})} (a_{x,y}^j)^2)^\beta$$

where, $b_{x,y}^i$ is Response-normalized activity and $a_{x,y}^i$ is the activity of a neuron computed by applying kernel i at position (x, y) and then applying the ReLU nonlinearity [4].

4.1.4 Overlapping Pooling

Traditional local pooling commonly employed in CNNs is when we set $s = z$ but, if we set $s < z$, we obtain overlapping pooling. This is what is used throughout AlexNet network, with $s = 2$ and $z = 3$. This scheme reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively, as compared with the non-overlapping scheme $s = 2; z = 2$, which produces output of equivalent dimensions.

4.1.5 Overall Architecture.

The AlexNet contains eight layers with weights; the first five are convolutional and the remaining three are fully connected. The output of the last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels [4].

The kernels of the second, fourth, and fifth convolutional layers are connected only to those kernel maps in the previous layer which reside on the same GPU. The kernels of the third convolutional layer are connected to all kernel maps in the second layer. The neurons in the fully connected layers are connected to all neurons in the previous layer.

Response-normalization layers follow the first and second convolutional layers. Max-pooling layers, of the kind described in Section 4.1.4, follow both response-normalization layers as well as the fifth convolutional layer. The ReLU non-linearity is

applied to the output of every convolutional and fully-connected layer. The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring [4]. Pictorial representation of AlexNet is shown in the following figure 3.2

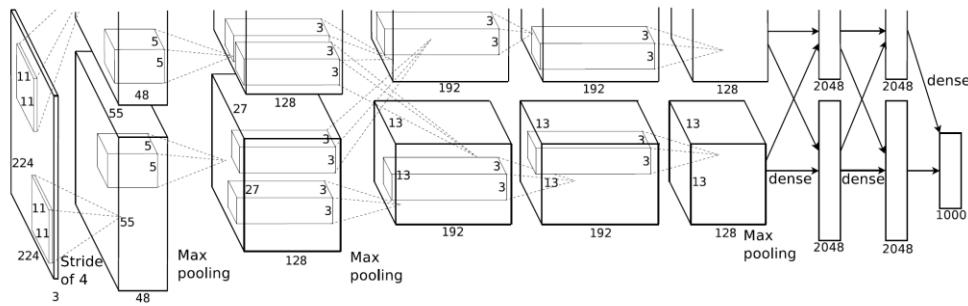


Figure 4.2. AlexNet Final Architecture [4].

4.2 Fine-tuning AlexNet for Road Objects Detection.

Transfer learning is a workflow that is commonly used in deep learning applications. In transfer learning, a network trained on a large collection of images, such as ImageNet [5], is used as the starting point to solve a particular detection task. In our case we used AlexNet discussed in section 4.1 and fine-tuned it to detect Road objects.

The advantage of using this approach is that the pre-trained network (AlexNet) has already learned a rich set of image features that are applicable to a wide range of images. This learning is transferable to the new task by fine-tuning the network.

Also transfer learning has an advantage that the number of images required for training and the training time is reduced; both of them being important constraints for this thesis.

In our approach first, we load a pretrained AlexNet and then this pre-trained AlexNet is fine-tuned for detection of Objects on road. We used the AlexNet as a CNN base to create a Region based Convolutional Neural Network for object detection. Detailed explanation of the process and RCNN architecture is given in the following chapter.

Chapter 5

REGION-BASED METHODS FOR OBJECT DETECTION

In this chapter we will talk about the Region based methods for object detection. In particular, R-CNN (Regional CNN), the original application of CNNs to this problem, along with its descendants Fast R-CNN, and Faster R-CNN.

In classification, there's generally an image with a single object as the focus and the task is to say what that image is. But when we look at the world around us, we carry out far more complex tasks. We see complicated sights with multiple overlapping objects, and different backgrounds and we not only classify these different objects but also identify their boundaries, differences, and relations to one another. *The goal of a Region based CNN is to take an image input to detected and localize the object in the image* [1].

We implemented all three above mentioned architectures for object detection and localization for a driving scenario to detect cars, pedestrians and traffic signs etc. and performed a time study to conclude which of the three architectures can be used as a detector for a real-world driving scenario for detection of objects present on road.

We will also discuss their architecture and describe how they perform the task of object detection and localization given the input image.

5.1 R-CNN

R-CNN was first introduced by Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik [1] in 2014. Region based Convolutional Network(R-CNN) combines two ideas: (1) one can apply high-capacity Convolutional Networks (CNN'S) to bottom-up region proposals in order to localize and segment objects and (2) When labelling data is

scarce, supervised pretraining for an auxiliary task, followed by domain-specific-fine-tuning, boosts performance significantly [1].

They combined region proposals with CNNs and called the resulting model an R-CNN or Region-based Convolutional Network.

5.1.1 R-CNN Architecture.

In R-CNN we input an image and then using the basic feature detection techniques such as edge detection etc. we get Region Proposals Also referred as Regions of interest. This process is also known as selective search [14].

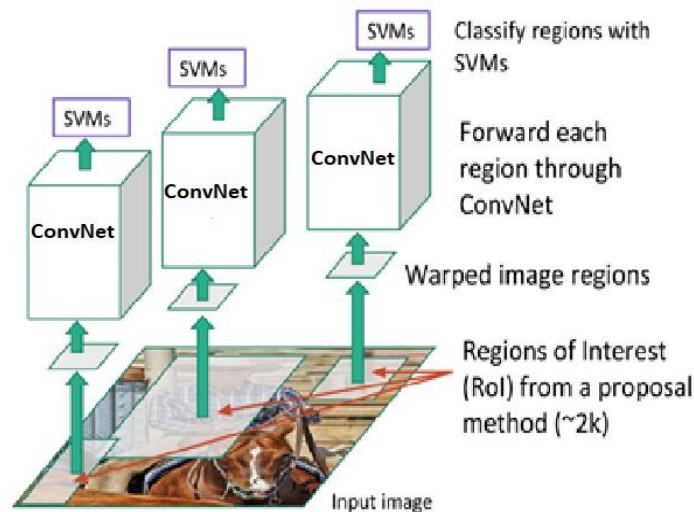


Figure 5.1 R-CNN Architecture [1]

In the next step after these region proposals are extracted these wrapped image regions go through a trained CNN in our implementation AlexNet and then on the final layer the classification is done Using a Support Vector Machine (SVM) classifier which classifies whether this is an object and if so what. Pictorial representation of above process is shown in Figure 5.2.

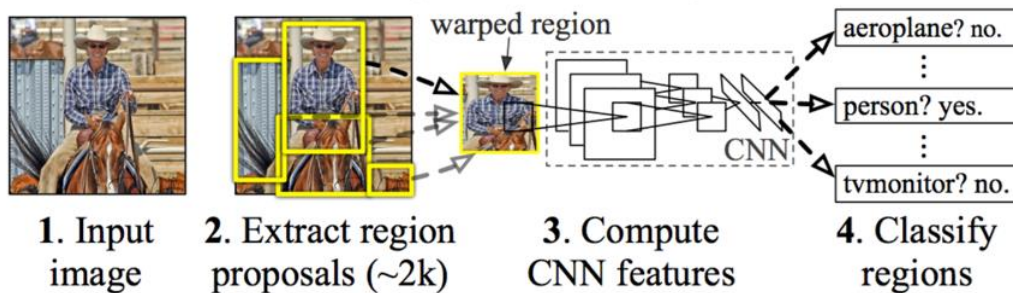


Figure 5.2 Procedure for Object detection and Localization in R-CNN [1].

5.1.2 Implementation of R-CNN for Road Object Detection.

We implemented the R-CNN architecture with AlexNet as base CNN to classify region proposals. We provided 5969 labelled images with label categories as Bus, Cars, Pedestrians, and Traffic Signs. Labelling was done with help of MATLAB R2017b Ground Truth Labeler application.

Using MATLAB R2017B we trained an RCNN which took approximately 30 hours to train the network.

5.1.3 Results

Time Study.

We used our trained RCNN to check how much time it takes for accurate detection of road objects in each Image and we found out it takes approximately 19 to 40 seconds to process each image. We also checked if the number of categories of objects of interest can also be a factor for detection time, so we created five different RCNN's and performed a time study and the results are as follows.

	1 Object of Interest	2 Objects of Interest	2 Different Objects of Interest	3 Objects of Interest
RCNN 1	15.586	20.25	20.19	22.03
RCNN 2	14.565	18.85	19.10	21.30
RCNN 3	17.432	19.59	18.90	23.35
RCNN 4	14.4031	19.85	20.25	22.15
RCNN 5	15.70	19.50	19.04	23.035

Table 5.1. Time study of RCNN's (Time in seconds)

From the time study we conclude that the number of objects of interest do not play much important part in increasing the detection time but the architecture itself takes a lot of time for object detection and localization given an input image. We also compared our results with the paper “Rich Feature Hierarchies for accurate object detection and semantic segmentation” [19] and found out similar pattern in detection time as shown in following Figure.

We concluded in accordance with [19] the reason for such high detection time is generation of region proposals i.e. around 2000 region proposals per Image input and the network also takes lot of time to train as It must train three different models separately - the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes.

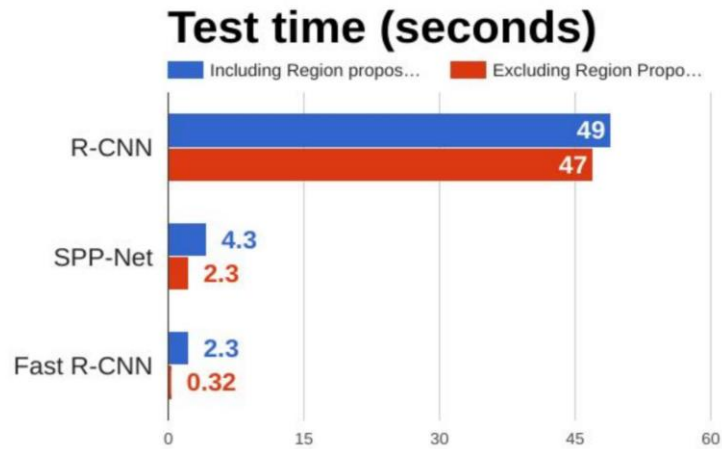


Figure 5.3 Time study Girshik et al. [19]

5.1.4 Conclusion

R-CNN works really well but is really quite slow for implementation in real time object detection. The reasons being

1. It requires a forward pass of the CNN (AlexNet) for every single region proposal for every single image that's around 2000 forward passes per image.
2. It must train three different models separately - the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes. This makes the pipeline extremely hard to train.

5.2 Fast R-CNN

In 2015, Ross Girshick, the first author of R-CNN [1], solved both these problems and the resulting architecture is called as Fast R-CNN [10].

In this architecture we input an image and then forward the Image through a CNN (AlexNet) which produces a feature map of images from where we get our Region proposals. In the next step RoI pooling is done. RoIPool shares the forward pass of a CNN for an image across its sub regions. Then, the features in each region are pooled (usually using max pooling). Hence taking just one pass of the original image as opposed to ~2000 in R-CNN discussed in section 5.1

Secondly, Fast R-CNN jointly trains the CNN, classifier, and bounding box regressor in a single model. Fast R-CNN replaced the SVM classifier with a softmax layer on top of the CNN to output a classification. It also added a linear regression layer parallel to the softmax layer to output bounding box coordinates. In this way, all the outputs needed came from one single network.

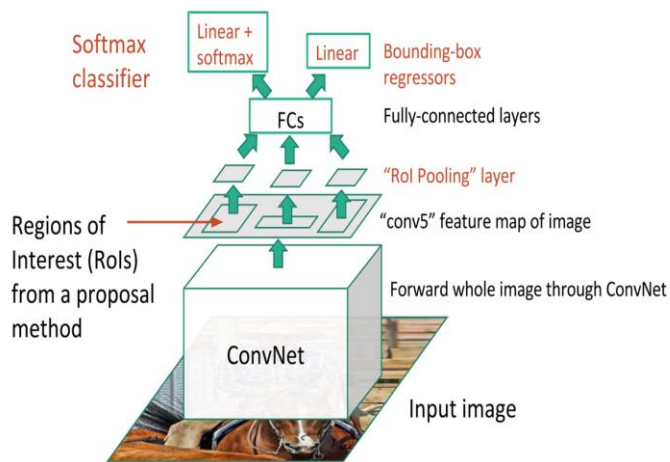


Figure 5.4 Fast R-CNN Architecture [10].

5.2.2 Implementation of Fast R-CNN for Road Object Detection.

In similar manner as R-CNN we implemented the Fast R-CNN architecture with AlexNet as base CNN to extract region proposals. We provided 5969 labelled images with label categories as Bus Cars, Pedestrians, and Traffic Signs. Labelling was done with help of MATLAB R2017b Ground Truth Labeler application.

Using MATLAB R2017B we trained an RCNN which took approximately 28 hours to train the network.

5.2.3 Results-Fast R-CNN

Time study

We performed the time study for Fast RCNN in the similar manner of RCNN as discussed in section 5.1.3. and we compared our results with the paper “Rich Feature Hierarchies for accurate object detection and semantic segmentation” [19] and found out similar pattern in detection time as shown in Figure 5.7.

	1 Object of Interest	2 Objects of Interest	2 Different Objects of Interest	3 Objects of Interest
Fast RCNN 1	.85	2.01	2.84	3.32
Fast RCNN 2	1.565	2.90	1.98	2.96
Fast RCNN 3	1.93	3.55	2.89	2.30
Fast RCNN 4	1.61	2.94	3.85	3.15
Fast RCNN 5	1.87	1.97	2.79	3.38

Table 5.2. Time study of Fast R-CNN (time in seconds)

5.2.4 Conclusion

The time study indicates that the Fast R-CNN are comparatively faster than the R-CNN. But if we observe carefully and in accordance with [19] the Runtime of the Fast R-CNN is dominated in computation of Region Proposals. To reduce that a new approach was proposed which is called Faster R-CNN which has also paved a way for Real-time Implementation of Region based methods for object detection.

Few Detection Results of our RCNN and Fast RCNN



Figure.5.5 Detection results (Image source [39], [40], [41], [42])

5.3 Faster R-CNN

In 2015, a team at Microsoft Research composed of Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, collectively created Faster R-CNN[11].

They rectified the remaining bottleneck in the Fast R-CNN process—the region proposer. As we saw, the very first step to detecting the locations of objects is generating a bunch of potential bounding boxes or regions of interest to test. In Fast R-CNN, these proposals were created using selective search , a fairly slow process that was found to be the bottleneck of the overall process.

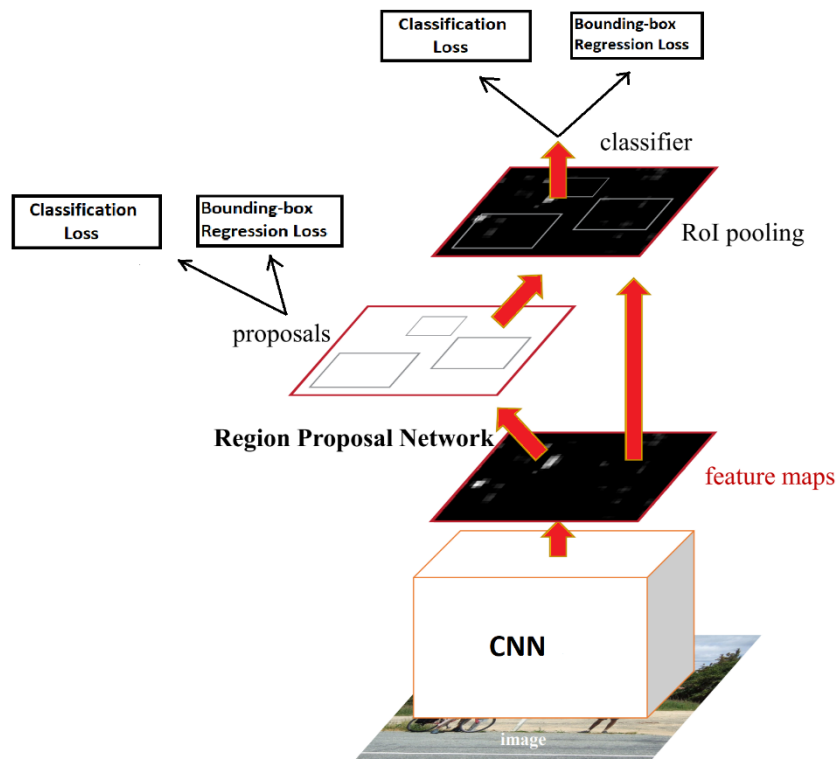


Figure 5.6 Faster R-CNN Architecture [11].

Similar to Fast R-CNN in this architecture Image input is given to a CNN from which we get a convolutional feature map.

Then it goes through a Region Proposal Network which is trained with the ground truth data to output the Region Proposals.

Generation of Region proposals

Faster R-CNN generates these region proposals from CNN features. Faster R-CNN adds a Fully Convolutional Network on top of the features of the CNN creating what's known as the Region Proposal Network.

The Region Proposal Network works by passing a sliding window over the CNN feature map and at each window, outputting k potential bounding boxes and scores for how good each of those boxes is expected to be.

Intuitively, we know that objects in an image should fit certain common aspect ratios and sizes. In such a way, we create k such common aspect ratios we call anchor boxes. For each such anchor box, we output one bounding box and score per position in the image.

And finally, the RoI pooling is done to match the Inputs of the CNN and then the image goes through the Fully Connected convolution layer.

5.3.2 Implementation of Faster R-CNN for Road Object Detection.

Similar to R-CNN and Fast R-CNN we implemented the Fast R-CNN architecture with AlexNet as base CNN to train a Region Proposal Network. We provided 5971 labelled images with label categories as Cars, Pedestrians, and Traffic Signs Bus and Traffic Light. Labelling was done with help of MATLAB R2017b Ground Truth Labeler application.

Using MATLAB R2017B we trained an Faster R-CNN which took approximately 23 hours to train the network.

5.3.3 Results- Faster R-CNN

Time Study

We performed the time study, but in slightly different manner than we did in RCNN and Fast RCNN as we concluded that the number of objects of interest in an particular image do not play a major role. We concluded that the approximate detection time per image is approximately 0.2 seconds. We then compared our results with Girshik et al. [19] and found similar results. The comparative time for detection for the region based Methods for object detection is shown in following figure.

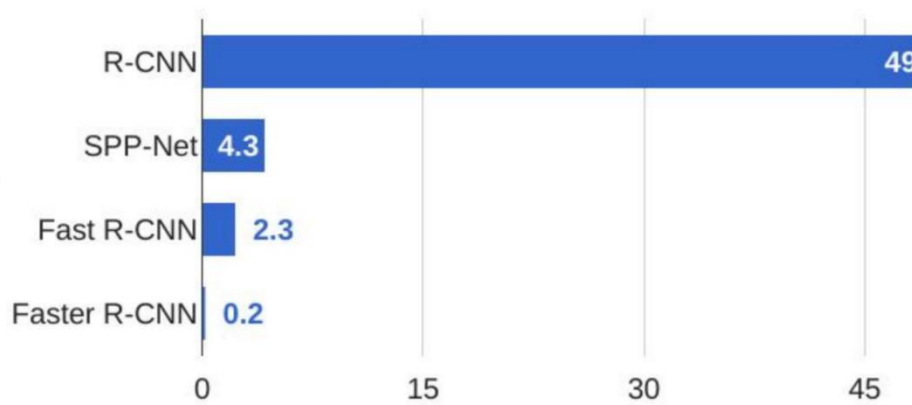


Figure 5.7 Comparison of Detection Time Per Image for Region Based CNN Architectures. Girshik et al. [19]

Few Detections of Faster R-CNN

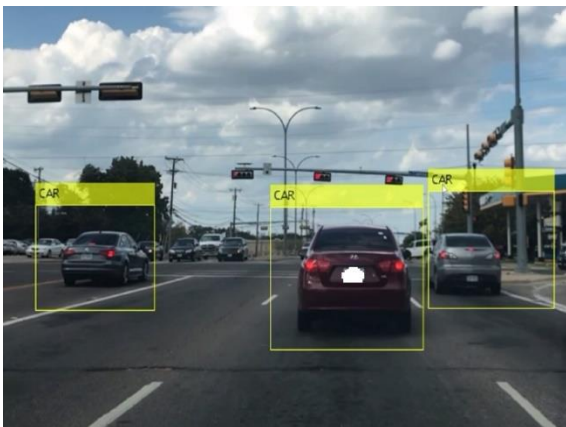
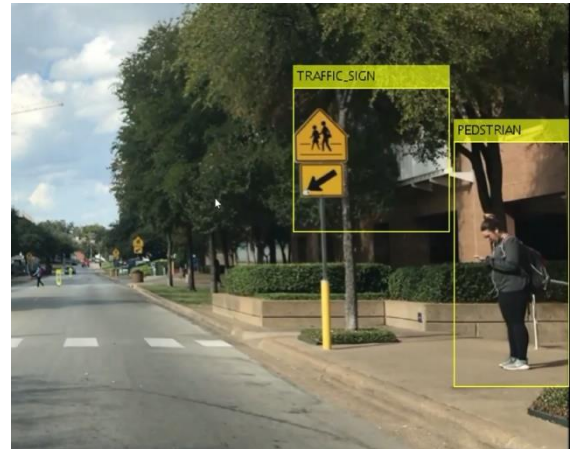
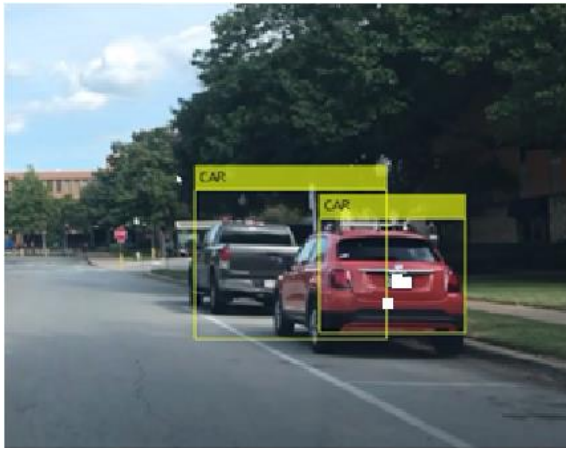


Fig.5.8 Detection Results Faster R-CNN (Result images source– Self generated data set)

5.3.4 Conclusion

With the help of the time study performed we were able to conclude that the Faster R-CNN can be used for real time object detection and localization. We also calculated the Mean Average Precision (mAP) of our Faster R-CNN which came out to approximately 70% which is satisfactory for the amount of the training data provided.

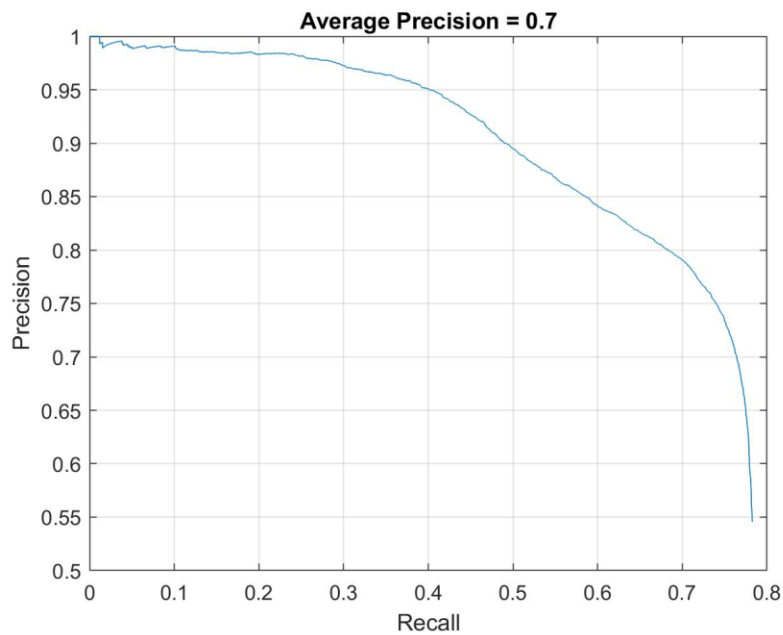


Fig 5.9 Mean Average Precision (mAP) of our Faster RCNN

We then applied our trained Faster R-CNN to measure the distance of objects in our case Cars, Pedestrian, and Traffic Signs etc. from the sensor (camera) mounted on the car which shows how object detection using computer vision paves a path towards ADAS applications and plays an important role towards vehicle autonomy.

Chapter 6

IMPLEMENTATION OF FASTER R-CNN FOR DEPTH ESTIMATION

6.1 Introduction

Stereo Vision, like human eyes is able to calculate distance from to images taken from different views. When looking from two different locations the same object will appear to be at two different locations which is called as disparity. Then using this disparity, the depth information is calculated using the stereo parameters of the camera.

6.2 Stereo Vision.

Analysis of video images in stereo has emerged as an important passive method for extracting the three-dimensional (3-D) structure of a scene [20]. A simplified stereo imaging system is shown in Figure 5.1 and summarized below [21] :

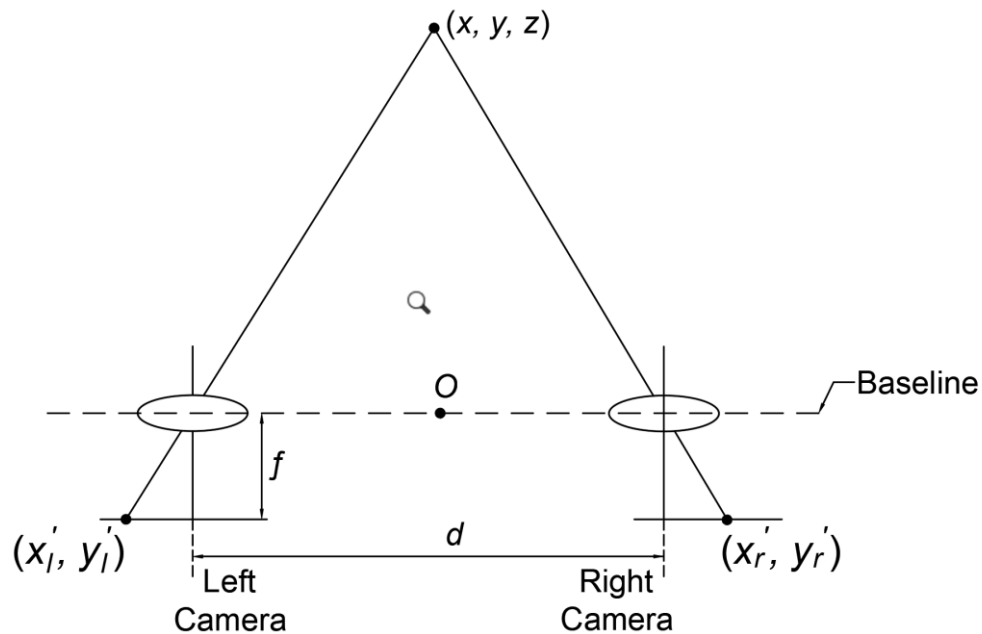


Figure 6.1 A Stereo Imaging System [21]

- Two cameras with their optical axes parallel and separated by a distance d .
- The line connecting the camera lens centers is called the baseline.
- The focal length of both cameras is f .
- Let the origin O of this system be mid-way between the lens centers.
- Let the x axis of the 3-D world coordinate system be parallel to the baseline.
- Consider a point (x, y, z) in 3-D world coordinates on an object.
- Let the point (x, y, z) have image coordinates (x'_l, y'_l) in the left plane.
- Let the point (x, y, z) have image coordinates (x'_r, y'_r) in the right plane.

The goal of stereo vision in this research is to estimate depth information from a pair of stereo images. With two cameras separated by a fixed distance, each camera receives a slightly different image of the same scene in the real world. If we can successfully determine which feature characteristics in the image from the left camera correspond with which in the image from the right camera, and if we know the stereo imaging geometry and camera focal length, it is possible to reconstruct the depth information.

Generally, the major stages involved in the stereo vision are pre-processing of images to obtain matching features, recovering the disparity between the images by a suitable stereo algorithm, and using geometry to recover the stereo depth.

We implemented this technique using following work flow-

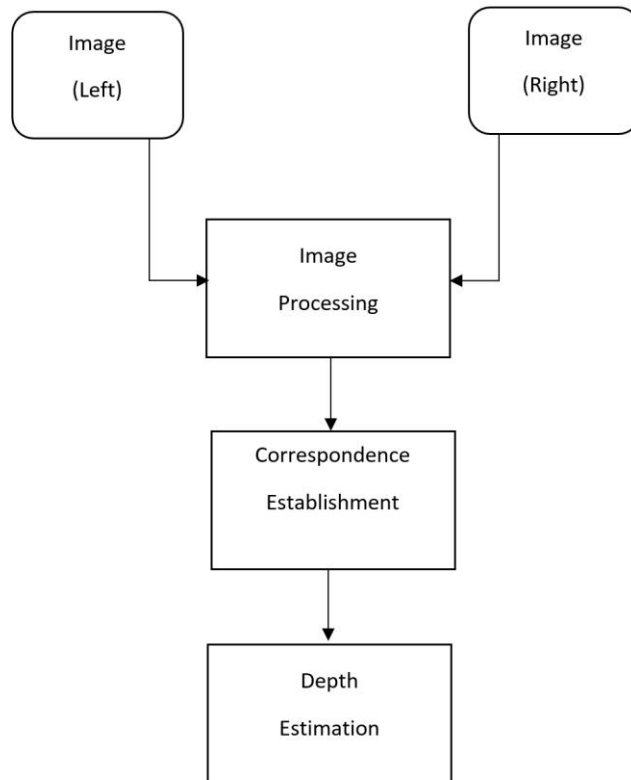


Figure 6.2. Depth Estimation Process Work Flow.

6.3 Camera Calibration

Geometric camera calibration estimates the parameters of a lens and image sensor of the camera used to capture an image or a video. These parameters are helpful in correction for lens distortion, measurement of object size or determining camera location in scene. This enables us to implement machine vision to detect and measure objects and in navigation system for robots and 3D scene reconstruction.

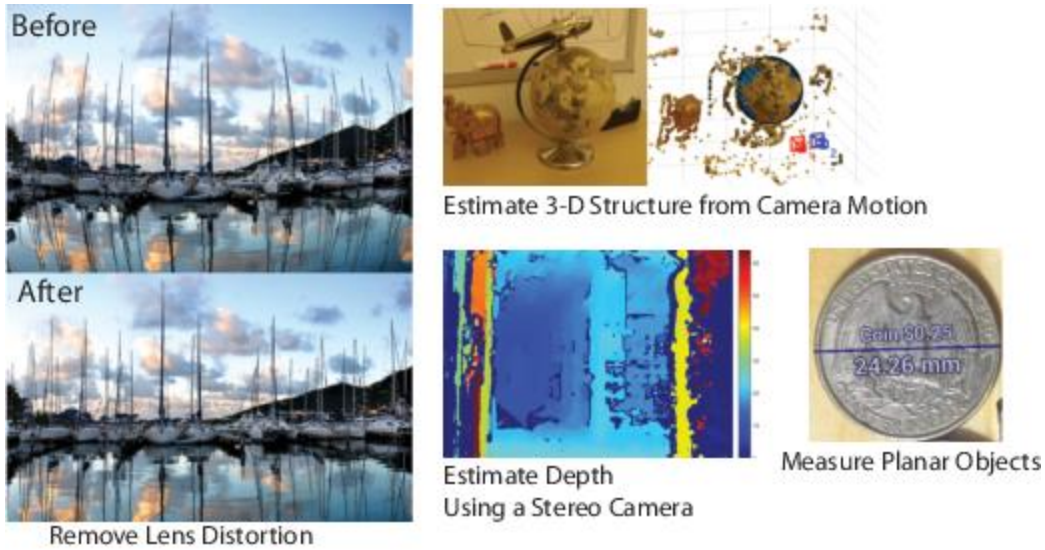


Figure 6.3 Examples of what you can do after calibrating the camera [25].

The camera parameters include intrinsic, extrinsic and distortion coefficients. The extrinsic parameters represent a rigid transformation from 3-D world coordinate system to the 3-D camera's coordinate system. The intrinsic parameters represent a projective transformation from the 3-D camera's coordinates into the 2-D image coordinates. Let 3D world coordinates be $[X, Y, Z]$ and $[X_c, Y_c, Z_c]$ be the camera co-ordinates and $[x, y]$ be the 2D image co-ordinates then the transformation can be represented as shown in Figure 5.4.

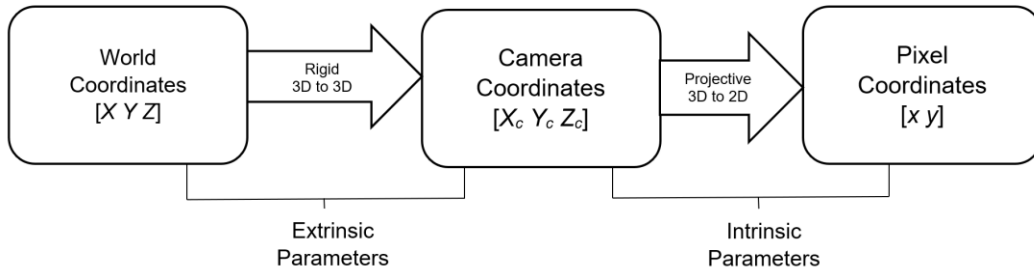


Figure 6.4 Intrinsic and Extrinsic parameters [25].

This transformation can be implemented by using images of calibration pattern like a checkerboard giving us the camera parameters.

The camera calibration process is performed using Camera Calibration Toolbox in MATLAB R2017b. Multiple sets of images captured from left and right lenses of a stereo camera are used to find identify points of correspondence and used to estimate the camera parameters.

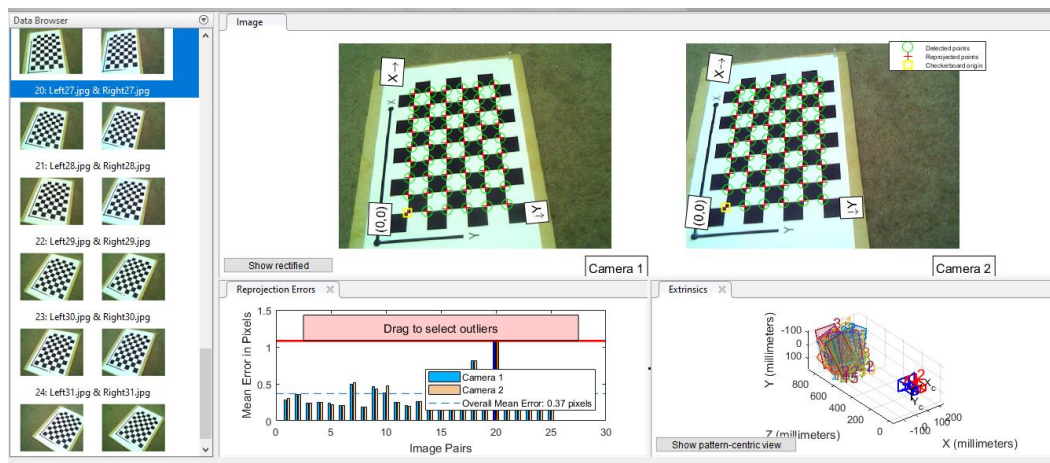


Figure 6.5 Camera Calibration Session in MATLAB R2017b

6.4 Disparity Mapping

As mentioned in [26], A system of stereo vision system consists of a stereo camera, namely, two cameras placed horizontally (i.e., one on the left and the other on the right). The two images captured simultaneously by these cameras are then processed for the recovery of visual depth information [27]. Depth information can be computed from a pair of stereo images by first computing the distance in pixels between the location of a feature in one image and its location in the other image. This gives us a

disparity map. It looks a lot like a depth map because pixels with larger disparities are closer to the camera, and pixels with smaller disparities are farther from the camera.

The block matching algorithm uses the local method of using constraints on small number of pixels surrounding the pixel of interest. The disparity at a point is estimated by comparing a small region about that point with congruent regions extracted from the other image. Block matching searches one image for the best corresponding region for a template in the other image. In our implementation we used Block matching for calculating the disparity.

6.5 Object Detection

The object detection is done using the trained Faster R-CNN which was discussed in Chapter 5. After the objects of interest are detected by the Faster R-CNN we generate a bounding box around it and then we find the centroid of the bounding box to calculate the disparity.

6.6 3D Reconstruction and Depth Estimation.

It is possible to estimate the 3D scene geometry with stereo vision, given only two images from the same scene. Rectification simplifies the stereo correspondence problem considerably which allows for a straight-forward computation of dense disparity maps, which are the basis of dense 3D reconstruction. Every value in the disparity map can be re-projected to a 3D point, with respect to camera co-ordinate frame.

When two images are acquired by a stereo camera system, every physical point M yields a pair of 2D projections m_1 and m_2 on two images [28]. If we know both the

intrinsic and extrinsic parameters of the stereo system, we can reconstruct the 3D location of the point M from m_1 and m_2 [29], [30].

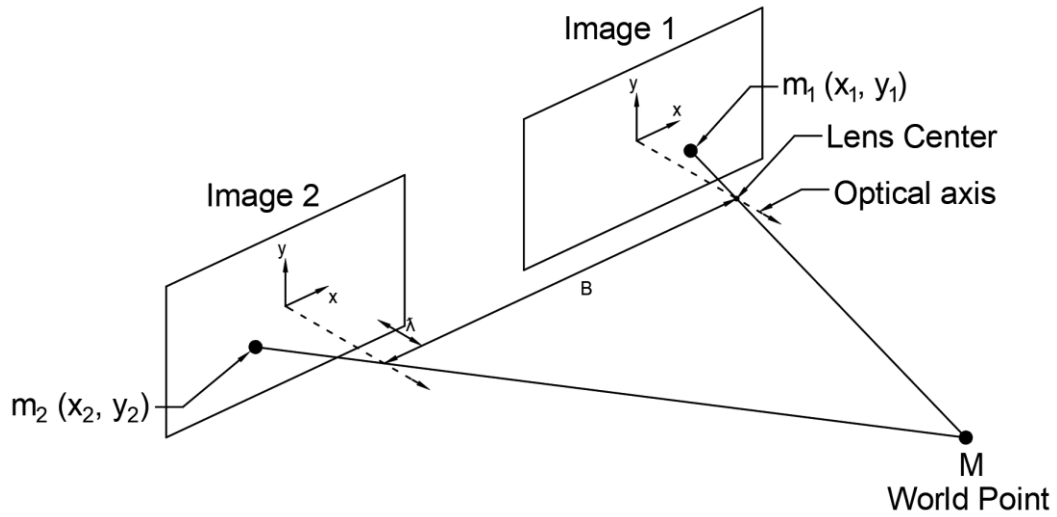


Figure 6.6 Stereo Camera Geometry [28].

The depth of a point M can be calculated by:

$$Z = f - \frac{fB}{x_2 - x_1}$$

where B is the baseline distance between two cameras and f is the focal length of the camera. We assume the parallel camera geometry for simplicity.

6.7 Results of Depth Estimation.

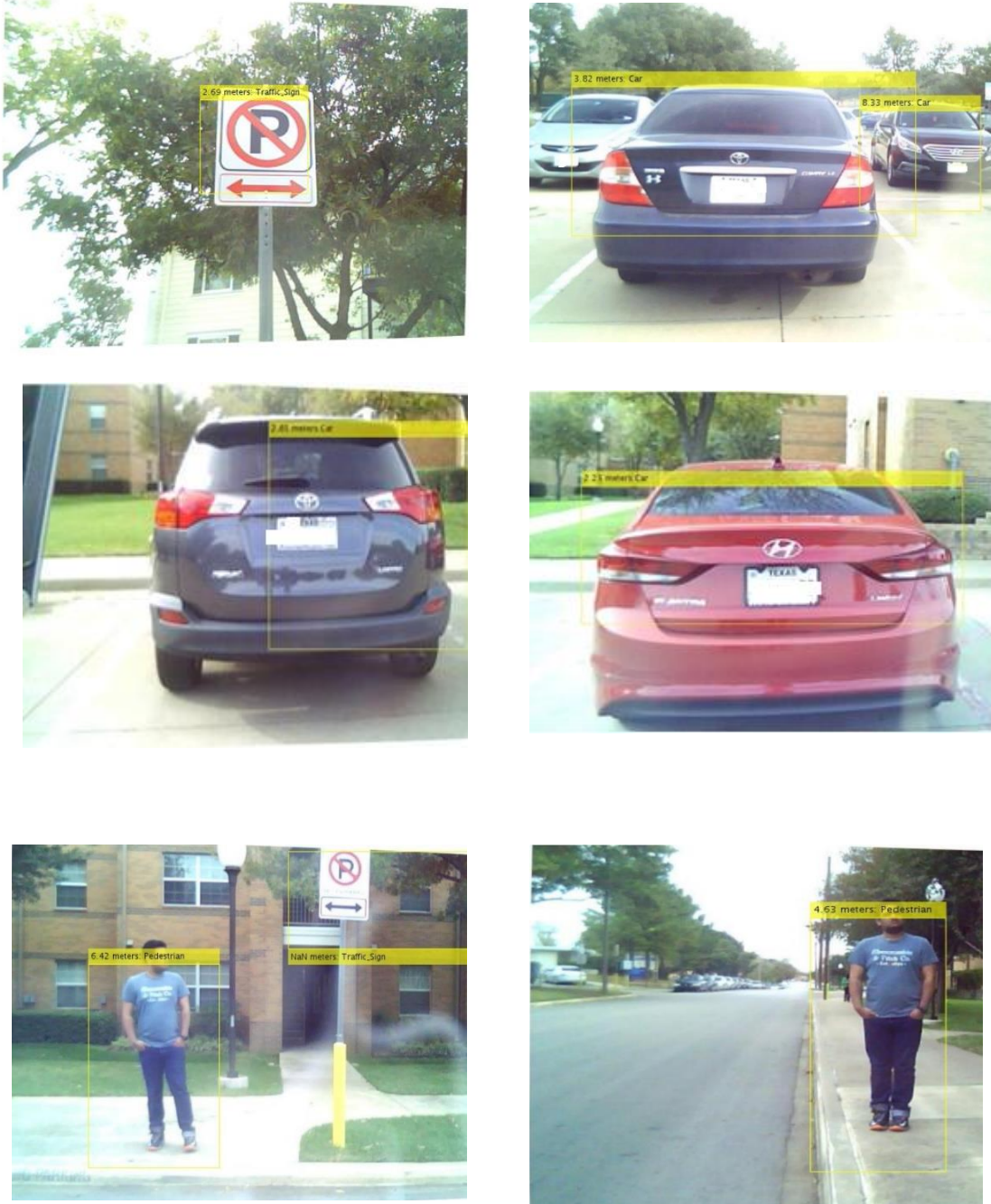


Figure 6.7. Results of Depth Estimation (Result images source– Self generated data set)

Chapter 7

CONCLUSION

In this work we presented a Faster Region Based Convolutional Neural Network (Faster RCNN) architecture that works well for the detection of objects which are present on road in a common driving scenario.

As part of developing and analyzing this approach we provided analysis of many architectural choices for the network, discussing best practices for training, and demonstrated the importance of fine-tuning, proposal generation and how depth of the CNN model effects the detection performance. We also did a comparative time study of Region based convolutional networks and concluded which of the region based methods for object detection and localization can lead to real-time object detection. We also concluded how training data quality and quantity can be a contributor to the networks mean average precision (mAP).

We also implemented our network for the depth estimation problem and saw how computer vision can be used to address certain important aspects of SLAM problem.

Chapter 8

FUTURE SCOPE

Object Detection and Recognition using Computer Vision for ADAS application has been an active area of research in last half decade with continuous innovations in both Algorithms for detection and sensor quality and capability. The Region based methods now have a successor which was introduced in 2017 called the Mask R-CNN which is a step further and does pixel level object segmentation instead of giving a bounding box output.

Furthermore, the ability to extract information beyond image structure from the knowledge available freely online, through understanding of the image, as well as the associated text is a promising way to build knowledge bases and accelerate the pace of visually sentient systems. New learning systems such as NEIL: Never Ending Image Learner [22] and LEVAN: Learning Everything about Anything [23] are some of the initial attempts in creating such systems more recently.

Also, there has been large investments in development of ADAS systems using computer vision by OEM'S which is accelerating the amount of research done in this sector as well as development of suitable hardware (Stereo Cameras) specifically for ADAS application

To conclude Object Detection and Recognition using computer vision is one of the most discussed and debated topic in the current decade and there is still a long way to go. The time is near when we will be seeing fully Autonomous Vehicles on the road and Computer Vision and Machine Learning will be one of the major factors driving that technology.

References

- [1] Ross Girshick, Jeff Donahue, Student Member, IEEE, Trevor Darrell, Member, IEEE, and Jitendra Malik, Fellow, IEEE. "Region-Based Convolutional Networks for Accurate Object Detection and segmentation" Copyright © 2016, IEEE.
- [2] D. Lowe, "Distinctive image features from scale-invariant keypoints", International Journal Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.
- [3] N. Dalal, B. Triggs, Proc. IEEE Conference Computer Vision Pattern Recognition., pp. 886-893, 2005.
- [4] A. Krizhevsky, I. Sutskever, G. Hinton, "ImageNet classification with deep convolutional neural networks", Proc. Advances in Neural Information Processing Systems, pp. 1106-1114, 2012.
- [5] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. Imagenet large scale visual recognition competition 2012 (ILSVRC2012) [Online]. Available: <http://www.image-net.org/challenges/LSVRC/2012/>, 2012.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, "ImageNet: A large-scale hierarchical image database", IEEE Conference. Computer Vision Pattern Recognition., pp. 248-255, 2009.
- [7] C. Szegedy, A. Toshev, D. Erhan, "Deep neural networks for object detection", Advances in Neural Information Processing Systems, pp. 2553-2561, 2013.
- [8] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, "Object detection with discriminatively trained part based models", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1627-1645, Sep. 2010.

- [9] T. Dean, J. Yagnik, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, "Fast accurate detection of 100000 object classes on a single machine", IEEE Conference Computer Vision Pattern Recognition, 2013.
- [10] Ross Girshick "Fast R-CNN" [arXiv:1504.08083v2](https://arxiv.org/abs/1504.08083v2) [cs.CV]
- [11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". [arXiv:1506.01497](https://arxiv.org/abs/1506.01497) [cs.CV]
- [12] R. Vaillant, C. Monrocq, Y. LeCun, "Original approach for the localisation of objects in images", IEE Proceedings Vision and Image Signal Processing, vol. 141, no. 4, pp. 245-250, Aug. 1994.
- [13] N. Dalal, B. Triggs, Proc. IEEE Conference. Computer Vision and Pattern Recognition, pp. 886-893, 2005.
- [14] J. Uijlings, K. van de Sande, T. Gevers, A. Smeulders, "Selective search for object recognition", International Journal Computer Vision, vol. 104, no. 3, pp. 154-171, 2013.
- [15] J. Hosang, R. Benenson, P. Dollár, B. Schiele, "What makes for effective detection proposals?", arXiv e-prints, 2015.
- [16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich "Going Deeper with Convolutions" [arXiv:1409.4842v1](https://arxiv.org/abs/1409.4842v1) [cs.CV].
- [17] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of 27th International Conference on Machine Learning, 2010.

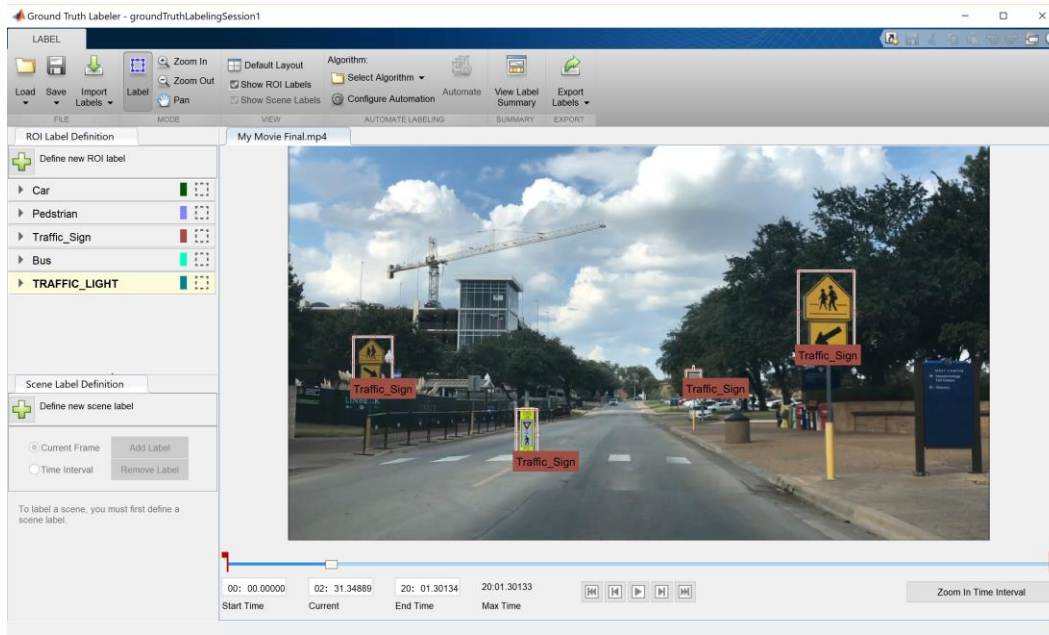
- [18] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classification. arXiv:1102.0183, 2011.
- [19] Ross Girshick, Jeff Donahue, Student Member, IEEE, Trevor Darrell, Member, IEEE, and Jitendra Malik, Fellow, IEEE." Rich Feature Hierarchies for accurate object detection and semantic segmentation" [arXiv:1311.2524v5](https://arxiv.org/abs/1311.2524v5) [cs.CV].
- [20] U. R. Dhond and J. K. Aggarwal, "Structure from Stereo-A Review," IEEE Transactions on Systems, Man and Cybernetics, vol. 19, pp. 1489-1510, Dec. 1989.
- [21] B. Fisher, Introduction to Stereo Imaging –Theory, University of Edinburgh, UK, https://users.cs.cf.ac.uk/dave/Vision_lecture/node11.html.
- [22] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In Proc. 14th International Conference on Computer Vision, volume 3, 2013.
- [23] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning.
- [24] <http://cs231n.github.io/convolutional-networks/>
- [25] <https://www.mathworks.com/help/vision/ug/camera-calibration.html>
- [26] Rostam Affendi Hamzah and Haidi Ibrahim, "Literature Survey on Stereo Vision Disparity Map Algorithms," Journal of Sensors, vol. 2016, Article ID 8742920, 23 pages, 2016. doi:10.1155/2016/8742920
- [27] B. H. Bodkin, Real-time mobile stereo vision [M.S. thesis], University of Tennessee, 2012.
- [28] H. Kim, Seung-jun Yang and Kwanghoon Sohn, "3D reconstruction of stereo images for interaction between real and virtual worlds," The Second IEEE and

- ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings., 2003, pp. 169-176.
- [29] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, The MIT Press, London, 2001
- [30] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, ch. 7, Prentice Hall, New Jersey, 1998.
- [31] M. Hansard, S. Lee, O. Choi, and R. P. Horaud, *Time-of-Flight Cameras: Principles, Methods and Applications*, Springer, 2012.
- [32] S. Foix, G. Alenyà, and C. Torras, "Lock-in time-of-flight (ToF) cameras: a survey," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, 2011.
- [33] M. Y. Kim, S. M. Ayaz, J. Park, and Y. Roh, "Adaptive 3D sensing system based on variable magnification using stereo vision and structured light," *Optics and Lasers in Engineering*, vol. 55, pp. 113–127, 2014.
- [34] S. Zhang, C. Wang, and S. C. Chan, "A new high resolution depth map estimation system using stereo vision and depth sensing device," in *Proceedings of the IEEE 9th International Colloquium on Signal Processing and Its Applications (CSPA '13)*, pp. 49–53, IEEE, Kuala Lumpur, Malaysia, March 2013.
- [35] W. Kazmi, S. Foix, G. Alenyà, and H. J. Andersen, "Indoor and outdoor depth imaging of leaves with time-of-flight and stereo vision sensors: analysis and comparison," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 88, pp. 128–146, 2014.
- [36] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, "Review of stereo vision algorithms and their suitability for resource-limited systems," *Journal of Real-Time Image Processing*, pp. 1–21, 2013.

- [37] X. Xiang, M. Zhang, G. Li, Y. He, and Z. Pan, "Real-time stereo matching based on fast belief propagation," *Machine Vision and Applications*, vol. 23, no. 6, pp. 1219–1227, 2012.
- [38] L. Nalpantidis and A. Gasteratos, "Stereovision-based fuzzy obstacle avoidance method," *International Journal of Humanoid Robotics*, vol. 8, no. 1, pp. 169–183, 2011.
- [39] "Safety Improvements on Tap in Minnesota Highway Plan" Published on December 27th, 2013, (last accessed on 11/27/2017).
<https://www.insurancejournal.com/news/midwest/2013/12/27/315550.htm>
- [40] Jeff Salton, "Volvo S60 details revealed ahead of Geneva premiere", Published on February 10th, 2010, (last accessed on 11/27/2017).
<https://newatlas.com/volvo-s60-production-debut/14128/>
- [41] Kelly Pleskot, "2015 Alfa Romeo 4C Priced at \$53900", Published on June 16th, 2014, (last accessed on 11/27/2017).
http://www.automotive.com/news/1406-2015-alfa-romeo-4c-priced-at-53900/photo_08.html
- [42] http://advision.com/traffic_signs.aspx (last accessed on 11/27/2017).

Appendix

Ground Truth labelling session in MATLAB R2017b



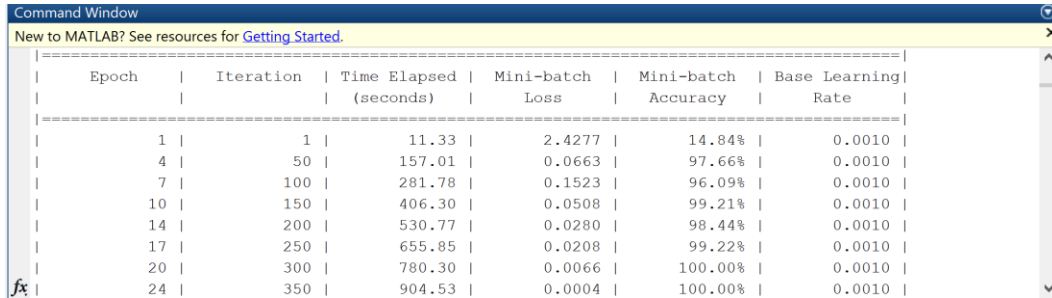
The above figure shows the labelling session in the MATLAB R2017b in which we can label the Ground truth data for the Neural Network. We have to manually label each and every category which we want train our network for. Though there are certain tools which help us to speed up the labelling process, but it is still a very time-consuming process

Training Data

	1	2	3	4	5	6	7	8	9	10	11	12
	imageFilename	Bus	Car	Pedstrian	Traffic_Sign							
1	'E:\finalLbels\...		[1158,709,1...]									
2	'E:\finalLbels\...		[1.1403e+0...]									
3	'E:\finalLbels\...		[1.1404e+0...]									
4	'E:\finalLbels\...		[1.1400e+0...]									
5	'E:\finalLbels\...		[1.1406e+0...]									
6	'E:\finalLbels\...		[1.1406e+0...]									
7	'E:\finalLbels\...		[1.1414e+0...]									
8	'E:\finalLbels\...		[1.1423e+0...]									
9	'E:\finalLbels\...		[1.1440e+0...]									
10	'E:\finalLbels\...		[1.1442e+0...]									
11	'E:\finalLbels\...		[1.1465e+0...]									
12	'E:\finalLbels\...		[1.1491e+0...]									
13	'E:\finalLbels\...		[1.1520e+0...]									
14	'E:\finalLbels\...		[1.1562e+0...]									
15	'E:\finalLbels\...		[1.1598e+0...]									
16	'E:\finalLbels\...		[1.1645e+0...]									
17	'E:\finalLbels\...		[1.1713e+0...]									
18	'E:\finalLbels\...		[1.1777e+0...]									

The labels are exported to the work space using the Ground Truth Labelling Application of MATLAB and then it is converted into Training Data on which the Final Neural Network is trained

Training the Neural Network In MATLAB R2017b



Command Window

New to MATLAB? See resources for [Getting Started](#).

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Mini-batch Accuracy	Base Learning Rate
1	1	11.33	2.4277	14.84%	0.0010
4	50	157.01	0.0663	97.66%	0.0010
7	100	281.78	0.1523	96.09%	0.0010
10	150	406.30	0.0508	99.21%	0.0010
14	200	530.77	0.0280	98.44%	0.0010
17	250	655.85	0.0208	99.22%	0.0010
20	300	780.30	0.0066	100.00%	0.0010
24	350	904.53	0.0004	100.00%	0.0010

Biographical Information

Sachit Kaul currently lives in Arlington, TX. He is originally from Jammu Kashmir, India. He has earned his B.E. in Mechanical Engineering from Dr.B.A.Marathawada University, Aurangabad, India before earning his Master's Degree (M.S.) in Mechanical Engineering from The University of Texas at Arlington in 2017. His Coursework is focused on Robotics, Control, Unmanned Vehicle Systems and Automotive Engineering.