

ROBUST VEHICLE DETECTION AND CLASSIFICATION FOR URBAN TRAFFIC ENVIRONMENT

by

Yumnah Hasan
Registration No: 39398

A thesis report submitted in partial fulfillment of the requirements for the degree of MS
Electrical Engineering

Examination Committee: Engr. Muhammad Umair Arif (Supervisor)

_____ (External)

Nationality: Pakistani

Previous Degree: Bachelor of Science in Electrical Engineering (Electronics)

FAST-NUCES University

Karachi, Pakistan

Bahria University Karachi Campus

13, National Stadium Road, Karachi

Pakistan

2017

Copyright @ 2017 Yumnah Hasan
all right reserved

**CERTIFICATE OF COMPLETION
OF THESIS WORK**

This is certified that **Yumnah Hasan** has successfully
completed her research thesis titled

“Robust Vehicle Detection and Classification for Urban Traffic Environment”

under my supervision.

The thesis meets the scholarly standards as set by

BAHRIA UNIVERSITY

Pakistan

Dated: _____

Research Supervisor (Signature)

Name: Engr. Muhammad Umair Arif

Affiliation: Bahria University Karachi Campus

Dept. of Electrical Engineering

CERTIFICATE OF ORIGINALITY

I certify that the intellectual contents of the thesis

“Robust Vehicle Detection and Classification for Urban Traffic Environment”

is the product of my own research work except, as cited properly and accurately in the acknowledgements and references, the material taken from any source such as research papers, research journals, books, internet, etc. solely to support, elaborate, compare and extend the earlier work. Further, this work has not been submitted previously for a degree at this or any other university.

The incorrectness of the above information, if approved at any stage, shall authorize the University to cancel my degree.

Signature: _____

Date: _____

Name of Research student: Yumnah Hasan

Declaration of Authentication

Title of report:

Robust Vehicle Detection and Classification for Urban Traffic Environment

Project supervisor's name:

Engr. Muhammad Umair Arif

We certify that the work given in this thesis is to the best of our awareness. All types of material and connections used and any help obtained in the construction of this thesis have been accepted by us. This thesis is less than 50,000 words in length without accommodating figures, tables, references and glossary. We have thus announced that nobody from us have submitted, either this whole material or a part of it, for any other degree at this material or any other institution.

Signature of Student

Acknowledgment

I wish with my deep heart to those people who put in valid contribution in to this thesis. I am first of all thankful to Allah almighty for giving me courage and strength to successfully complete the thesis. I am extremely thankful to my supervisor Engr. Muhammad Umair Arif for his guidance in resolving the problems throughout the thesis work. Without his help and guidelines, it was unmanageable to write this thesis. Dr. _____ is also acknowledged for managing time from her busy scheduled to become an external examination committee member. I am also very thankful to Dr. Haroon Rasheed to give me his precious time to review my thesis. I am also thankful to my parents, husband Mr. Taha Zafar, and brother Dr. Syed Raheel Hasan, who shown great commitment, sympathy and kindness for me during the entire thesis work completion.

Abstract

In computer vision there are practical applications available in the domain of traffic monitoring and surveillance. Detection and classification of vehicles play a significant role in in traffic monitoring. In the detection and classification phase there are different challenges present which have immense impact on the accomplishment of these tasks which includes the variant attributes such as illumination, camera viewing angle, shape, cluttered background, color, speed of vehicle, size, pose, shadows, environmental conditions, and occlusion. In the detection part, native complex datasets namely TOLL PLAZA and NIPA with different urban complex attributes are utilized for the implementation of this research analysis. Total 1516 vehicles are assed in NIPA dataset whereas there 376 vehicles are present in the TOLL PLAZA dataset. The analysis of insight performance is done on the comparative study of two different techniques namely Haar cascade classifier using cascade of boosted classifier and Blob analysis. The advantages of Haar classifier includes effective extraction of discernible regions of interest by using haar features in urban complex traffic scenes. The results of detection showed 83.7% and 88.3% accuracy for NIPA and TOLL PLAZA datasets by using Haar cascade classifiers respectively. In comparison with blob analysis produced 43.8% and 65.7% true detection accuracy for NIPA and TOLL PLAZA datasets respectively. For vehicle classification techniques of Neural Networks (NN) and Deep Neural Networks (DNN) is performed on three complex urban datasets. Scale Conjugate Gradient Backpropagation (SCG) is used for classical approach while Inception V3 and MobileNet model is used for DNN implementation. Total 6 categories of vehicles are assed in this study namely Bike, Bus, Rickshaw, Sedan, Van and Hatchback. The combine training stage for each algorithm utilizes 2400 samples and 12 samples of each dataset are used for testing. The results acquired after the processing reveals that the MobileNet model has the highest accuracy in true classifications among the other two. SCG has the lowest precision while Inception V3 model also has good classifications rate. There are some false and misclassifications observed between Hatchback, Sedan and Van due to their similar structure.

Table of Contents

Acknowledgement	vi
Abstract	vii
Table of Contents	viii
List of Figures	x
List of Tables	xii
Thesis Outline	xiii
1 Introduction.....	1
1.1 Problem Statement.....	1
1.2 Objectives	1
1.3 Challenges	2
1.4 Motivation	2
1.5 Contribution	2
2 Literature Review	3
2.1 Object Detection by using Blob Analysis	5
2.1.1 Introduction to Blob Analysis	5
2.1.2 Concept of Region and Blob	5
2.1.3 Fundamental steps of Blob Analysis	6
2.1.3.1 Extraction	7
2.1.3.2 Refinement	7
2.1.3.3 Analysis	8
2.2 Object Detection by using Haar Cascade Classifier	8
2.2.1 Haar-Like Feature	8
2.2.2 Cascade Classifier	8
2.3 Object Classification by using Neural Networks.....	10
2.3.1 Neural Networks	10

2.4 Object Classification by using Deep Neural Networks.....	11
2.4.1 Deep Neural Networks.....	11
2.4.2 TensorFlow	13
2.4.3 Digit	13
2.3.4 Caffe	14
2.3.5 Torch	14
2.3.6 Maxnet	14
2.3.7 Darknet	14
3 Methodology	15
3.1 System Settings	15
3.2 Vehicle Detection on Urban Datasets	15
3.2.1 Blob Analysis	15
3.2.2 Haar Cascade Classifier	18
3.3 Vehicle Classification on Urban Datasets	23
3.3.1 Urban Datasets	23
3.3.2 Classical Neural Networks Approach	26
3.3.2.1 Scale Conjugate Gradient Method	27
3.3.3 Deep Neural Networks Approach	29
3.3.3.1 Inception V3 Model.....	30
3.3.3.2 MobileNet Model	31
4 Results and Discussion	37
4.1 Outcomes of Vehicle Detection	37
4.2 Result Validation	41
4.3 Outcomes of Vehicle Classification	45
5 Conclusion and Future Work	65
5.1 Conclusion	65
5.2 Future Work	65
References.....	66
Appendix A (Milestone Achieved).....	69

List of Figures

Figure 2.1: Block Diagram of Blob Analysis	5
Figure 2.2: Detailed Block Diagram of Blob Analysis	6
Figure 2.3: Block Diagram of Haar Cascade Classifier	9
Figure 2.4: The Basic Structure of Neural Networks.....	10
Figure 2.5: A Single Node Example of Neural Networks.....	11
Figure 2.6: CNN Basic Structure.....	12
Figure 2.7: Example of Data Flow Graph $c = a*b$	13
Figure 3.1: Work Flow Block Diagram of Blob Analysis	16
Figure 3.2: Positive samples for Training a Cascade Classifier [NIPA Dataset]	19
Figure 3.3: Positive samples for Training a Cascade Classifier [TOLL PLAZA Dataset]	19
Figure 3.4: Negative Samples for Training a Cascade Classifier [NIPA Dataset]	21
Figure 3.5: Negative Samples for Training a Cascade Classifier [TOLL PLAZA Dataset].....	21
Figure 3.6: Cascade Classifier Consists of N Stages	22
Figure 3.7: Work Flow Block Diagram of Haar Cascade Classifier	23
Figure 3.8: Confusion Matix of Training Stage	28
Figure 3.9: Training, Validation, Test and all ROC curve.....	29
Figure 3.10: Schematic Diagram of Inception V3	30
Figure 3.11: Basic Structure of Inception V3 Model	31
Figure 3.12: Standard Convolution, Depthwise Convolution and Pointwise Convolution.....	32
Figure 3.13: Image from Video captured at University Road.....	32

Figure 3.14: Testing Samples of Urban Dataset I	33
Figure 3.15: Testing Samples of Urban Dataset I I.....	33
Figure 3.16: Testing Samples of Urban Dataset III	34
Figure 3.17: Training Samples of Urban Dataset I	34
Figure 3.18: Training Samples of Urban Dataset II	35
Figure 3.19: Training Samples of Urban Dataset III	35
Figure 3.20: Extraction of Surf Features	36
Figure 4.1: Detection results obtained from Blob Analysis [NIPA Dataset]. Lanes are labeled as 1, 2 and 3.....	39
Figure 4.2: Detection results obtained from Haar Cascade Classifier [NIPA Dataset]. Lanes are labeled as 1, 2 and 3.....	39
Figure 4.3: Detection results obtained from Blob Analysis [TOLL PLAZA Dataset]. Lanes are labeled as 1 and 2.....	40
Figure 4.4: Detection results obtained from Haar Cascade Classifier [TOLL PLAZA Dataset]. Lanes are labeled as 1 and 2	40
Figure 4.5: ROC Curve for NIPA Dataset (a) Lane 1 (b) Lane 2 (c) Lane 3.....	43
Figure 4.6: ROC Curve for TOLL PLAZA Dataset (a) Lane 1 (b) Lane 2.....	44
Figure 4.7: Overall efficiency of compared Algorithms for NIPA Dataset.....	44
Figure 4.8: Overall efficiency of compared Algorithms for TOLL PLAZA Dataset.....	45
Figure 4.9: Classification Result obtained from Inception V3 model of Bike for Urban Dataset I.....	46
Figure 4.10: Classification Result obtained from Inception V3 model of Hatchback for Urban Dataset I	47
Figure 4.11: Classification Result obtained from Inception V3 model of Sedan for Urban Dataset I	47
Figure 4.12: Classification Result obtained from Inception V3 model of Hatchback for Urban Dataset I	48
Figure 4.13: Classification Result obtained from Inception V3 model of Van for Urban Dataset I.....	48
Figure 4.14: Classification Result obtained from Inception V3 model of Bike for Urban Dataset II	49

Figure 4.15: Classification Result obtained from Inception V3 model of Bus for Urban Dataset II	49
Figure 4.16: Classification Result obtained from Inception V3 model of Hatchback for Urban Dataset II	50
Figure 4.17: Classification Result obtained from Inception V3 model of Rickshaw for Urban Dataset II	50
Figure 4.18: Classification Result obtained from Inception V3 model of Sedan for Urban Dataset II	51
Figure 4.19: Classification Result obtained from Inception V3 model of Van for Urban Dataset II	51
Figure 4.20: Classification Result obtained from Inception V3 model of Bike for Urban Dataset III	52
Figure 4.21: Classification Result obtained from Inception V3 model of Bus for Urban Dataset III	52
Figure 4.22: Classification Result obtained from Inception V3 model of Hatchback for Urban Dataset III	53
Figure 4.23: Classification Result obtained from Inception V3 model of Rickshaw for Urban Dataset III	53
Figure 4.24: Classification Result obtained from Inception V3 model of Sedan for Urban Dataset III	54
Figure 4.25: Classification Result obtained from MobileNet model of Bike for Urban Dataset I.....	55
Figure 4.26: Classification Result obtained from MobileNet model of Hatchback for Urban Dataset I.....	55
Figure 4.27: Classification Result obtained from MobileNet model of Rickshaw for Urban Dataset I.....	56
Figure 4.28: Classification Result obtained from MobileNet model of Sedan for Urban Dataset I.....	56
Figure 4.29: Classification Result obtained from MobileNet model of Van for Urban Dataset I.....	57
Figure 4.30: Classification Result obtained from MobileNet model of Bike for Urban Dataset II.....	57

Figure 4.31: Classification Result obtained from MobileNet model of Hatchback for Urban Dataset II.....	58
Figure 4.32: Classification Result obtained from MobileNet model of Rickshaw for Urban Dataset II.....	58
Figure 4.33: Classification Result obtained from MobileNet model of Sedan for Urban Dataset II.....	59
Figure 4.34: Classification Result obtained from MobileNet model of Van for Urban Dataset II.....	59
Figure 4.35: Classification Result obtained from MobileNet model of Bike for Urban Dataset III.....	60
Figure 4.36: Classification Result obtained from MobileNet model of Hatchback for Urban Dataset III.....	60
Figure 4.37: Classification Result obtained from MobileNet model of Rickshaw for Urban Dataset III.....	61
Figure 4.38: Classification Result obtained from MobileNet model of Sedan for Urban Dataset III.....	61
Figure 4.39: Classification Result obtained from MobileNet model of Van for Urban Dataset III.....	62
Figure 4.40: Classification Result obtained from MobileNet model of Bike for Urban Dataset III.....	62
Figure 4.41: Classification Result obtained from MobileNet model of Bus for Urban Dataset III.....	63

List of Tables

Table 3.1: Highlights of NIPA Dataset.....	17
Table 3.2: Highlights of TOLL PLAZA Dataset.....	20
Table 3.3: Samples Of Used Features Prototypes	20
Table 3.4: Highlights of Urban Dataset I	24
Table 3.5: Highlights of Urban Dataset II	25
Table 3.6: Highlights of Urban Dataset III	26
Table 4.1: Ground Truth-NIPA Dataset.....	37
Table 4.2: Qualitative Results Obtained For Blob Analysis [NIPA Dataset].....	37
Table 4.3: Qualitative Results Obtained For Blob Analysis [TOLL PLAZA Dataset].....	38
Table 4.4: Qualitative Results Obtained For Cascade Classifier # 1 [NIPA Dataset].....	38
Table 4.5: Qualitative Results Obtained For Cascade Classifier # 2 [NIPA Dataset].....	38
Table 4.6: Ground Truth-TOLL PLAZA Dataset.....	38
Table 4.7: Qualitative Results Obtained For Cascade Classifier [TOLL PLAZA Dataset]	38
Table 4.9: Vehicle Classification Results for Urban Dataset I	63
Table 4.10: Vehicle Classification Results for Urban Dataset II	64
Table 4.11: Vehicle Classification Results for Urban Dataset III	64

Thesis Outline

In the first chapter we introduce all parts of thesis module and describe problem statement, motivation, and contribution in the thesis. The second chapter reviews the literature of the complete system. The third chapter includes the methodology of thesis and describes the algorithms for detection as well as classification of vehicles and their simulation results. The chapter four describes the results and classifies the best detection and classification technique for vehicle in urban complex environment. In chapter five the conclusion and future work are presented.

Chapter 1

1 Introduction

Intelligent Transportation System (ITS) utilizes vision based techniques in this modern era. The methods are more acceptable in the advance world due to the number of advantages which include non-disturbing nature, development of flexible algorithms on continuous basis, easy installation setup, more cost effective, low maintenance and huge information accessibility for analysis. The basic aim in road monitoring and surveillance identification of different region of interest areas in the traffic environment.

Traffic based exposure data is important for designing highway infrastructure management system, safety analysis, resource allocation and other applications. Exposure data not only contain vehicle detection per unit time but also the classification according to its volume into different vehicle classes. In previous research studies there are many techniques proposed for the detection and classification of vehicle. However, still more research is required in the identification of robust and classical approaches inorder to enhance the true detection and classification rate in variant weather conditions and complex traffic environments. The complexity further increases in the detection and classification of vehicle because of the variant attributes like orientation, color, size, occlusion, background clutter, shape, shadows, speed of vehicles and camera viewing angle.

In this thesis we have performed detection and classification of vehicles in urban complex environment. The datasets used in this thesis are unique with varying complexity.

1.1 Problem Statement

- Robust detection and classifications research is being carried out for varying complexity of data sets. There is still room for enhancements for new datasets.
- Comparison of latest techniques is still not reported.

1.2 Objectives

Based on the problem found in existence study, the main objectives are:

- Study vehicle detection and classification techniques in detail.
- Implement some techniques on different urban complex datasets Like NIPA, TOOL PLAZA and URBAN DATASETS and perform their comparison.
- Enhance best selected technique for better results for local complex environments. To check varietal differential/nation variation.

1.3 Challenges

There are many challenges in real time vehicle detection and classification for urban complex environment.

- Weather conditions.
- Poor road illuminations.
- Occlusion from other vehicles.
- High speed of vehicles.
- Shadow occurrence.
- Camera viewing angle and position.

1.4 Motivation

- Vehicle detection and classification plays an important role for reducing traffic congestion on highways and urban environments.
- This also helps in reducing road accidents, Identifying suspicious vehicles for security purposes, Identifying illegal heavy traffic on urban roads and adjusting traffic signal timings.
- Futuristic planning like building new roads, bridges, underpasses or subway systems are also done on the basis of vehicle detection, classification and counting.
- Vision based sensors are less costly and an extremely effective solution for vehicle detection.

1.5 Contribution

- Existing computer vision techniques for detection and classification of vehicles are implemented and their comparison are done for novel traffic datasets.

Chapter 2

2 Literature Review

The authors (P.Viola et al., **2001**) introduced a basic algorithm for face detection. Although later, this can be further extended and applied on different objects. The approach is considered as a classical and competitive real-time object detection model. The authors evaluate rectangular features named as Haar like by using an integral of image. When the training phase completed, faces are differentiated from non-facial images by using cascade classifiers. This framework provided us the baseline of our proposed research model for detection.

Basically, two types of vehicle detection approaches are discussed by authors (X. Zhuang et al., **2016**) which include appearance based detection and motion based detection. A motion of differential foreground from background is identified by using a motion based approach in paper (A. Broggi et al., **2003**). An adaptive model for the extraction of foreground is used but the identification of pedestrian and vehicle is not upto the mark. In research paper, author (A. Jazayeri et al., **2010**) utilized a technique known as Markov Model in order to separate the background model and moving vehicle in a video sequence. The modeling of a moving vehicle image is done on the basis of horizontal position of joint distribution, displacement per unit time and characteristics of scene. Authors in paper (J. P. Jodoin et al., **2014**), proposed a tracker that possess a capacity to handle multiple objects of different sizes and shapes. Object detection for moving objects is done by using background subtraction method. A state machine and an interface blob is designed for the handling for segmentation and occlusion. The rate of detection of motion based techniques are comparatively high because no complex computation is required for the detection of motion. On the other side, the error margin of false detection is usually high in these algorithms due to increased sensitivity in noise and they do not consider any other characteristic of vehicle.

The level of complexity is higher in appearance based method as compare to the motion based method. In these methods the characteristics of scenes possess low dependencies. A group of IBM Research department in paper, implemented a detection system based on multi-view of motion-let classifier on urban traffic including congested scenes (S. Pankati et al., **2011**). A motion-let algorithm is proposed by the same group (R.Feris et al., **2011**) where clustering is done on the basis motion-let approach by using large scale feature extraction on crowded and occluded scenes. The false detection is reported in this paper because of the less number of negative images present in their dataset. Another author in paper (N.C Mithun et al., **2012**), implemented method based on time-spatial images (TSIs) for object detection and classification. Each image is acquired from a virtual detection line present on the frames of a video. The accuracy of true detection is increased by using TSIs. The identification of occluded vehicles is also done and pixel dependencies are decreased between the still and moving objects. The researchers (Y.Chen et al., **2014**), proposed a novel Bayesian fusion algorithm which is consist of Gaussian mixture model for urban complex traffic environment. The method is quite computationally expensive and has low efficiency for occluded vehicles. The state of the art technique is proposed in paper (R. Feris et al., **2011**), where

the detection is done on the basis of automatic extracted appearance information. The variant conditions are presented inside the implemented dataset for this algorithm. A separate detector is trained for each set. The dataset consists of two types which are heavily crowded scenes and low crowded scenes. The occlusion is reduced by training a detector on the basis of many occluded images. Poison image reconstruction through gradient is used for occlusion handling. The accuracy of results are further increased by introducing parallel feature selection over multiple planes which include grey scale, blue, green and red channels with classical adaboost detection. The results showed higher accuracy in true detection on the congested scenes.

For the classification of vehicle four different types of cars are studied in this paper (H. Huttunen et al., **2016**) which includes Bus, Truck, Van and small car. Two techniques are considered in this paper that are deep neural network and SVM (support vector machine using SIFT). The results show 97% accuracy. Previously research is divided into two types that are manufacturer type and vehicle type. Now in present days vehicle type is gaining more attention as compare to vehicle manufacturer type. Deep neural network and a support vector machine with scale invariant feature transform (SIFT) image feature are implemented. DNN has 40% fewer error than SVM. In paper, (Qiling et al., **2015**) deep neural network is implemented for vehicle detection on satellite images. A DNN is trained for the classification of vehicle or non-vehicle. Image path is extracted by using graph based super pixel segmentation. The proposed method has first step training of DNN networks by using labeled data. Then images are converted into patches set by using super pixel for new satellite image. Geographical Information System (GIS) is used to extract road segments. Detection is for only vehicles on road. Results shows DNN has excellent performance as compare to alternative approaches. Authors (Z. Dong et al., **2014**) proposed a method which is feature based vehicle type classification is implemented. Different classes of vehicles are identified by applying convolutional neural network. The results proved that proposed method has very effective vehicle classification findings. In paper, (Zhang et al., **2013**) authors have developed a frame work in which classification is done by using the vector quantization representation's reconstruction error. The results have 92% accuracy. Author (F.M de S. Mastro et al., **2013**) in paper proposed neural networks used for car type classification. Network input consists of different images. Features like width, height, and perimeter. Two neural networks are used for classification. 69% accuracy is reported for 100 vehicles. Another author (M. Kafai et al., **2012**) proposed a technique based on Bayesian network for vehicle type classification. The classification accuracy is 95.7% of 177 vehicle based dataset. To enhance the accuracy a new logic for ILD using back propagation neural network is introduced in this paper (Yong-Kul Ki et al., **2006**). Five classified vehicles are considered as output. The detection rate is achieved from the results is 91.5 %. In this paper, (Wei Wu et al., **2002**) two different techniques parameterized model and neural networks based solutions are provided for vehicle classification. The results reveal more than 91% accuracy of vehicle detection achieved from parameterized model technique.

Caffe has direct access to deep neural network. It is different from other CNN models because it is completely written in C++ and reference model are given. Caffe stores data in the form of blobs and it is of 4 dimensional arrays (Jia, Yangqing, et al., **2014**). In this paper author proposed a fusion method that integrates two and three deep convolutional neural networks (CNN). The results proved that fusion method of two deep CNN has better accuracy than individual model around

80.3% and also three deep CNN has more accuracy than two deep CNN techniques which is around 81.146% (Lavinia et al., 2016). In paper, (Kadrev et al., 2016) authors have compare three different techniques which is recognized by CNN. KNN training is more time consuming process, so it has greater efficiency. Authors in paper (Zhou et al., 2016) have used feature extraction and fixed tuning. Fine tuning also produces good results on classification of normal images. Fine tune Alexnet model has poor result on dark images on late fusion on dark and transformed images the accuracy of result is about 85%. In this paper, (Redmon et al., 2016) an algorithm based on joint training is implemented which is termed as YOLOv2 and YOLO9000. In this paper, (Culibrk et al., 2006) a novel approach of neural network is introduced for motion object segmentation in a video sequence by using background modeling. Promising results are obtained when implemented on hardware FPGA for execution. In this paper, (Xie et al., 2016) they have performed image based detection of vehicle. For this purpose they have created DNN implementations for discriminating different types of vehicles bus, truck, van and car automatically. Experimental results proved that LIDE-C optimized technique has promising results than the others. In this paper, (Luckow et al., 2016) they have developed a dataset which automatically allows learning and detecting different features of vehicles. Tensor flow has best accuracy 94% with all of its version. In this paper, (LeCun et al, 2004) they have proposed a solution of objection recognition problem in different pose, light and surrounding environment. Error rate of unseen objects in a uniformed background is around 13% for SVM and 7 % for convolutional Networks.

2.1 Object Detection by using Blob Analysis

2.1.1 Introduction to Blob Analysis

Blob analysis is a basic method of computer vision based technique for the analysis of consistent image regions. In this method the objects being investigated are clearly visible from the background. Different set of Blob Analysis techniques provides an opportunity to acquire desired results of vast range of visual inspection problems. The technique has some major advantages which includes high flexibility and good performance. But there are some limitations as well which includes relation requirement of clear background-foreground and pixel precision.

2.1.2 Concept of Region and Blob

- Region: It is defined as the image pixels subsets.
- Blob: It is described as linked region. Single spilt region into blobs filter is used to obtain from different region.

2.1.3 Fundamental steps of Blob Analysis

The fundamentals steps of Blob Analysis are presented in figure 2.1 and, the detailed block diagram of blob analysis is shown in figure 2.2.



Figure 2.1: Block Diagram of Blob Analysis

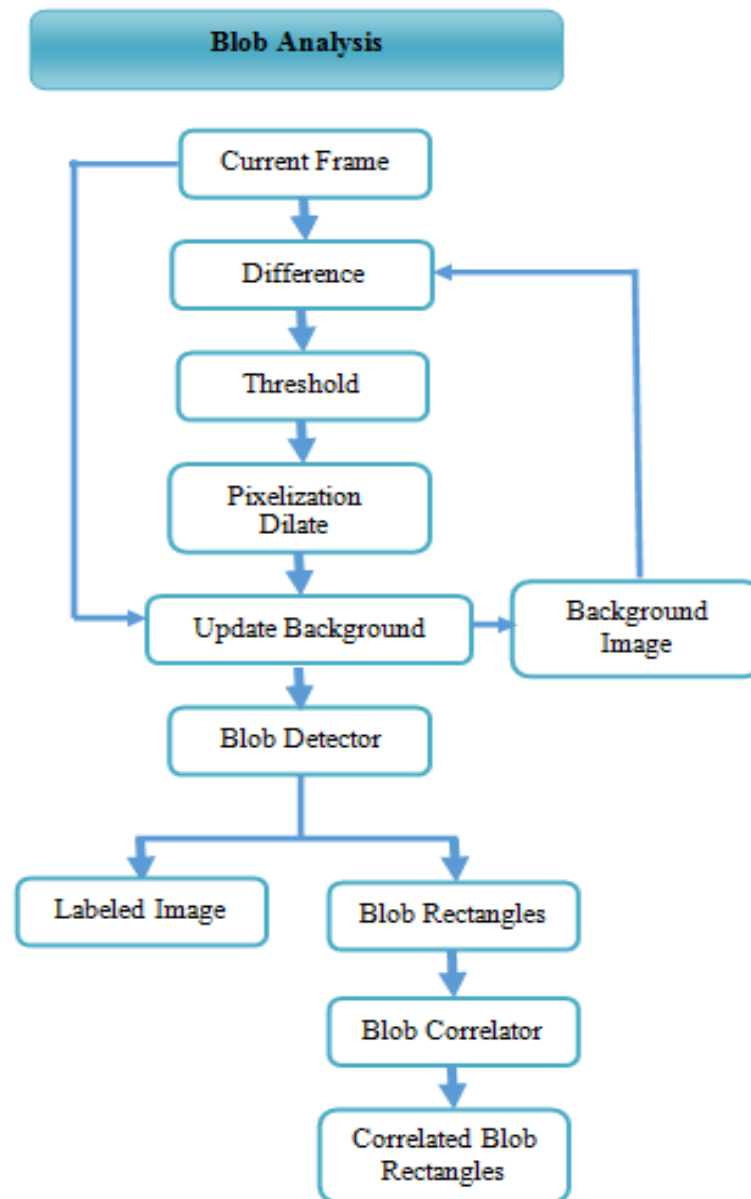


Figure 2.2: Detailed Block Diagram of Blob Analysis

1. **Extraction** – In the first step an image thresholding technique is applied to acquire a region that corresponds to the object which is to be inspected.
2. **Refinement** – The extracted region is consist of various kind of noises like different lighting conditions or poor image quality. In this step the region is improved by using region transformation techniques.
3. **Analysis** – Final results are obtain through this refined region. If there are multiple objects in the same region then it will be split into individual blob and examined separately.

2.1.3.1 Extraction

Two techniques are used to extract regions from an image:

- **Image Thresholding** – generally used techniques that evaluate a region as a set of pixels that satisfies particular condition dependent on the specific operator (e.g regions that are brighter than given value or brighter than the average value of brightness in their neighborhood). The final data is always single region may be consist of multiple objects.
- **Image Segmentation** – A set of blobs in this specialized methods are evaluated corresponding to the areas in the image that meet specific condition. The final resultant data is always in the form of an array of connected regions (blobs).

2.1.3.2 Refinement

- **Region Morphology:** For Region transformation a classic technique is known as region morphology. The main concept of this toolset is the utilization of a structuring element called as the kernel. The kernel is comparatively a small shape that is repeatedly centered at every pixel with dimensions of the region that is being transformed. Each pixel is either included to the resulting region or not, relying on operation-specific condition on the minimum number of kernel pixels that have to overlap with real input region pixels (in the given position of the kernel).
- **Dilation:** It is define from one of the basic morphological transformations. In this technique pixel **P** within the area of the region is being transformed and further added to the resulting region in the case if structuring element centered at P overlaps with at least one pixel that is from input region. Moreover, circular kernel such transformation is same as the uniform expansion of the region in all direction.
- **Erosion:** A dual operation dilation is erosion. In this technique, each pixel **P** within the dimensions of the region being transformed is added to the final result region in the case if structuring element centered at P is completely present in the region pixels. Moreover, circular kernel such transformation is same as the uniform reduction of the region in all direction.

- **Closing:** The real power of region morphology includes in the composite operators that is Closing and Opening. The cases like blind region expansion performed by dilation, the empty spaces in the transformed region are filled in. In the result, the expanded region no longer corresponds to the object being evaluated. However, erosion operator is applied to bring the expanded region back to its actual dimensions. The basic idea is that the gaps that are completely filled during dilation will stay filled after erosion as well. The technique in which erosion is applied to the resultant.
- **Opening:** It is defined as the dual operation of opening. In this method the region transformed is basically eroded and then dilated. The final results restores the form of initial region, with the exception of small areas that are eliminated during the process.

2.1.3.3 Analysis

Finally when the region is obtained that related to the object or the objects being evaluated are then analyzed for meaningful information. Image analysis includes processing an image into basic components so that statistical data is being extracted. Image analysis included different operations like finding shapes, detecting edges, eliminating noise, counting objects and measuring other image properties and characteristics of an object (Adaptive Vision, **2017**).

2.2 Object Detection by using Haar Cascade Classifier

2.2.1 Haar-Like Feature

A Haar like feature takes neighboring rectangular regions at a particular location in a detection window, adds the pixel intensities in every region and computes the difference between the sums. The resultant difference is then used to categorize subsections of an image. An example would be detection of human faces generally from the areas around the eyes that are usually darker than the areas on the cheeks (S. Soo, **2014**).

2.2.2 Cascade Classifier

Initially a classifier known as cascade if boosted classifier working with haar-like features is trained with some few hundred sample views of a specific object like face or car that are known as positive samples and are scaled to the same size 20x20. Furthermore, random images of the same size are known as negative samples.

When the classifier is trained, it can be implemented to a region of interest (of the same size as utilized while training) in an input image. The classifier results into a “1” i.e “T” if there an object present for example car or face, and “0” i.e “F” on the other case as shown in figure 2.3. In order to find the object in an image, a search window is used across the image and counter verify every location using the classifier. The resizing of classifier can also be done so that the objects of interest at different sizes can easily be find out, which is more flexible rather than resizing an image itself. Therefore, an object with unknown dimension in the image is being figure out by using the scan procedure which is to be performed several times at different scales.

Cascade term in classifier is to be used for the resultant classifier which includes several easy classifier or stages that are applied successively to a region of interest until at some stage the candidate is refused or passed all the stages. The positives samples used at each stage is calculated by using the below mentioned function in equation (2.1).

$$\text{number of positive samples} = \text{floor}(\text{totalPositiveSamples} / (1 + (\text{NumCascadeStages} - 1) * (1 - \text{TruePositiveRate}))) \quad (2.1)$$

Next word “boosted” means that the classifier at each stage of the cascade are complex themselves and they are built out of basic classifier using one of four different boosting methods. Recently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost are available. In basic classifier the decision tree classifier has at least two leaves. Input to the basic classifier is Haar like features and are calculated (OpenCV Documentation, **2017**).

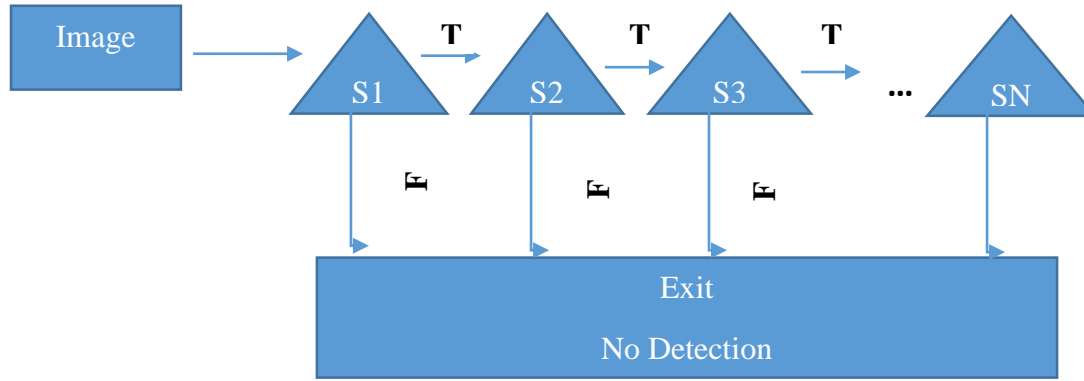


Figure 2.3: Block Diagram of Haar Cascade Classifier

A strong classifier is basically a linear combination of weighted simple weak classifiers in this algorithm as shown in equation (2.2).

$$h(x) = \text{sgn}\left(\sum_{j=1}^M \alpha_j h_j(x)\right) \quad (2.2)$$

Feature f_i is used as a feature function of weak classifier as shown in equation (2.3).

$$\begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases} \quad (2.3)$$

θ_j is the threshold value, and during the training polarity is determined $s_j \in \pm 1$, and also the coefficient α_j .

For input positive and negative training N images are labeled (x^i, y^i) . If i is the image face $y^i = 1$, if not $y^i = -1$.

In first step we have assigned a weight $w^i = \frac{1}{N}$ to every image i . For every feature f_i having $j = 1, \dots, M$. This is required to have sum one by normalizing the weights. Then feature application on each image is done during the training. Then optimal threshold and polarity θ_j, s_j that reduces the weighted classification error as shown in equation (2.4).

$$\theta_j, s_j = \arg \min_{\theta, s} \sum_{i=1}^N w_j^i \epsilon_j^i \text{ where } \epsilon_j^i = \begin{cases} 0 & \text{if } y^i = h_j(\mathbf{x}^i, \theta_j, s_j) \\ 1 & \text{otherwise} \end{cases} \quad (2.4)$$

Weight is assigned from α_j to h_j which is inversely related to the error rate. Therefore, best classifier can be created. For the next iteration weights w_{j+1}^i are reduced for the images i which are accurately classified. Now lastly the final classifier is set to as shown in equation (2.5) (Wikipedia, 2017).

$$h(x) = \text{sgn} \left(\sum_{j=1}^M \alpha_j h_j(x) \right) \quad (2.5)$$

2.3 Object Classification by using Neural Networks

2.3.1 Neural Networks

Neural networks (NN) based techniques are studied in this thesis. Basically, neural networks possess an ability to learn and adapt complex patterns. Artificial neural networks are based on the basic principle of human brain. They usually divided into two type feedforward and feedback networks. Previously, network connection are formed without loop concept but after this one or more loops formation during the network connection concept has formed. Nowadays, commonly used approach of feedforward networks include layered functions in which neurons are arranged on different layers having the same direction and connection related to each other. A neuron is define as an input vector $\{x_1, \dots, x_k\}$ to a scalar output y through a weight vector $\{w_1, \dots, w_k\}$ and a nonlinear function f as shown in equation (2.6).

$$y = f \left(\sum_{i=0}^k w_i x_i \right) = f(w^T x) \quad (2.6)$$

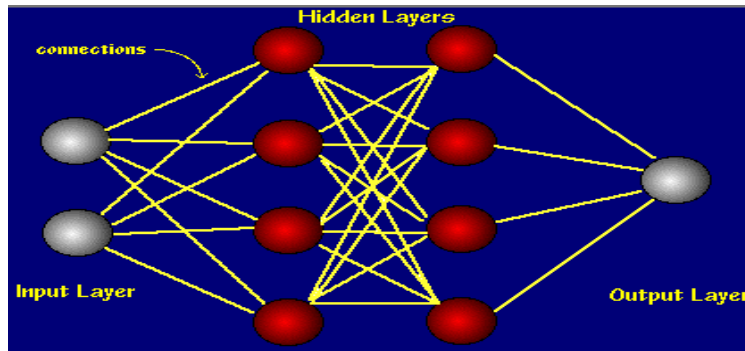


Figure 2.4: The basic structure of Neural Networks

In figure 2.4 the basic structure of Neural Networks is presented. Generally, the network is organized in the form of layers. These layers are made up of interconnected nodes which possess activation function. The patterns are described in the form of input layer which communicates with to one or more hidden layers where weighted connections are used for actual processing. These hidden layers are connected to output layer which presents the output.

There are different kinds of learning rule utilized by neural networks one of them is delta rule. The delta rule is commonly utilized by Backpropagational neural networks. The term backpropagation is define as the backwards propagation of error. In this rule supervised learning takes place with each cycle or epoch (a new input pattern is used to represent network each time) through a forward activation flow of outputs, and weighted adjustments of back propagation error. In simple words a pattern in neural networks initially makes a random guess. After that it will see how far the answer was from the real one and do suitable adjustment to its connection weights as shown in figure 2.5.

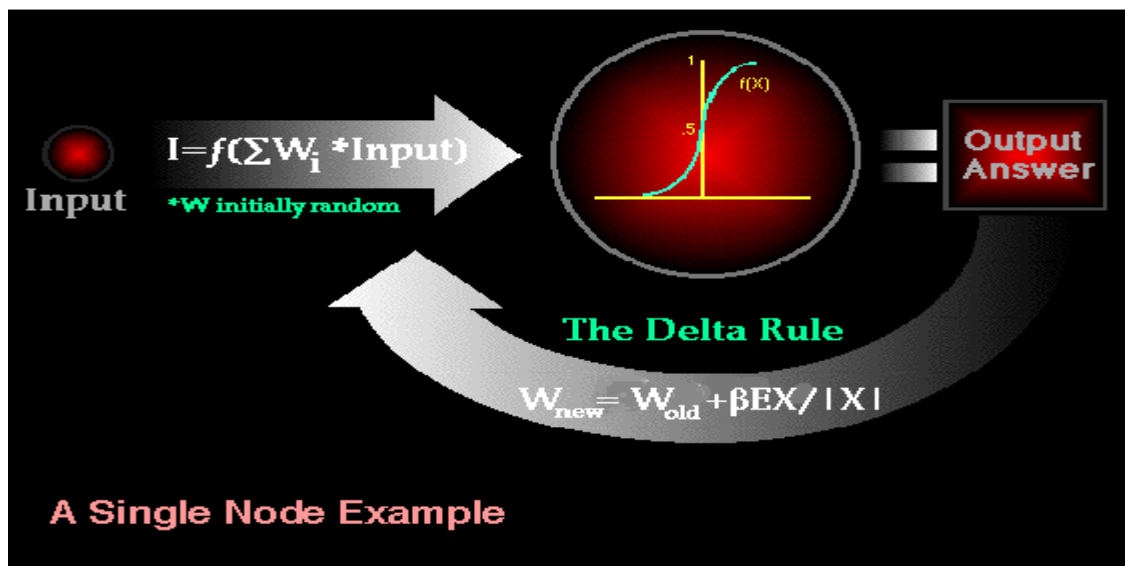


Figure 2.5: A single node example of Neural Networks

2.4 Object Classification by using Deep Neural Networks

2.4.1 Deep Neural Networks

Deep learning linked to neural networks having multiple hidden layers that can increasingly abstract information of the input data. Although the concept of deep neural networks are gaining more attention in the modern era. More complex features from the input images now can easily be learn sequentially through many hidden layers present in deep neural networks. The limitations of classic neural network approach includes higher training time, greater number of epochs required, hardly one or two hidden layer present for the computations. These shortcomings can be overcome by using deep neural networks. The main advantage of deep neural networks includes high accuracy in the results due to their computational architecture. The types of deep neural network is known as convolutional neural networks (CNN). The working principle of CNN is

same as NN described above in section 2.3 but the difference in CNN architecture is, it makes assumptions that the inputs are images which gives us option to encode certain properties into the architecture. This makes the forward function more efficient to implement. The basic structure of CNN is shown in figure 2.6.

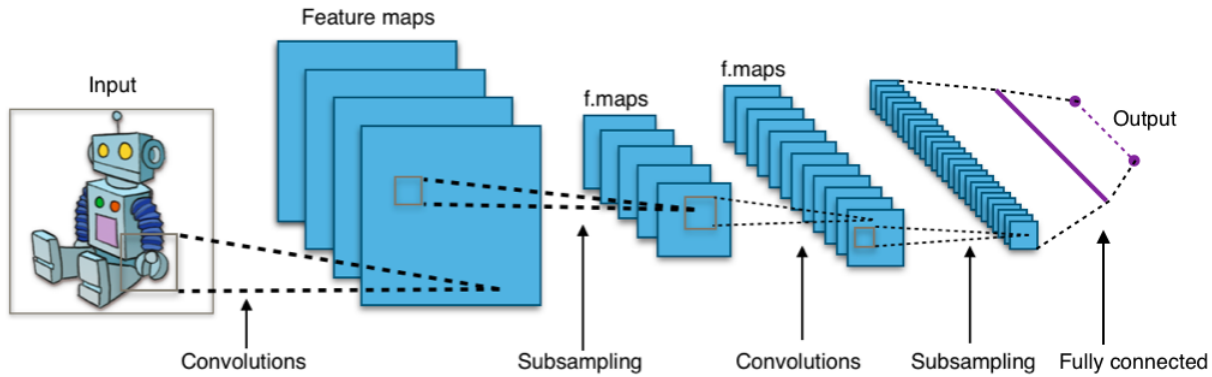


Figure 2.6: CNN basic structure

Convolution operation can be explained by using the example let suppose we are tracking a moving car by using a sensor $x(t)$ having a position at time t . Here “ x ” and “ t ” are real values but sensor is providing us noisy values. We would like to remove this noise by using an average of different obtained measurements. More recent values has more importance. Hence weighted average values by a weighted function $w(a)$ where “ a ” is the age. So if we proceed to the every value of “ t ” we will get “ s ” smoothing function which is shown in equation (2.6).

$$s(t) = \int x(a) w(t - a) da \quad (2.6)$$

The smoothing function can be expressed in the form of convolution operation and it can be represented in the form of asterisk as shown in equation (2.7).

$$s(t) = (x * w)(t) \quad (2.7)$$

Here “ x ” is referred as input and “ w ” is the kernel and the output is termed as feature map. In CNN first we have convolution layer where the convolution occurs. Then we have activation layers or also known as detection layer which is used to activate neuron same as in Artificial Neural Networks (ANN) which is followed by a pooling layer. Pooling is basically interchanges the outputs with the summarized statistics from the near outputs (I. G. Y. Bengio et al., **2016**). Max pooling is generally considered as the most commonly used method for this. In this method a maximum value is picked from the matrix or the portion of matrix. At the last dropout layer is present which is used to avoid overfitting of data. During training, every node is either dropout by a factor of $1 - p$ probability or kept based on probability p . However, p is set when we are creating CNN architecture. Hence a concise network is created which is more computationally fast and reduces the possibility of overfitting.

Previously different functions are presented in the form of mathematical expressions like deep learning algorithms and other architectural components which include initialization of a function, normalization methods and transformation. But nowadays computer based programs are largely implemented and preferable in this modern world. This need results into the growing trend of open source libraries and machine learning tools. There are different machine learning platform available few are discussed below.

2.4.2 Tensor Flow

Tensor Flow is a programming framework in which you speak to calculations as charts. Hubs in the chart are called operations (short for operations). An operation takes at least zero Tensors, plays out some calculation, and produces at least zero Tensors. A Tensor is a written multi-dimensional cluster. For instance, you can speak to a scaled down clump of pictures as a 4-D exhibit of skimming point numbers with measurements [batch, stature, width, channels] (Wikipedia, **2017**). The multidimensional arrays which are also known as tensors are defined by using vertices of graphs while the nodes represent numerical functions between them. For instance if two numbers “a” and “b” are multiplied and their result is “c”. Form the figure 2.7 it can be observed that a line from “a” and “b” to “c” represents the vertices and forming a node at “c” which denotes the operation that is to be performed on “a” and “b”. Generally, other libraries are implemented on single platform while Tensor Flow can execute on more than one device such as CPUs and GPUs. It also has CUDA additional option in case of processing is graphics based. It is designed by Google Brain’s second generation is released on November 9, 2015. It is available in Linux, MacOS X and Android and iOs. It is highly flexible and efficient in terms of performance as compare to other machine learning tools

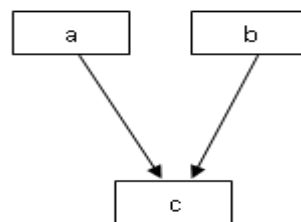


Figure 2.7: Example of Data Flow Graph $c = a * b$

2.4.3 Digit

Digit is an open source library used for reducing complexity in deep learning algorithms like organizing, learning, and constructing deep neural networks on different GPU-systems. It is a web based system designed for interactive problem solution platform. The researchers can easily design their DNN (deep neural networks) on the basis of their real time network properties. Other than debugging and programming it mainly focuses on the designing and training networks. This web application allows detecting, segmenting and classifying different objects by implementing your network. Hence, it is a server based system so sharing of techniques, results; datasets and their specifications are quite more easy and simple. It has reduced training time and high accuracy of results (NVIDIA Accelerated Computing, **2017**).

2.4.4 Caffe

An open source library named Caffe is developed by Berkeley Vision and Learning Center (BVLC) and community contributors. It was released under C++ Library BSD License with python interface. It is a tool for designing deep learning algorithms. Its main focus is on expression, speed and modularity. The key features of this library are flexible architecture which allows defining models by configuration irrespective of any hard code. It has also fabulous speed which makes it suitable for industrial purposes. It can process around 60M images in a day with a single GPU. Time required for inference is around 1 ms/image and for training its around 4 ms/image (Caffe, 2017).

2.4.5 Torch

Torch is an open source library for deep machine learning and it is based on Lua programming language. It uses core language LuaJIT, underlying C/CUDA implementation. It uses package where tensors and mathematical computations are defined. It supports different operations and basic functions like resizing, transposing, slicing, indexing, duplicating, sharing memory space and numerical operations. Its GPU is fast and has multiple port option available for iOS, android and FPGA. Its interface is quite simple, flexible, easy and fast in process designing and implementation complex neural network algorithms (Wikipedia, 2017).

2.4.6 Maxnet

A mesh network consists of different nodes each node is connected with one another including the network itself. A triggered signal is usually send to every other node so that each node competes with another. This network is basically a feedback network having single layer with n-neurons. It works for finding a maximum function. After each iteration the number of neurons comparatively decreases and will reach to unity in an activation mode. The highest output produces by the neuron is known as “winner”.

2.4.7 Darknet

Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install and supports CPU and GPU computation. Prior detection systems repurpose classifiers or localizers to perform detection. In this library a model is applied to an image at different locations and scales. High scoring regions or areas of the image are considered as detections.

Chapter 3

3 Methodology

3.1 System Settings

Two variant datasets are used for the implementation of vehicle detection algorithms which are titled as NIPA and TOLL PLAZA (M. U. Arif et al., **2016**). The details of each dataset is provided in table 3.1 and table 3.2. The road signs and markings are comparatively low in these datasets. The range of vehicles lie between motorbikes to large trucks.

There are three and two lanes present in NIPA and TOLL PLAZA dataset respectively. The vehicles are consist of different shapes and having different directions of flow and speed. For NIPA dataset, the camera viewing angle is providing less resolution and side view of the traffic environment. The cast shadow size is also small. The complexity increased due to the vehicle posture along with cluttering of background. For NIPA dataset total 1516 vehicles are evaluated and for TOLL PLAZA dataset 376 vehicles are used for the implementation of selected algorithms.

The system used for the testing of algorithms is Core i-5 2.30 GHz having 8 GB RAM platform. Training is done by using OpenCV library. Receiver Operating Characteristics Curves are plotted by using VLFeat. ROCs are basically used to demonstrate the performance of a binary classifier. Matlab® 2014 and Visual Studio 2013 are used as a programming terminal. The estimated time consumed during the training of a classifier is around 4-6 hours.

For vehicle classification task techniques were implemented on core i-3 2.10 GHz, 4 GB RAM platform. The training and testing of Neural Networks is done on Matlab® 2015 by using “Neural Networks Pattern Recognition Tool (nprtool)” which is mainly design for pattern recognition problems on 64 bit windows 7 operating system. Whereas, results of Deep Neural Networks are obtained by using TensorFlow platform on Ubuntu 16.04 operating system.

3.2 Vehicle Detection on Urban Datasets

3.2.1 Blob Analysis

In this technique detection of vehicles is performed by estimating the motion of each detected vehicle using blob analysis. An example for motion based object detection is available in MATLAB computer vision toolbox. An insight was adapted from it to understand the dynamics and build our own algorithm that can be tailored towards complex urban traffic datasets.

Blobs which are basically defined as the linked group of foreground pixels. These blobs are taken into consideration with the moving objects in this method. The state of the tracked vehicle is maintained by using an array tracks of moving vehicles. In a frame short lived tracks are created due to the noisy detections. To cater for this the algorithm initiates vehicle tracking only after it is followed for some number of frames. Similarly, when no detections are associated with a particular track for several consecutive frames, the algorithm assumes that the vehicle has left the field of view thus deleting the track. A track may also get deleted as noise if it was tracked for a short time and marked invisible for most of the frames. Binary motion segmentation is obtained by using foreground detector. Afterwards, noisy pixels are removed and holes are filled in the remaining

blobs by applying morphological operations on the resultant binary mask. Kalman filter is then used to predict the centroid of each track in the current frame and updates its bounding box accordingly. Kalman filter is basically used for detecting regions of object in the upcoming frames. Prediction and correction are the two basic steps of this method. This filter provides an optimal solution to the noisy data and reduces mean square errors like speed, velocity etc. Assigning object detections in the current frame to existing tracks is done by minimizing cost function. The cost function is defined as the negative log-likelihood of a detection corresponding to a track. The cost minimization involves following two steps:

Step 1: The cost of assigning all detections to each existing track is computed. This cost utilizes Euclidean distance between the centroid of the detection and the predicted centroid of the track. Furthermore, it contains Kalman filter confidence prediction.

Step 2: Assignment problem represented by the cost matrix and the cost of not assigning any detections to a track is solved. It uses the Munkres version of the Hungarian algorithm to compute an assignment which minimizes the total cost.

In the last part, the algorithm identifies assigned tracks and delete lost tracks. It also deletes recently created tracks that have been invisible for too many frames. The block diagram of blob analysis is shown in figure 3.1.

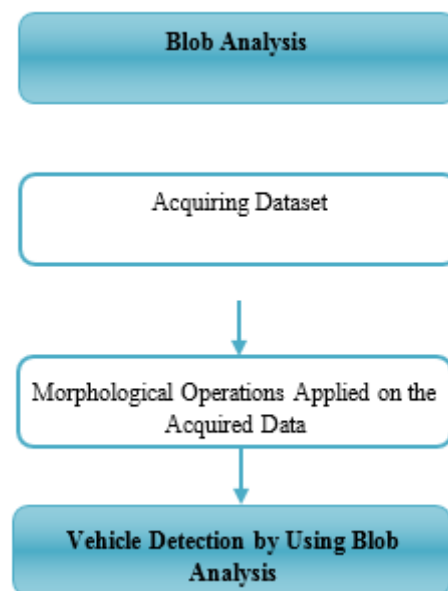



Figure 3.1: Work Flow Block diagram of Blob Analysis

TABLE 3.1: HIGHLIGHTS OF THE NIPA DATASET (M. U. Arif et al., 2016)

Video Frame	
Sequence Type	Outdoor
Video Length	00:08:16
Image Size	640 x 420
Shadow Strength	Low
Shadow Size	Small
Object Class	Vehicle
Object Size	Small
Object Speed (Pixels)	Slow/Fast
Noise Level	Medium
Camera position	Side view

3.2.2 Haar Cascade Classifier

For TOLL PLAZA one trained classifier is used while for NIPA two different trained classifiers are analyzed. The results reveals that classifier trained for TOLL PLAZA has good results which directly eliminates the need of second classifier. However, for NIPA dataset results showed that first classifier is producing less accurate results due to which another classifier is required to be trained for making the results accuracy more efficient. The positive and negative images are used for training the cascade function. The following paragraphs will provide the details related to the procedures and techniques implemented in training a Haar Cascade Classifier and it's testing on the variant datasets. For each datasets, a set of positive and negative samples are used for training a classifier. Negative images set are downloaded from GitHub (GitHub, **2016**). For each datasets total of 2000 negative images as shown in figure 3.4 and figure 3.5 are saved, these images are tabulated in a text file for path determination and image file extraction. Next, positive images for each datasets are taken by cropping a positive sample vector from the image sequence using object marker tool as shown in figure 3.2 and figure 3.3.

Viola and Jones introduced Haar cascade classifier which utilizes Haar like features (P. Viola et al., **2001**). Haar features prototypes (P. Menezes et al., **2004**) were used and are shown in table 3.3. These 14 features include 2 center-surround features, 4 edge features and 8 line features. The features are classified by its shape, position and the scale (which is not actually same scale that is used for detection stage although these two scales are multiplied) as shown in table 3.3 for particular classifier. Let assume a case where third line feature the output is calculated as the difference between the sum of image pixels under the rectangle including the entire feature (two white stripes and one black stripe in the center) and the sum of the image pixels under the black stripe multiplied by 3 so that the difference in the size of areas is to be compensated. Integral of images is used to rapidly calculate the sum of pixels over a rectangular regions (OpenCV Documentation, **2017**).

A complete and rich set of features is generated by scaling these prototypes independently in vertical and horizontal direction. The cascade function is trained from a set of positive and negative images. The ensuing paragraphs will provide the details regarding the procedures and techniques involved in training a Haar cascade classifier and it's testing on the datasets.

Haar cascade classifier utilizes two different trained classifiers for NIPA dataset. In the second part the TOLL PLAZA dataset uses one trained Haar cascade classifier. The trained cascade classifier for each dataset is unique on the basis of positive and negative samples and number of stages. The false positive rate can be decreased by increasing the number of stages. In the first case for NIPA dataset the cascade classifier # 1 has 168 positives, 500 negatives and 16-stages, whereas the cascade classifier # 2 has 580 positives, 1500 negatives and 16-stages. In the second case the cascade classifier has 1000 positives, 2000 negatives and 18-stages for the TOLL PLAZA dataset.

Basically, samples are taken for each classifier where number of positive and negative images are present. Random images are taken for the negative, which are not belong to the region of detection. However, the positive samples correspond to cropped template of objects of interest.

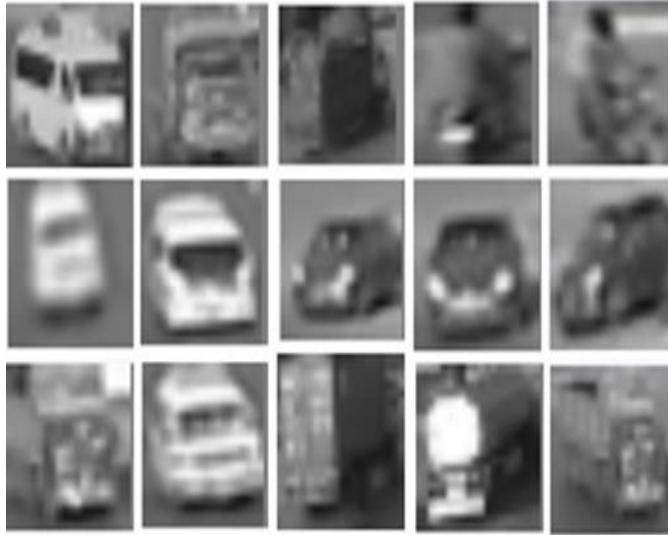


Figure 3.2: Positive samples for training a Cascade Classifier [NIPA Dataset].

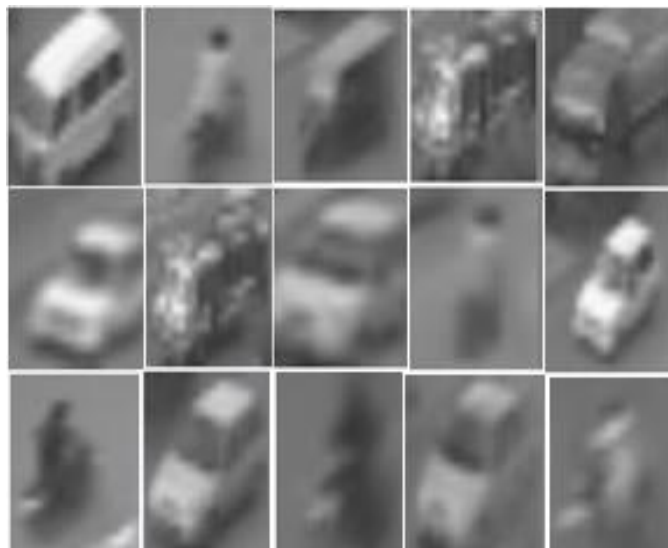


Figure 3.3: Positive samples for training a Cascade Classifier [Toll Plaza Dataset].

TABLE 3.2: HIGHLIGHTS OF THE TOLL PLAZA DATASET (M. U. Arif et al., 2016)


Video Frame	
Sequence Type	Outdoor
Video Length	00:20:00
Image Size	720x576
Shadow Strength	High
Shadow Size	Large
Object Class	Vehicle
Object Size	Medium
Object Speed (Pixels)	Slow
Noise Level	Medium
Camera position	Front view

TABLE 3.3: SAMPLES OF USED FEATURES PROTOTYPES (P. Menezes et al., 2004)



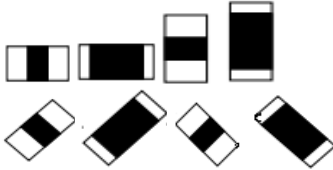
Edge Features	
Center-surround Features	
Line Features	



Figure 3.4: Negative Samples for Training a Cascade Classifier [NIPA Dataset].



Figure 3.5: Negative Samples for Training a Cascade Classifier [Toll Plaza Dataset].

Thereafter, a vector file is created by using positive and negative samples. The sample image size is important so as to define number of cascade classifier stages and to maintain appropriate resolution that in-turn will be used for feature extraction. The cascade classifier is based on stages as shown in figure 3.6.

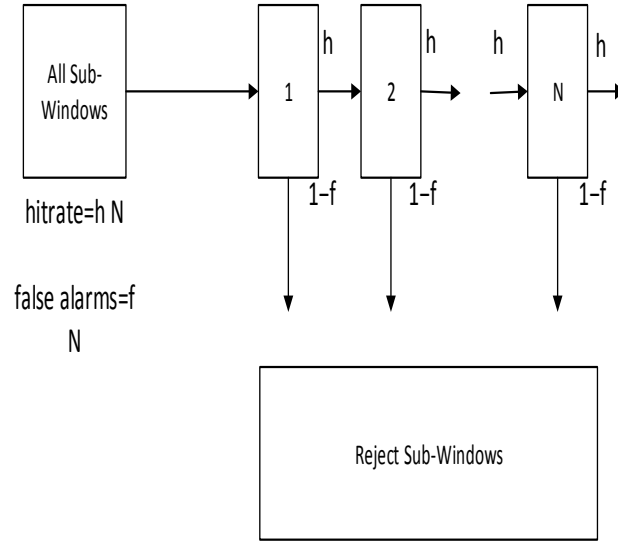


Figure 3.6: Cascade Classifier Consists of N Stages (P. Menezes et al. 2004).

To train each classifier applied technique is based on Boosting which utilizes weighted average of the decision made by decision stumps (one level decision tree). The present location of sliding window (which may be positive or negative) is determined by the labels present in each stage of classifier. Zero detection is indicated by a negative label indicates and then the classifier moves to the next label. Alternatively, positive label indicates object found and the detected region is fed to the next stage. The detector in the final stage classifies the object with positive label.

The numbers of stages for NIPA dataset are 16. Therefore, the false alarm rate for a cascade classifier having 16 stages is approximately $0.516 = 1.525 \times 10^{-5}$ and a hit rate of approximately 0.99916. The time required by a cascade classifier to train with 580 positives images, 1500 negatives images and 16 stages on Core i5 is around 4 to 5 hours. While training a cascade classifier for TOLL PLAZA dataset having 2000 negative images and 1000 positive images on the same system configurations required 6 to 7 hours. The false alarm rate is around $0.518 = 3.814 \times 10^{-6}$ and a hit rate of around 0.99918 having 18 stages in actual. Finally to train the classifier in both cases Haar training tool is executed. Finally, cascade classifier was converted into an XML file by using Haar converter tool in C++. Figure 3.7 shows the block diagram of stated methodology of haar cascade classifier.

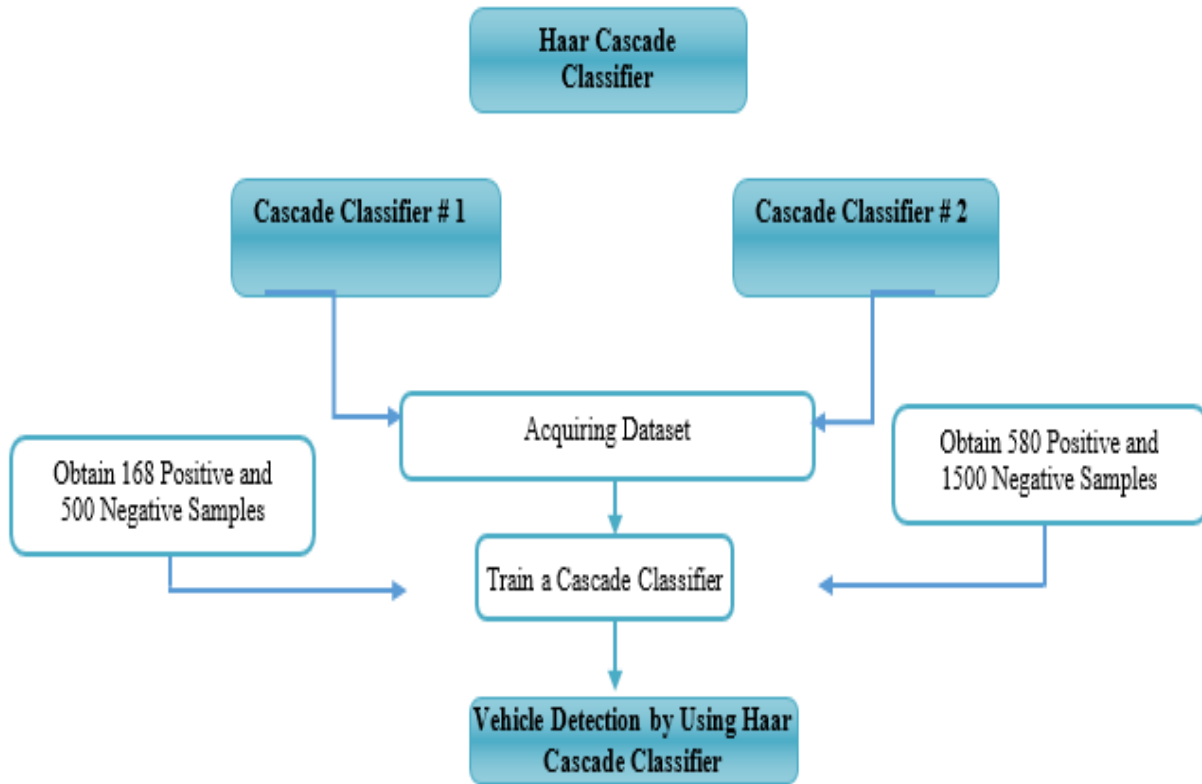


Figure 3.7: Work Flow Block Diagram of Haar Cascade Classifier

3.3 Vehicle Classification on Urban Datasets

3.3.1 Urban Datasets

Three different complex datasets are used in order to evaluate the methods for vehicle classification which are shown in table 3.4, 3.5 and 3.6. Each dataset has 6 different categories of vehicles present in it which are Bus, Motorbike, Rickshaw, Sedan, Van and Hatchback. In the first dataset which is shown in table 3.4 contains medium noise level with moderate object speed. The strength of shadow is also at average level but shadow size is usually large in length. In the second dataset which is shown in table 3.5 has high noise level but object speed is medium. The strength of shadow is medium and shadow size is small. In the last dataset which is shown in table 3.6 possess high noise level and medium object speed. The shadow strength and size is medium. Different attributes of each dataset makes it variant in complexity. The camera viewing angle of each video is from top view.

For training a classifier in each technique, 150 samples of Motorbike, Sedan, Rickshaw and Hatchback present in each dataset are used. However, 100 samples from each dataset of Van and Sedan are used. The samples used for training as shown in figure 3.17, 3.18 and 3.19. Total 2400 samples are used for training purpose. Similarly, for testing a classifier 12 samples from each datasets as shown in figure 3.14, 3.15 and 3.16 are used for all implemented algorithms. The image sample taken from video at university is shown in figure 3.13. The sequence of classes for training and testing are arranged in this order Bike is class 1, Hatchback is class 2, Rickshaw is class 3, Sedan is class 4, Van is class 5, and Bus is class 6.

TABLE 3.4: HIGHLIGHTS OF THE URBAN DATASET I



Video Frame	
Sample shadow of one vehicle	
Sequence Type	Outdoor
Video Length	00:00:57
Shadow Strength	Medium
Shadow Size	Large
Object Type	Sedan, Bike, Rickshaw, Bus, Van, Truck, SUV, Hatchback
Object Speed	Medium
Noise Level	Medium
Camera position	top view

TABLE 3.5: HIGHLIGHTS OF THE URBAN DATASET II





Video Frame	
Sample shadow of one vehicle	
Sequence Type	Outdoor
Video Length	00:01:00
Shadow Strength	Medium
Shadow Size	Small
Object Type	Sedan, Bike, Rickshaw, Bus, Van, Truck, SUV, Hatchback
Object Speed	Medium
Noise Level	High
Camera position	top view

TABLE 3.6: HIGHLIGHTS OF THE URBAN DATASET III

Video Frame	
Sample shadow of one vehicle	
Sequence Type	Outdoor
Video Length	00:01:00
Shadow Strength	Medium
Shadow Size	Medium
Object Type	Sedan, Bike, Rickshaw, Bus, Van, Truck, SUV, Hatchback
Object Speed	Medium
Noise Level	High
Camera position	top view

3.3.2 Classical Neural Networks Approach

In the classical Neural Networks technique six categories of vehicles are assed for classification. Each dataset contain training and testing samples of six classes namely Motorbike, Hatchback, Rickshaw, Sedan, Van, and Bus. In this algorithm feature extraction is done by using speeded up robust features (SURF) technique as shown in figure. 3.20. It is basically known as local feature detector and generally used for image recognition and classification applications. This technique uses integer approximation of Hessian blob detector for acquiring interest points in any image. The sum Haar wavelet response around the point of interest is used for feature descriptor. SURF was introduced by Herbert Bay (Herbert Bay et al., **2008**) and it is partially inspired from Scale Invariant Feature Transform (SIFT) descriptor (Wikipedia, **2017**). This technique is divided into two parts which are training a classifier and testing a classifier on real time data.

For the first part i.e training a classifier an input file of 50x2400 order is created where 50 represents number of features and 2400 are the number of samples. Furthermore, target file of order 6x2400 for the same input is created where 6 represents number of classes and 2400 are the number of samples. After importing these files into nprtool, training samples are randomly divided into three categories. 70% samples are used for training. 15% samples are used to validate that the network and the last 15% samples are used for completely independent test of network generalization. For training a Neural Networks Scaled Conjugate Gradient Backpropagation technique is used. The basic training of multilayers network includes division of data into three subsets. In the first subset, network weights and bias are updated and gradient is also calculated in order to obtain training set. The second subset is used for validation set and its error is continuously monitored during the training state. During the starting phase validation error reduces like the training set error. Although, when there is an over fitting of data begins, the validation error starts to increases. Generally, network weights and biases are stored at the lowest value of validation set error. For the comparison of different models test error is used but it does not play any role during the training process. If the test set error approaches a minimum at a different iteration number as compare to the validation set error, this would be the indication of a poor division of the dataset. The next step after division of samples is to set the number of neurons which is 25 in this training. Finally for training a classifier, the standard network is used. It is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer. When training is finished then we obtained confusion matrix of training stage which are shown in figure 3.8. The matrix is basically plotted between the predicted and actual class. The training outputs are quite accurate, the high numbers of correct responses are represented in the green squares and the low numbers of incorrect responses are shown in the red squares. Over all accuracy is of training is 98.0% and the error rate is 2.0%. The Receiver Operating Characteristic Curve (ROC) is a plot between true positive rate and false positive rate when the threshold is changed. In figure 3.9 the ROC curves we obtained for training state reveals that the points shown in the upper-left corner have high accuracy of sensitivity and specificity.

In the second part, for testing a classifier SURF algorithm is used for extraction of 50 features of each sample image. Test file is created of order 50x12 where 50 is the features and 12 are the number of samples. The resultant network is then tested the real time data by using the new input file which is our test file for evaluation of training.

3.3.2.1 Scale Conjugate Gradient Method

Scale Conjugate Gradient (SCG) is a supervised learning feed forward neural networks algorithm. It belongs to the class of Conjugate Gradient Method (CGM). The parameters of SCG are shown in equation (3.1).

$$s_k = \frac{E^j(w_k + \sigma_k \cdot p_k) - E^j(w_k)}{\sigma_k} + \lambda_k \cdot p_k \quad (3.1)$$

λ_k are calculated from their respective values at step $k - 1$, SCG mainly consists of two quantities namely, σ_k and λ_k . The values are not critical but should possess the condition σ_1 and λ_1 . As

$P = \exp(-\Delta E / T)$ be a vector from the space $T = T \cdot \text{deg}$, where N is define as the sum of number of weights and of the number of biases of the network. Suppose E be the error function we want to reduce. SCG can be differentiated from CGMs in two main points:

- Every iteration K of a CGM calculates w_i , where R^N a latest conjugate direction is, and $w_{k+1} = w_k + \alpha_k \cdot p_k$ is the dimension of the step in this direction. In real, p_k is a function of α_k , the error function in Hessian matrix which is the matrix of the second derivatives. As compare to the other CGMs which ignore the complex computations of the Hessian and approximate α_k with a time consuming line search process. SCG works on the simple approximation term $E^j(w_k)$, a plus point of the computation of α_k .
- s_k as the Hessian is not always accurate, which forbids the algorithm from attaining high performance. SCG utilizes a scalar α_k which is meant to be regulate the indefiniteness of the Hessian.

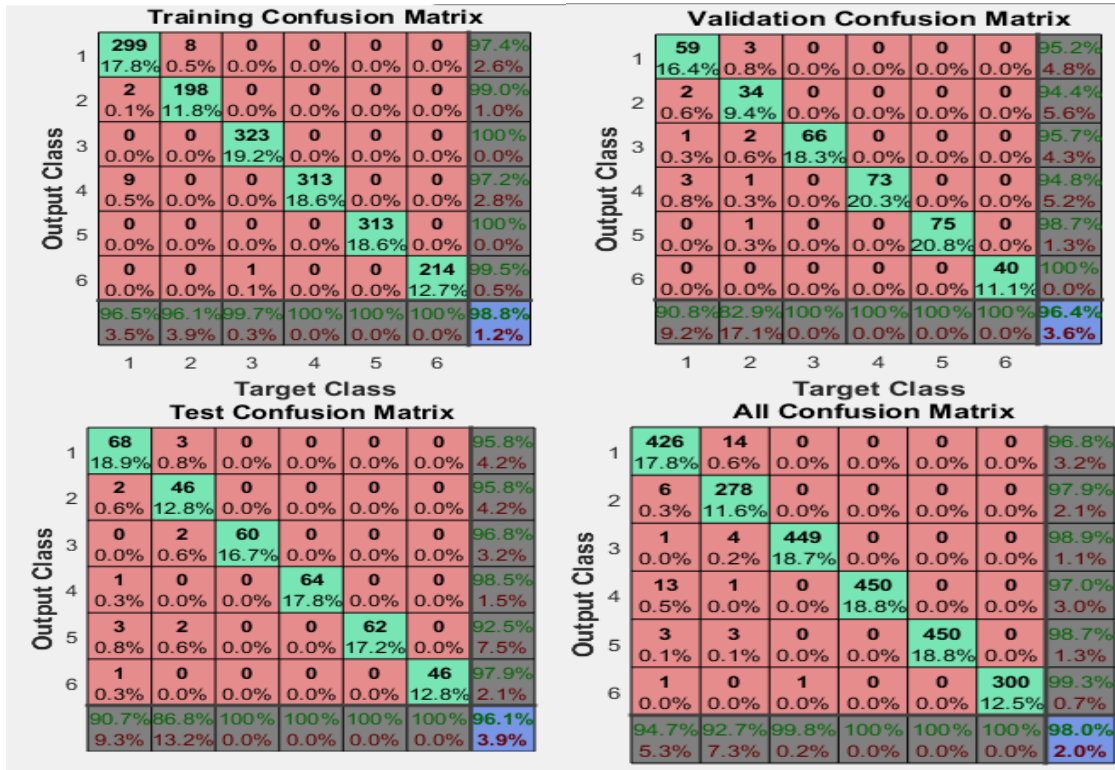


Figure 3.8: Confusion Matrix of training stage

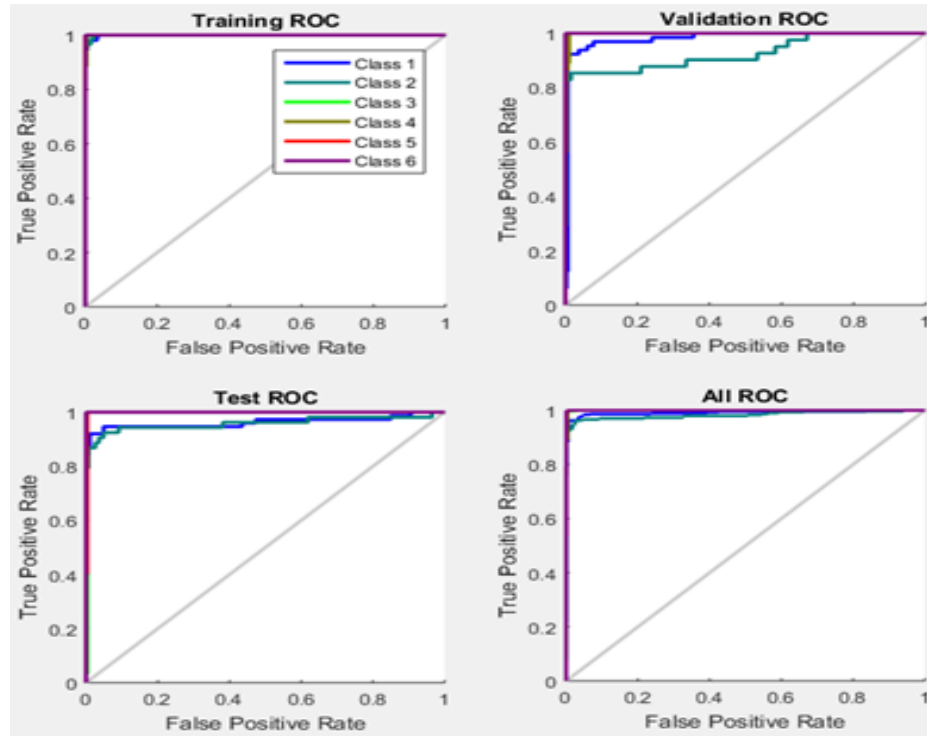


Figure 3.9. Training, Validation, Test and All ROC Curve

3.3.3 Deep Neural Networks Approach

In this approach TensorFlow utilizes Inception V3 and MobileNet models for the implementation. The basic adaptations of Inception V3 network are reduction of convolutions maximum to 3x3, increment in the general depth of the networks, and width increase technique at each layer is used to enhance feature combination. Although MobileNet architecture is consists of depth wise separable convolutions except for the first layer which is a full convolution (Howard et al., 2017). In TensorFlow, training step includes setup of different parameters which are bottleneck directory, training steps and models directory. Before the final output layer, another layer is present which is informally known as “Bottleneck”. It is responsible for the classification of objects. As soon as the bottleneck step is completed then real training of last layer is started. The step outputs are shown in the following sections:

- Training accuracy: It indicates the percentage of correct labeling of images with their true class that were used for training.
- Validation accuracy: It shows the percentage of randomly selected group of images from unique set that were labeled correctly.
- Cross Entropy: It represents the loss function that provides an overview of how well the learning process is executing. Lowest values are considered better.

The actual evaluation of performance is done on the basis of dataset that is not provided during the training state which is a real time data. The validation accuracy indicates the performance efficiency. If the value of validation accuracy is low but the training accuracy is high this shows an over fitting of data. It shows the network learn particular features of training images that do not

play any role in the classification of images. The main concept is to maintain cross entropy as low as possible. 4,000 training steps are executed by default. Each step chooses 10 images at random from the training set, finds their bottlenecks from the cache, and feeds them into the final layer to get predictions. Those predictions are then compared against the actual labels to update the final layer's weights through a back-propagation process. As the process continues, the reported accuracy improve. After all the training steps are complete, the script runs a final test accuracy evaluation on a set of images that are kept separate from the training and validation pictures. This test evaluation provides the best estimate of how the trained model will perform on the classification task.

3.3.3.1 Inception V3 Model

Transfer learning belongs to the machine learning methods where a pre-trained neural network is used. For image recognition model which is known as inception V3. The basic features of inception V3 model are the number of convolutions are reduced to 3x3, general depth of the networks are increased, and feature combination is improved by utilizing width increase technique at each layer. The schematic diagram of model as shown in figure 3.10 and it is mainly consists of two sections:

1. Extraction of feature by using CNN.
2. Classification by using fully connected layers and softmax layers.

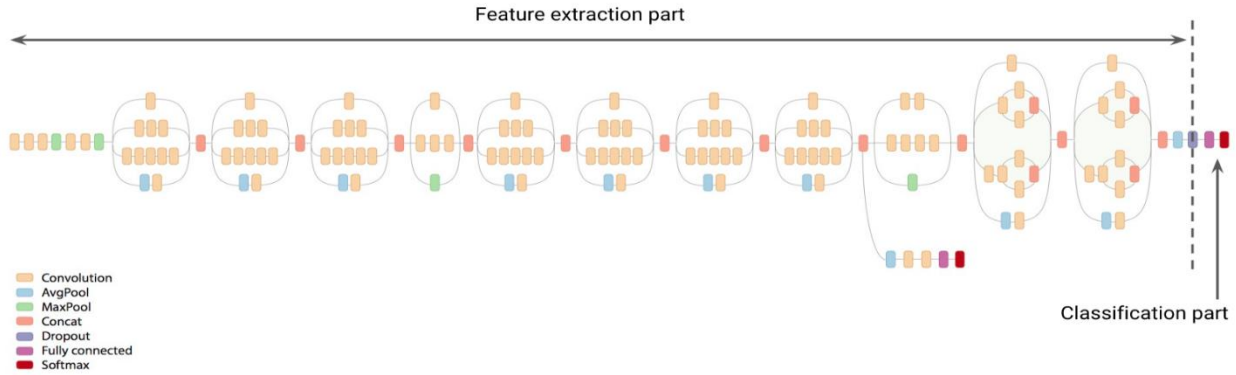


Figure 3.10. Schematic Diagram of Inception V3 (Szegedy, Christian, et al., 2016)

The softmax layer which is the last layer is retrained y using softmax regression where probabilities are picked out from the extracted features of images. The sum of weights detected by the intensity of pixels, with added bias are used to calculate evidences as shown in equation (3.2).

$$evidence_i = \sum_j W_{i,j} x_j + b_j \quad (3.2)$$

Here, an input “x” is given for the class “i” and “W_i” is the weight “b_i” is the bias and lastly “j” is the summing index of pixels in input. The evidences are used to generate the probabilities by using softmax function as shown in equation (3.3).

$$y = \text{softmax} (evidence) \quad (3.3)$$

We used the basic structure of inception V3 as shown in figure 3.11.

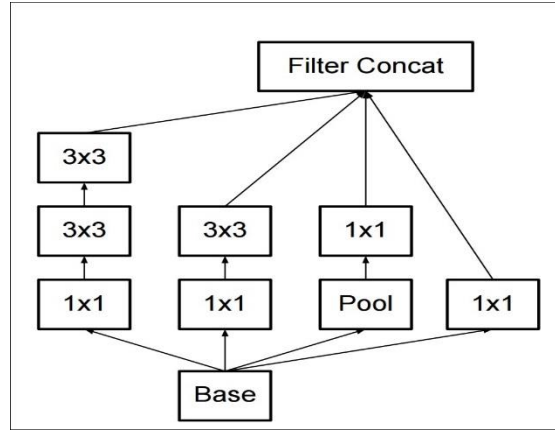


Figure. 3.11. Basic Structure of Inception V3 model

3.3.3.2 MobileNet Model

In MobileNet model it consists of depthwise separable convolutions which belongs to the factorized convolution. In this type of convolution standard factorization is used into a depthwise convolution and 1x1 point wise convolution. The standard and depthwise convolution are shown in figure 3.12. In depthwise convolution a single filter is implemented on each input. Where as in pointwise convolution a 1x1 to reunite the outputs of depthwise convolution. Both filters and combine inputs for new set of outputs done by a standard convolution in one step. These are splitted into two layers by depthwise convolution layer to perform filtering and combining. This factorization mainly contributes towards less computational time and model size. In standard convolution takes an input $Dg \times Dg \times M$ feature map F and generates $Df \times Df \times N$ feature map G . Where Df is used for the width and height of square feature map input. M shows the number of channels for input. On the other hand, Dg is used for the width and height of square feature map output. N shows the number of channels for output. For standard convolution the output feature map is shown in equation (3.4).

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \quad (3.4)$$

Depthwise convolution is consists of two layers depthwise convolution and pointwise convolution. A single filter for per channel input is used in depthwise convolution. In pointwise convolution a 1x1 convolution is performed to generate the linear combination of output of the depthwise layer. For both layers Batchnorm and Rectified Linear Unit (ReLU) nonlinearities is used in MobileNets for both layers (Howard, Andrew G., et al., 2017). For depthwise convolution a single filter per input channel is shown in equation (3.5).

$$G_{k,l,m} = \sum_{i,j} K_{i,j,m} \cdot F_{k+i-1,l+j-1,m} \quad (3.5)$$

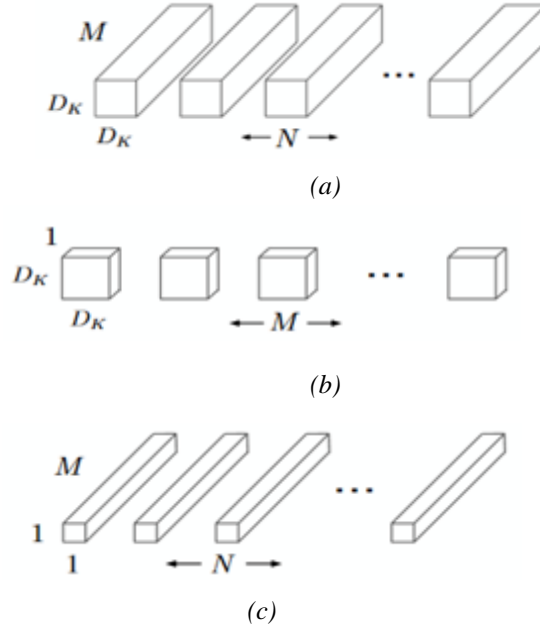


Figure 3.12 (a) standard Convolution (b) Depthwise Convolution (c) Point wise Convolution



Figure 3.13: Image from Video captured at University Road

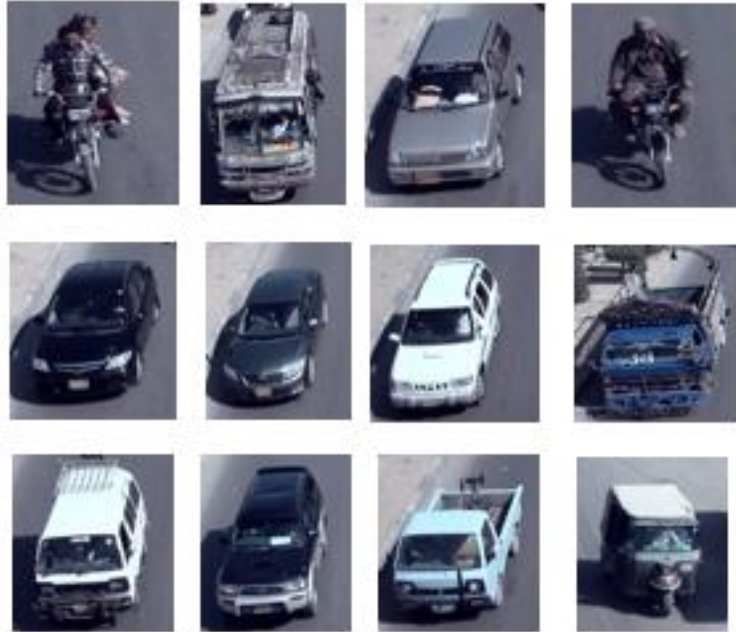


Figure 3.14: Testing Samples of Urban Dataset I

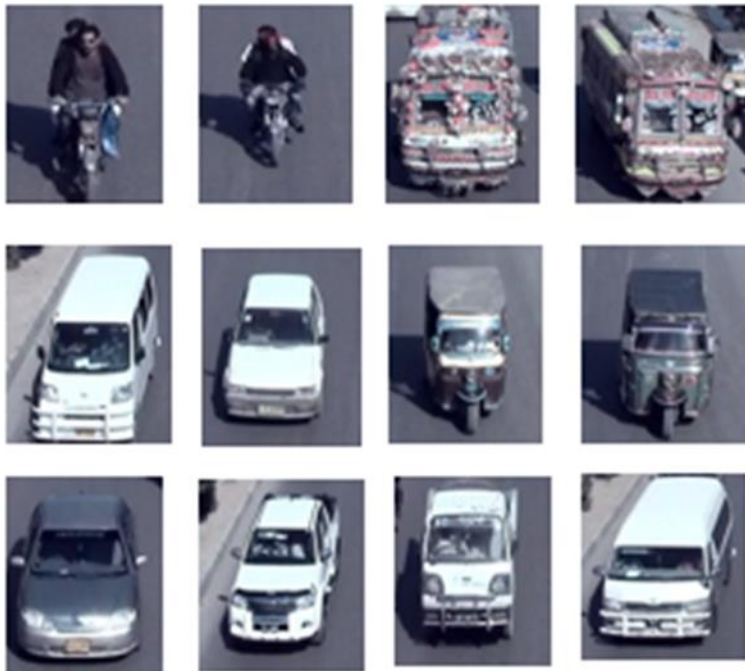


Figure 3.15: Testing Samples of Urban Dataset II



Figure 3.16: Testing Samples of Urban Dataset III



Figure 3.17: Training Samples of Urban Dataset I



Figure 3.18: Training Samples of Training Urban Dataset II



Figure 3.19: Training Samples of Training Urban Dataset III



Figure 3.20: Extraction of SURF Features

Chapter 4

4 Results and Discussion

4.1 Outcomes of Vehicle Detection

In Table 4.1 details about each lane of NIPA dataset is shown. The performance of blob analysis is quite low and detects only 665 vehicles out of 1516 as shown in table 4.1. Haar classifier # 1 detects 1242 out of 1516 vehicles as shown in table 4.2. Similarly, detection results using blobs analysis and Haar classifier are laid over input images and shown in figure 4.1 and figure 4.2 for visual inspection.

The ground truth of TOLL PLAZA dataset is shown in table 4.4. The detection results obtained from Haar cascade classifier in comparison with blob analysis are better. The numbers of total detected vehicles by using blob analysis are 247 out of 376 as shown in table 4.5. However, Haar cascade classifier detected 332 vehicles out of 376 as shown in table 4.6. The results of each method are tabulated in Table V. The pictorial results of blob analysis and Haar classifier are shown in figure 4.3 and figure 4.4. The detection results obtained from the trained Haar cascade classifier for NIPA and TOLL PLAZA datasets have 83.7% and 88.3% accuracy respectively. In contrast blob analysis has detection accuracy of 43.8% for NIPA and 65.7% for TOLL PLAZA datasets.

TABLE 4.1: GROUND TRUTH - NIPA DATASET

NIPA				
Category	Lane 1	Lane 2	Lane 3	Total
Total	687	422	407	1516

TABLE 4.2: QUALITATIVE RESULTS OBTAINED FOR BLOB ANALYSIS [NIPA DATASET]

Blob Analysis				
Category	Lane 1	Lane 2	Lane 3	Total
Total	216	228	221	665
Correct vehicle detection	31.4%	54.0%	54.2%	43.8%

TABLE 4.3: QUALITATIVE RESULTS OBTAINED FOR CASCADE CLASSIFIER # 1 [NIPA DATASET]

Cascade Classifier # 1				
168 positives, 500 negatives, 16-stages				
Category	Lane 1	Lane 2	Lane 3	Total
Total	477	396	369	1242
Correct vehicle detection	69.4%	93.8%	90.6%	81.9%

TABLE 4.4: QUALITATIVE RESULTS OBTAINED FOR CASCADE CLASSIFIER # 2 [NIPA DATASET]

Cascade Classifier # 2				
580 positives, 1500 negatives, 16-stages				
Category	Lane 1	Lane 2	Lane 3	Total
Total	507	404	359	1270
Correct vehicle detection	73.8%	95.7%	88.2%	83.7%

TABLE 4.5: GROUND TRUTH – TOLL PLAZA DATASET

TOLL PLAZA			
Category	Lane1	Lane2	Total
Total	122	254	376

TABLE 4.6: GROUND TRUTH – TOLL PLAZA DATASET

Blob Analysis			
Category	Lane1	Lane2	Total
Total	104	143	247
Correct Vehicle Detection	85.2%	56.3%	65.7%

TABLE 4.7: QUALITATIVE RESULTS OBTAINED FOR CASCADE CLASSIFIER [TOLL PLAZA DATASET]

Cascade Classifier			
1000 positives, 2000 negatives, 18-stages			
Category	Lane1	Lane2	Total
Total	111	221	332
Correct Vehicle Detection	90.9%	87.0%	88.3%



Figure 4.1: Detection results obtained from Blob Analysis [NIPA Dataset]. Lanes are labeled as 1, 2 and 3.



Figure 4.2: Detection results obtained from Haar Cascade Classifier [NIPA Dataset]. Lanes are labeled as 1, 2 and 3.



Figure 4.3: Detection results obtained from Blob Analysis [TOLL PLAZA Dataset]. Lanes are labeled as 1 and 2.



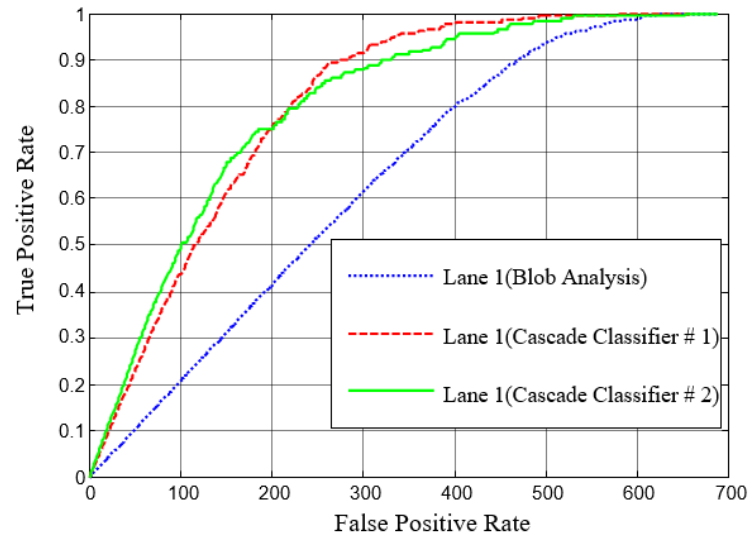
Figure 4.4: Detection results obtained from Haar Cascade Classifier [TOLL PLAZA Dataset]. Lanes are labeled as 1 and 2.

4.2 Result Validation

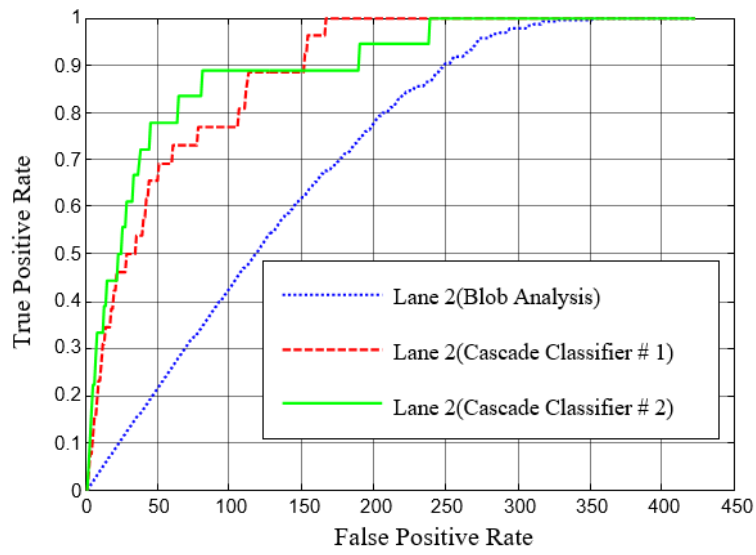
Receiver Operating Characteristics curves (ROC) is used for the analysis of algorithms implemented for vehicle detection. For NIPA dataset, the acquired results reveal that Haar cascade classifier has the highest accuracy as compared to blob analysis. Detection results are enhanced in cascade classifier 2 by increasing the number of positive images and stages. ROCs of stated algorithms for lane 1, lane 2 and lane 3 are shown in figure 4.5. The detection results of each lane are analyzed individually. It has been noted that correct detection rate for lane 1 has comparatively low accuracy as compared to lane 2 and lane 3 for both methods. This is due being the farthest lane and having camera viewing angle which resulted in increased complexity and occlusion. Similarly, for TOLL PLAZA dataset, Haar cascade classifier performed better as compared to Blob analysis and reflected by ROCs in figure 4.6. Moreover, overall efficiency of compared algorithms for NIPA and TOLL PLAZA datasets are shown in figure 4.7 and figure 4.8 respectively. Due to comparatively slow vehicle speeds and frontal view angle of the dataset thereby having better appearance and motion information both Haar classifier and blob analysis performed better in terms of detection rate as compared to NIPA results. The performance curves for each lane are compared by using the ROC (Receiver Operating Characteristics) curves. This curve uses false positive rate which means that there is a vehicle but no detection on x axis and true positive rate which means that there is a vehicle and it was detected on the y axis having T as the varying parameter. The graph between these two quantities is known as ROC curve. Generally, this curve is used to evaluate the performance of proposed algorithms. Continuous random variable “ X ” termed as “score” is used for the class prediction of each instance for binary classification. The threshold “ T ” is given and for true detection condition is defined as “ $X > T$ ” and for false in the other cases. Probability density function $f_1(x)$ uses for the instance which actually belongs to the class for “ X ” and $f_0(x)$ otherwise. Hence true positive rate and false positive rate are shown in equation (4.1) and (4.2) respectively.

$$TPR(T) = \int_T^{\infty} f_1(x) dx \quad (4.1)$$

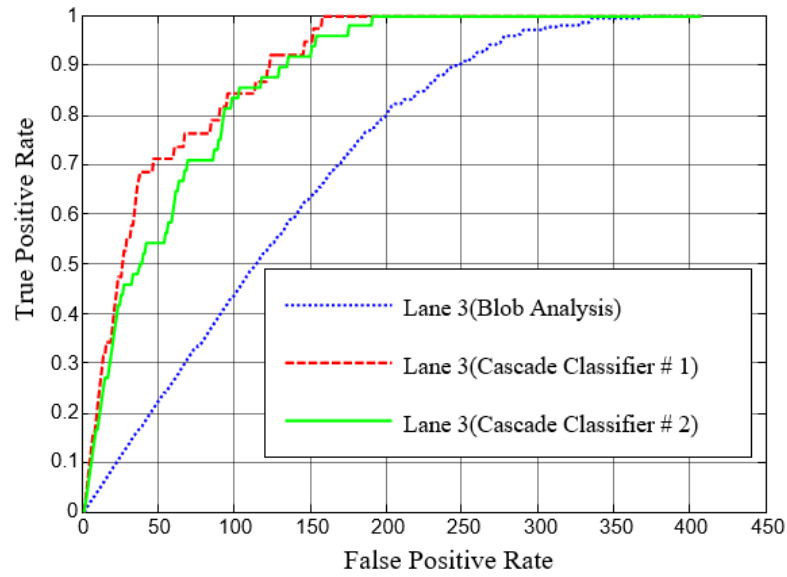
$$FPR(T) = \int_T^{\infty} f_0(x) dx \quad (4.2)$$



(a)

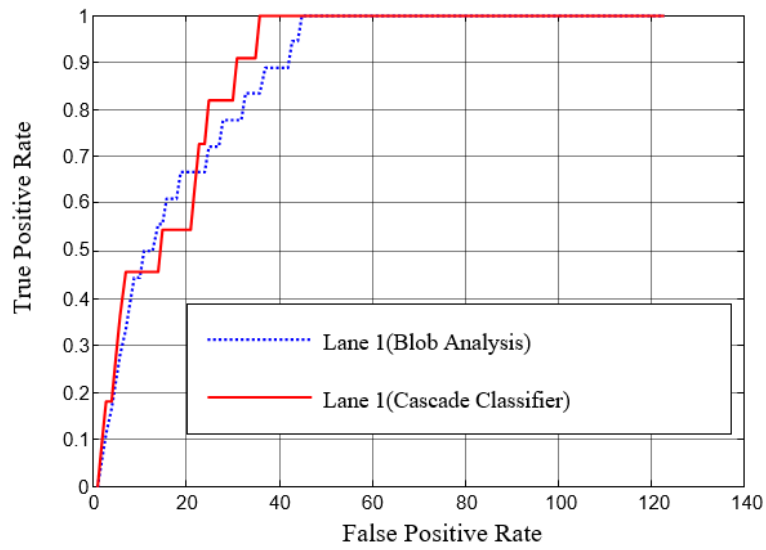


(b)

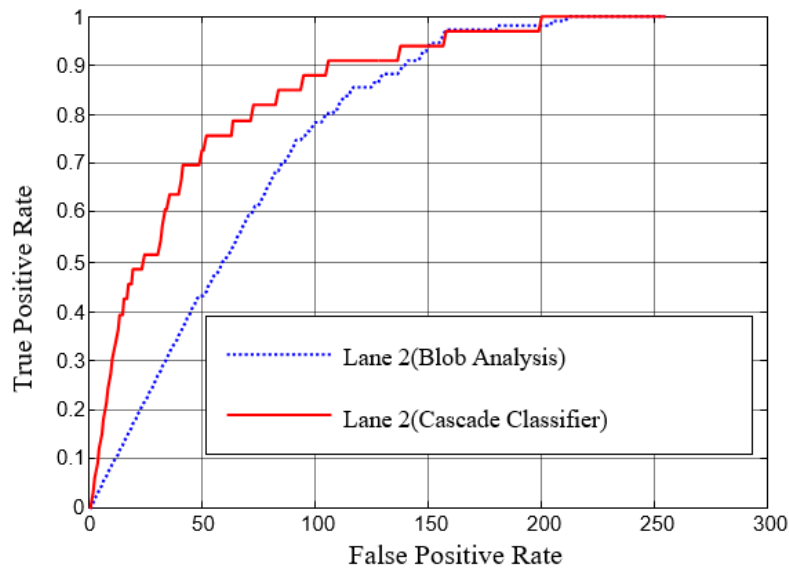


(c)

Figure 4.5: ROC Curve for NIPA Dataset (a) Lane 1 (b) Lane 2 (c) Lane 3



(a)



(b)

Figure 4.6: ROC Curve for TOLL PLAZA Dataset (a) Lane 1 (b) Lane 2

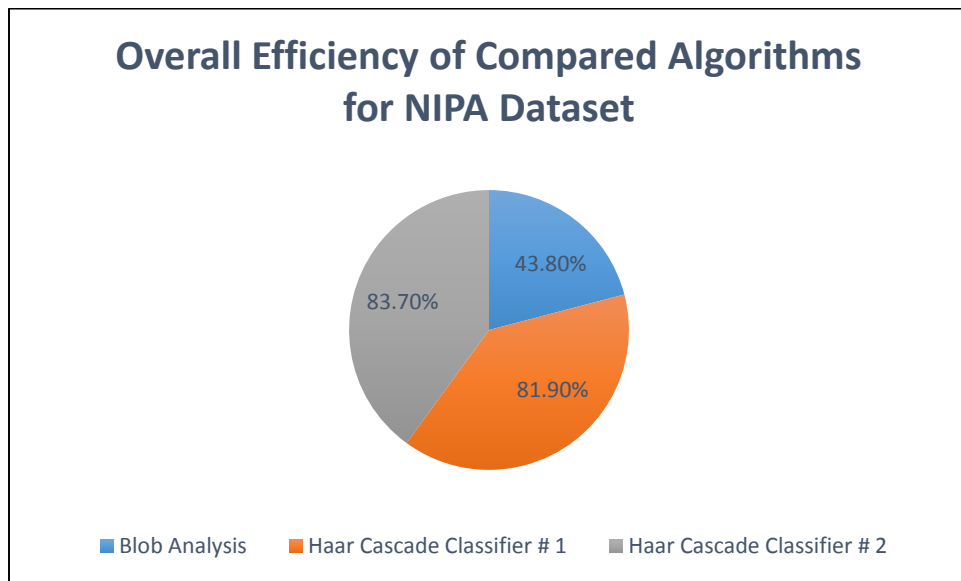


Figure 4.7: Overall efficiency of compared Algorithms for NIPA Dataset

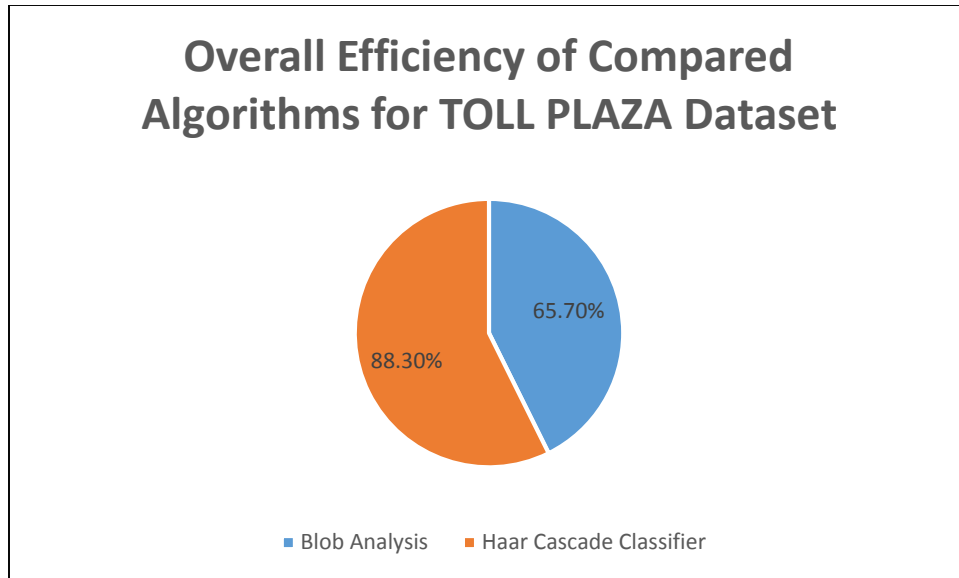


Figure 4.8: Overall efficiency of compared Algorithms for TOLL PLAZA Dataset

4.3 Outcomes of Vehicle Classification

For the classification of the vehicle we have worked on Neural Networks technique which is Scaled Conjugate gradient (SCG) Backpropagation and Deep Neural Networks techniques which are Inception V3 and MobileNet models by using different platforms namely Neural Networks Pattern Recognition Tool box in Matlab and TensorFlow respectively. The outcomes from inception V3 model are shown in figure 4.9 to figure 4.24 for each dataset. The outcomes from MobileNet model are shown in figure 4.25 to figure 4.41 for each dataset. Results obtain from MobileNet are more promising than Inception V3 and SCG Backpropagation. The results obtain from NN and DNN techniques for each datasets are shown in Table 4.8, Table 4.9 and Table 4.10 revealing the correct classification on the basis of their trained categories. The score of classified vehicle obtained is out of 1. There are some observations obtained from the results based on each individual dataset. The details are given below:

- Urban Dataset I:

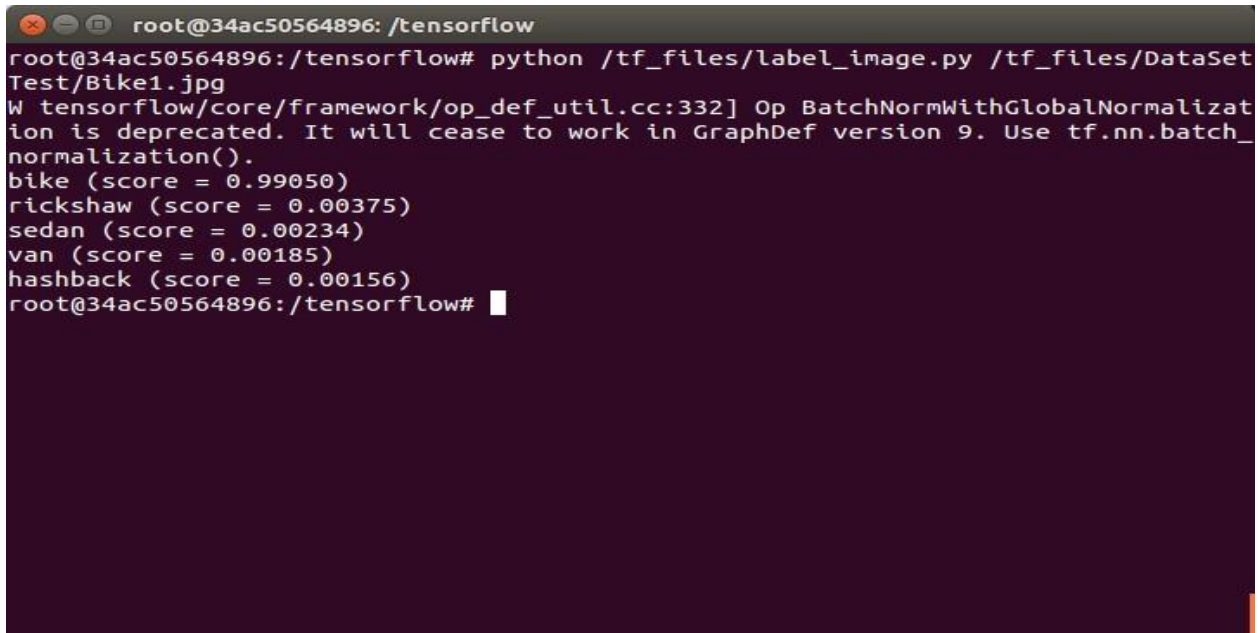
SCG has the lowest classification rate it identified only one class which is Bike. However, the results obtain from Inception V3 and MobileNet are of higher accuracy but some false classifications are observed in between Sedan and Van due to their similar structure. Overall highest accuracy is occurred in MobileNet model.

- Urban Dataset II:

In the urban dataset II, the results acquired from SCG are better as compare to the dataset I and correctly classifies 4 classes. Although in MobileNet again produces promising results but there is a misclassification occurred in class Van. Surprisingly Inception V3 has attain higher accuracy in the second dataset and classifies all the classes correctly.

- Urban Dataset III:

In the last dataset SCG classifies 5 classes successfully but MobileNet has achieved the highest accuracy among all three methods. Low accuracy results are obtain in HatchBack and Van in both Inception V3 and MobileNet. Hatchback and Van look similar from the front that is why the system is generating low accuracy results between them.

A terminal window with a dark background and light text. The prompt is 'root@34ac50564896: /tensorflow'. The command executed is 'python /tf_files/label_image.py /tf_files/DataSetTest/Bike1.jpg'. The output shows a deprecation warning for 'Op BatchNormWithGlobalNormalization' and then lists five classification results with their scores: 'bike (score = 0.99050)', 'rickshaw (score = 0.00375)', 'sedan (score = 0.00234)', 'van (score = 0.00185)', and 'hashback (score = 0.00156)'. The prompt returns to 'root@34ac50564896: /tensorflow#'.

```
root@34ac50564896: /tensorflow# python /tf_files/label_image.py /tf_files/DataSetTest/Bike1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_normalization().
bike (score = 0.99050)
rickshaw (score = 0.00375)
sedan (score = 0.00234)
van (score = 0.00185)
hashback (score = 0.00156)
root@34ac50564896: /tensorflow#
```

Figure 4.9: Classification Result obtained from Inception V3 of Bike for Urban Dataset I

```
root@34ac50564896: /tensorflow
root@34ac50564896:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/back1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
hashback (score = 0.96262)
rickshaw (score = 0.01582)
sedan (score = 0.01458)
van (score = 0.00521)
bike (score = 0.00177)
root@34ac50564896:/tensorflow#
```

Figure 4.10: Classification Result obtained from Inception V3 of Hatchback for Urban Dataset I

```
root@34ac50564896: /tensorflow
root@34ac50564896:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/Rickshaw1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
rickshaw (score = 0.97885)
hashback (score = 0.01258)
sedan (score = 0.00448)
van (score = 0.00234)
bike (score = 0.00175)
root@34ac50564896:/tensorflow#
```

Figure 4.11: Classification Result obtained from Inception V3 of Rickshaw for Urban Dataset I

```
root@34ac50564896: /tensorflow
root@34ac50564896:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/sedan2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
hashback (score = 0.89963)
sedan (score = 0.08002)
bike (score = 0.00813)
rickshaw (score = 0.00680)
van (score = 0.00542)
root@34ac50564896:/tensorflow#
```

Figure 4.12: Classification Result obtained from Inception V3 of Sedan for Urban Dataset I

```
root@34ac50564896: /tensorflow
root@34ac50564896:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/Van1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
van (score = 0.67421)
hashback (score = 0.26228)
rickshaw (score = 0.04083)
sedan (score = 0.01134)
bike (score = 0.01134)
root@34ac50564896:/tensorflow#
```

Figure 4.13: Classification Result obtained from Inception V3 of Van for Urban Dataset I


```

root@46b68537bf2d: /tensorflow
2017-02-11 04:44:17.512149: Step 470: Validation accuracy = 100.0%
2017-02-11 04:44:18.005908: Step 480: Train accuracy = 100.0%
2017-02-11 04:44:18.005977: Step 480: Cross entropy = 0.076197
2017-02-11 04:44:18.056068: Step 480: Validation accuracy = 100.0%
2017-02-11 04:44:18.553974: Step 490: Train accuracy = 99.0%
2017-02-11 04:44:18.554040: Step 490: Cross entropy = 0.081468
2017-02-11 04:44:18.604377: Step 490: Validation accuracy = 100.0%
2017-02-11 04:44:19.053203: Step 499: Train accuracy = 100.0%
2017-02-11 04:44:19.053271: Step 499: Cross entropy = 0.058100
2017-02-11 04:44:19.102429: Step 499: Validation accuracy = 100.0%
Final test accuracy = 95.6%
Converted 2 variables to const ops.
root@46b68537bf2d: /tensorflow# python /tf_files/label_image.py /tf_files/Dataset
Test/bike1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bike (score = 0.98710)
rickshaw (score = 0.00444)
van (score = 0.00443)
hashback (score = 0.00206)
sedan (score = 0.00111)
bus (score = 0.00086)
root@46b68537bf2d: /tensorflow#

```

Figure 4.14: Classification Result obtained from Inception V3 of Bike for Urban Dataset II

```

root@46b68537bf2d: /tensorflow
bike (score = 0.00069)
root@46b68537bf2d: /tensorflow# python /tf_files/label_image.py /tf_files/Dataset
Test/bus1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bus (score = 0.74999)
van (score = 0.15605)
hashback (score = 0.06236)
rickshaw (score = 0.01748)
sedan (score = 0.00770)
bike (score = 0.00642)
root@46b68537bf2d: /tensorflow# python /tf_files/label_image.py /tf_files/Dataset
Test/bus2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bus (score = 0.91520)
hashback (score = 0.03467)
van (score = 0.02393)
rickshaw (score = 0.01007)
sedan (score = 0.00922)
bike (score = 0.00691)
root@46b68537bf2d: /tensorflow#

```

Figure 4.15: Classification Result obtained from Inception V3 of Bus for Urban Dataset II


```
root@46b68537bf2d: /tensorflow
rickshaw (score = 0.00162)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/back1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
van (score = 0.47027)
hashback (score = 0.36147)
sedan (score = 0.14018)
bus (score = 0.01917)
bike (score = 0.00454)
rickshaw (score = 0.00437)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/back2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
sedan (score = 0.54201)
hashback (score = 0.43274)
van (score = 0.02195)
bus (score = 0.00145)
rickshaw (score = 0.00116)
bike (score = 0.00069)
root@46b68537bf2d:/tensorflow#
```

Figure 4.16: Classification Result obtained from Inception V3 of Hatchback for Urban Dataset II

```
root@46b68537bf2d: /tensorflow
bike (score = 0.00642)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/bus2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bus (score = 0.91520)
hashback (score = 0.03467)
van (score = 0.02393)
rickshaw (score = 0.01007)
sedan (score = 0.00922)
bike (score = 0.00691)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/rickshaw1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
rickshaw (score = 0.94167)
hashback (score = 0.02717)
van (score = 0.01340)
sedan (score = 0.00887)
bus (score = 0.00612)
bike (score = 0.00277)
root@46b68537bf2d:/tensorflow#
```

Figure 4.17: Classification Result obtained from Inception V3 of Rickshaw for Urban Dataset II

```

root@46b68537bf2d: /tensorflow
bus (score = 0.00086)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/bike2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bike (score = 0.98051)
van (score = 0.01198)
rickshaw (score = 0.00260)
sedan (score = 0.00173)
hashback (score = 0.00165)
bus (score = 0.00153)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/sedan1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
sedan (score = 0.97298)
hashback (score = 0.02482)
van (score = 0.00134)
bus (score = 0.00032)
rickshaw (score = 0.00031)
bike (score = 0.00024)
root@46b68537bf2d:/tensorflow#

```

Figure 4.18: Classification Result obtained from Inception V3 of Sedan for Urban Dataset II

```

root@46b68537bf2d: /tensorflow
bike (score = 0.00277)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/rickshaw2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
rickshaw (score = 0.90005)
hashback (score = 0.07434)
sedan (score = 0.01033)
van (score = 0.00928)
bike (score = 0.00386)
bus (score = 0.00213)
root@46b68537bf2d:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/van1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
van (score = 0.84666)
hashback (score = 0.12629)
sedan (score = 0.02130)
bus (score = 0.00299)
bike (score = 0.00155)
rickshaw (score = 0.00121)
root@46b68537bf2d:/tensorflow#

```

Figure 4.19: Classification Result obtained from Inception V3 of Van for Urban Dataset II


```

root@13a09f23e48e: /tensorflow
2017-02-11 06:52:25.921252: Step 470: Cross entropy = 0.040647
2017-02-11 06:52:25.969904: Step 470: Validation accuracy = 100.0%
2017-02-11 06:52:26.458519: Step 480: Train accuracy = 100.0%
2017-02-11 06:52:26.458584: Step 480: Cross entropy = 0.042250
2017-02-11 06:52:26.507867: Step 480: Validation accuracy = 100.0%
2017-02-11 06:52:27.001649: Step 490: Train accuracy = 100.0%
2017-02-11 06:52:27.001713: Step 490: Cross entropy = 0.033296
2017-02-11 06:52:27.050131: Step 490: Validation accuracy = 100.0%
2017-02-11 06:52:27.488520: Step 499: Train accuracy = 100.0%
2017-02-11 06:52:27.488586: Step 499: Cross entropy = 0.038988
2017-02-11 06:52:27.536208: Step 499: Validation accuracy = 100.0%
Final test accuracy = 100.0%
Converted 2 variables to const ops.
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/bike1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bike (score = 0.97156)
rickshaw (score = 0.01055)
bus (score = 0.00649)
hashback (score = 0.00583)
sedan (score = 0.00557)
root@13a09f23e48e:/tensorflow#

```

Figure 4.20: Classification Result obtained from Inception V3 of Bike for Urban Dataset III

```

root@13a09f23e48e: /tensorflow
bus (score = 0.02075)
bike (score = 0.00832)
rickshaw (score = 0.00532)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/bus1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bus (score = 0.62276)
hashback (score = 0.29906)
sedan (score = 0.05852)
rickshaw (score = 0.00984)
bike (score = 0.00982)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/bus2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
bus (score = 0.94252)
hashback (score = 0.02873)
rickshaw (score = 0.01134)
bike (score = 0.00893)
sedan (score = 0.00848)
root@13a09f23e48e:/tensorflow#

```

Figure 4.21: Classification Result obtained from Inception V3 of Bus for Urban Dataset III

```

root@13a09f23e48e: /tensorflow
bus (score = 0.01516)
bike (score = 0.00486)
rickshaw (score = 0.00336)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/back1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
hashback (score = 0.86779)
sedan (score = 0.11341)
bus (score = 0.01195)
rickshaw (score = 0.00410)
bike (score = 0.00274)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/back2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
sedan (score = 0.52093)
hashback (score = 0.44468)
bus (score = 0.02075)
bike (score = 0.00832)
rickshaw (score = 0.00532)
root@13a09f23e48e:/tensorflow#

```

Figure 4.22: Classification Result obtained from Inception V3 of Hatchback for Urban Dataset III

```

root@13a09f23e48e: /tensorflow
bus (score = 0.01487)
rickshaw (score = 0.00460)
bike (score = 0.00289)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/rickshaw1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
hashback (score = 0.64924)
sedan (score = 0.24229)
rickshaw (score = 0.07714)
bus (score = 0.01844)
bike (score = 0.01289)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/rickshaw2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
rickshaw (score = 0.48400)
hashback (score = 0.39155)
bus (score = 0.06306)
bike (score = 0.03178)
sedan (score = 0.02961)
root@13a09f23e48e:/tensorflow#

```

Figure 4.23: Classification Result obtained from Inception V3 of Rickshaw for Urban Dataset III


```
root@13a09f23e48e: /tensorflow
bus (score = 0.00750)
hashback (score = 0.00467)
sedan (score = 0.00257)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/sedan1.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
sedan (score = 0.61221)
hashback (score = 0.37260)
bus (score = 0.01105)
bike (score = 0.00242)
rickshaw (score = 0.00173)
root@13a09f23e48e:/tensorflow# python /tf_files/label_image.py /tf_files/DataSet
Test/sedan2.jpg
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalizat
ion is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_
normalization().
sedan (score = 0.67993)
hashback (score = 0.29669)
bus (score = 0.01516)
bike (score = 0.00486)
rickshaw (score = 0.00336)
root@13a09f23e48e:/tensorflow#
```

Figure 4.24: Classification Result obtained from Inception V3 model of Sedan for Urban Dataset III

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban1/Bike 1.jpg"
bike 0.999743
bus 0.000105169
rickshaw 7.44722e-05
hashback 5.04265e-05
sedan 2.55656e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.25: Classification Result obtained from MobileNet model of Bike for Urban Dataset I

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban1/Hashback 1.jpg"
hashback 0.803766
sedan 0.195433
rickshaw 0.000484542
bus 0.000230509
bike 8.23039e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.26: Classification Result obtained from MobileNet model of Hatchback for Urban Dataset I

```
dell@Dell: /media/dell/Data/Urban_Combined
dell@Dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban1/Rickshaw 2.jpg"
rickshaw 0.884099
sedan 0.111004
bike 0.00425252
bus 0.000352945
hashback 0.00026582
dell@Dell:/media/dell/Data/Urban_Combined$
```

Figure 4.27: Classification Result obtained from MobileNet model of Rickshaw for Urban Dataset I

```
dell@Dell: /media/dell/Data/Urban_Combined
dell@Dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban1/Sedan 1.jpg"
sedan 0.739202
hashback 0.2606
bus 9.67403e-05
rickshaw 6.80661e-05
bike 3.18924e-05
dell@Dell:/media/dell/Data/Urban_Combined$
```

Figure 4.28: Classification Result obtained from MobileNet model of Sedan for Urban Dataset I

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban1/Van 1.jpg"
sedan 0.846854
bus 0.101466
hashback 0.0256068
van 0.0203481
rickshaw 0.00362224
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.29: Classification Result obtained from MobileNet model of Van for Urban Dataset I

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban2/Bike 2.jpg"
bike 0.999481
hashback 0.000356603
rickshaw 8.35959e-05
sedan 4.04793e-05
bus 3.87235e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.30: Classification Result obtained from MobileNet model of Bike for Urban Dataset II


```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer
=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_m
ean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/v
al/Urban2/Hashback 1.jpg"
sedan 0.503773
hashback 0.494375
bus 0.00129304
rickshaw 0.000420152
van 7.57663e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.31: Classification Result obtained from MobileNet model of Hashback for Urban Dataset II

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer
=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_m
ean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/v
al/Urban2/Rickshaw 2.jpg"
rickshaw 0.988044
sedan 0.0114343
bike 0.000353527
hashback 0.000130851
bus 3.69545e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.32: Classification Result obtained from MobileNet model of Rickshaw for Urban Dataset II

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban2/Sedan 2.jpg"
sedan 0.989125
hashback 0.0107481
rickshaw 7.38988e-05
bus 5.035e-05
bike 1.9516e-06
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.33: Classification Result obtained from MobileNet model of Sedan for Urban Dataset II

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban2/Van 1.jpg"
sedan 0.72147
hashback 0.269848
bus 0.00703552
rickshaw 0.00105724
bike 0.00051417
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.34: Classification Result obtained from MobileNet model of Van for Urban Dataset II

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer
=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_m
ean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/v
al/Urban2/Bus 1.jpg"
bus 0.97481
sedan 0.0232326
hashback 0.00150863
rickshaw 0.000351725
bike 8.08048e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.35: Classification Result obtained from MobileNet model of Bus for Urban Dataset II

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer
=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_m
ean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/v
al/Urban3/Bike 2.jpg"
bike 0.998564
bus 0.00112284
rickshaw 0.000210745
hashback 5.10778e-05
sedan 5.04146e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.36: Classification Result obtained from MobileNet model of Bike for Urban Dataset III

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban3/Hashback 2.jpg"
sedan 0.598149
hashback 0.400359
bus 0.00101257
rickshaw 0.000462839
bike 1.5671e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.37: Classification Result obtained from MobileNet model of Hatchback for Urban Dataset III

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban3/Rickshaw 1.jpg"
rickshaw 0.913588
sedan 0.0738102
bike 0.00662238
bus 0.00508887
hashback 0.000889609
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.38: Classification Result obtained from MobileNet model of Rickshaw for Urban Dataset III

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban3/Sedan 2.jpg"
sedan 0.848554
hashback 0.150755
rickshaw 0.000426737
bus 0.000241948
bike 2.15757e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.39: Classification Result obtained from MobileNet model of Sedan for Urban Dataset III

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_mean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/val/Urban3/Van 1.jpg"
sedan 0.801338
hashback 0.198539
rickshaw 8.0253e-05
bus 1.9904e-05
bike 1.53477e-05
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.40: Classification Result obtained from MobileNet model of Van for Urban Dataset III

```
dell@dell: /media/dell/Data/Urban_Combined
dell@dell:/media/dell/Data/Urban_Combined$ python3 label_image.py --output_layer=
final_result --labels=retrained_labels.txt --graph=retrained_graph.pb --input_m
ean=-127 --input_std=127 --input_width=224 --input_height=224 --image="Dataset/v
al/Urban3/Bus 1.jpg"
bus 0.988248
hashback 0.00534747
sedan 0.00408609
rickshaw 0.00193804
bike 0.000316127
dell@dell:/media/dell/Data/Urban_Combined$
```

Figure 4.41: Classification Result obtained from MobileNet model of Bus for Urban Dataset III

TABLE 4.8: VEHICLE CLASSIFICATION RESULTS FOR URBAN DATASET I

Technique	Bike	Hatchback	Rickshaw	Sedan	Van	Bus
SCG	97%	1.6%	13%	3%	0.54%	5e-5%
Inception V3	99%	96%	97%	8%	67%	0%
MobileNet	99%	80%	88%	73%	2%	98%

TABLE 4.9: VEHICLE CLASSIFICATION RESULTS FOR URBAN DATASET II

Technique	Bike	Hatchback	Rickshaw	Sedan	Van	Bus
SCG	0%	78%	7.48e-2%	88%	34%	99%
Inception V3	98%	43%	94%	97%	84%	91%
MobileNet	99%	49%	98%	98%	0%	97%

TABLE 4.10: VEHICLE CLASSIFICATION RESULTS FOR URBAN DATASET III

Technique	Bike	Hatchback	Rickshaw	Sedan	Van	Bus
SCG	93%	95%	84%	85%	45%	0.0011%
Inception V3	97%	44%	48%	67%	0%	94%
MobileNet	99%	40%	91%	84%	80%	98%

Chapter 5

5 Conclusion and Future Work

5.1 Conclusion

This thesis provides an insight on the performance of motion (blob analysis) and appearance (Haar cascade classifier) based methods for vehicle detection in native environment. The dataset includes complex urban traffic scenes where road markings and signs are significantly low. In addition, type of vehicles ranges from motorbikes to long trucks having varying speeds thereby posing increased detection challenges. The detection results obtained from the trained Haar cascade classifier are more promising than blob analysis. Some false and misdetection are occurring due to varying vehicle speed, cast shadows, background clutter and camera viewing angle causing detection complexity and occlusion. On the other hand the classification results obtained from NN and DNN models are very interesting and reveals the true classification of the defined categories. There are some false classifications which also occurred due to the similar structure of vehicles. The handling of datasets in Inception V3 and MobileNet is more efficient as compare to SCG Backpropagation. The highest accuracy is achieved in MobileNet model it is more robust and classifies with higher rate as compare to the other two. The computational speed and accuracy of results are quite promising despite of the complexity present the evaluated datasets.

5.2 Future Work

As a future work, we plan to incorporate shadow modeling and elimination and enhance the feature pool that will help improve vehicle detection in the subject multifaceted environment. Moreover, multiple features in 3D plane can be a good choice in future to enhance the accuracy of detection. The addition of multiple features on multiple planes will eliminate and reduce the false detection in the complex environments. We plan to add more features in multiple planes so that it will give more precise detailing for the classification methods. The similar structures can be differentiated on the basis of their unique aspects so that false classifications can be reduced. Different climatic conditions is also one of the biggest challenge that needs to be address in future methods. The classification results can be further improved by defining the more details of vehicle so that the similar structures can be differentiated and avoid false results.

References:

- A. Jazayeri, H. Cai, J. Y. Zheng, M. Tuceryan, "Motion Based Vehicle Detection Identification in Car video", IEEE Intell. Veh. Symp., 23, (3), pp. 493-499, 2010.
- Adaptive vision. (2017, July 15). Retrieved from Adaptive Vision website: http://docs.adaptive-vision.com/current/studio/machine_vision_guide/BlobAnalysis.html
- Caffe. (2017, July 15). Retrieved from Caffe Website: <http://caffe.berkeleyvision.org/>
- Culibrk, Dubravko, et al. "A neural network approach to bayesian background modeling for video object segmentation." VISAPP (1). 2006.
- F.M de S. Mastros et al. Hierarchical classification of vehicle images using nn with conditional adaptive distance Neural Information Processing, ser. Lecture Notes in Computer Science, M. Lee, A. Hirose, Z.-G. Hou, and R. Kil, Eds. Springer Berlin Heidelberg, 2013, vol. 8227, pp. 745–752.
- Github. (2016, June 18). Retrieved from Github Website: <https://github.com/sonots/tutorial-haartraining/tree/master/data/negatives>.
- Heikki Huttunen et al. Car Type Recognition with Deep Neural Networks. Tampere University of Technology, Finland Visy Oy, Tampere, Finland IEEE Intelligent Vehicles Symposium (IV) Gothenburg, Sweden, June 19-22, 2016.
- Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv: 1704.04861 (2017).
- Herbert Bay et al. "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359, 2008.
- I. G. Y. Bengio and A. Courville, Deep learning. Book in preparation for MIT Press, <http://www.deeplearningbook.org>, 2016.
- Jia, Yangqing, et al. "Caffe: Convolutional architecture for fast feature embedding." Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014.
- J. P. Jodoin, G. A. Bilodeau, N. Saunier, "Urban Tracker: Multiple Object Tracking in Urban Mixed Traffic", IEEE Winter Conference on Applications of Computer Vision, 2014.
- Kadrev, Georgi, Georgi Kostadinov, and Petko Ruskov. "Expansion of a CNN-based image classifier's scope using transfer learning and k-NN. Technical report." Intelligent Systems (IS), 2016 IEEE 8th International Conference on. IEEE, 2016.
- Lavinia, Yukhe, Holly H. Vo, and Abhishek Verma. "Fusion Based Deep CNN for Improved Large-Scale Image Action Recognition." Multimedia (ISM), 2016 IEEE International Symposium on. IEEE, 2016.
- LeCun et al. "Learning methods for generic object recognition with invariance to pose and lighting." Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. Vol. 2. IEEE, 2004

Luckow, Andre, et al. "Deep learning in the automotive industry: Applications and tools." Big Data (Big Data), 2016 IEEE International Conference on. IEEE, 2016.

LeCun et al. "Learning methods for generic object recognition with invariance to pose and lighting." Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. Vol. 2. IEEE, 2004.

M. Kafai et al. Dynamic bayesian networks for vehicle classification in video M. Kafai et al. Industrial Informatics, IEEE Transactions on, vol. 8, no. 1, pp. 100–109, 2012.

M. U. Arif, Z. Lodhi, M. Khan, R. H. Raza, "Detection & Classification of Vehicles in Varying Complexity of Urban Traffic Scenes", Electronic Imaging, Video Surveillance and Transportation Imaging Applications, pp. 1-9(9), 2016.

N. C. Mithun, N. U. Rashid, S. M. Rahman, "Detection and Classification of Vehicles from Video Using Multiple Time-Spatial Images", IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 3, 2012.

NVIDIA Accelerated Computing. (2017, July 18). Retrieved from NVIDIA Accelerated Computing: <https://developer.nvidia.com/digits>

OpenCV Documentation. (2017, July 15). Retrieved from OpenCV Documentation Website: http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

P. Viola, M. J. Jones, "Robust Real-time Object Detection," International Journal of Computer Vision 57(2)", 137–154, 2001.

P. Menezes , J. C. Barreto , J. DiasFace, "Face Tracking based on Haar-like Features and Eigenfaces." IFAC/EURON Symposium on Intelligent Autonomous Vehicles. Vol. 500, 2004.

Qiling et al. Deep Neural Networks-based Vehicle Detection in Satellite Images. Fujian Key Laboratory of Sensing and Computing for Smart City School of Information Science and Engineering, Xiamen University, Xiamen, China Bioelectronics and Bioinformatics (ISBB), 2015 International Symposium, date of Conference: 14-17 Oct. 2015, IEEE Publisher.

R. Feris, J. Petterson, B. Siddiquie, L. Brown, S. Pankanti, "Large Scale Vehicle Detection in Challenging Urban Surveillance Environment", Applications of Computer Vision (WACV) IEEE Workshop, 2011.

R. Feris B. Siddiqui, Y. Zhai, "[Attribute-based Vehicle Search in Crowded Surveillance Videos", CMR'11 Proceedings of the 1st ACM International Conference on Multimedia Retrieval Article No. 18 ACM New York, NY, USA, 2011.

Redmon, Joseph, and Ali Farhadi. "YOLO9000: Better, Faster, Stronger." arXiv preprint arXiv:1612.08242 (2016).

S. Pankati, L. Brown, J. Connell, A. Datta, Q. Fan, R. S. Feris, N. Haas, Y. Li, N. Ratha, H. Trinh, "Practical Computer Vision: Example Techniques and Challenges", IBM J RES & DEV, vol 55 Paper 3, 2011.

Soo, Sander. "Object detection using Haar-cascade Classifier." Institute of Computer Science, University of Tartu (2014).

Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

Wei Wu et al. A method of vehicle classification using models and neural networks. Vehicular Technology Conference, 2001, VTC 2001 Springer IEEE VTS 53rd, date of Conference: 6-9 May 2001, date Added to IEEE Xplore: 07 August 2002.

Wikipedia. (2017, July 15). Retrieved from Wikipedia Website: <https://en.wikipedia.org/wiki/TensorFlow>.

Wikipedia. (2017, July 15). Retrieved from Wikipedia Website: [https://en.wikipedia.org/wiki/Torch_\(machine_learning\)](https://en.wikipedia.org/wiki/Torch_(machine_learning)).

Wikipedia. (2017, September 19). Retrieved from Wikipedia Website: https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

Wikipedia. (2017, August 08). Retrieved from Wikipedia Website: https://en.wikipedia.org/wiki/Speeded-up_robust_features.

X. Zhuang, W. Kang, Q. Wu, "Real-time vehicle detection with foreground-based cascade classifier", IET Image Processing, vol 10, issue 4, 2016.

Xie, Renjie, et al. "Resource-constrained implementation and optimization of a deep neural network for vehicle classification." Signal Processing Conference (EUSIPCO), 2016 24th European. IEEE, 2016.

Yong-Kul Ki et al. Vehicle-Classification Algorithm for Single-Loop Detectors Using Neural Networks. IEEE Transactions on Vehicular Technology (Volume: 55, Issue 6, Nov. 2006)

Y. Chen, G. Qin, "Video-Based Vehicle Detection and Classification in Challenging Scenarios", International Journal on Smart Sensing and Intelligent Systems, Vol.7, No.3, 2014.

Zhen Dong et al. Vehicle Type Classification Using Unsupervised Convolutional Neural Network. Pattern Recognition (ICPR), 2014 and International Conference, date of Conference: 24-28 Aug. 2014, date Added to IEEE Xplore: 08 December 2014.

Zhang et al. Vehicle classification with confidence by classified vector quantization B. Intelligent Transportation Systems Magazine, IEEE, vol. 5, no. 3, pp. 8–20, 2013.

Zhou, Yiren, et al. "Image-based Vehicle Analysis using Deep Neural Network: A Systematic Study." arXiv preprint arXiv: 1601. 01145 (2016).

Appendix A

Milestone Achieved:

Published Research Paper:

1. Yumnah Hasan, Muhammad Umair Arif, Amad Asif and Rand Hammad Raza "Comparative analysis of vehicle detection in urban traffic environment using Haar cascaded classifiers and blob statistics," 2016 Future Technologies Conference (FTC), San Francisco, CA, 2016, pp. 547-552. doi: 10.1109/FTC.2016.7821660 ". Published in IEEE.

Accepted Research Papers:

1. Abdul Rafay, Adnan Iqbal, Yumnah Hasan "Recognition of Finger Print Biometric System Access Control for Car Memory Settings through Artificial Neural Network " has been accepted for inclusion in the Future of Information and Communications Conference (FICC) 2018 to be held from 5-6 April 2018 in Singapore. It is technically sponsored by IEEE.
2. Yumnah Hasan, Anser Ahmed, Muhammad Umair Arif "A Comparative Study of Neural and Deep Neural Networks Based Vehicle Classification Techniques" has been accepted for inclusion in the Future of Information and Communications Conference (FICC) 2018 to be held from 5-6 April 2018 in Singapore. It is technically sponsored by IEEE.

Evidences of Published and Accepted papers:

Please see number 70-75.