TRAFFIC INTENSITY PREDICTION BASED ON DEEP LEARNING

A PROJECT REPORT

Submitted by

PRIYA.K(960219104077)

RESHMA.R(960219104081)

SABITHA.N(960219104086)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ARUNACHALA COLLEGE OF ENGINEERING FOR WOMEN

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**TRAFFIC INTENSITY PREDICTION BASED ON DEEP LEARNING**" is the bonafide work of "**PRIYA.K(960219104077),RESHMA.R(960219104081),SABITHA.N (960219104086)**" who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.T.V.Chithra,M.E.,Ph.D.,	Mrs. J.Caroline Misbha,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Associate Professor and Head,	Assistant Professor,
Department of CSE,	Department of CSE,
Arunachala College of Engineering	Arunachala College of
for Women,	Engineering for Women,
Vellichanthai-629203	Vellichanthai-629203

Submitted for the B.E**.** project viva-voce Examination held at Arunachala College of Engineering for women on _______________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we thank the **Lord Almighty** who had showered his blessings on us by providing us with good health and knowledge, and were with us in each and every step of this project work.

It is a great pleasure to express deep sense of gratitude to our respected **Chairman Dr.T.KRISHNASWAMY, M.E., Ph.D.**, of Arunachala College of Engineering for Women for all the efforts and administration in educating us in his prestigious institution.

We wish to express our sincere thanks to our **Principal Dr.S.JOSEPH JAWHAR, M.E., Ph.D., M.B.A (HR)., M.Sc(Psychology)., M.I.E., M.I.S.T.E.,** for having granted to do the project in this organisation.

We solicit our thanks to our respected **HOD Dr.T.V.Chithra, M.E., Ph.D.**, for her constant encouragement and valuable suggestions throughout the completion of this project.

We express our sincere gratitude to **Mrs. J.Caroline Misbha M.E, Assistant Professor** and our supervisor in continuously guiding me to complete our project work successfully.

We express our heartfelt thanks to our staffs and parents for their moral support and constant encouragement. We also thank each and everyone who had helped us directly or indirectly. At this junction we gratefully thank our friends and classmates for extending their valuable suggestion in bringing out this project.

ABSTRACT

The purposes are to analyze the safety and real-time performance of intelligent transportation under Deep learning and Artificial Neural Network. and solve urban transportation safety problems. The existing problems of complex traffic data collection and single factor analysis are analyzed. This project helps to predict and classify traffic intensity based on historical traffic data. Deep Learning (ANN) is a popular supervised learning algorithm that is widely used for classification and regression tasks in various fields, including traffic prediction. Intelligent traffic intensity prediction and classification using deep learning has been applied in various real-world scenarios, including urban traffic management, freeway traffic prediction, and public transportation planning.

It has been shown to be an effective and accurate method for predicting and classifying traffic intensity, and can be used to improve traffic flow, reduce congestion, and enhance transportation safety. Meantime, the proposed algorithm can effectively inhibit the spread of congestion, and achieve the effect of timely evacuation for traffic congestion

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the fast social and economic development, urbanization gets accelerated continuously, and the noticeable increase in motor vehicles as a means of transportation has aroused various transportation safety concerns, such as car accidents and road congestion. Statistics suggest that from 2010 to 2020, car accidents in China have caused the deaths of more than 80,000 people each year on average, bringing a major impact on the health and well-being of more than 350,000 people. Urban car accidents occupy 1/3 of the total in China, ranking the first worldwide posing a severe threat to the safety of people's lives and properties.Car accidents have now become the fundamental causes of traffic congestion and road network paralysis. Hence, as various intelligent algorithms, such as Artificial Intelligence (AI), and Deep Learning (DL), smart urban transportation has become a hot topic all around the world.

To ensure urban transportation safety, road infrastructure construction should be strengthened, traffic control measures should be formulated, and intelligent transportation must be developed. Only depending on road infrastructure construction and traditional traffic control measures brings pressure on a city's economy, with very limited effects.

Hence, intelligent transportation is particularly urgent regarding the constantly occurring transportation safety issues. To provide prerequisites and guarantees for acquiring loads of instant, reliable, and dynamic transportation data, which can accurately predict the traffic flow, breaking through the major bottlenecks that restrict urban Traffic Flow Prediction (TFP). Various traffic detection and information communication approaches get popularized, so that the accuracy, breadth, and annotation of traffic data have been improved continuously, and transportation has shifted from a data-indigence to a data-abundant era. Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data.

In short, under the background of traffic big data, mining the hidden laws in traffic data, understanding real-time traffic information, mastering the characteristics and changing laws of traffic flow, and improving traffic congestion are the prerequisite and foundation for intelligent transportation, as well as the guarantee to solve safety problems in urban transportation.

1.1 MACHINE LEARNING

Machine learning is the scientific field dealing with the ways in which machines learn from experience. For many scientists, the term “machine learning” is identical to the term “artificial intelligence”, given that the possibility of learning is the main characteristic of an entity called intelligent in the broadest sense of the word.

The purpose of machine learning is the construction of computer systems that can adapt and learn from their experience. A more detailed and formal definition of machine learning is given by Mitchel: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

With the rise of Machine Learning approaches, we have the ability to find a solution to this issue, we have developed a system using data mining which has the ability to predict whether the patient has diabetes or not.

Furthermore, predicting the disease early leads to treating the patients before it becomes critical. Data mining has the ability to extract hidden knowledge from a huge amount of diabetes-related data. Because of that, it has a significant role in diabetes research, now more than ever.

The aim of this research is to develop a system which can predict the diabetic risk level of a patient with a higher accuracy. This research has focused on developing a system based on three classification methods namely, Support Vector Machine, Logistic regression and Artificial Neural Network algorithms.

Supervised Learning

In supervised learning, the system must “learn” inductively a function called target function, which is an expression of a model describing the data. The objective function is used to predict the value of a variable, called dependent variable or output variable, from a set of variables, called independent variables or input variables or characteristics or features. The set of possible input values of the function, i.e., its domain, are called instances. Each case is described by a set of characteristics (attributes or features).

A subset of all cases, for which the output variable value is known, is called training data or examples. In order to infer the best target function, the learning system, given a training set, takes into consideration alternative functions, called hypothesis and denoted by h. In supervised learning, there are two kinds of learning tasks: classification and regression.

Classification models try to predict distinct classes, such as e.g., blood groups, while regression models predict numerical values. Some of the most common techniques are Decision Trees (DT), Rule Learning, and Instance Based Learning (IBL), such as k-Nearest Neighbours (k-NN), Genetic

Algorithms (GA), Artificial Neural Networks (ANN), and Support Vector Machines (SVM).

Unsupervised Learning

In unsupervised learning, the system tries to discover the hidden structure of data or associations between variables. In that case, training data consists of instances without any corresponding labels. Association Rule Mining appeared much later than machine learning and is subject to greater influence from the research area of databases.

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Reinforcement Learning

The term Reinforcement Learning is a general term given to a family of techniques, in which the system attempts to learn through direct interaction with the environment so as to maximize some notion of cumulative reward.

It is important to mention that the system has no prior knowledge about the behaviour of the environment and the only way to find out is through trial and failure (trial and error). Reinforcement learning is mainly applied to autonomous systems, due to its independence in relation to its environment.

1.2 TRAFFIC INTENSITY PREDICTION

Automatic traffic intensity prediction can be achieved using machine learning algorithms that analyze historical traffic data to forecast future traffic conditions. There are several steps involved in this process, including data collection, preprocessing, feature extraction, and model training and evaluation.

First, traffic data needs to be collected from various sources, such as traffic sensors, cameras, and GPS devices. This data can include information such as vehicle speed, volume, and occupancy, as well as weather and road conditions.

Next, the collected data needs to be preprocessed to clean and transform it into a usable format for analysis. This step can include removing missing data, converting data types, and normalizing data.

After preprocessing, relevant features need to be extracted from the data to use as inputs for the machine learning model. These features could include time of day, day of the week, weather conditions, and historical traffic patterns.

Finally, the machine learning model can be trained and evaluated using the extracted features and historical traffic data. The model can then be used to make predictions about future traffic intensity based on current and historical data.

Overall, automatic traffic intensity prediction can help improve traffic management and reduce congestion by allowing authorities to anticipate and plan for traffic flow.

1.3 PROBLEM STATEMENT

The key of intelligent transportation is to organically integrate AI algorithms with traditional transportation models, perceiving, optimizing, and adjusting traffic conditions in real-time, improving the current situation of urban traffic congestion, maximizing the efficiency of road utilized, and guaranteeing people's travel safety simultaneously. At present, intelligent transportation often comprises traffic information collection and management systems, intelligent transportation systems, and electronic toll systems.

To ensure urban transportation safety, road infrastructure construction should be strengthened, traffic control measures should be formulated, and intelligent transportation must be developed. Only depending on road infrastructure construction and traditional traffic control measures brings pressure on a city's economy, with very limited effects. Hence, intelligent transportation is particularly urgent regarding the constantly occurring transportation safety issues.

1.4 OBJECTIVE

- The main objective of this system is to improve transportation safety by detecting and classifying safety events in real-time, predicting future events, and analyzing driver behavior.
- The system should be capable of analyzing transportation safety data in real-time to provide quick responses to safety events.

CHAPTER 2

LITERATURE SURVEY

2.1 R. F. Berriel, A. T. Lopes, A. F. de Souza, and T. Oliveira-Santos, "Deep Learning Based Large-Scale Automatic Satellite Crosswalk Classification," IEEE Geoscience and Remote Sensing Letters, vol. 14, pp. 1513–1517, Sept 2019.

High-resolution satellite imagery has been increasingly used on remote sensing classification problems. One of the main factors is the availability of this kind of data. Even though, very little effort has been placed on the zebra crossing classification problem. In this paper, crowdsourcing systems are exploited in order to enable the automatic acquisition and annotation of a large-scale satellite imagery database for crosswalks related tasks. Then, this dataset is used to train deep-learning-based models in order to accurately classify satellite images that contains or not zebra crossings.

A novel dataset with more than 240,000 images from 3 continents, 9 countries and more than 20 cities was used in the experiments. Experimental results showed that freely available crowdsourcing data can be used to accurately train robust models to perform crosswalk classification on a global scale.

Advantages

- High Accuracy
- Trained on very large dataset.

Disadvantages

- Interoperability
- Not suitable for small dataset.

2.2 Big Data Analytics in Intelligent Transportation Systems: A Survey Li Zhu, Fei Richard IEEE 2019.

In this work, It shows a subsystem to handle pedestrians in crosswalks using deep neural networks for the IARA autonomous car, which relies on camera and LIDAR data fusion. Crosswalks' positions were manually annotated in IARA's map. Pedestrians are detected in the camera image using a convolutional neural network (CNN). Then, pedestrians' positions in the map are obtained by fusing their positions in the image with the LIDAR point cloud. Subsequently, if a pedestrian position is inside the crosswalk area, the crosswalk is set as busy. Finally, a busy crosswalk message is published to the High-Level Decision Maker subsystem. This subsystem selects the car's behavior according to the crosswalk condition and propagates this decision down through the control pipeline, in order to make the car drive correctly through the crosswalk area.

The Pedestrian Handler subsystem was evaluated on IARA, which was driven autonomously for various laps along a real and complex circuit with various crosswalks. In all passages through crosswalks, the Pedestrian Handler dealt with pedestrians as expected, i.e., without any human intervention.

Advantages

- Improved safety

Disadvantages

- Complexity
- Complex Architecture

2.3 S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Advances in neural information processing systems, pp. 91–99, 2017.

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet and Fast R-CNN have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, they introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features-using the recently popular terminology of neural networks with 'attention' mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model, the detection system has a frame rate of 5 fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image.

Advantages

- speed and Efficiency
- Nearly cost-free region proposals

Disadvantages

- Complexity and Computational requirements
- Not so accurate in real-time

2.4 An Edge Traffic Flow Detection Scheme Based on Deep Learning in an Intelligent Transportation System Bin li et al 2019 IEEE.

Traffic light recognition (TLR) is an integral part of any intelligent vehicle, which must function in the existing infrastructure. Pedestrian and sign detection have recently seen great improvements due to the introduction of learning-based detectors using integral channel features. A similar push have not yet been seen for the detection sub-problem of TLR, where detection is dominated by methods based on heuristic models. Evaluation of existing systems is currently limited primarily to small local datasets.

In order to provide a common basis for comparing future TLR research an extensive public database is collected based on footage from US roads. The database consists of both test and training data, totaling 46,418 frames and 112,971 annotated traffic lights, captured in continuous sequences under a varying light and weather conditions. The learning-based detector achieves an AUC of 0.4 and 0.32 for day sequence 1 and 2, respectively, which is more than an order of magnitude better than the two heuristic model-based detectors.

Advantages

- Experimental results are promising

Disadvantages

- Not suitable for high illuminant condition

2.5 N. Fairfield and C. Urmson, "Traffic Light Mapping and Detection," 2011. IEEE.

The outdoor perception problem is a major challenge for driver-assistance and autonomous vehicle systems. While these systems can often employ active sensors such as sonar, radar, and lidar to perceive their surroundings, the state of standard traffic lights can only be perceived visually. By using a prior map, a perception system can anticipate and predict the locations of traffic lights and improve detection of the light state. The prior map also encodes the control semantics of the individual lights. This paper presents methods for automatically mapping the three-dimensional positions of traffic lights and robustly detecting traffic light state onboard cars with cameras. We have used these methods to map more than four thousand traffic lights, and to perform onboard traffic light detection for thousands of drives through intersections.

Advantages

- Robustly detecting traffic light stat

Disadvantages

- Sensitivity to hyperparameters

2.6 D. Barnes, W. Maddern, and I. Posner, "Exploiting 3D Semantic Scene Priors for Online Traffic Light Interpretation," in Intelligent Vehicles Symposium (IV), 2018.

In this paper, present a probabilistic framework for increasing online object detection performance when given a semantic 3D scene prior, which we apply to the task of traffic light detection for autonomous vehicles. Previous approaches to traffic light detection on autonomous vehicles have involved either precise knowledge of the relative 3D positions of the vehicle and the traffic light (requiring accurate and expensive mapping and localization systems), or a classifier-based approach that searches for traffic lights in images (increasing the chance of false detections by searching all possible locations for traffic lights). It combine both approaches by explicitly incorporating both prior map and localization uncertainty into a classifier-based object detection framework, generating a scale-space search region that only evaluates parts of the image likely to contain traffic lights, and weighting object detection scores by both the classifier score and the 3D occurrence prior distribution.

Advantages

- Reduces computation time, cost-effective

Disadvantages

- Limited contextual information

2.7 R. De Charette and F. Nashashibi, "Traffic Light Recognition using Image Processing Compared to Learning Processes," 2009 IEEE.

In this paper it represents a real-time traffic light recognition system for intelligent vehicles. The method proposed is fully based on image processing. Detection step is achieved in grayscale with spot light detection, and recognition is done using our generic adaptive templates. The whole process was kept modular which make our TLR capable of recognizing different traffic lights from various countries. To compare our image processing algorithm with standard object recognition methods also developed several traffic light recognition systems based on learning processes such as cascade classifiers with AdaBoost. This system was validated in real conditions in our prototype vehicle and also using registered video sequence from various countries (France, China, and U.S.A.). It was noticed high rate of correctly recognized traffic lights and few false alarms. Processing is performed in real-time on 640x480 images using a 2.9 GHz single core desktop computer. Since traffic scenes are complex and include lots of information, keeping constant attention on the traffic signs is not an easy task for drivers. Therefore, some traffic data (signs) can be missed for several causes such as the complexity of the road scene, the high number of visual information, or even the driver's stress or visual fatigue.

Advantages

- No complex structures used

Disadvantages

- Do not focus on attribute of traffic lights

2.8 R. De Charette and F. Nashashibi, "Real Time Visual Traffic Lights Recognition Based on Spot Light Detection and Adaptive Traffic Lights Templates," in Intelligent Vehicles Symposium (IV), 2016.

This paper introduces a new real-time traffic light recognition system for on-vehicle camera applications. This approach has been tested with good results in urban scenes. Thanks to the use of our generic "adaptive templates" it would be possible to recognize different kinds of traffic lights from various countries. Our approach is mainly based on a spot detection algorithm therefore able to detect lights from a high distance with the main advantage of being not so sensitive to motion blur and illumination variations.

The detected spots together with other shape analysis form strong hypothesis we feed our adaptive templates matcher with. Even though it is still in progress, our system was validated in real conditions in our prototype vehicle and also using registered video sequences. They noticed a high rate of correctly recognized traffic lights and very few false alarms. Processing is performed in real-time on 640 times 480 images using a 2.9 GHz single core desktop computer.

Advantages

- Not so sensitive to motion blur and illumination variations

Disadvantages

- Less accurate on adverse conditions

2.9 G. Siogkas, E. Skodras, and E. Dermatas, "Traffic Lights Detection in Adverse Conditions using Color, Symmetry and Spatiotemporal Information," Computer Vision Theory and Applications (VISAPP), 2012.

This paper proposes the use of a monocular video camera for traffic lights detection, in a variety of conditions, including adverse weather and illumination. The system incorporates a color pre-processing module to enhance the discrimination of red and green regions in the image and handle the "blooming effect" that is often observed in such scenes.

The fast radial symmetry transform is utilized for the detection of traffic light candidates and finally false positive results are minimized using spatiotemporal persistency verification. The system is qualitatively assessed in various conditions, including driving in the rain, at night and in city roads with dense traffic, as well as their synergy. It is also quantitatively assessed on a publicly available manually annotated database, scoring high detection rates.

Advantages

- Trained on a larger dataset

Disadvantages

- Not suitable for inclined roads in real-time

2.10 M. Weber, P. Wolf, and J. M. Zöllner, “DeepTLR: A single Deep Convolutional Network for Detection and Classification of Traffic Lights,” in Intelligent Vehicles Symposium (IV), 2016.

Recent improvements in object detection are driven by the success of convolutional neural networks (CNN). They are able to learn rich features outperforming hand-crafted features. So far, research in traffic light detection mainly focused on hand-crafted features, such as color, shape or brightness of the traffic light bulb. This paper presents a deep learning approach for accurate traffic light detection in adapting a single shot detection (SSD) approach. SSD performs object proposals creation and classification using a single CNN.

The original SSD struggles in detecting very small objects, which is essential for traffic light detection. By the adaptations it is possible to detect objects much smaller than ten pixels without increasing the input image size. We present an extensive evaluation on the DriveU Traffic Light Dataset (DTLD). It reach both, high accuracy and low false positive rates. The trained model is real-time capable with ten frames per second on a Nvidia Titan Xp.

Advantages

- Reached notable results in real-time

Disadvantages

- Processing time is higher

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Traffic flow prediction is a crucial task for sustainable smart cities, as it enables the efficient management of transportation resources and reduces congestion, air pollution, and greenhouse gas emissions. One approach to traffic flow prediction is to use deep learning models that can learn from historical data and make accurate predictions based on spatiotemporal features.

One popular deep learning model for spatiotemporal data is the attention-based model. Attention-based models have shown excellent performance in various tasks, including natural language processing and computer vision, and have recently been applied to spatiotemporal data.

The attention-based deep learning model for traffic flow prediction can be trained on historical traffic data, including traffic volume, speed, and other relevant features. The model can learn the relationships between traffic patterns and various spatiotemporal features such as weather, time of day, and road conditions.

The model's attention mechanism can help identify the most relevant spatiotemporal features for predicting traffic flow, allowing the model to focus on the most critical information. This attention-based approach can help improve the model's accuracy, particularly in situations where there are many spatiotemporal features to consider.

The input to the model can be a sequence of spatiotemporal features, with the output being a predicted traffic flow for each time step in the sequence. The model can be trained using supervised learning, where the predicted traffic flow is compared to the actual traffic flow, and the model's parameters are updated to minimize the prediction error.

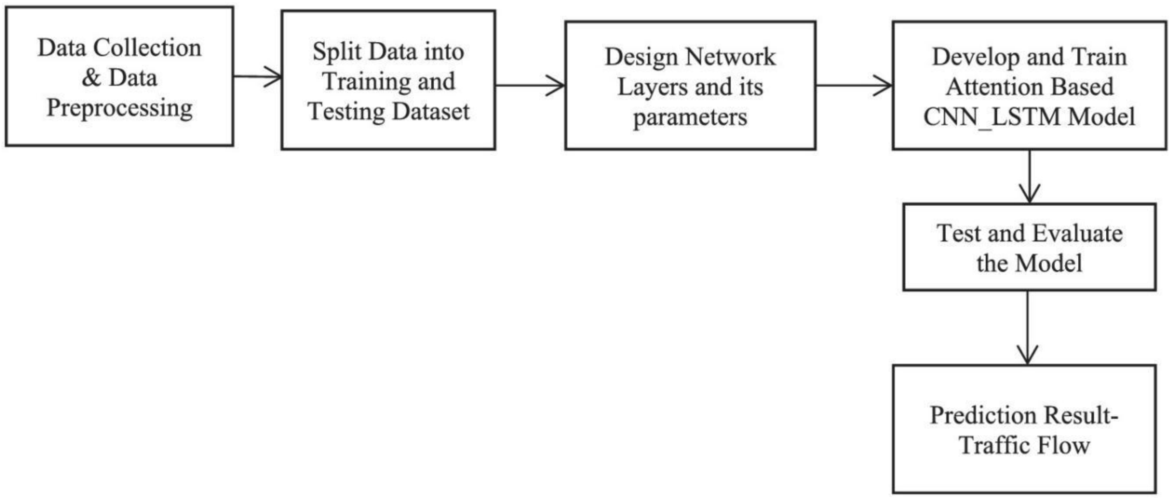

Data Collection & Data Preprocessing
Split Data into Training and Testing Dataset
Design Network Layers and its parameters
Develop and Train Attention Based CNN_LSTM Model
Test and Evaluate the Model
Prediction Result- Traffic Flow

Fig 3.1: Block Diagram of Existing System

Therefore, an attention-based deep learning model for traffic flow prediction using spatiotemporal features can help smart cities manage transportation resources more efficiently and reduce traffic congestion, air pollution, and greenhouse gas emissions. The model's attention mechanism can help identify the most relevant spatiotemporal features for predicting traffic flow, making it a powerful tool for sustainable smart cities.

3.1.1 Disadvantages

While attention-based deep learning models for traffic flow prediction using spatiotemporal features have many benefits, there are also some drawbacks that should be considered.

Data requirements: One major drawback of attention-based deep learning models is their data requirements. These models require large amounts of high-quality data to train effectively. In the case of traffic flow prediction, obtaining high-quality data can be challenging, especially in developing countries where the data collection infrastructure may not be well-established.

Model interpretability: Attention-based models are often considered black-box models, making it difficult to understand how the model makes its predictions. In traffic flow prediction, understanding the factors that contribute to traffic flow is essential for policymakers to make informed decisions. Therefore, models that are more interpretable may be more appropriate in this context.

Generalization: Attention-based models can sometimes be overfit to the training data, meaning they may not generalize well to new, unseen data. This can be especially problematic in traffic flow prediction, where changes in traffic patterns or infrastructure could significantly impact traffic flow.

Computationally expensive: Attention-based models can be computationally expensive to train and deploy, which may limit their practical use in some contexts. In particular, real-time traffic flow prediction may require models that can make predictions quickly and efficiently, which could be challenging with attention-based models.

In summary, attention-based deep learning models for traffic flow prediction using spatiotemporal features have several drawbacks, including data requirements, model interpretability, generalization, and computational cost. However, these limitations can be addressed with appropriate data Collection and preprocessing, model architecture design, and training techniques.

3.2 PROPOSED SYSTEM

Real-time intelligent automatic transportation safety based on Artifical Neural Network can be a promising approach to enhance transportation safety.

Artificial Neural Network is a machine learning algorithm that can be used for classification, regression, and outlier detection.

The Artificial Neural Network algorithm can be trained using labeled data from different transportation safety scenarios such as accident prediction, driver behavior analysis, and road condition analysis. Artificial Neural Network algorithm uses a non-linear mapping function to map the input data into a higher-dimensional space where the data can be easily classified. It then finds a hyperplane that separates the different classes of data with maximum margin.

The real-time intelligent automatic transportation safety system can use Artificial Neural Network to analyze real-time data from different transportation safety sensors such as cameras, radars, and lidars. The ANN algorithm can detect and classify different transportation safety events such as lane departure, collision warning, and pedestrian detection.

The system can also use ANN to predict future transportation safety events based on historical data. For example, it can predict the likelihood of an accident happening at a particular location based on the historical accident data and current road conditions.

The ANN algorithm can also be used for driver behavior analysis to detect unsafe driving practices such as distracted driving, speeding, and aggressive driving. The system can then issue warnings to the driver or take control of the vehicle to prevent accidents.

The real-time intelligent automatic transportation safety based on ANN algorithm can significantly improve transportation safety by detecting and classifying transportation safety events in real-time, predicting future events, and analyzing driver behavior.

Real-time intelligent automatic transportation safety based on ANN algorithm is a system that uses machine learning techniques to predict and prevent road accidents. Artificial Neural Network (ANN) is a supervised learning algorithm that can classify and predict data based on input features. Artificial Neural Network is widely used in classification, regression, and other machine learning tasks.

The system consists of two main components: data acquisition and prediction. In the data acquisition phase, sensors are used to collect data on road conditions, weather, vehicle speed, and other relevant parameters. This data is then preprocessed and input into the ANN algorithm to predict the likelihood of an accident occurring.

In the prediction phase, the ANN algorithm analyzes the input data and classifies the current road condition as safe or risky. If the road condition is identified as risky, the system sends an alert to the driver or the transportation control center, informing them of the potential danger. The system can also provide recommendations for safe driving based on the current road conditions.

3.2.1 ARTIFICIAL NEURAL NETWORK

The invention of these Neural Networks took place in the 1970s but they have achieved huge popularity due to the recent increase in computation power because of which they are now virtually everywhere. In every application that you use, Neural Networks power the intelligent interface that keeps you engaged. Artificial Neural Networks are a special type of machine learning algorithms that are modelled after the human brain. That is, just like how the neurons in our nervous system are able to learn from the past data, similarly, the ANN is able to learn from the data and provide responses in the form of predictions or classifications.

ANNs are nonlinear statistical models which display a complex relationship between the inputs and outputs to discover a new pattern. A variety of tasks such as image recognition, speech recognition, machine translation as well as medical diagnosis makes use of these artificial neural networks.

An important advantage of ANN is the fact that it learns from the example data sets. Most commonly usage of ANN is that of a random function approximation. With these types of tools, one can have a cost-effective method of arriving at the solutions that define the distribution. ANN is also capable of taking sample data rather than the entire dataset to provide the output result. With ANNs, one can enhance existing data analysis techniques owing to their advanced predictive capabilities.

Artificial Neural Networks Architecture

The functioning of the Artificial Neural Networks is similar to the way neurons work in our nervous system. The Neural Networks go back to the early **1970s** when **Warren S McCulloch and Walter Pitts** coined this term. In order to understand the workings of ANNs, let us first understand how it is structured. In a neural network, there are three essential layers –

Input Layers

The *input layer* is the first layer of an ANN that receives the input information in the form of various texts, numbers, audio files, image pixels, etc.

Hidden Layers

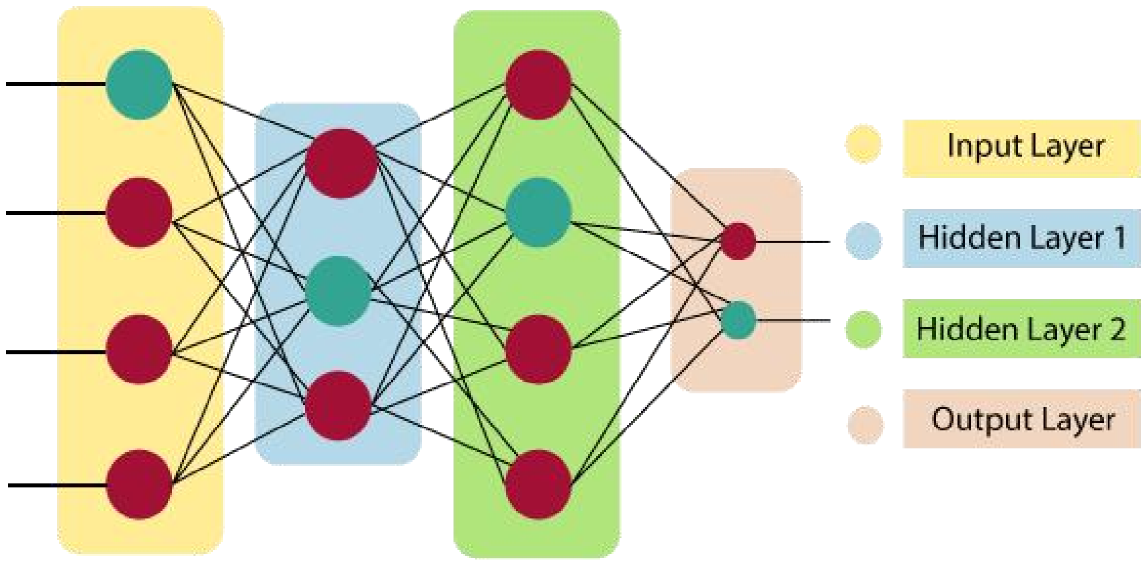
In the middle of the ANN model are the *hidden layers*. There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These hidden layers perform various types of mathematical computation on the input data and recognize the patterns that are part of.

Output Layer

In the *output layer*, we obtain the result that we obtain through rigorous computations performed by the middle layer.

In a neural network, there are multiple parameters and hyperparameters that affect the performance of the model. The output of ANNs is mostly dependent on these parameters. Some of these parameters are weights, biases, learning rate, batch size etc. Each node in the ANN has some weight.

Each node in the network has some weights assigned to it. A transfer function is used for calculating the weighted sum of the inputs and the bias.



3.2.2 Advantages

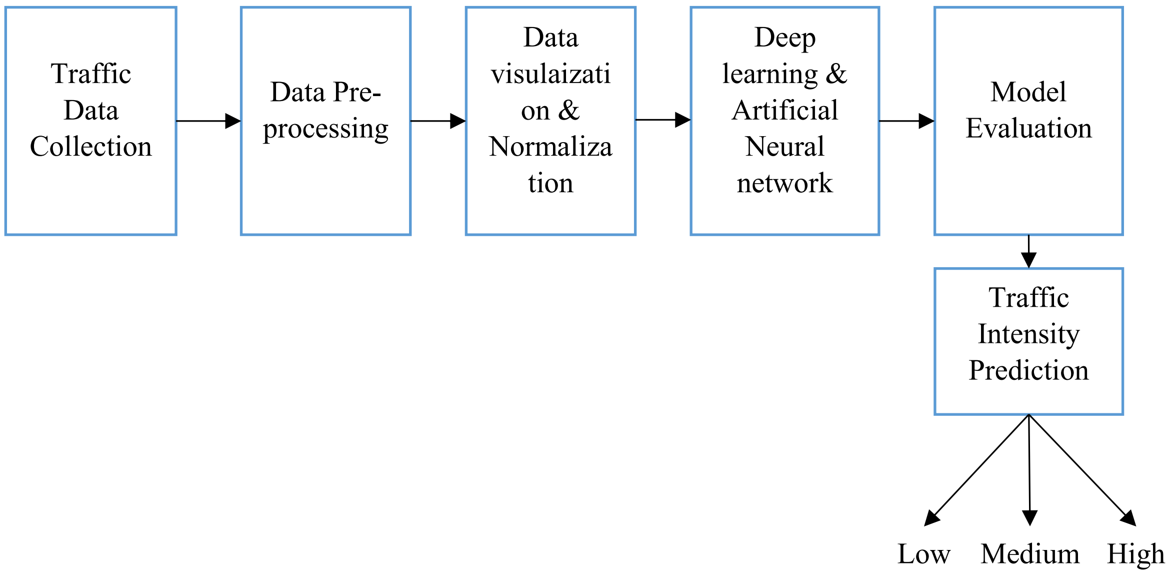
- A neural network can implement tasks that a linear program cannot.
- When an item of the neural network declines, it can continue without some issues by its parallel features.
- A neural network determines and does not require to be reprogrammed.
- It can be executed in any application

CHAPTER 4

SYSTEM DESIGN

4.1 BLOCK DIAGRAM

The block diagram of the proposed system has been given in Fig 4.1.



Traffic Data Collection
Data Pre-processing
Data visulaization & Normalization
Deep learning & Artificial Neural network
Model Evaluation
Traffic Intensity Prediction
Low
Medium
High

Fig 4.1: Block Diagram of Proposed System

4.2 MODULES

The modules involved in the proposed system are:

- Data Collection
- Data Preprocessing
- Data Visualization and Normalization
- Deep learning and Artificial Neural Network
- Model Evaluation
- Traffic Intensity Prediction

Data Collection

This module involves collecting traffic data from various sources, such as cameras, loop detectors, and GPS devices. The data can include traffic volume, speed, flow, and other relevant parameters.

Data Preprocessing

This module involves cleaning, filtering, and normalizing the collected data to remove noise, outliers, and inconsistencies. This is an essential step to ensure the quality and reliability of the data.

Feature Extraction

This module involves extracting relevant features from the preprocessed data. The features can include statistical measures such as mean, variance, and standard deviation, as well as other parameters such as time of day, day of week, and weather conditions.

ANN Model Training

This module involves training the ANN model using the preprocessed data and extracted features. The ANN model is trained to predict and classify traffic intensity based on the historical data.

Model Evaluation

This module involves evaluating the performance of the trained ANN model using test data. The evaluation can include metrics such as accuracy, precision, recall, and F1-score.

Traffic Intensity Prediction

This module involves using the trained ANN model to predict traffic intensity for new data. The predicted traffic intensity can be further classified into different levels, such as low, moderate, and high.

CHAPTER 5
SYSTEM DESCRIPTION

5.1 Hardware Requirements

System	:	Pentium IV 3.5 GHz or Latest Version.
Hard Disk	:	40 GB.
Monitor	:	14’ Color Monitor.
Mouse	:	Optical Mouse.
Ram	:	1 GB.

5.2 Software Requirements

Operating system	:	Windows 10
Coding Language	:	Python Programming
Tools used	:	Jupyter Notebook

5.2.1 Operating System: Windows 10

Microsoft Windows is a group of several graphical operating system families, all of which are developed, marketed, and sold by Microsoft. Each family caters to a certain sector of the computing industry.

Active Windows families include Windows NT and Windows Embedded; these may encompass subfamilies, e.g., Windows Embedded Compact (Windows CE) or Windows Server. Defunct Windows families include Windows 9x, Windows Mobile and Windows Phone.

Microsoft introduced an operating environment named Windows on November 20, 1985, as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUIs).

Microsoft Windows came to dominate the world's personal computer (PC) market with over 90% market share, overtaking Mac OS, which had been introduced in 1984. Apple came to see Windows as an unfair encroachment on their innovation in GUI development as implemented on products such as the Lisa and Macintosh (eventually settled in court in Microsoft's favor in 1993). On PCs, Windows is still the most popular operating system.

However, in 2014, Microsoft admitted losing the majority of the overall operating system market to Android, because of the massive growth in sales of Android smartphones. In 2014, the number of Windows devices sold was less than 25% that of Android devices sold.

This comparison however may not be fully relevant, as the two operating systems traditionally target different platforms. Still, numbers for server use of Windows (that are comparable to competitors) show one third market share, similar to for end user use.

Fig 5.1: Windows OS

Microsoft, the developer of Windows, has registered several trademarks each of which denote a family of Windows operating systems that target a specific sector of the computing industry. As of 2014, the following Windows families are being actively developed.

5.2.2 Jupyter Notebook

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in

2014 by Fernando Pérez. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter.

Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab.Project Jupyter's operating philosophy is to support interactive data science and scientific computing across all programming languages via the development of open-source software.

According to the Project Jupyter website, "Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license".

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context.

A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Fig 5.2: Jupyter Notebook

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- IPython
- ØMQ
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

The Jupyter Notebook has become a popular user interface for cloud computing, and major cloud providers have adopted the Jupyter Notebook or derivative tools as a frontend interface for cloud users. Examples include Amazon's SageMaker Notebooks, Google's Colaboratory and Microsoft's Azure Notebook.

Colaboratory (also known as Colab) is a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive. Colab was originally an internal Google project; an attempt was made to open source all the code and work more directly upstream, leading to the development of the "Open in Colab" Google Chrome extension, but this eventually ended, and Colab development continued internally.

As of October 2019, the Colaboratory UI only allows for the creation of notebooks with Python 2 and Python 3 kernels; however, an existing notebook whose kernelspec is IR or Swift will also work, since both R and Swift are installed in the container. Julia language can also work on Colab (with e.g., Python and GPUs; Google's tensor processing units also work with Julia on Colab).

5.2.3 Python Programming:

Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected.

It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Fig 5.3: Python Programming

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity.[60] When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SOFTWARE IMPLEMENTATION

Implementation includes all those activities that take place to convert from the old system to the new. The old system consists of manual operations, which is operated in a very different manner from the proposed new system.

A proper implementation is essential to provide a reliable system to meet the requirements of the organizations. An improper installation may affect the success of the computerized system.

6.1.1 Implementation Methods

There are several methods for handling the implementation and the consequent conversion from the old to the new computerized system.

The most secure method for conversion from the old system to the new system is to run the old and new system in parallel. In this approach, a person may operate in the manual older processing system as well as start operating the new computerized system.

This method offers high security, because even if there is a flaw in the computerized system, we can depend upon the manual system. However, the cost for maintaining two systems in parallel is very high. This outweighs its benefits.

Another commonly method is a direct cut over from the existing manual system to the computerized system. The change may be within a week or within a day. There are no parallel activities. However, there is no remedy in case of a problem. This strategy requires careful planning.

A working version of the system can also be implemented in one part of the organization and the personnel will be piloting the system and changes can be made as and when required. But this method is less preferable due to the loss of entirety of the system.

The implementation plan includes a description of all the activities that must occur to implement the new system and to put it into operation. It identifies the personnel responsible for the activities and prepares a time chart for implementing the system.

The implementation plan consists of the following steps.

- List all files required for implementation.
- Identify all data required to build new files during the implementation.
- List all new documents and procedures that go into the new system.

The implementation plan should anticipate possible problems and must be able to deal with them.

The usual problems may be missing documents; mixed data formats between current and files, errors in data translation, missing data etc.

CHAPTER 7

SYSTEM TESTING

System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error. A good test is one that has a probability of finding an as yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing. Testing is the vital to the success of the system.

In the code testing the logic of the developed system is tested. For this every module of the program is executed to find an error. To perform specification test, the examination of the specifications stating what the program should do and how it should perform under various conditions. Unit testing focuses first on the modules in the proposed system to locate errors. This enables to detect errors in the coding and logic that are contained within that module alone. Those resulting from the interaction between modules are initially avoided. In unit testing step each module has to be checked separately.

System testing does not test the software as a whole, but rather than integration of each module in the system. The primary concern is the compatibility of individual modules. One has to find areas where modules have been designed with different specifications of data lengths, type and data element name. Testing and validation are the most important steps after the implementation of the developed system. The system testing is performed to ensure that there are no errors in the implemented system. The software must be executed several times in order to find out the errors in the different modules of the system.

Validation refers to the process of using the new software for the developed system in a live environment i.e., new software inside the organization, in order to find out the errors. The validation phase reveals the failures and the bugs in the developed system. It will be come to know about the practical difficulties the system faces when operated in the true environment. By testing the code of the implemented software, the logic of the program can be examined. A specification test is conducted to check whether the specifications stating the program are performing under various conditions. Apart from these tests, there are some special tests conducted which are given below:

Peak Load Tests: This determines whether the new system will handle the volume of activities when the system is at the peak of its processing demand. The test has revealed that the new software for the agency is capable of handling the demands at the peak time.

Storage Testing: This determines the capacity of the new system to store transaction data on a disk or on other files. The proposed software has the required storage space available, because of the use of a number of hard disks.

Performance Time Testing: This test determines the length of the time used by the system to process transaction data.

7.1 Test Plan

In this phase the software developed Testing is exercising the software to uncover errors and ensure the system meets defined requirements. Testing may be done at 4 levels

- Unit Level
- Module Level
- Integration & System
- Regression

Unit Testing:

A Unit corresponds to a screen /form in the package. Unit testing focuses on verification of the corresponding class or Screen. This testing includes testing of control paths, interfaces, local data structures, logical decisions, boundary conditions, and error handling. Unit testing may use Test Drivers, which are control programs to co-ordinate test case inputs and outputs, and Test stubs, which replace low-level modules. A stub is a dummy subprogram.

Validation Testing:

In this requirement established as part of software requirements analysis are validated against the software that has been constructed. Validation testing provides final Assurance that software meets all functional, behavioral and performance requirements. Validation can be defined in many ways but a simple definition is that validation succeeds when software Function in a manner that can be reasonably by the customer.

1. Validation test criteria
2. Configuration review
3. Alpha and Beta testing (conducted by end user)

Module Level Testing:

Module Testing is done using the test cases prepared earlier. Module is defined during the time of design.

Integration & System Testing:

Integration testing is used to verify the combining of the software modules. Integration testing addresses the issues associated with the dual problems of verification and program construction. System testing is used to verify, whether the developed system meets the requirements. System testing is actually a series different test whose primary purpose is to full exercise the computer base system. Where the software and other system elements are tested as whole. To test computer software, we spiral out along streamlines that broadens the scope of testing with each turn.

The last higher-order testing step falls outside the boundary of software Engineering and in to the broader context of computer system engineering. Software, once validated, must be combining with order system Elements (e.g. hardware, people, databases). System testing verifies that all the elements Mesh properly and that overall system function/performance is achieved.

1.Recovery Testing

2.Security Testing

3.Stress Testing

Regression Testing:

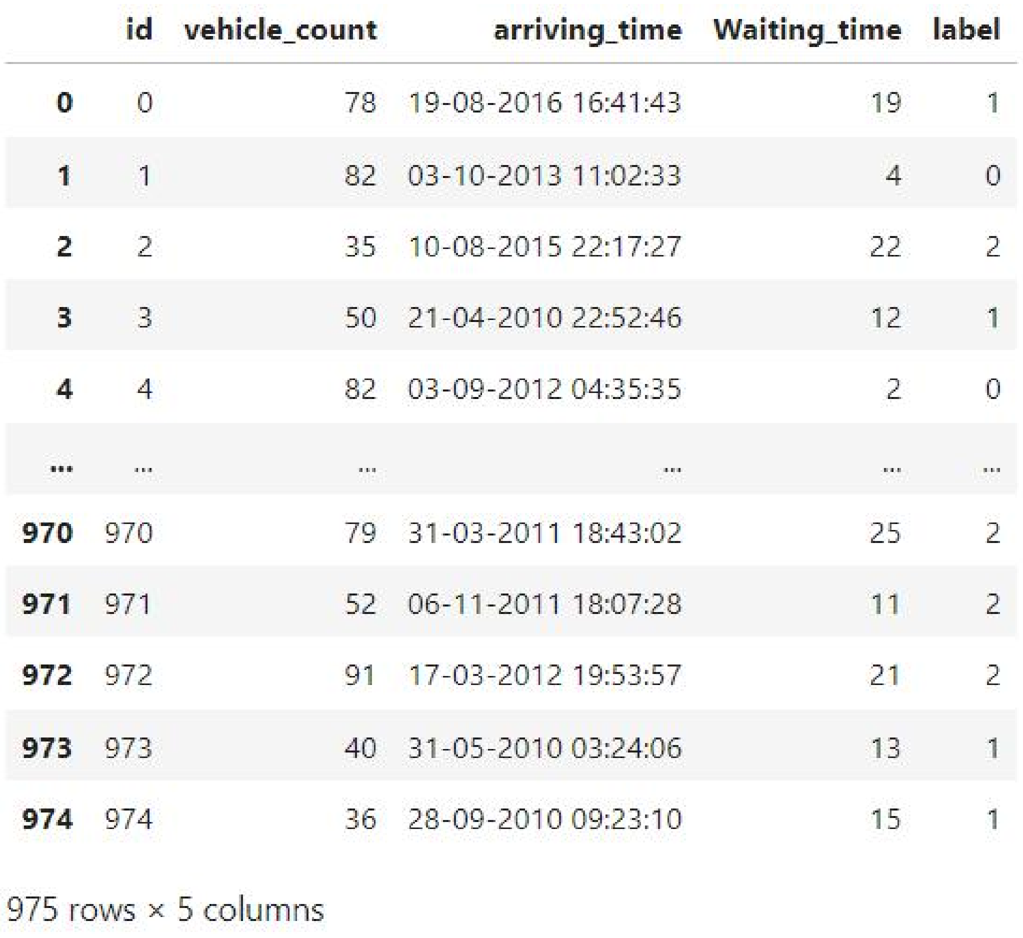
Each modification in software impacts unmodified areas, which results serious injuries to that software. So, the process of re-testing for rectification of errors due to modification is known as regression testing.

Installation and Delivery: Installation and Delivery is the process of delivering the developed and tested software to the customer. Refer the support procedures.

Acceptance and Project Closure: Acceptance is the part of the project by which the customer accepts the product. This will be done as per the Project Closure, once the customer accepts the product, closure of the project is started. This includes metrics collection, PCD, etc.

CHAPTER 8

RESULTS AND DISCUSSION

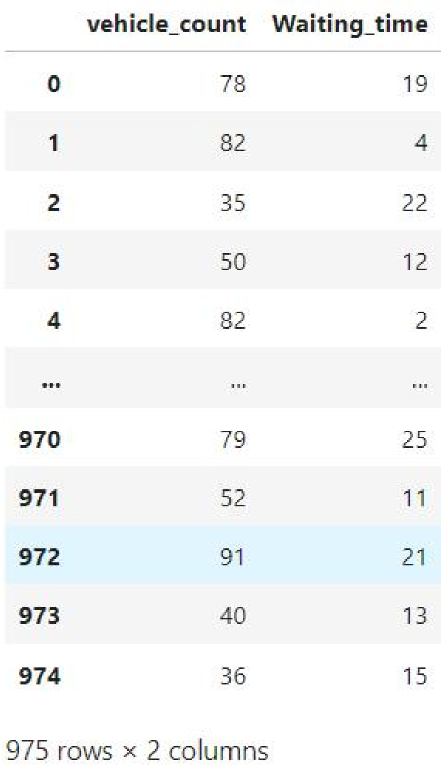


	id	vehicle_count	arriving_time	Waiting_time	label
0	0	78	19-08-2016 16:41:43	19	1
1	1	82	03-10-2013 11:02:33	4	0
2	2	35	10-08-2015 22:17:27	22	2
3	3	50	21-04-2010 22:52:46	12	1
4	4	82	03-09-2012 04:35:35	2	0
...	...	...	...	...	...
970	970	79	31-03-2011 18:43:02	25	2
971	971	52	06-11-2011 18:07:28	11	2
972	972	91	17-03-2012 19:53:57	21	2
973	973	40	31-05-2010 03:24:06	13	1
974	974	36	28-09-2010 09:23:10	15	1

975 rows × 5 columns

Fig 8.1: Head and Tail Rows of Dataset

To view a small sample of a Series or the Data Frame object, use the head() and the tail() methods. head() returns the first n rows(observe the index values). The default number of elements to display is five, but you may pass a custom number. tail() returns the last n rows(observe the index values).

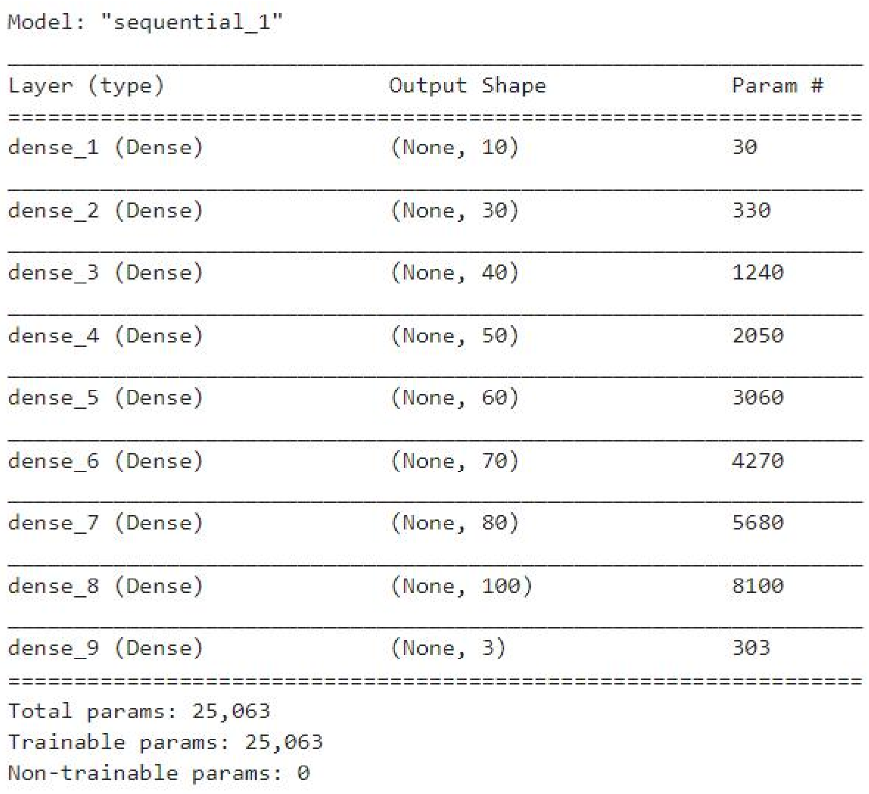


	vehicle_count	Waiting_time
0	78	19
1	82	4
2	35	22
3	50	12
4	82	2
...	...	...
970	79	25
971	52	11
972	91	21
973	40	13
974	36	15

975 rows × 2 columns

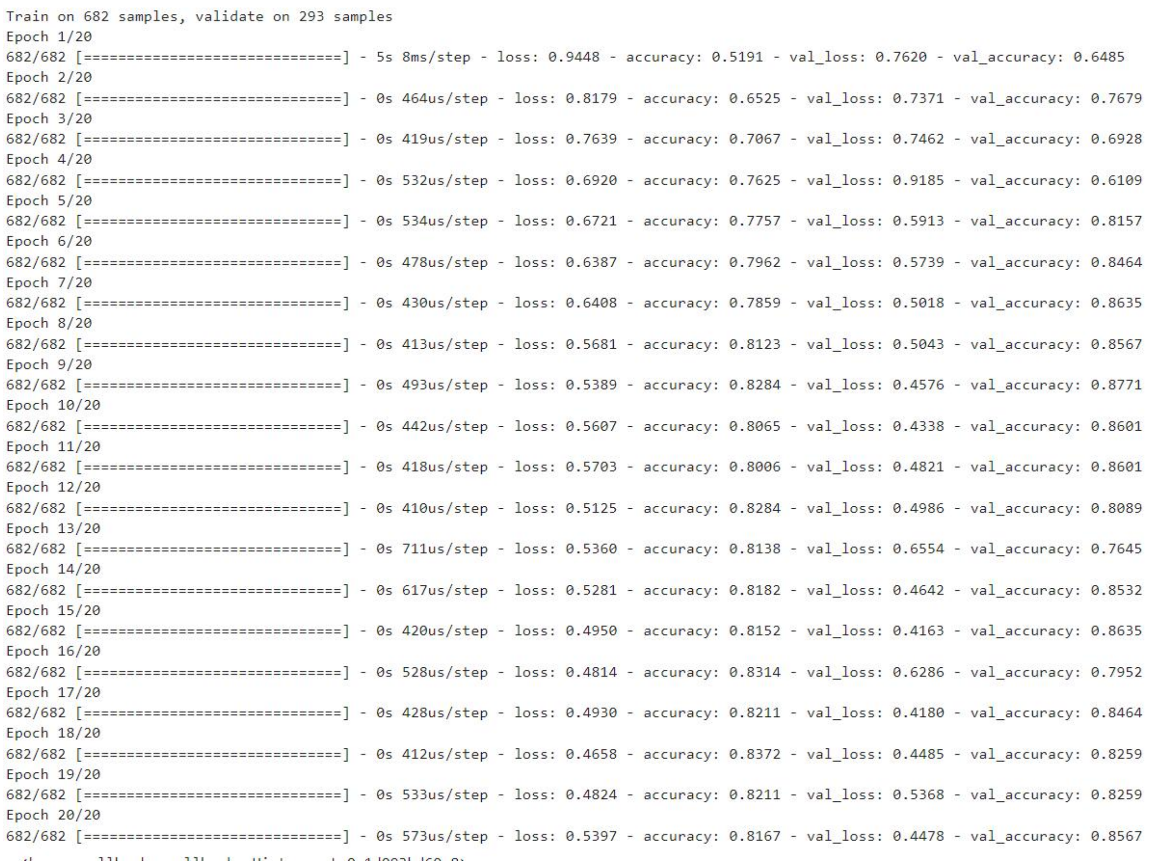
Fig 8.2: Vehicle Count and Waiting Time

Vehicle counting is a computer vision solution that automates the process of vehicle detection and classification. Vehicle counting software focuses on keeping track of the number and type of vehicles that enter and leave through a particular route for accurate monitoring of traffic.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                30
_________________________________________________________________
dense_2 (Dense)              (None, 30)                330
_________________________________________________________________
dense_3 (Dense)              (None, 40)                1240
_________________________________________________________________
dense_4 (Dense)              (None, 50)                2050
_________________________________________________________________
dense_5 (Dense)              (None, 60)                3060
_________________________________________________________________
dense_6 (Dense)              (None, 70)                4270
_________________________________________________________________
dense_7 (Dense)              (None, 80)                5680
_________________________________________________________________
dense_8 (Dense)              (None, 100)               8100
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 303
=================================================================
Total params: 25,063
Trainable params: 25,063
Non-trainable params: 0
```

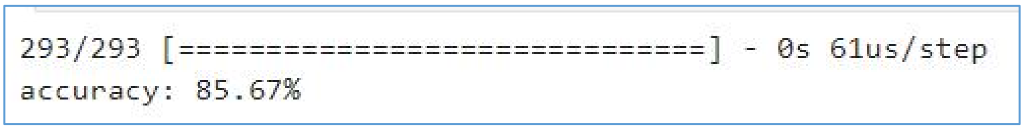
Fig 8.3: Model Summary



```
Train on 682 samples, validate on 293 samples
Epoch 1/20
682/682 [==============================] - 5s 8ms/step - loss: 0.9448 - accuracy: 0.5191 - val_loss: 0.7620 - val_accuracy: 0.6485
Epoch 2/20
682/682 [==============================] - 0s 464us/step - loss: 0.8179 - accuracy: 0.6525 - val_loss: 0.7371 - val_accuracy: 0.7679
Epoch 3/20
682/682 [==============================] - 0s 419us/step - loss: 0.7639 - accuracy: 0.7067 - val_loss: 0.7462 - val_accuracy: 0.6928
Epoch 4/20
682/682 [==============================] - 0s 532us/step - loss: 0.6920 - accuracy: 0.7625 - val_loss: 0.9185 - val_accuracy: 0.6109
Epoch 5/20
682/682 [==============================] - 0s 534us/step - loss: 0.6721 - accuracy: 0.7757 - val_loss: 0.5913 - val_accuracy: 0.8157
Epoch 6/20
682/682 [==============================] - 0s 478us/step - loss: 0.6387 - accuracy: 0.7962 - val_loss: 0.5739 - val_accuracy: 0.8464
Epoch 7/20
682/682 [==============================] - 0s 430us/step - loss: 0.6408 - accuracy: 0.7859 - val_loss: 0.5018 - val_accuracy: 0.8635
Epoch 8/20
682/682 [==============================] - 0s 413us/step - loss: 0.5681 - accuracy: 0.8123 - val_loss: 0.5043 - val_accuracy: 0.8567
Epoch 9/20
682/682 [==============================] - 0s 493us/step - loss: 0.5389 - accuracy: 0.8284 - val_loss: 0.4576 - val_accuracy: 0.8771
Epoch 10/20
682/682 [==============================] - 0s 442us/step - loss: 0.5607 - accuracy: 0.8065 - val_loss: 0.4338 - val_accuracy: 0.8601
Epoch 11/20
682/682 [==============================] - 0s 418us/step - loss: 0.5703 - accuracy: 0.8006 - val_loss: 0.4821 - val_accuracy: 0.8601
Epoch 12/20
682/682 [==============================] - 0s 410us/step - loss: 0.5125 - accuracy: 0.8284 - val_loss: 0.4986 - val_accuracy: 0.8089
Epoch 13/20
682/682 [==============================] - 0s 711us/step - loss: 0.5360 - accuracy: 0.8138 - val_loss: 0.6554 - val_accuracy: 0.7645
Epoch 14/20
682/682 [==============================] - 0s 617us/step - loss: 0.5281 - accuracy: 0.8182 - val_loss: 0.4642 - val_accuracy: 0.8532
Epoch 15/20
682/682 [==============================] - 0s 420us/step - loss: 0.4950 - accuracy: 0.8152 - val_loss: 0.4163 - val_accuracy: 0.8635
Epoch 16/20
682/682 [==============================] - 0s 528us/step - loss: 0.4814 - accuracy: 0.8314 - val_loss: 0.6286 - val_accuracy: 0.7952
Epoch 17/20
682/682 [==============================] - 0s 428us/step - loss: 0.4930 - accuracy: 0.8211 - val_loss: 0.4180 - val_accuracy: 0.8464
Epoch 18/20
682/682 [==============================] - 0s 412us/step - loss: 0.4658 - accuracy: 0.8372 - val_loss: 0.4485 - val_accuracy: 0.8259
Epoch 19/20
682/682 [==============================] - 0s 533us/step - loss: 0.4824 - accuracy: 0.8211 - val_loss: 0.5368 - val_accuracy: 0.8259
Epoch 20/20
682/682 [==============================] - 0s 573us/step - loss: 0.5397 - accuracy: 0.8167 - val_loss: 0.4478 - val_accuracy: 0.8567
```

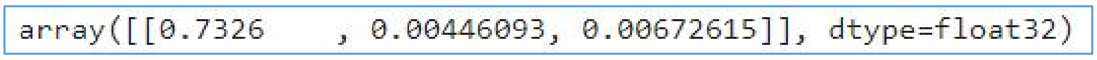
Fig 8.4: Training Epochs

An epoch is when all the training data is used at once and is defined as the total number of iterations of all the training data in one cycle for training the machine learning model. Another way to define an epoch is the number of passes a training dataset takes around an algorithm



```
293/293 [==============================] - 0s 61us/step
accuracy: 85.67%
```

Fig 8.5: Accuracy Score



```
array([[0.7326    , 0.00446093, 0.00672615]], dtype=float32)
```

Fig 8.6: Prediction

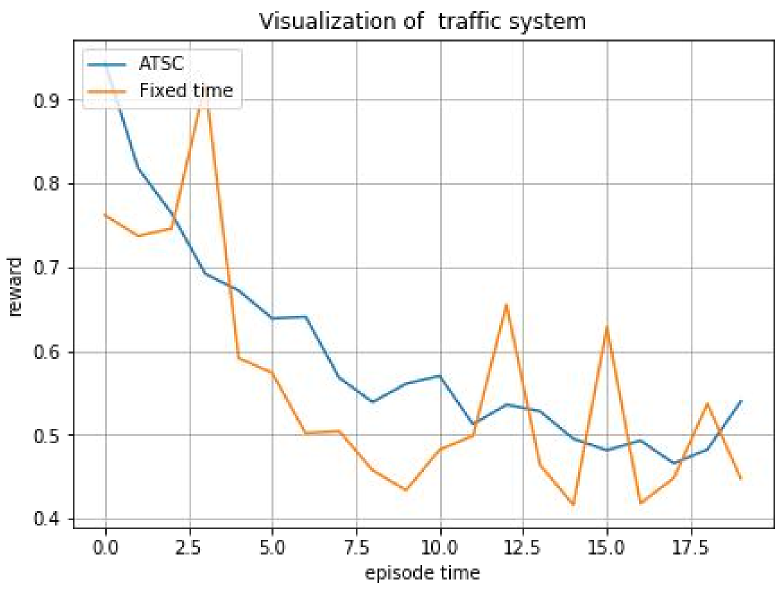


Fig 8.7: Visualization of Traffic System

Devices typically used include speed bumps, barricades to block streets, turn prohibitions, stop signs, and raised pavement markers. Traffic restraint also includes programs to foster bicycle and pedestrian travel

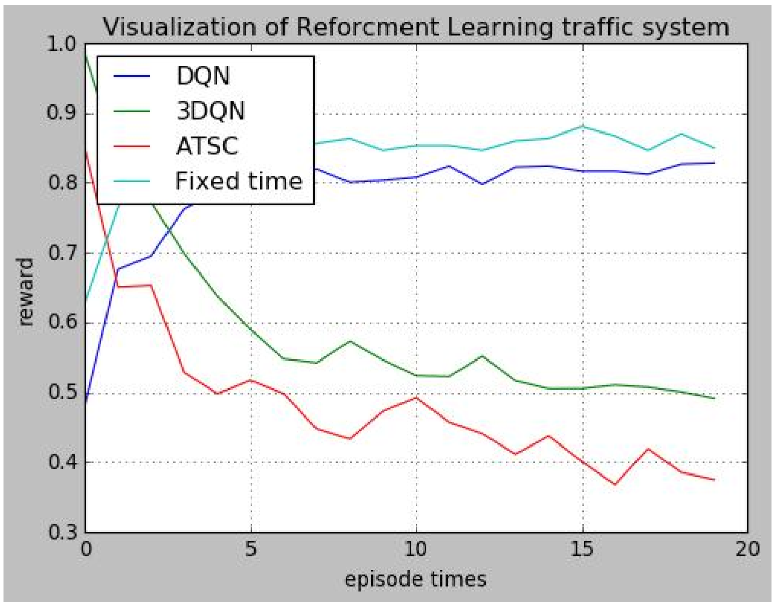


Fig 8.8 (a): Visualization of Reinforcement Learning Traffic System

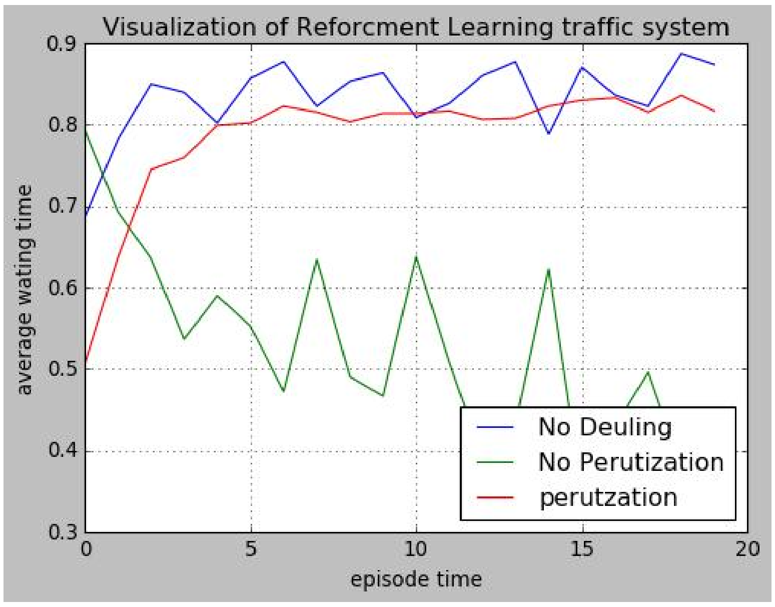


Fig 8.8 (b): Visualization of Reinforcement Learning Traffic System

Recently, reinforcement learning (RL) has been widely used for traffic light control. Reinforcement learning defines traffic light control as a Markov decision process (MDP) and learns an optimal control strategy through continuous iteration with the environment.

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

With the rapid development of urbanization, some social problems, such as traffic congestion, have emerged. The real-time intelligent automatic transportation safety based on Artificial Neural Network algorithm is a promising approach to enhance transportation safety. The system can analyze transportation safety data in real-time, detect and classify safety events, predict future events, to prevent accidents and save lives. With the potential to be implemented on a large scale, real-time intelligent automatic transportation safety based on ANN algorithm can help create a safer transportation environment for everyone. Subsequently, deep learning can be combined with real-time analysis and prediction to explore the related functional application and task model.

9.2 FUTURE WORK

The system can be enhanced by integrating it with other emerging technologies such as 5G, IoT, and edge computing to improve data collection, processing, and transmission. It can be enhanced by adopting a cloud-based architecture that allows for scalable, flexible, and cost-effective deployment.

REFERENCES

[1] R. F. Berriel, A. T. Lopes, A. F. de Souza, and T. Oliveira-Santos, “Deep Learning Based Large-Scale Automatic Satellite Crosswalk Classification,” IEEE Geoscience and Remote Sensing Letters, vol. 14, pp. 1513–1517, Sept 2017.

[2] R. Guidolini, L. G. Scart, L. F. R. Jesus, V. B. Cardoso, C. Badue, and T. Oliveira-Santos, “Handling Pedestrians in Crosswalks Using Deep Neural Networks in the IARA Autonomous Car,” in 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, July 2018.

[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Advances in neural information processing systems, pp. 91–99, 2015.

[4] M. P. Philipsen, M. B. Jensen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, “Traffic Light Detection: A Learning Algorithm and Evaluations on Challenging Dataset,” in International Conference on Intelligent Transportation Systems (ITSC), 2015.

[5] N. Fairfield and C. Urmson, “Traffic Light Mapping and Detection,” in International Conference on Robotics and Automation (ICRA), 2011.

[6] D. Barnes, W. Maddern, and I. Posner, “Exploiting 3D Semantic Scene Priors for Online Traffic Light Interpretation,” in Intelligent Vehicles Symposium (IV), 2015.

[7] R. De Charette and F. Nashashibi, “Traffic Light Recognition using Image Processing Compared to Learning Processes,” in International Conference on Intelligent Robots and Systems (IROS), 2009.

[8] R. De Charette and F. Nashashibi, “Real Time Visual Traffic Lights Recognition Based on Spot Light Detection and Adaptive Traffic Lights Templates,” in Intelligent Vehicles Symposium (IV), 2009.

[9] G. Siogkas, E. Skodras, and E. Dermatas, “Traffic Lights Detection in Adverse Conditions using Color, Symmetry and Spatiotemporal Information,” in International Conference on Computer Vision Theory and Applications (VISAPP), 2012.

[10] M. Weber, P. Wolf, and J. M. Zöllner, “DeepTLR: A single Deep Convolutional Network for Detection and Classification of Traffic Lights,” in Intelligent Vehicles Symposium (IV), 2016.