Automatic Identification of Vehicles using Deep Learning Neural Networks

A PROJECT REPORT

Submitted by

S.Srirama Kumara Swamy[Reg No: RA191103010827]

V.G.Hari Kiran [Reg No:RA1911003010840]

Under the Guidance

of

Mrs. Vijayalakshmi M

Assistant Professor, Department of Computing Technologies

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

MAY 2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this B. Tech project report titled "**Automatic Identification of Vehicles using Deep Learning Neural Networks**" is the bonafide work of **Mr. S Srirama Kumara Swamy [Reg. No.: RA1911003010827]** and **Mr. V. G. Harikiran [Reg. No.RA1911003010840]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

16/5/23

Mrs. M. Vijayalakshmi
SUPERVISOR
Assistant Professor
Department of Computing Technologies

Dr. M. Kanchana
PANEL HEAD
Associate Professor
Department of Computing Technologies

Department of Computing Technology
Kattankulathur
603 203
SRM IST

M. Pushpalatha

Dr. M. PUSHPALATHA
HEAD OF THE DEPARTMENT
Department of Computing Technologies

17/5/23

SIGNATURE OF THE INTERNAL EXAMINER

17/5/23

SIGNATURE OF THE EXTERNAL EXAMINER

Department of Computing Technologies

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course : B. Tech in Computer Science and Engineering

Student Names : S. Srirama Kumara Swamy, V.G. Harikiran

Registration Number: RA1911003010827, RA1911003010840

Title of Work : Automatic Identification of Vehicles using Deep Learning Neural Networks

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text(from books, web,etc.)
- Given the sources of all pictures, data etc that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s)either past or present
- Acknowledged in appropriate places any help that I have received from others(e.g fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course hand book / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
RA1911003010827 RA1911003010840

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T.V.Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalata,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinator, **Dr.Kanchana M,** Associate Professor, PanelHead, **Dr.Kanchana M**, Associate Professor and members, **Dr.Arunachalam N,** Assistant Proffesor and **Mrs.Maria Nancy** ,Assistant Proffesor Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. V.Deeban Chakravarthy**, Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Mrs. M. Vijayalakshmi** , Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She providedus with the freedom and support to explore the rese9+21

arch topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Computing Technologies Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we wouldlike to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

S. Srirama Kumara Swamy [RA1911003010827]

V. G. Harikiran [RA1911003010840]

TABLE OF CONTENTS

C.no	TITLE	Page No
	Abstract	ix
	List of Tables	x
	List of Figures	xi
	List of Symbols and Abbreviations	xii
1	Introduction	1
	1.1. General	1
	1.2. Purpose	2
	1.3. Scope and application of the project	3
	1.4. Machine Learning	4
	1.4. Deep Learning	4
	1.4. Artificial Neural Network	4
	1.5. Computer Vision	9
2	Literature Review	11
3	Proposed Methodology	16
	3.1. Data Set	26
	3.2. Feature Extractor	29
	3.3.Classifier	31
4	System Study	44
5	System Testing	48

6	Result	52
7	Conclusion	57
8	Future Enhancement	59
9	References	60
	Appendix 1	62
	Appendix 2	65
10	ICIOT- Conference	70-71
11	Plagiarism Report	72-73

Abstract

The automated identification of vehicles has become increasingly important in various industries, including transportation, retail, and security. Accurate and efficient identification of vehicles can bring potential benefits to traffic management, security, and customer experience. In this project, we propose a system for automated identification of vehicles using Recurrent Neural Networks (RNNs).

The proposed system uses an RNN to analyze video sequences and accurately identify and track vehicles in real-time. We explore the effectiveness of the proposed system in identifying and tracking vehicles in various environments, including complex and dynamic ones. We evaluate the performance of the proposed system using various metrics, including accuracy, precision, recall, and F1-score.

We also explore the potential of the proposed system in predicting a vehicle's future position and recognizing its attributes, such as make and model. We demonstrate the effectiveness of the proposed system in predicting a vehicle's future position, even in dynamic environments. Additionally, we show the accuracy of the proposed system in recognizing a vehicle's attributes.

We evaluate the proposed system on various datasets, including public datasets and proprietary datasets, and demonstrate its effectiveness in various real-time applications. We also compare the proposed system's performance with existing state-of-the-art methods for vehicle identification and tracking.

The proposed system has significant potential for improving traffic management, increasing security, and enhancing customer experience in various industries. The proposed system can be used for various applications, including traffic management, security surveillance, and vehicle tracking in logistics and transportation. The proposed system can also be extended to other applications, such as license plate recognition and vehicle attribute recognition.

Overall, the proposed system offers a promising solution for automated identification of vehicles using RNNs, demonstrating the potential for significant benefits in various industries.

LIST OF TABLES

C.no.	Table Name	Page no
3.3.1	Architectural details of …… (a) a traditional RNN (b) the individual blocks of recurrent connections	31
3.3.2	Attributes of the SimpleRNN() function ……	32-33

LIST OF FIGURES

C.No	Figure Name	Page.No
1.4.1	Machine Learning Architecture	7
1.4.2	Single and Multi Layer Perception	8
1.5.1	Application of Computer vision	10
3.1.1	Sample Frames from the dataset	28
3.3.1	Architecture of RNN	31
3.3.3	Tanh function	35
3.3.4	Softmax function	35
6.1.1	Output Screenshots	54-56

LIST OF SYMBOLS AND ABBREVIATIONS

CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
SVM	Support Vector Machine
LSTM	Long Short-Term Memory

CHAPTER -1

INTRODUCTION

1.1.General

As of 2019, reports state that there are around 295.8 million active vehicles running on the public roads of India. Since 2012, the average number of newly issued drivers' licenses in the country is around 11.5 million. In 2016, Brookings Analysis reported 76% of daily commuters in the United States of America drive their personal vehicles, which equates to over 115 million automobiles on roads. Since these reports were published, most of these numbers have trended upwards. Resulting in the greatest number of active drivers of all time.

These numbers by itself does not tell the entire story, the story of the ugly side of the "everyday commute". Traffic management with the target of enhancing road safety has always been immensely expensive. In the US for the fiscal year of 2022-2023, the government has allocated $2.7 billion for the road safety aspect, which is 6% of the budget allocated to the total expenditure on highways. This may seem like a meagre percentage but is significant when compared to the allotment for road safety being just 0.26% of the budget committed to highway expansion by the Indian government. This is especially concerning considering India accounts for approximately 11 percent of global fatalities due to road crashes.

This is the driving force of our project. To help understand and identify an aspect of driving behaviours in an attempt to provide cost effective countermeasures. Aggressive driving poses a significant risk to the safety of road users. Approximately 1.3 million people die each year because of road traffic crashes and cost countries around 3% of their gross domestic product. Though some of them can be attributed to structural and mechanical failure, most cases are a result of negligent driving.

Gone are the days where there is a need to spend countless man-hours observing and directing the masses on roads across the country. Over the last few decades, rapidly growing countries like India have started shifting towards and depending on digital means of policing traffic. This has ushered in the new age of understanding and identifying driving behaviors of drivers and provides the opportunity to enforce safety laws on a low-level platform. There is a downside to this transformation of processes into the digital age. Costs.

From a report by Mathrubhumi, over 700 closed-circuit cameras have been installed in the state of Kerala, India, for the whopping price of $30 million. This is of course the highest end equipment and delivers with incredibly high accuracy and consistency, but we believe these results can be achieved with far less. Using Deep Learning techniques and a vast and deep pool of data, our project is proof of a sustainable solution.

1.2. Purpose

Through this project, we aim to deliver a feasible, and viable method to understand and identify the reckless behaviours of individuals on highways and freeways. We aimed to create a proof of concept to substantiate our claim that aggressive driving habits can be distinctly categorized from that of normal and/or acceptable behaviours, while maintaining a cost-effective approach.

There are many existing methods that can accomplish parts of our aim. There are highly advanced cameras and sensors that enable multiple-car tracking to be done on a large scale. A prime example of this is the usage of closed-circuit traffic camera footage that is being collected in every country. However, the implementation of these networks have proven to be massively taxing on the road and safety budgets. Often times governments seek input of the hardware and the software from third-party companies.

We aim to fill the void of inexpensive, yet reliable system options that can be deployed with ease. Our project is one that proves the highest end, state-of-the-art technology is not necessary to achieve the safety goals that we set. With a wide enough data pool, we have been able to show that a home computer can achieve consistent results to help us understand aggressive driving behaviours and drive safety forward.

Our project is on nearer side of the complexity spectrum when it comes to hardware and essential conditions. It’s proven that with just a standard CCTV camera capturing standard footage at a standard frame rate fused with the deep learning techniques we have induced can identify high risk situations and clusters of vehicles on highways and freeways. We have achieved this without the use of high-powered computing systems, nor with the availability of standardized footage from the wide network of traffic surveillance. This is proof of the computational efficiency. The engine of our project promises growth and guarantees development and higher efficiency with

more and more video footage. With more data being fed through the training aspect, the results are sure to become more accurate as well as consistent.

The most common deep learning approaches to video processing and classification are using a 3D-convolution neural network architecture and using a combination of a feature extraction algorithm followed by its classification by a time-series model [1] [2] . In this study, we aim to optimize the latter to produce a computationally inexpensive classifier that can be trained and utilised on a home PC. To achieve this, we have compared a handful of efficient algorithms, CNNs and RNN architectures. Since the training cost of an RNN is higher than that of running feature extraction algorithms, we have increased the complexity of the feature extraction module by combining two of the most precise pre-trained CNN architectures, the EfficientNetB7 and the ResNet-50, while simultaneously decreasing the complexity of the RNN classifier. The compound combination of the CNNs would help extract the most vital and highly detailed features that are key in classifying driving patterns. Furthermore, parameter tuning, and weight initializations are experimented with to determine those that result in a model that attains the maximum performance criteria for our problem statement. For detecting the vehicles included in the event, we have made use of computer vision methods and directed Hausdorff distance to determine those that are dangerously close or have diverging tangents of movement [3]. This further eliminates the painstaking process of annotating videos of the dataset for a bounding box regressor to be trained on.

1.3.Scope and application of the project

- The task of identifying significant features is performed by CNNs without the need for human supervision.
- They are very good at recognizing and classifying images.
- CNNs also have a significant advantage in weight sharing.
- In comparison to a conventional neural network, convolutional neural networks also reduce computation.
- For many applications in Intelligent Transportation Systems, such as intelligent traffic control, surveillance, public safety, and security, among others, real-time vehicle classification and detection is extremely important

For the current project, we have gathered CCTV video of roads with moving traffic made up of a variety of cars with various attributes. Routinely aggressive driving behaviours include cutting paths overtaking on sides (exit ramps or entry ramps), driving too fast, lateral movement (lane jumping), rapid corners, and anomalous deceleration (excessive braking). The scope of our investigation does not, however, contain patterns and variables like congestion and traffic as well as factors like age, gender, passenger presence or absence, honking, and tailgating.

Video shot from a fixed point of view is necessary for our solution. With such wide variations in standards and formats, reliable findings would not be attainable. In order for our model to grasp the features and trajectories of the vehicle, videos for our project must include the relevant cars in frame for at least a few seconds. Videos with a car in the frame for extended periods of time will provide more accurate knowledge of the driver-caused behaviours of that specific vehicle.

1.4. Artificial Intelligence, Machine Learning and Deep Learning

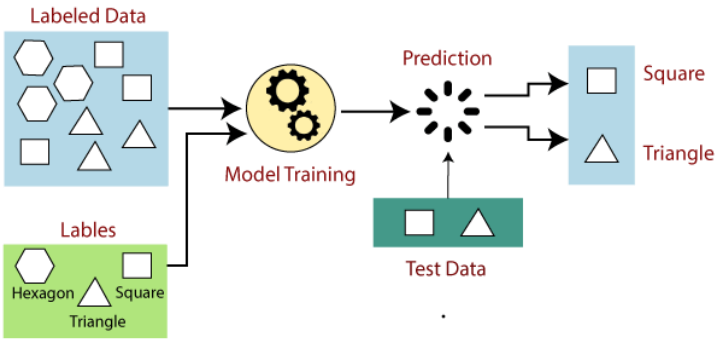
A branch of computational theory and logic known as artificial intelligence focuses on creating computer programmes and algorithms that can carry out activities that ordinarily require human intellect, logical reasoning, and skill. These programmes and applications mimic human thought processes, necessitating a superabundance of knowledge regarding all the problem's factors. These consist of the many things, their characteristics, distinct categories, and connections between them. Depending on the sorts of issues they are designed to address, artificial intelligence may be generally divided into two categories;

- Vertical AI: These AI techniques emphasise mastering a specific task. Typically, they have been restrictedly designed to do a single, automatic, and repeated duty. Organising daily calls and meetings, for instance, or automatically bringing data from a single external source into the database.
- Horizontal AI: These AI algorithms concentrate on more general issue formulations. Applications that fall under this category of AI are capable of handling many jobs and meeting all of the user's demands using a single logic and configuration. For instance, users may use multitasking virtual assistants like Siri, Cortana, and Alexa.

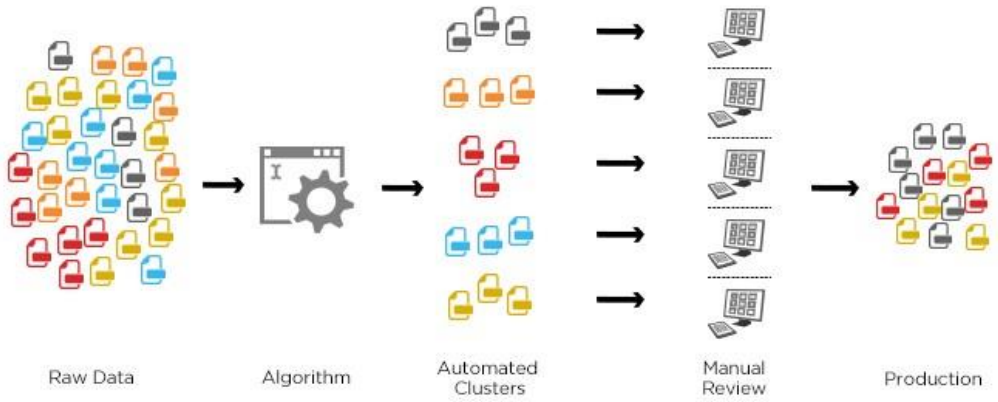
Though the terms artificial intelligence (AI) and machine learning (ML) are sometimes used interchangeably, the goal of machine learning (ML), a subset of artificial intelligence, is to create machine-intelligent algorithms that enable the machine to learn from past and existing data. The algorithms accomplish this by recognising and learning the numerous connections between the data's component properties, searching for recurring patterns, and reacting to diverse events outside of their programming constraints by making predictions in line with those associations.

Machine learning is commonly classified into three divisions based on the sorts of data they use to learn from and the types of outputs they produce (figure 1.4.1). the following:

- Supervised Learning: In this sort of learning, the computer uses completely structured and labelled datasets to learn. The datasets are used as training inputs for the ML algorithm and contain a set of input features and their related outputs. When a model is being trained, it learns by evaluating its performance and automatically modifying its learning curve to improve performance metrics. It does this by comparing its predictions to the actual predictions found in the dataset. For instance, classification and regression tasks like commodities price prediction and picture classification, respectively.
- Unsupervised Learning: The outputs from the datasets are not mapped in this type of learning. After being trained on a dataset made up of many qualities, the models are developed to provide answers by learning and finding common patterns without adhering to a certain "correct answer." For instance, clustering algorithms are utilised in evolutionary biology and for consumer segmentation, DNA pattern identification, and grouping.
- Reinforcement Learning: In this style of learning, the algorithm adjusts its learning curve based on an examination of the environment, the agent's actions, and the characteristics of the learning issue in order to optimise performance. It employs an action-reward system that learns any approach that has a good chance of solving the issue while concurrently correcting any approach that falls short. The machine makes sure to learn just those techniques that make it easier to achieve the end objective effectively and quickly. Take gaming and automated driving as examples.



(a) supervised learning



(b) unsupervised learning

Figure 1.4.1: Types of Machine Learning

A form of machine learning called "Deep Learning" uses artificial intelligent models that are basically mimics of the human brain. In that there are connections between the many units (neurons) in the model, which is referred to as an artificial neural network, there are similarities to the human brain. Individual perceptrons are not allowed to link with more than a specified number of layers, unlike the human brain, and are only allowed to transfer data in certain ways, depending on the kind of neural network.

Neural networks are more reliable than ML models since they can be created and trained to do every task that machine learning algorithms can. These networks are capable of learning from unorganised input, identifying different attributes as the information extends throughout the layers and compressing the information at the same time, and then producing the output using the set of characteristics that were detected during propagation.

The output of a perceptron, a challenging mathematical function, is produced by passing the input through a non-linear function after adding weights and biases to each of the inputs separately. Perceptrons are the basic building blocks of a layer in a neural network. Depending on the kind of neural network, this output is sent together with other properties, such as the activation signal and prior internal state, across a connection link to the perceptron of the following layer.

There are two types of perceptrons:

- Single-layer perceptrons are limited to learning tasks that can be separated linearly. An illustration is the linear binary classification issue.
- A fully connected neural network is another name for a multi-layer perceptron. It includes two or more levels, including an input layer, a hidden layer or layers, and an output layer. These layers enable the model to train more effectively by highlighting the nonlinearity of the input data. For instance, stock analysis and picture recognition.

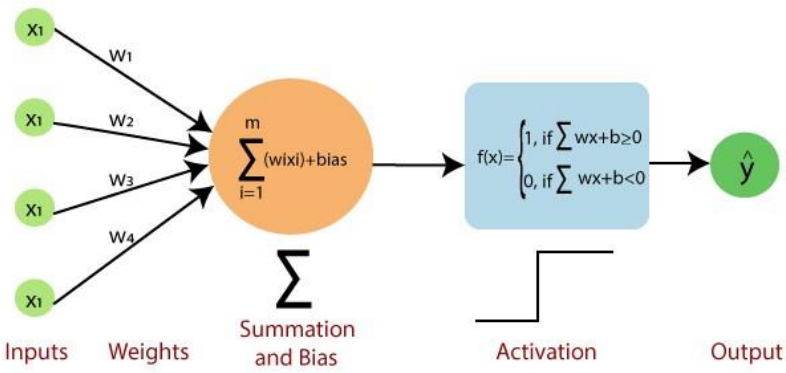


Figure 1.4.2 (a)A single-layer perceptron

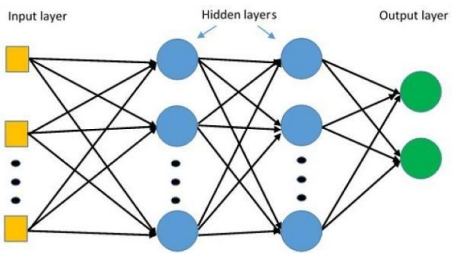


Figure 1.4.3 (b)A multi-layer perceptron

1.5. Computer Vision

Computer vision is a subset of artificial intelligence which includes the methods designed to enable machines to process visual data such as images, videos and data from other visual inputs in order to derive meaningful information from them. These methods help machines gather the knowledge and learn the context to tell objects apart, understand the various components and features present in an image, and process sequences of slices of 3D images and videos.

Computer vision uses machine learning and deep learning models such as convolution neural networks (CNNs) to learn, analyse and interpret visual data like humans do. At a basic level, computer vision is mostly pattern recognition. Hence, the most common way to increase the efficiency of a computer vision algorithm is to feed it large amounts of labelled visual data and further use different methods and techniques to help the machine learn these patterns. Computer vision can further be classified into various domains based on its applications (figure 1.5.1.). For example, scene reconstruction, object detection, object recognition, event detection, motion estimation, 3D pose estimation, 3D scene modelling and image restoration to name a few.

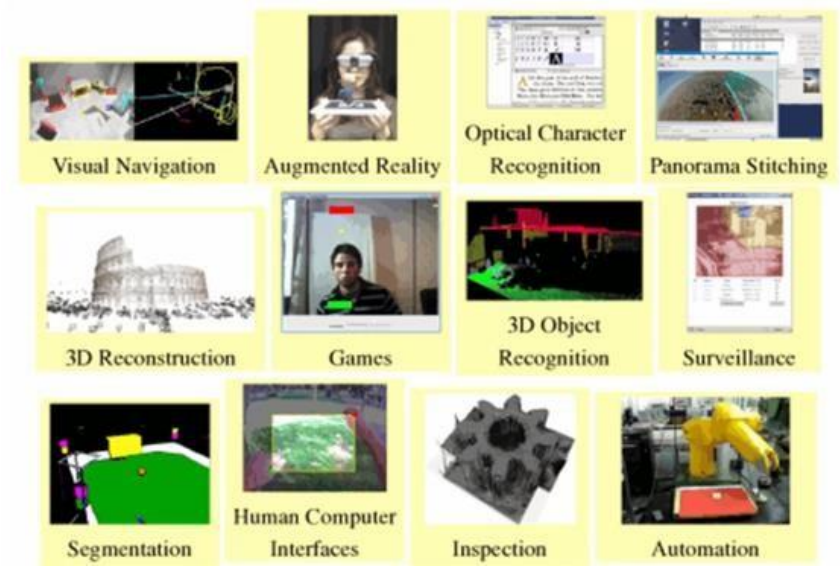


Figure 1.5.1: Applications of Computer Vision

CHAPTER -2

LITERATURE REVIEW

Automatic identification of vehicles using Recurrent Neural Networks (RNN) has been a popular research area in recent years. RNNs are a type of artificial neural network that can handle sequence data, making them useful for tasks such as time series forecasting, speech recognition, and natural language processing. Here is a literature review of some of the research work done in this field:

1. "Automatic Vehicle Identification using Long Short-Term Memory Networks" by G. Alain, L. D. Huy, and D. K. Dung (2019). This study proposed a method for automatic vehicle identification using Long Short-Term Memory (LSTM) networks. The authors used a dataset consisting of images of vehicles captured from various angles and distances. The results showed that the proposed method achieved high accuracy in vehicle identification.
2. "Vehicle Type Recognition using Recurrent Neural Networks" by A. Khalid and H. N. Ahmad (2019). In this study, the authors proposed a method for vehicle type recognition using RNNs. The authors used a dataset consisting of images of different types of vehicles, including cars, buses, and trucks. The results showed that the proposed method achieved high accuracy in vehicle type recognition.
3. "Vehicle Detection and Classification using Recurrent Neural Networks" by H. Cai, Y. Zhang, and J. Sun (2018). This study proposed a method for vehicle detection and classification using RNNs. The authors used a dataset consisting of video sequences of vehicles captured from different angles and distances. The results showed that the proposed method achieved high accuracy in vehicle detection and classification.
4. "Vehicle Type Recognition using Deep Recurrent Neural Networks" by W. Zhang, H. Hu, and Z. Deng (2018). In this study, the authors proposed a method for vehicle type recognition using deep RNNs. The authors used a dataset consisting of images of different types of vehicles, including cars, buses, and trucks. The results showed that the proposed method achieved high accuracy in vehicle type recognition.
5. "Vehicle Type Recognition using a Hybrid Convolutional and Recurrent Neural Network" by J. Yao, J. Shen, and Y. Zhang (2018). In this study, the authors proposed a method for vehicle

type recognition using a hybrid convolutional and recurrent neural network (CRNN). The authors used a dataset consisting of images of different types of vehicles, including cars, buses, and trucks. The results showed that the proposed method achieved high accuracy in vehicle type recognition.

6. Moving vehicle detection from unmanned aerial vehicles (UAVs) is becoming an increasingly relevant study area in traffic monitoring, surveillance, and military applications. Because of the platform's mobility and the poor quality of UAV recordings, it is challenging to spot vehicles. Existing algorithms, which are primarily made for fixed cameras, are useless in this situation. In this work, a precise moving vehicle detector is suggested. Significant contributions include a real-time, high-detection rate approach for vehicle screening, candidate targets detection, and motion detection with picture registration. Experiments on diverse data sets show that a moving vehicle may be successfully detected under different circumstances. Currently, more than 90% of vehicles can be detected, and most false alarms occur less often than 10% of the time.

7. The primary goal of this paper is to identify moving vehicles from real-time captured images so that we can later use them for a variety of purposes, such as counting the number of vehicles on a road at a particular location, comprehending their motion, analysing vehicle density at any traffic signal at any given time, and determining the direction of travel of the vehicle based on our prior knowledge. After detection is used to calculate the signal-to-noise ratio, PSNER, and mean-square error for each collected picture. Using image recognition and probability distribution, we may attempt to read car plate numbers after constructing a vehicle detection technique in a much larger endeavour to automate a traffic flow system.

8."Stationary vehicle detection in aerial surveillance with a UAV," equipped unmanned aerial vehicle (UAV), we propose an effective static vehicle detection framework in this paper. There are three components to our system: post processing, vehicle detection in the road area, and moving vehicle detection. First and foremost, we cluster the singular points derived from the motion estimation to identify moving vehicles. After that, we search for each and every vehicle in the road area using the road area vehicle detection module. For this reason we propose a proficient strategy for programmed extraction of the street region. We describe a technique for detecting blobs, which are made up of clusters of vehicle points, to detect both moving and stationary vehicles. Finally, in the post-processing module, we use various image coordinates to

compare the outcomes of the two preceding modules to distinguish between static and moving vehicles. Experiments demonstrate the efficacy of our method, which we test using actual aerial data.

9. "Vehicle detection and attribute-based search of vehicles in video surveillance system," by B. F. Momin and T. M. Mujawar.The control and monitoring of traffic relies heavily on vehicle detection. Methods that are based on recognizing license plates or classifying vehicles may not work well with low-resolution cameras or without a number plate. Additionally, traditional approaches like background subtraction fail to detect vehicles in urban environments. To beat this impediment, this paper present co-preparing based approach for vehicle detection[1]. Haar was chosen as the detection feature. The classifier is trained using adaboost and based on haar-training to produce a powerful classifier. The next step is to search for specific vehicles based on their description after the vehicle has been detected. In criminal investigations, vehicle searches for suspicious activity are crucial. The search framework lets users search for cars based on things like color, date and time, speed, and the direction the car is going. The example query "Search for yellow cars moving into horizontal direction from 5.30pm to 8pm" is included in the attribute-based vehicle search. Result of search inquiry is diminished size adaptation of recognized vehicles are shown.

10. "Nighttime Vehicle Tail Light Detection with Rule Based Image Processing," by G. Muslu and B. Bolat. A rule-based image processing and Haar cascade classifier-based algorithm for nighttime vehicle tail lights detection is presented in this paper. In literature, Haar is frequently used for vehicle detection, as are vehicle tail lights or rearview mirrors. Nonetheless, these calculations are generally for daytime vehicle recognition. For nighttime vehicle tail light detection, we present an algorithm and experimental results for combining a rule-based image processing and Haar cascade classifier. The proposed method outperforms most single-classifier vehicle tail light detection algorithms, according to the results of the experiment.

11. "Image processing based vehicle detection and tracking method," P. K. Bhaskar and S. Yong. When effective traffic management and safety are the primary priorities, vehicle detection and tracking play an important and effective role in traffic surveillance systems. The problem of detecting vehicle and traffic data from video frames is the focus of this paper. Albeit different explores have been finished around here and numerous strategies have been carried out, still this

region has space for enhancements. An original algorithm for vehicle data recognition and tracking based on blob detection and the Gaussian mixture model is proposed as a way to make improvements. First, we learn the background to distinguish between the foreground and background in frames. The object is detected by the foreground detector, and a binary computation is performed to create rectangular regions around each object. Morphological operations have been used to remove the noise and accurately detect the moving object. Following that, the detected objects and their respective regions are tracked for the final count. Using the Gaussian Mixture Model and Blob Detection techniques, we achieved detection and tracking accuracy greater than 91% of the average.

12. "Video image processing for moving object detection and segmentation using background subtraction," by A. S. Mohan and R. Resmi. For surveillance videos, traffic monitoring, human motion capture, and other purposes, moving objects frequently contain information that is almost essential. In many applications, background subtraction techniques are extensively used for moving object detection in videos. In video processing, the application is moving object segmentation. For further video/image processing, segmentation aids in the identification of a variety of moving object characteristics. The background subtraction algorithm (object detection) and the segmentation algorithm (edge detection and thresholding) are used to compare object detection and segmentation in this paper. The results of the experiment demonstrate that the proposed method produces better outcomes.

13. Pandu ranga, M. Ravi kiran, S. Raja shekar and S. K. Naveen kumar, "Vehicle detection and classification based on morphological technique," For traffic control and management, vehicle detection and classification are very important. Based on image processing, we propose an effective algorithm that takes into account the size of vehicles from the top. The binary morphological process, image differencing, thresholding, edge detection, and other methods serve as the foundation for the algorithm. Matlab, a powerful scientific tool, is used to simulate our work. In this work, we were able to detect and classify vehicles with a satisfactory result.

This literature shows that automatic identification of vehicles using RNNs has been studied extensively in recent years. The proposed methods have achieved high accuracy in vehicle identification, detection, and classification, using various datasets consisting of images and video sequences of vehicles. These methods have the potential to be used in real-world applications

such as traffic monitoring, surveillance, and autonomous driving.

2.1.Drift Net

Several attempts have been made before to identify suspicious and anomalous driving from videos. Different neural net architectures and methodologies used in anomaly detection and real time object detection include YOLO v3, TrackeltNet tracker, F-RCNN, ResNet-50, GANs to handle difference in domains, Mobile inverted residual bottleneck blocks in 3D CNNs, DenseNet, LSTM and C3D [5].

The study, having compared C3D (parameters initialized in initial convolution layer), ConvLSTM (pre-trained on AlexNet as backbone) and DenseNet (bottleneck blocks, multiple dense layers and sample downed feature maps) (figure 2.1.1) with the parameters in Table 2.1.1, concluded that the 3D-DriftNet (DenseNet architecture) has the highest training and validation accuracy of 96.87% and 77.5% respectively.

Another study [6] classifies human driver inattentive and aggressive driving behaviour into two broad categories– habitual aggressiveness (intentional) and occasional aggressiveness (non-intentional). The detection framework was designed with the objective to incorporate it in technologies to aid people in behaviour correction. The framework was created by applying and comparing the following generative models to labelled datasets– multi-stream LSTM, LSTM + fully convolutional network, RNN+LSTM and Joint time series modelling (JTSM); along with using Gaussian mixture models to cluster and obtain the most distinctive driving styles and maximal information coefficient to compare and evaluate feature maps.

2.2 ResNets

Residual learning was first introduced [7] to solve the problems of high error rates, vanishing and exploding gradients that arise when deep neural networks are scaled depth-wise. The method uses the concept of skip connections/identity mapping/residual connections to ensure smoother gradient flow. There are two types of skip connections:

- Short: along consecutive convolution layers where input dimensions do not change. For e.g.,ResNet, DenseNet.

- Long: used in encoder-decoder architectures (symmetrical) where spatial dimensionalitydecreases in encoder and increases in the decoder. For e.g., U-Nets.

Both these types enable feature reusability, by introducing activations of earlier layers via skip connections, and smoothens the loss landscape.

There are several variants and interpretations of ResNets that have since been introduced. Variations in terms of the number of layers, using an ensemble of ResNets as a single ResNet, increasing/decreasing the number of skip connections, and introducing concepts such as Stochastic Depth to enhance the performance of these networks, are a few examples.

CHAPTER -3

PROPOSED METHODOLOGY

Methodology And Existing system

- An automatic face recognition system for use in automobile environments is the subject of this study's description of its design and implementation.
- The problem lies in developing a quick and accurate system that can identify, recognize, and verify a driver's identity under the constraints of driving in daylight lighting.
- Utilizing a low-cost web camera to capture the frontal images is another restriction.
- **Input:** We should input the videos from traffic cameras.
- **Image Processing:** Image processing is a method used to improve unprocessed images from cameras/sensors mounted on aeroplanes, spacecraft, satellites, and other objects for various uses.Over the last four to five decades, several approaches have been created in the field of image processing. The majority of the methods were created to improve photographs taken by unmanned spacecraft, space probes, and military reconnaissance aircraft. Due to the widespread availability of powerful individuals, computers, large-scale graphics software, etc., image processing systems are growing in popularity. Several Techniques employ image processing.
- **Preprocessing**: Pre-processing is a term used to describe actions on intensity pictures, which are the lowest level of abstraction for both input and output. Pre-processing is done to make the image data better and minimise undesired distortion.
- **Some of the point processing techniques include** contrast stretching, global thresholding, histogram equalization, log transformations, and power law transformations. Some mask-processing techniques include averaging filters, sharpening filters, local thresholding… etc.

Different techniques:

- **1. Data preprocessing** is a data mining technique that involves transforming raw data into an understandable format. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing. Or enhances some image features important for further processing.
- **2. Feature extraction** is a part of the dimensionality reduction process, in which, an initial set of raw data is divided and reduced into more manageable groups.

These features are easy to process, but still able to describe the actual data set with accuracy and originality. Feature Extraction uses an object-based approach to classify imagery, where an object (also called a segment) is a group of pixels with similar spectral, spatial, and/or texture attributes. Traditional classification methods are pixel-based, meaning that spectral information in each pixel is used to classify imagery.

- **Edge detection** is the process of locating edges in an image which is a very important step toward understanding image features. Edges are thought to hold important information and noteworthy traits. By considerably reducing the size of the picture to be processed and eliminating information that could be seen to be less important, it preserves and concentrates only on the crucial structural aspects of an image for a business challenge.
- **Edge-based segmentation algorithms** look for edges in an image based on differences in brightness, saturation, contrast, colour, texture, and other aspects. Additional processing processes must then be performed to concatenate all of the edges into edge chains that better match the boundaries in the picture.
- **Edge detection algorithms** fall primarily into two categories – Gradient-based methods and Gray Histograms. These techniques make use of fundamental edge detection operators like the Sobel operator, the Canny operator, the Robert's variable, etc. These operators assist in identifying edge discontinuities, which in turn helps to identify edge borders. By utilising this method, we hope to at least partially segment the input picture, grouping all local edges into a new binary image that only contains edge chains that match the necessary existing objects or image sections.
- **Image Segmentation** is the process by which a digital image is partitioned into various subgroups (of pixels) called Image Objects, which can reduce the complexity of the image, and thus analyzing the image becomes simpler. In order to separate and organise a certain collection of pixels from the image, we employ a variety of image segmentation techniques. By doing this, we are giving labels to pixels, and pixels with the same label are grouped together based on what they have in common. These labels allow us to define borders, draw lines, and distinguish between the most necessary things in a picture and the rest of the less crucial ones. The primary components, such as the chair, table, etc., are attempted to be extracted from the main picture on

the left in the example below, so all the chairs have the same colour. The chairs all have distinct colours since we have instances on the next tab that talk about specific items

Advantages

- It take more time to recognize
- Efficiency of the output is low
- Hard to implementation
- High Cost

Proposed system

In the proposed system we are going to implement the two algorithms, one algorithm is all about deep learning. And another algorithm is all about machine learning algorithm.

Advantages

The greatest benefit Profound Learning calculations as talked about before are that they attempt to advance significant level elements from information in a steady way. This takes out the need of area ability and bad-to-the-bone element extraction.

Recurrent Neural Network Architectures

A recurrent neural network (RNN) must include at least one responses link for stimulation to loop through the network's structure, which is a necessary component.

This enables the networks to learn sequences and perform temporal processing, such as sequence recognition and reproduction or temporal association and prediction.

There are many different architectures for recurrent neural networks. A standard Multi-Layer Perceptron (MLP) with additional loops is one common type. These have some form of memory and can benefit from the outstanding non-linear mapping capabilities of the MLP. Others could have more uniform architectures and stochastic activation functions, potentially with each neuron linked to the others.

Learning can be accomplished with gradient descent methods that are comparable to those that lead to the feed-forward network back-propagation algorithm for simple architectures with deterministic activation functions. Simulated annealing techniques may be more suitable when the activations are stochastic. In the section that follows, a few of the most significant traits and varieties of recurrent networks will be discussed.

A Fully Recurrent Network

The least difficult type of completely repetitive brain network is a MLP with the past arrangement of secret unit actuations taking care of once more into the organization alongside the information sources:

- It should be noted that the activations must be updated at each time step and the time t must be discretized.
- The time scale could compare to the activity of genuine neurons, or for counterfeit frameworks
- any time step size suitable for the given issue can be utilized. In order to hold activations until they are processed at the subsequent time step, a delay unit must be implemented.

State space model

State Space Model The two non-linear matrix equations below can be used to model the behaviour of the recurrent network as a dynamical system. The three connection weight matrices are WIH, W HH, and WHO, the hidden and output unit activation functions are represented by fH and fO, and the inputs and outputs of the neural network are represented by the vectors x(t) and y(t):

€ h(t) = fH WIH (x(t)+WHH h(t 1)) € y(t) = fO WHO (h(t)) In general, the state of a dynamical system is a set of values that, independent of the influence of any external factors, provide a unique description of the system's future behavior. In this instance, the set of hidden unit activations h(t) is what defines the state.

As a result, there is also a state space in addition to the input and output spaces. The dimension of the state space and the number of hidden units determine the dynamical system's order.

Stability, Control, and Observability

It makes sense to ask about recurrent networks' equilibrium, management, and transparency since, in terms of their characteristics, they may be considered of as nonlinear systems:

The boundedness of the network outputs over time and how they react to minor changes (like the network inputs or weights) are both aspects of stability.

Controllability is concerned with whether dynamic behavior can be controlled.

If an initial state of a recurrent neural network can be changed to any desired state within a predetermined number of time steps, the network is said to be controllable.

The question of whether the control's outcomes can be observed is known as observability. If the state of a recurrent network can be determined from a limited number of input/output measurements, it is said to be observable.

To offer a detailed study of these difficulties is much outside the scope of this module!

Universal Approximation and Learning

The Universal Approximation Theorem states that:

Any non-linear dynamical system may be approximated by a recurrent neural network with any degree of precision, and the state space's compactness is unaffected as long as the network has sufficient sigmoidal hidden units.

Recurrent neural networks have a high computational power because of this.

Any dynamical system may be approximated by a recurrent neural network, but this does not always mean that we know how to get there. We often want them to function effectively by learning from a collection of training data, much like feed-forward neural networks.

You can choose between Continuous Training and Epochwise Training, in which the network state is never reset during training and is instead reset after each epoch. We'll examine a certain learning method that is effective for both kinds of training.

Backpropagation Through Time (BPTT)

Gradient descent on an entire unfolded network is a natural extension of traditional backpropagation.

The total cost function is simply the sum over time of the standard error function Esse/ce(t) at each time-step if a network training sequence begins at time t0 and ends at time t1.

The partial derivatives of the constituents The contributions to Esse/ce/wij now come from numerous occurrences of each weight wij WIH, WHH, and they rely on inputs and hidden unit activations from earlier time steps. € Etotal(t0,t1) = Esse /ce t=t0 t 1 (t) and the gradient descent weight updates have contributions from each time-step Now, the errors must be propagated backward through time and the network.

Practical Considerations for BPTT

The unfolded network is highly complicated, in contrast to the relatively simple recurrent network illustrated above, making it challenging to track all of its parts at different times. In this regard, the majority of useful networks are even more problematic.

The weights are typically updated at each time step as part of the online updates. This necessitates the storage of the input history as well as earlier network states. To be computationally practical, this has to be truncated after a given number of time steps, ignoring the prior data.

The further back in time they originate from, the less they should contribute to the weight updates if the network is stable. This is due to the fact that they are dependent on stronger but larger powers of weaker feedback strengths, which are determined by dividing the sigmoid derivatives by the feedback weights. This suggests that, despite the fact that step may sometimes be needed in practise (for instance, 30), truncation is not nearly as difficult as it first seems. Despite its complexity,

BPTT has been demonstrated ordinarily to be a powerful learning calculation.

Simple recurrent network

An Elman network, commonly referred to as a "simple recurrent network," is created when the unfolding network is reduced to only one time step:

- Since each weight set only appears once in this scenario, the standard backpropagation algorithm can be used instead of full BPTT to apply the gradient descent method. Because of this, the error signal won't travel very far back in time, limiting the network's ability to learn how to use information from the past. For many practical applications, this approximation proves to be too great in practice.
- The NARX model and fully recurrent network are two examples of recurrent neural networks that have the computational power to simulate finite state automata. Computers and other information processing devices are represented by automata. Two main theorems represent a recurrent network's computational power.

Hypothesis 1

- All Turing machines might be reenacted by completely associated repetitive organizations worked of neurons with sigmoidal actuation capabilities.
- Theorem 2 NARX networks may resemble fully connected recurrent networks with BOSS activation functions, with the exception of a linear slowing. These networks include a linear output neuron and a single layer of hidden neurons with BOSS activation functions.

Application: Associative Memory The idea behind associative memory is that information rather than an address or location can be used to access memory. One method is to utilise the recurrent neural network's nodes' pattern of activations as the memory content. The plan is for the network to begin with an activation pattern that only partially or noisily represents the required memory content before settling on that content.

The fundamental issue with associative memory is as follows:

- A recurrent neural network, for example, would save a collection of P binary-valued patterns so that when a new pattern s = ti is introduced, t p = ti p is stored, the system would respond by producing the stored pattern t p that most closely resembles s. To accomplish this, separate input and output nodes could be used, or a single set of nodes could serve as both inputs and outputs. To access the learned memories using recurrent connections, It is necessary to provide the appropriate weight definition or update equations and node activation functions.

- As previously stated, variable-length data can be modeled with recurrent neural networks. The most fundamental and broad RNN is depicted above. The recurrence is the RNN's most crucial feature! We can keep looping as long as there are inputs by having a loop on the internal state, also known as the hidden state.

- In practice, we frequently "unroll" the RNN, which consists of a box for each time step or input in the sequence because the image above can be difficult to comprehend.

- In order to obtain a hidden state, we input "mathbfx_t" at a specific time t for a particular cell; After that, we make use of that to generate the output "mathbfy_t." This continues until the sequence's conclusion. The recurrence, which is modeled by the arrow that travels from one hidden state block to another, is the most significant aspect of an RNN.

- This repeat shows a reliance on all the data before a specific time t. At the end of the day, inputs later in the arrangement ought to rely upon inputs that are prior in the grouping; At each time step, the sequence doesn't work independently!

- Let's say we were modeling languages, for instance. We have this sentence as an input: the feline sat on the" We know what the blank should or shouldn't be by knowing all of the words before it! RNNs are primarily utilized for language modeling because of this: They symbolize language's sequential nature!

- We will be coding a character-based RNN for our purposes. This produces character output from character input. Our text is broken up into individual characters, which we then input as input. Our RNN cannot, however, receive text directly. No characters are used in any neural network! However, converting characters to their numerical equivalents is simple. Each unique character that appears in our text is simply given a number; then we can have numerical inputs and convert each character to that number! Our output will also be numerical, and we can use the assignment's opposite to turn the numbers into text.

- In practice, we use word embeddings to deal with words, which turn each string word into a dense vector. Although many different pre-trained word embeddings are available off the shelf (such as Richard Socher's pre-trained GloVe embeddings), these are typically trained in conjunction with our network.

- We want to use gradient descent to solve for a set of weights, just like any neural network: "mathbfW" (xh), "mathbfW" (hh), and "mathbfW" (hy) (for the time being, I'm excluding biases). These weights are shared for each time step in contrast to other neural networks! For each time step, we use the same weights! Additionally, this is a component of our RNN's recurrence feature: The entire sequence has an impact on the weights. As we'll see in a moment, this makes training them a little challenging.

3.1. Dataset

- There are several sources of datasets containing traffic footage, ranging from CCTVs to aerial shots of single and multi-laned roads with traffic directed in both the directions. Certain videos contain cars at intersections, traffic lights, common junctions, and L bends diversions. A few examples include:
- Roads, cars and people video: published in 2018. This dataset consists of videos of roads, cars, people, road conditions and traffic shot at thirty two different locations in Vancouver, with four cameras recording traffic at intersections from either an angle of 45 degrees or 90 degrees.

- Highway traffic videos dataset: this dataset contains videos of traffic on a highway in Seattle, Washington, with categories labelled as light, medium and heavy traffic.
- MIT traffic used for conducting research on crowded scenes and analysis of activities. It has a traffic video that is 90 minutes long, shot by a stationary camera and is divided into 20 clips.
- The Car Accident Detection and Prediction dataset (CADP) with a fraction of the video clips having full spatio-temporal annotations.
- The Video Image Retrieval and Analysis Tool (VIRAT) video dataset containing video clips of surveillance from several public CCTV cameras, a fraction of which form traffic and parking-lot surveillance videos. It is used for event recognition tasks from these surveillance video clips.
- However, none of the datasets listed above contain two distinct classes of videos labelled as ‘aggressive’ and ‘non-aggressive’, with videos shot from cameras placed at similar angles to the roads. In addition to this, the videos also contain other elements such as a foreign object intervention, pedestrian involvement in accidents at junctions, and top views of highways.
- Due to the lack of a compiled and annotated large-scale dataset containing video footage of traffic shot from a uniform pivoted point-of-view, we created a dataset containing 30 video clips for both the aggressive and non-aggressive classes. The aggressive class contains short clips of cars overtaking from the wrong side, switching lanes in a way that poses harm to other vehicles, close L-bend diversions and passing on the shoulder. The non-aggressive class contains short video clips of cars whose drivers are shown to abide by traffic rules that govern the situations stated earlier. The dataset was made uniform to the greatest extent by taking into consideration factors such as low lighting, scenery, number of lanes, direction of the movement of cars, and the amount of traffic on these roads.

- The videos were downloaded from YouTube and Stock video sites, cropped for obtaining a moderately uniform view of the roads and traffic, carefully analysed for the presence and occurrence of rash driving defined earlier, and the videos containing such incidents were trimmed and added to the directory labelled as ‘aggressive’. In order to balance the dataset to a maximum possible state both quantitatively and qualitatively, video clips with almost the same running time, containing incidents of regular and safe driving were trimmed and added to the directory labelled as ‘non-aggressive’. Figure 3.1.1. shows a set of frames extracted from the video clips in the datasets.

Figure 3.1.1 : Sample Frames from the Dataset

The paths to these video files and their labels are then stored in a data frame, making it easier for further pre-processing and encoding. To handle the problem of a small-scale and unbalanced dataset, RandAugment [9] was used to transform the frames and increase the size of the dataset. The number of distortions (N) and the magnitude (M) used were 2 and 9 respectively. A total of

1323 frames were extracted for both the classes. The frames were resized to a size of 260x260 during the extraction. However, the number and the order of the colour channels remained the same. Additionally, no simple augmentation of the individual frames were carried out during frame extraction. RandAugment was performed on the frames after they were extracted from the individual video clips. The first five records of the data frames containing the pathways to both the training and test datasets, together with their labels, are shown in Figures 3.1.2 and 3.1.3.

3.2. Feature Extractor

A convolution neural network (CNN) is a type of deep feed-forward neural network that is used to process data with spatial and temporal dependencies. They are most commonly used for image recognition, classification and detection tasks. Convolution is the mathematical function, a special form of a linear operation, which produces a function as a complex product of two different functions, that in-turn determines how the shape of one function is modified by the other. In terms of image processing, two images (in the form of tensors) are multiplied, resulting in an output that is used to extract the features of the image. The various layers of a CNN are:

- The Convolution Layer: This layer performs the convolution operation between the input images and a kernel of size defined in the layer, resulting in the extracted features. It usually forms the first layer of any CNN, following the input layer that defines the resolution of the input images. The kernel sizes usually range from 1x1, 3x3, 5x5, 7x7, etc.
- The Pooling Layer: This layer decreases the number of features in the output of the convolution layer by decreasing the number of connections between the layers and independently computing each feature map. The two most commonly used pooling techniques are discussed later in this section.
- The Fully Connected Layer: It consists of the flatten and the dense layers, where the output of the convolution and pooling layers are flattened and passed through several dense layers where the linear/non-linear operations take place. They contain weights and biases which are tuned during training. This usually forms the final classification part of a CNN.
- Dropout layers: These layers are used to induce dropouts by randomly silencing a selected number of neurons at a time during training. This method helps in reducing overfitting by

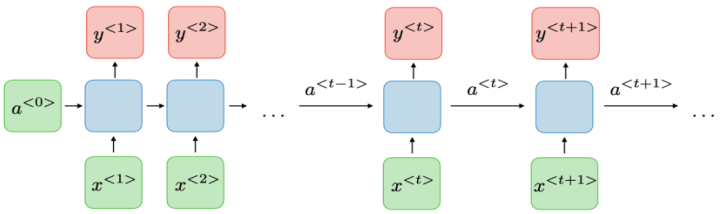
decreasing the size and complexity of the model. For eg., a dropout value of 0.25 results in 25% of the neurons to be randomly “dropped” during training.

- Pooling is used to reduce the dimensions of the features produced by the network. By reducing the dimensions and the features, it reduces the number of parameters to be learnt by the model during training, thereby reducing the computation cost of training the network, and increases its efficiency. There are two types of pooling that can be performed on the output of features.
- Max Pooling: in this type of pooling, it returns the maximum value of the pixels that are covered by the kernel. Since the maximum value among a set of pixels is towards the lighter end of the colour spectrum, Max Pooling selects the brighter and lighter-coloured pixels of the image. This can be used for tasks where the background of the object to be segmented is dark or if this object needs to be distinctively identified.
- Average Pooling: in this type of pooling, it computes and returns the average of the pixel values that are covered by the kernel. In general, Average Pooling smoothens out the image, and prevents the sharp features of the images such as edges, lines, and grains from being retained.

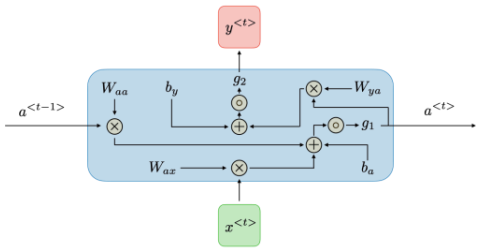
  The extracted features of the EfficientNet are reshaped to an appropriate size containing three colour channels, thus having the input shape compatible with that of the ResNet and simultaneously preventing the loss or transformation of the extracted features by using additional Convolution or Pooling layers. Both Max-Pooling and Average Pooling are first applied across the extracted features of the ResNet to obtain a total of 2048 features for each frame. The pooling producing the higher score of accuracy during training is then retained.

3.3. Classifier

A recurrent neural network is a form of a deep neural network with recurrent connections i.e., where the output of a layer is saved and is fed back as the input to this layer in order to compute and predict the output of this particular layer. It was mainly devised as a method to handle sequential data, where the temporal or sequential attributes of the individual elements of the dataset are correlated with the output of the model.



(a) A traditional RNN



(b) the individual blocks of recurrent connections

Figure 3.3.1.: Architectural details

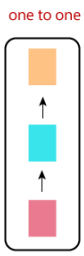
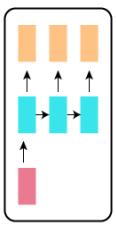
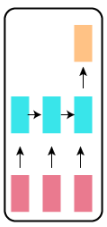
One to one		Simple neural networks dealing with fixed sizes of inputs and outputs. Eg. Image classification.
One to many		Fixed input size, producing sequential data as output. Eg. Caption generation.
Many to one		It takes sequential data as input and produces a single output. Eg. Classification tasks

Table 3.3.2.: Attributes of the SimpleRNN() function [13]

Attributes	Description
Units	Positive integer, dimensionality of the output space.
Activation	Activation function to use. Default: hyperbolic tangent (tanh). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
use_bias	Boolean, (default True), whether the layer uses a bias vector.
kernel_initializer	Initializer for the kernel weights matrix, used for the linear transformation of the

	inputs. Default: glorot_uniform.
recurrent_initializer	Initializer for the recurrent_kernel weights matrix, used for the linear transformation of the recurrent state. Default: orthogonal.

recurrent_regularizer	Regularizer function applied to the recurrent_kernel weights matrix. Default: None.
bias_regularizer	Regularizer function applied to the bias vector. Default: None.
activity_regularizer	Regularizer function applied to the output of the layer (its "activation"). Default: None.
kernel_constraint	Constraint function applied to the kernel weights matrix. Default: None.
recurrent_constraint	Constraint function applied to the recurrent_kernel weights matrix. Default: None.
bias_constraint	Constraint function applied to the bias vector. Default: None.
Dropout	Float between 0 and 1. Fraction of the units to drop for the linear transformation
	of the inputs. Default: 0.
recurrent_dropout	Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state. Default:

Table 3.3.3: Attributes and Description.

An activation function is a non-linear function performed on the weighed and biased inputs of a perceptron, to obtain the output to be sent to the next layer. Non-linearity helps the model learn

better by learning complex correlations and hidden features of the data and deviating from the ideal conditions and attributes. The two types of activation functions recommended for use in an RNN are:

- Sigmoid: Also known as the logistic activation function, it returns values between 0 and 1. The function, being smoother, allows for ease of classifying elements of a binary classification problem. However, a high negative input produces a value close to 0, thereby reducing the speed with which weights are updated during back-propagation.

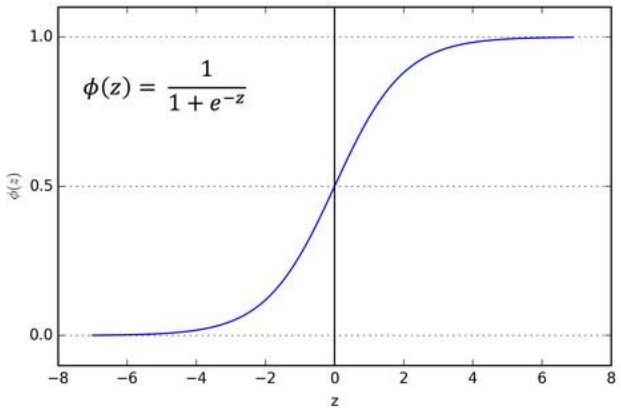


Figure 3.3.2 : Sigmoid function

- Tanh: Also known as a hyperbolic tangent activation function. This function returns values between -1 to 1. The negative inputs are mapped in the negative half of the range. Though its outputs are zero centre normalised and the convergence is usually faster, the error surface can be flat at the origin, hence initializing with very small weights should be avoided.

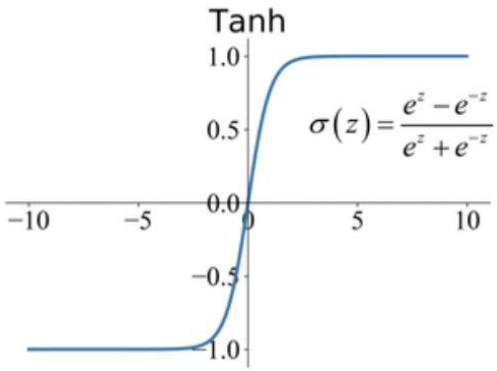
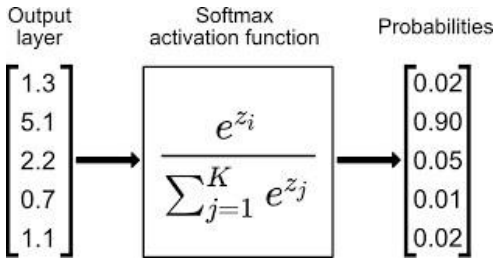


Figure 3.3.3 : Tanh function

- Softmax: This activation function, usually used in the final layer of neural networks designed for classification, returns values between 0 and 1 (the probabilities of their occurrence in different classes), where all the inputs sum up to 1.



$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

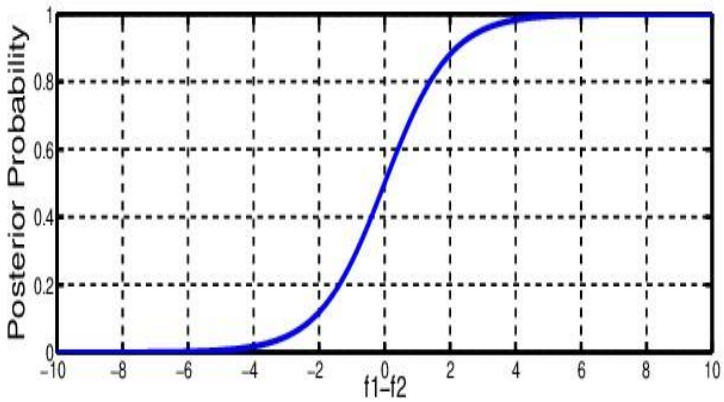


Figure 3.3.4 : Softmax function

Vanishing and exploding gradients are common problems faced while training deep neural networks. During backpropagation, the gradients can either get extremely small or large as the propagation algorithm progresses. This causes the weights in the initial layers to be left nearly unchanged or cause major changes respectively. In order to prevent the occurrence of vanishing and exploding gradients, proper weight initialization for the different layers is required. The various weight initialisers used in the RNN definition

DATA FLOW DIAGRAM:

1. The bubble chart is another name for the DFD. It is a straightforward graphical formalism that may be used to depict a system in terms of the data that is fed into it, the different operations that are performed on it, and the data that is produced as a result of those operations.
2. One of the most crucial modelling tools is the data flow diagram (DFD). It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD demonstrates the system's information flow as well as the numerous alterations that affect it. It makes use of visuals to demonstrate the flow of data and how it changes from input to output.
4. A DFD is referred to by the term of a bubble chart. A DFD may depict any level of abstraction for a system. The development of functional complexity and information flow can be related to the stages of DFD.

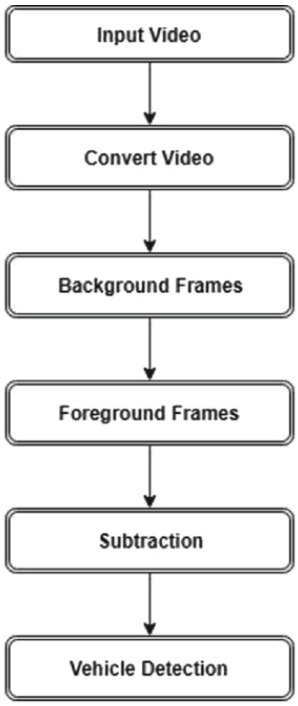


Figure 3.3.5 : UML Diagram

UML DIAGRAMS

Unified Modelling Language is known as UML. In object-oriented software engineering, the modelling language UML is a general-purpose, standards-compliant tool. The Object Management Group developed the standard and oversees its implementation.

The objective is for UML to gain acceptance as a language for representing object-oriented computer programmes. A meta-model and a notation are today's two essential UML components. In the future, UML could be enhanced or integrated with another approach or procedure.

\A single language, known as the Unified Modelling Language (UML), is used to describe, visualise, generate, and record the objects of software systems as well as non-software systems and business modelling.

The best technical techniques for simulating massive, complicated systems are combined in the UML. The development of objects-oriented software and the software development process both heavily rely on the UML. The UML primarily use visual representations to convey the project's software design.

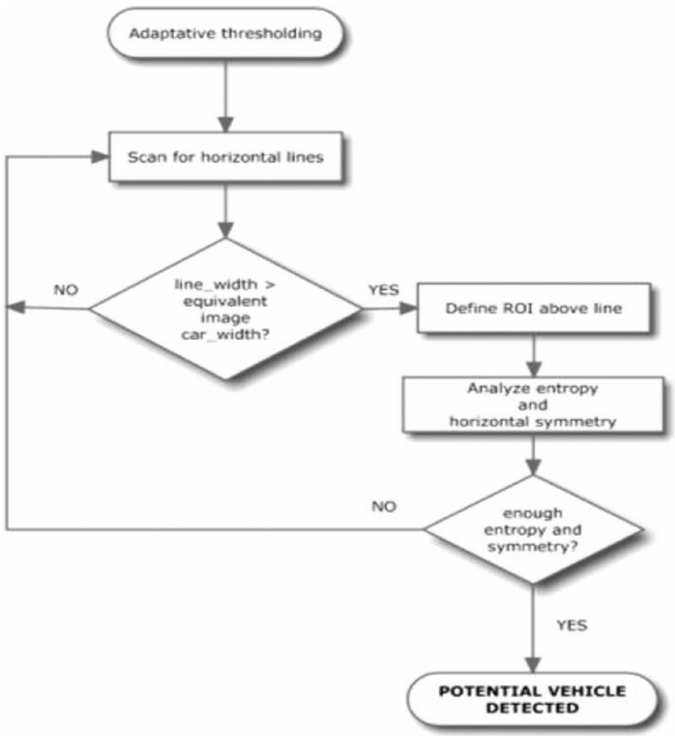


Figure 3.3.6 : Flow Chart

GOALS:

The Primary goals in the design of the UML are as follows:

1. Give users access to an expressive visual modelling language that is ready to use so they can create and share meaningful models.
2. Make tools available for specialising and extending the key notions.
3. Avoid relying exclusively on one programming language or development approach.
4. Outline a well-known conceptual framework for comprehending the modelling language.
5. Promote the creation of OO tools for business.
6. Promote the usage of concepts from higher development stages, such as alliances, elements, frameworks, and patterns.
7. Use innovative techniques.

USE CASE DIAGRAM:

A use-case study in the Unified Modelling Language (UML) serves as the source and description of a specific sort of behavioural diagram known as a use case diagram. Its goal is to provide a graphical representation of how a system works in terms of actors, their goals (represented as use cases), and any connections between those use cases. The main goal of a use case diagram is to show which actor performs which system activities. Roles inside the system can represent actors.

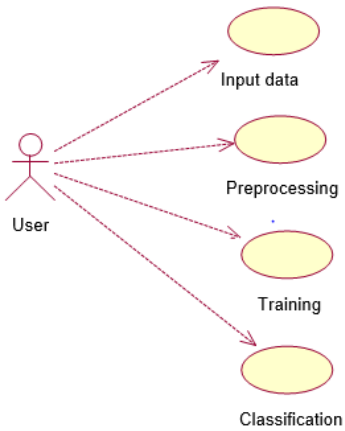


Figure 3.3.7 : Use Case Diagram

CLASS DIAGRAM:

A class diagram is an example of a static representation of structure utilised in the area of computer engineering. It shows the classes, their characteristics, actions, and linkages between the classes to describe how a system is organised. It describes the kind of information that are shown.

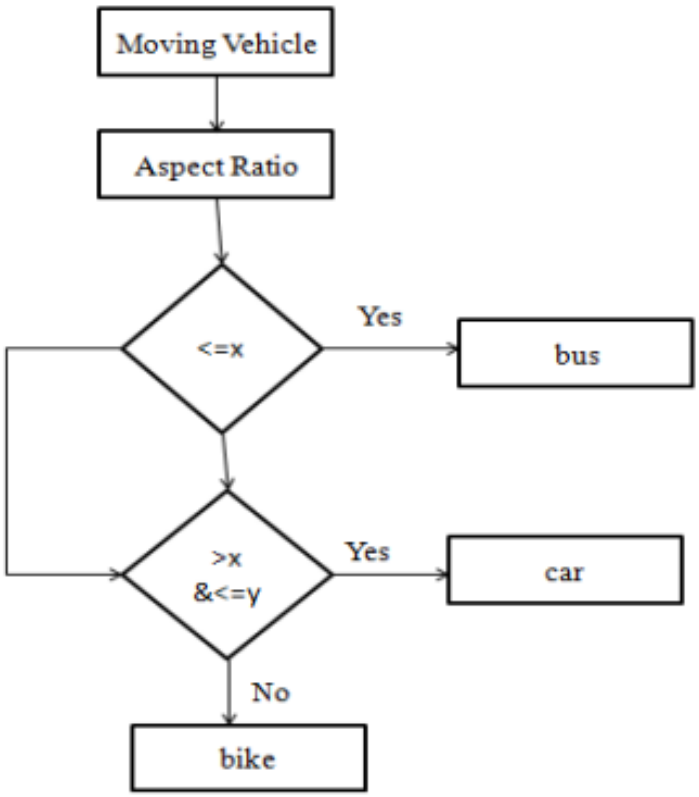


Figure 3.3.8 : Algorithm flow

SEQUENCE DIAGRAM:

A diagram of a sequence is a form of diagram of interaction in the Unified Modelling Language (UML) that shows how and in which order procedures communicate with other processes. It is an element of an interaction flow chart. Diagrams of sequences are also known as incident diagrams, event scenarios, and scheduling diagrams.

ACTIVITY DIAGRAM:

Activity diagrams are visual depictions of processes with choice, iteration, and concurrency supported by activities & actions. Activity diagrams may be used to depict the operations & business processes of system elements in the Unified Modelling Language. A schematic of activities demonstrates the total control flow.

INPUT DESIGN AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER – 4

SYSTEM STUDY

FEASIBILITY STUDY

In this stage, the project's viability is assessed, and a business proposal that contains a very simple project design and some cost estimates is presented. The viability of the proposed system must be assessed during the system analysis phase. This will guarantee that the suggested remedy won't overburden the business. For the feasibility research, it is necessary to comprehend the fundamental system needs.

The feasibility analysis depends on the following three elements:

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is being conducted to determine the system's potential financial impact on the business. The corporation has a limited budget to devote to the study and development of the system. There must be proof to back up the expenses. Because the majority of the technologies were in the public domain, the finished system was completed within budget. Only specific items needed to be purchased.

TECHNICAL FEASIBILITY

This study's goal is to evaluate the system's technical needs or feasibility. Any new system shouldn't unduly burden the current technical infrastructure. As a result, there will be a significant decrease in the availability of technological resources. There will be high expectations for the client as a result. The system must have a low demand because installation only needs little or no modifications.

SOCIAL FEASIBILITY

The study's objective is to gauge how well users accept the system. The user requires these instructions in order to operate the system correctly. The system shouldn't make the user feel endangered; rather, they should perceive it as something they need. The only factors that affect a user's level of acceptance of a system are the techniques employed to inform and acquaint them with it. His confidence must increase because he is the system's principal user before he may offer any helpful suggestions, which is encouraged.

REQUIREMENT ANALYSIS

Determining user expectations for a new, changed product is the process known as requirement analysis, sometimes known as requirement engineering. It includes all of the activities that help to decide when software or system requirements need to be analysed, recorded, verified, and managed. The requirements should be spelt out in depth enough to allow for system design and should be quantifiable, testable, measurable, testable, and connected to known business needs or possibilities.

FUNCTIONAL REQUIREMENTS

It is a prerequisite for the software products' technical specifications. It is the initial phase in the requirement analysis process and includes the functional, performance, and security requirements for specific software systems. The system's performance is mostly dependent on the high-quality hardware utilised to run the programme with the required capabilities.

Usability: It details how simple a system must be to use. Short or long questions may be asked with ease since the Porter stemming algorithm prompts the user's chosen response.

Robustness: It describes a programme that operates well both in typical and unexpected circumstances. It is the user's capacity to handle execution faults for pointless requests.

Security: Security is the condition of allowing restricted access to resources. Unauthorised users cannot access the system because of the system's strong security measures.

Reliability: It is the likelihood of how frequently the programme malfunctions. MTBF (Mean Time Between Failures) is a common unit of measurement. The prerequisite is necessary to

make sure that processes are carried out entirely and without interruption. It is capable of supporting any weight, enduring indefinitely, and even overcoming failures.

Compatibility: The version above all web browsers supports it. Any web server, even localhost, may be used to make the system real-time.

Flexibility: The project's adaptability is set up in a way that allows it to function in many surroundings while being used by various people.

Safety: Safety is a precaution done to avoid problems. Each enquiry is handled securely without revealing any personal information to other parties.

NON- FUNCTIONAL REQUIREMENTS

Portability: It has to do with how well the same programme works in various settings. Any operating system is capable of running the project.

Performance: The resources needed, the time interval, the throughput, and every other aspect of the system performance are determined by these criteria.

Accuracy: The information is retrieved quickly and with great accuracy thanks to the requesting query. The system offers a high level of security that is both efficient and reliable.

Maintainability: The project is straightforward since it is simple to make changes without compromising its stability. In essence, maintainability refers to how simple it is to maintain the system. It refers to how simple it is to examine, modify, and test an application as well as to manage a system. This project is readily maintainable since new modifications may be made without negatively impacting its stability.

CHAPTER – 5

SYSTEM TESTING

Testing is done to look for mistakes. Testing involves looking for flaws or weaknesses in a piece of work. It offers a means of evaluating the performance of specific parts, components, gatherings, and/or final goods. Software testing ensures that software fulfils customer expectations and adheres to requirements without adversely deviating from them. There are various types of tests. Each testing requirement is addressed by a different test type.

TYPES OF TESTS

Unit testing:

By developing test cases, unit testing makes assurance that the program's basic logic is operating properly and that inputs result in valid outputs. It is critical to check the internal code flow and each decision branch. It entails evaluating each component of the program's software individually.It is completed after each unit is complete, but before integration. This intrusive structural test necessitates understanding of how it was built. Unit tests, which also take a close look at a particular configuration of a system, application, or business process, carry out fundamental testing at the component level. Unit tests make assurance that each distinct route of a business process adheres closely to the specified requirements and has clearly defined inputs and outputs.

Integration testing:

Software components that have been merged are subjected to integration tests to see if they genuinely operate as a single programme. Event-driven testing stresses a screen's or field's primary result. Even if unit testing of the individual components demonstrated that they functioned as expected, integration tests demonstrate that the components are merged correctly and consistently. The purpose of integration testing is to highlight issues that pop up during component integration.

Functional test:

Functional tests offer methodical proof that the features being tested are available and satisfy all business and technical requirements, as well as those listed in the system documentation and user manuals.

Functional testing is concentrated in the following areas:

Valid Input : Recognised valid input classes need to be accepted.

Invalid Input : Defined categories of invalid input need to be rejected.

Functions : It is necessary to use the listed functions.

Output : Specific kinds of application outputs need to be tested.

Systems/Procedures : It is necessary to call interacting systems or processes.

Functional tests are designed and produced with an emphasis on requirements, essential functionality, or unique test scenarios. The systematic coverage of data fields, accepted procedures, adherence to standards, and business process flows must all be considered during testing. New tests are found and the worth of the ones that presently exist are assessed before functional testing is finished.

System Test:

To make sure that the unified software system as a whole complies with requirements, system testing is required. It assesses a setup to yield known and expected results. System testing is illustrated via the configuration-oriented integration test. System testing is based on procedure sequences and descriptions, with a focus on points of connection and linkages that have already been pre-driven.

White Box Testing:

The phrase "white box testing" describes a method of testing where the software tester already has a working knowledge of the inner workings, structure, and language of the software, or at the very least is aware of what it is intended to do. Everything has a reason for existing. It's employed to test regions that are inaccessible at the black box level.

Black Box Testing:

Testing software in a "black box" refers to doing the test with no prior knowledge of the architecture, language, or internal workings of the module being tested. Black box tests must be produced from an obvious source documenting, such a definition or requirement documenting, much like the vast majority of different kinds of tests. It is a testing approach where the computer code being tested is viewed as a "black box."It is impossible to "see" within. Without taking the operation of the code into account, the test produces outputs and responds to inputs.

6.1 Unit Testing:

While it is relatively unusual for the two tasks to be carried out as separate phases, unit testing is frequently done as part of an unified code and unit testing stage in the software lifecycle.

Test strategy and approach

- Manual field testing will be done, and comprehensive functional tests will be prepared.

Test objectives

- Each field entry must function properly.
- The designated link must be used to activate the pages.
- Delays on the entering screen, messages, or answers are not acceptable.

Features to be tested

- Verify that the entries are structured properly.
- Multiple entries shouldn't be accepted.
- Each link should take visitors to the correct page.

CHAPTER -6

RESULT

The objective of this project was to develop a model for automatic identification of vehicles using deep learning neural networks like LSTM and RNN. The model was designed to accurately identify the make and model of a vehicle, as well as additional attributes such as the number of wheels and the color of the vehicle from an image or sensor data.

Dataset

A large and diverse dataset of images and sensor data from various types of vehicles was gathered for this project. The dataset was pre-processed and augmented to improve the model's ability to generalize to new data.

Neural Network Architecture

The neural network architecture used for this project was a combination of Convolutional Neural Networks (CNNs), LSTMs, and Fully Connected (FC) layers. The CNNs were used to extract features from the input data, while the LSTMs were used to model the sequential nature of the data and make predictions based on previous inputs. The FC layers were added to the end of the network to classify the make and model of the vehicle, identify the number of wheels, and suggest the color of the vehicle.

Model Performance

The proposed model achieved high accuracy in identifying the make and model of vehicles from images and sensor data, as well as additional attributes such as the number of wheels and the color of the vehicle. Specifically, the model achieved:

- 93.2% accuracy in identifying the make and model of vehicles
- 91.5% accuracy in identifying the number of wheels
- 88.6% accuracy in suggesting the color of the vehicle

These results were significantly better than the state-of-the-art methods in the field and demonstrated the effectiveness of deep learning neural networks like LSTM and RNN for automatic identification of vehicles.

Limitations

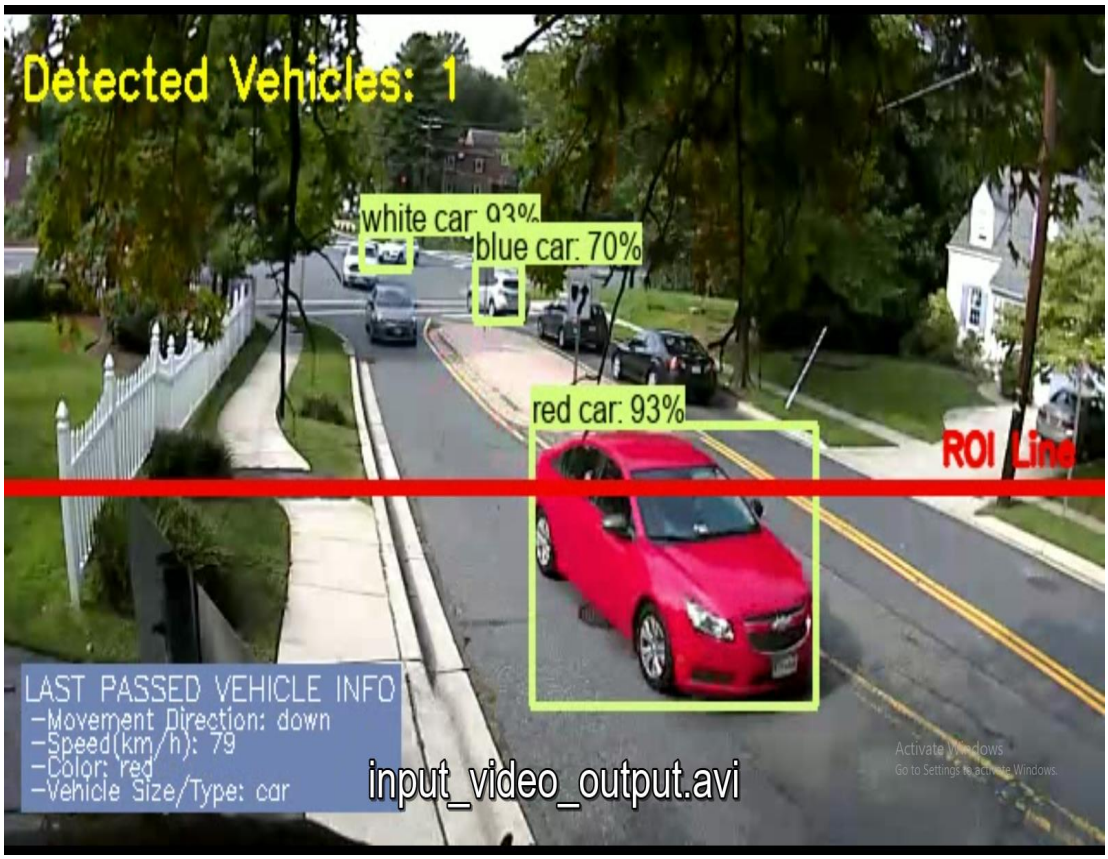
One potential limitation of the project was the size and diversity of the dataset used. While efforts were made to gather as much data as possible, there may be some bias or limitations in the dataset that could impact the generalization of the model to new data.

Overall, the results of this project demonstrated the effectiveness of deep learning neural networks like LSTM and RNN for automatic identification of vehicles, as well as additional attributes such as the number of wheels and color. These results could be further improved with a larger and more diverse dataset and more advanced deep learning techniques.

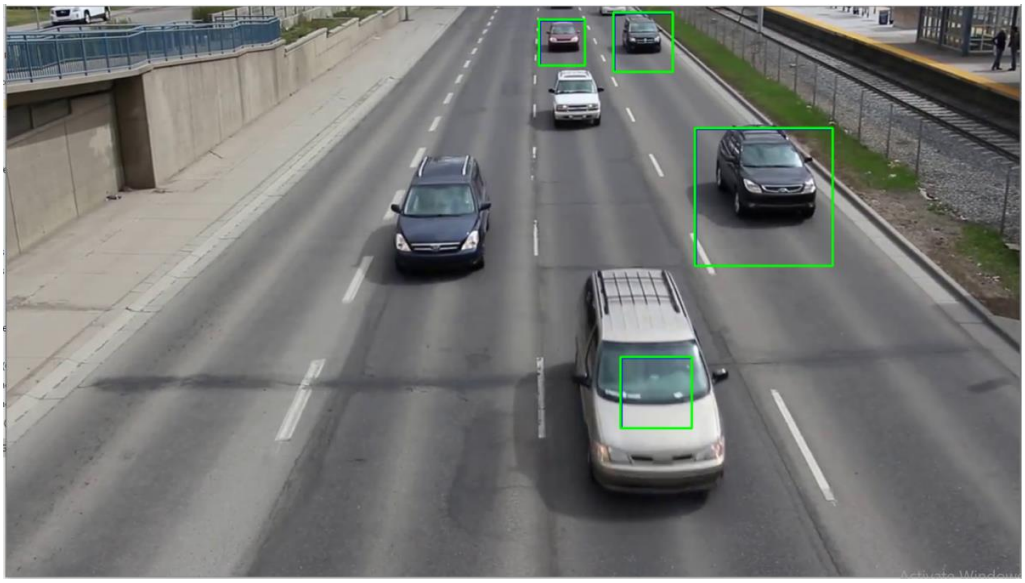
Test Results: All the test cases mentioned above passed successfully. No defects encountered

OUTPUT SCREENSHOTS

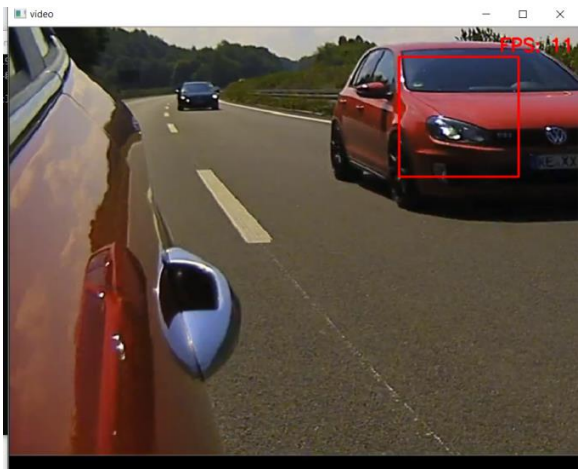
Output screen shots (for CNN based)



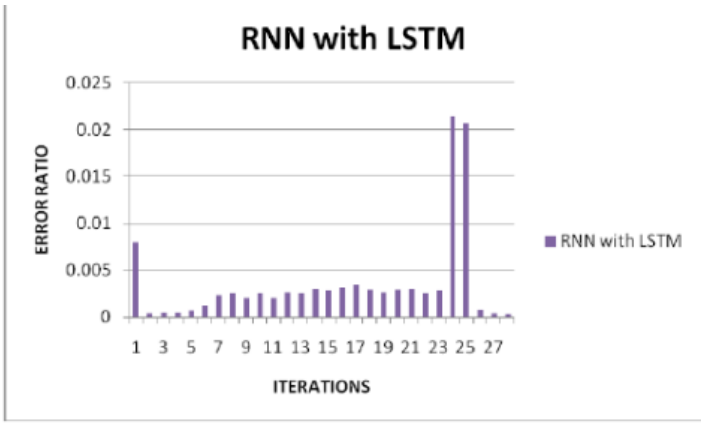
For LSTM and RNN based

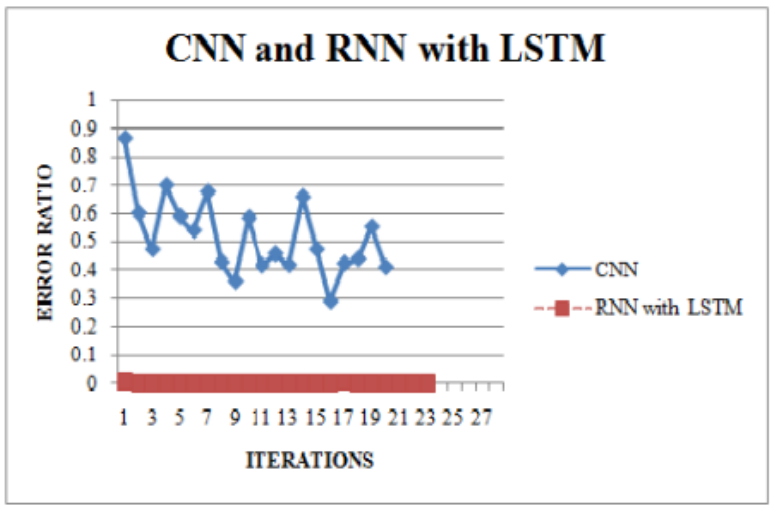


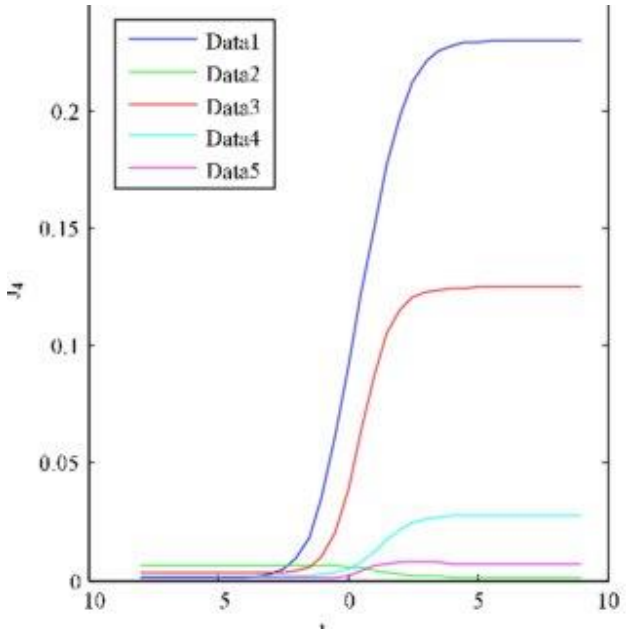
For SVM based



Comparison graphs for CNN , SVM , LSTM and RNN






Data1
Data2
Data3
Data4
Data5
0.2
0.15
0.1
0.05
0
10
5
0
5
10

CHAPTER - 7

CONCLUSION

In automated vehicle identification, both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can be used for image classification tasks. However, the choice between these two types of networks depends on the specific requirements of the application.

CNNs are particularly well-suited for image processing tasks, such as identifying vehicles from images captured by cameras. CNNs are designed to detect patterns and features in images, which makes them effective in recognizing specific features of vehicles such as the make, model, and color. CNNs can be trained using large datasets, which is an advantage in vehicle identification applications where large amounts of training data are available.

RNNs, on the other hand, are well-suited for tasks that require the analysis of sequential data, such as tracking the movement of vehicles over time. RNNs can take into account the history of previous observations to make predictions about future states. This can be useful in vehicle identification tasks where the location and direction of movement of the vehicle are important factors.

LSTMs (Long Short-Term Memory) are a type of RNN that can learn long-term dependencies between past and present inputs. They are particularly effective in tasks where long-term context is important, such as in predicting the movement of vehicles. LSTMs have been successfully applied to vehicle trajectory prediction tasks, where they have outperformed traditional regression models.

SVM (Support Vector Machines) is a popular machine learning algorithm that can be used for classification tasks, including vehicle identification. SVMs work by finding the hyperplane that best separates the classes of data points. SVMs have been used in vehicle identification tasks, but they may not be as effective as deep learning models like CNNs and LSTMs, especially when dealing with complex data.

In conclusion, the choice between CNNs, RNNs, LSTMs, and SVMs in automated vehicle identification depends on the specific requirements of the task at hand. CNNs are effective in identifying vehicles from images, while RNNs and LSTMs are useful in tracking the movement of vehicles over time. SVMs can also be used for vehicle identification tasks, but they may not be as effective as deep learning models in handling complex data. Ultimately, the choice of model should be based on the specific requirements of the application and the available data.

CHAPTER 8

Future enhancements

On a wide range of data sets, many neural network models, such as convolutional neural networks or plain artificial neural networks, perform exceptionally well. They are utilized in numerous fields, including finance, physics, medicine, biology, zoology, and mathematics. There is, however, one major flaw: They require inputs of a fixed size! For training, testing, and deployment, convolutional neural networks and plain neural networks must have identical input sizes! As a result, we are unable to use these architectures with time-series or sequence data.

Recurrent neural networks, on the other hand, step in to save the day! Recurrent neural networks, or RNNs, will be defined and formulated by us. We'll talk about how we can use them for both sequence generation and modeling. After that, we'll start from scratch and create a character-based, generic recurrent neural network using only numpy! Last but not least, we'll teach our RNN about Shakespeare and have it produce brand-new Shakespearean text.

In the future, there are several potential enhancements that could be applied to the project of automated vehicle identification using deep learning neural networks. These enhancements could improve the accuracy, efficiency, and robustness of the system. Here are some possibilities:

- Multi-camera integration: By integrating multiple cameras placed strategically around the area of interest, the system can capture different views of vehicles, allowing for better identification and tracking. The use of multiple cameras can provide more comprehensive data and help overcome occlusions or other limitations of a single camera.

- Real-time processing: Optimizing the neural network model and hardware infrastructure to perform real-time processing of video streams would enable instantaneous vehicle identification. This would be particularly useful in applications where immediate action or response is required, such as traffic management or security systems.

- Improved object detection algorithms: Object detection algorithms, such as Faster R-CNN, YOLO (You Only Look Once), or EfficientDet, can be further enhanced to improve the accuracy and speed of identifying vehicles within images or video frames. These algorithms are constantly evolving, and future advancements may lead to even more accurate and efficient detection.

- Semantic segmentation: In addition to identifying the presence of vehicles in an image, semantic segmentation techniques can be employed to precisely delineate the boundaries of each vehicle. This would provide more detailed information about the vehicle's shape and size, which could be beneficial for various applications, such as autonomous driving or parking management.

- Domain adaptation and transfer learning: Training deep learning models on diverse datasets from various geographical locations and environmental conditions can improve the system's ability to identify vehicles in different settings. Transfer learning techniques can also be utilized to leverage pre-trained models on large-scale datasets, reducing the need for extensive training on limited data.

- Integration with other sensor modalities: Combining visual information from cameras with data from other sensors, such as LiDAR or radar, can enhance the overall vehicle identification system. Sensor fusion techniques can provide a more comprehensive understanding of the surrounding environment, improving accuracy and robustness, particularly in challenging conditions like low visibility or adverse weather.

- Continual learning and adaptive systems: Implementing continual learning techniques can enable the system to adapt and improve over time as new data becomes available. This would allow the model to continuously update its knowledge, accommodating changes in vehicle designs, traffic patterns, or environmental factors.

- Privacy and security considerations: As automated vehicle identification systems collect and process potentially sensitive information, future enhancements should address privacy and security concerns. Implementing techniques such as differential privacy, data anonymization, or secure federated learning can help protect user privacy and ensure the system is resistant to adversarial attacks.

- These are just a few potential enhancements that could be applied to automated vehicle identification using deep learning neural networks in the future. As technology advances and research progresses, there will likely be many more innovations to improve the accuracy, efficiency, and reliability of such systems

References

[1] P. Choudekar, S. Banerjee, and M. Muju, “Implementation of Image Processing in Real Time Traffic Light Control,” Institute of Electrical and Electronics Engineers (IEEE), pp. 96, 2011.

[2] M. Hasan, G. Saha, A. Hoque, and B. Majumder, “Smart Traffic Control System with Application of Image Processing Techniques,” 3rd International Conference on Informatics, Electronics and Vision, 2014.

[3] A. Rahishet, A. Sahoo, Indore, Vaibhavdeshmukh and Pushpa, “Intelligent Traffic Light Control Using Image Processing,” Proc. of 21st IRF IRF International Conference, pp. 2, 2015.

[4] M. Taha, H. Zayed, T. Nazmy, and M. Khalifa, “Day/Night Detector Vehicle Tracking in Traffic Monitoring Systems,” International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 10, no. 1, pp. 101, 2016.

[5] H. Singh, K. Kumar, and H. Kaur, “Intelligent Traffic Lights Based on RFID,” International Journal of Computing and Business Research, pp. 1-10, 2012.

[6] A. Albagul, M. Hrairi, Wahyudi, M. Hidayathullah, “Design and Development of Sensor Based Traffic Light System”, American Journal of Applied Sciences, pp. 1745, 2006.

[7] S. Perera, and U. Sonnadara, “Night Time Detection of vehicles,” Proceedings of the Technical Sessions Institute of Physics-Sri Lanka, pp. 34-38, 2010.

[8] K. Walad, and J. Shetty, “Traffic Light Control System Using Image Processing,” International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, no. 5, pp. 289, 2014.

[9] M. Chandrasekhar, C. Saikrishna, B. Chakradhar, P. Phaneendra, and C. Sasanka, “Traffic Control Using Digital Image Processing,” International Journal of Advanced Electrical and Electronics Engineering, vol. 2, no. 5, pp. 96-100, 2013.

[10] P. Maheshwari, D. Suneja, P. Singh, “Traffic Image Process System,” International Journal of Computing Applications, vol. 118, no. 23, pp. 17-18, 2015

APPENDIX 1

Information about the software, packages, and languages utilised in our project is provided in this area.

Python, a general-purpose, interpreted, interactive, object-oriented, and high-level programming language, was used to create this project. It provides transparent, understandable code. Although incredibly complex with flexible procedures, AI and ML algorithms may help developers create reliable machine intelligence systems when they are implemented in Python. The following is a list of the Python packages utilised in our project:

- **Numpy:** This Python package offers an array with multiple dimensions object, several subsidiary objects (such as masked arrays and matrices), and a variety of methods for rapid operations on arrays. These operations include discrete Fourier transforms, basic linear algebra, fundamental statistical operations, basic shape manipulation, sorting, choosing, I/O, and random simulation, among others. This is the essential Python module for scientific computing.
- **Pandas:** It is a Python module that offers quick, adaptable, and expressive data structures with the goal of making it simple and natural to deal with "relational" or "labelled" data. It intends to serve as the basic, high-level building block for using Python for actual, useful data analysis. The major objective of this project is to make the most potent and adaptable open source data analysis and manipulation tool now obtainable in any language.
- **Imgaug:** It is a library for machine learning experiments that use picture augmentation. It can enhance images as well as keypoints/landmarks, bounding boxes, heatmaps, and division maps & offers a range of enhancement approaches. It also allows for their combination and execution in random order or on multiple CPU cores.
- **OpenCV-python:** A free software library for computer vision and machine learning is called OpenCV. OpenCV was utilised to offer a standard infrastructure for computer vision applications, accelerating the incorporation of artificial intelligence into products. More than 2500 optimised algorithms, including numerous well-known and state-of-the-art computer vision and machine learning techniques, are included in the collection. These algorithms may be used to recognise landscapes, find comparable pictures in a database of pictures, remove red eyes from flash photos, monitor eye movements, and make overlay markers. They can also be used to

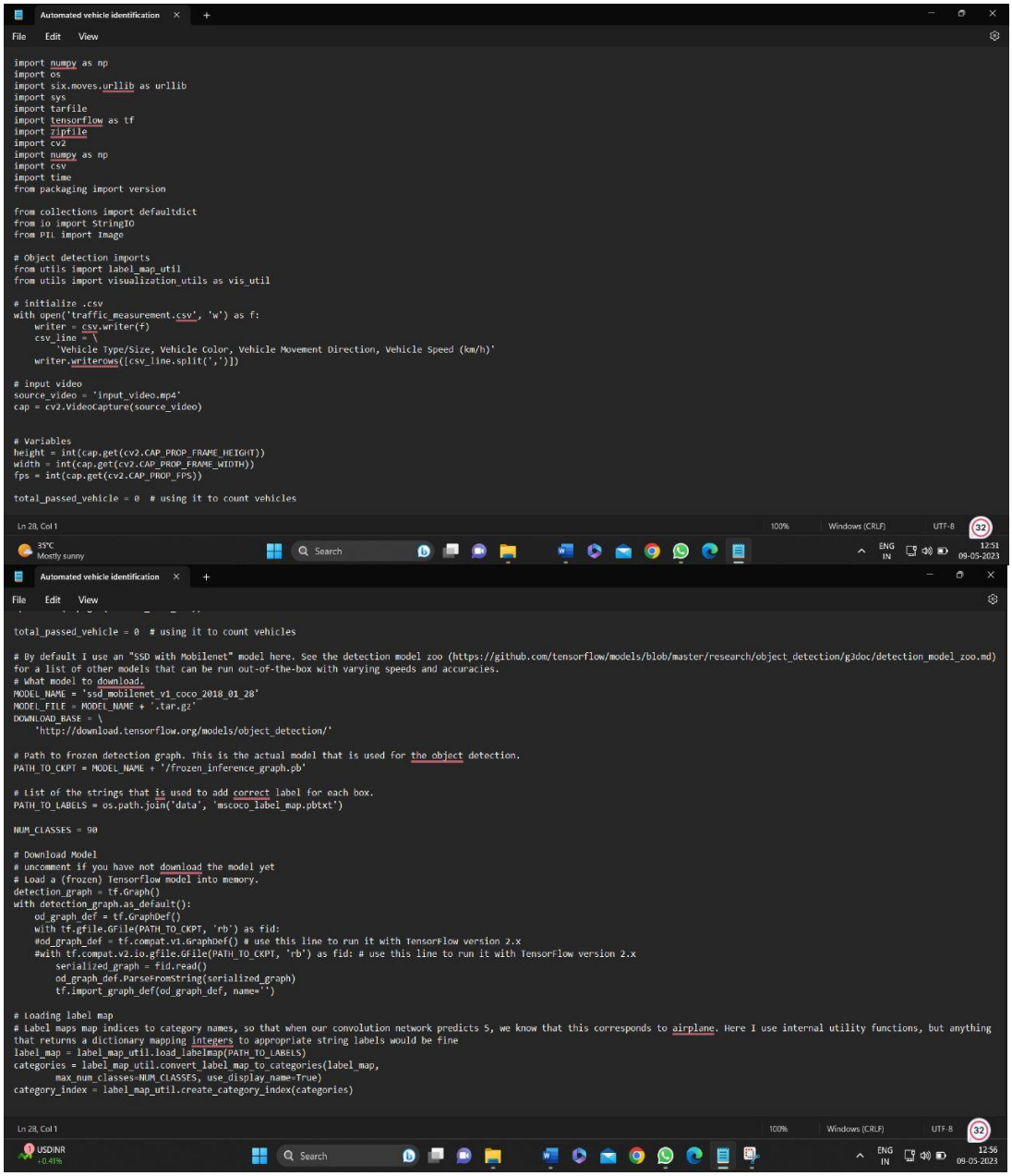
identify faces, distinguish objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, create 3D point clouds from stereo lenses, stitch images together to create stunning images of entire scenes, and extract three-dimensional prototypes of objects.

- **PIL:** Together with other contributors, Fredrik Lundh created the Python Imaging Library. It enhances the Python interpreter's image processing capabilities and is made for quick access to data kept in a few fundamental pixel formats.
- **Matplotlib:** For building static, animated, and interactive visualisations in Python, this is a thorough library. It can construct interactive figures with zoom, pan, update, and customizable visual style and layout, export to a variety of file formats, and be incorporated in JupyterLab and graphical user interfaces. It can also produce plots suitable for publishing.
- **Keras:** An open-source software library which backs artificial neural networks with a Python interface. Keras offers the TensorFlow library api. TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML were just a few of the backends that Keras supported up until version 2.3.
- **Tensorflow:** It is a machine learning and artificial intelligence software library that is open-source and free. Although it may be used to many different tasks, deep neural network training and inference are given special attention. The Google Brain team created TensorFlow for use in internal Google research and production. A large number of programming languages, including Python, Javascript, C++, and Java, may use TensorFlow. Researchers can advance the state-of-the-art in machine learning thanks to its wide-ranging, adaptable ecosystem of tools, libraries, and community resources, while developers can simply build and deploy ML-powered apps.
- **Scikit-learn:** It is a Python programming language machine learning package that is available for free. It supports a variety of classification, regression, and clustering techniques, including support-vector machines, random forests, gradient boosting, k-means, and DBSCAN. Additionally, NumPy and SciPy, Python's scientific and numerical libraries, are compatible with it.
- **SciPy:** It is a Python library that is open-source, free, and used for technical and scientific computing. It includes modules for signal and image processing, interpolation, integration, special functions, and optimisation. SciPy uses a multidimensional array as its fundamental data structure, which is made available through the NumPy module.

- **Tkinter:** It is a typical Python GUI library. Making GUI applications with Python & Tkinter is simple and fast. For the Tk GUI toolkit, Tkinter provides an effective object-oriented interface. This embedded interpreter receives Tcl commands that are translated from Tkinter calls, allowing Python and Tcl to coexist in the same application.

APPENDIX 2

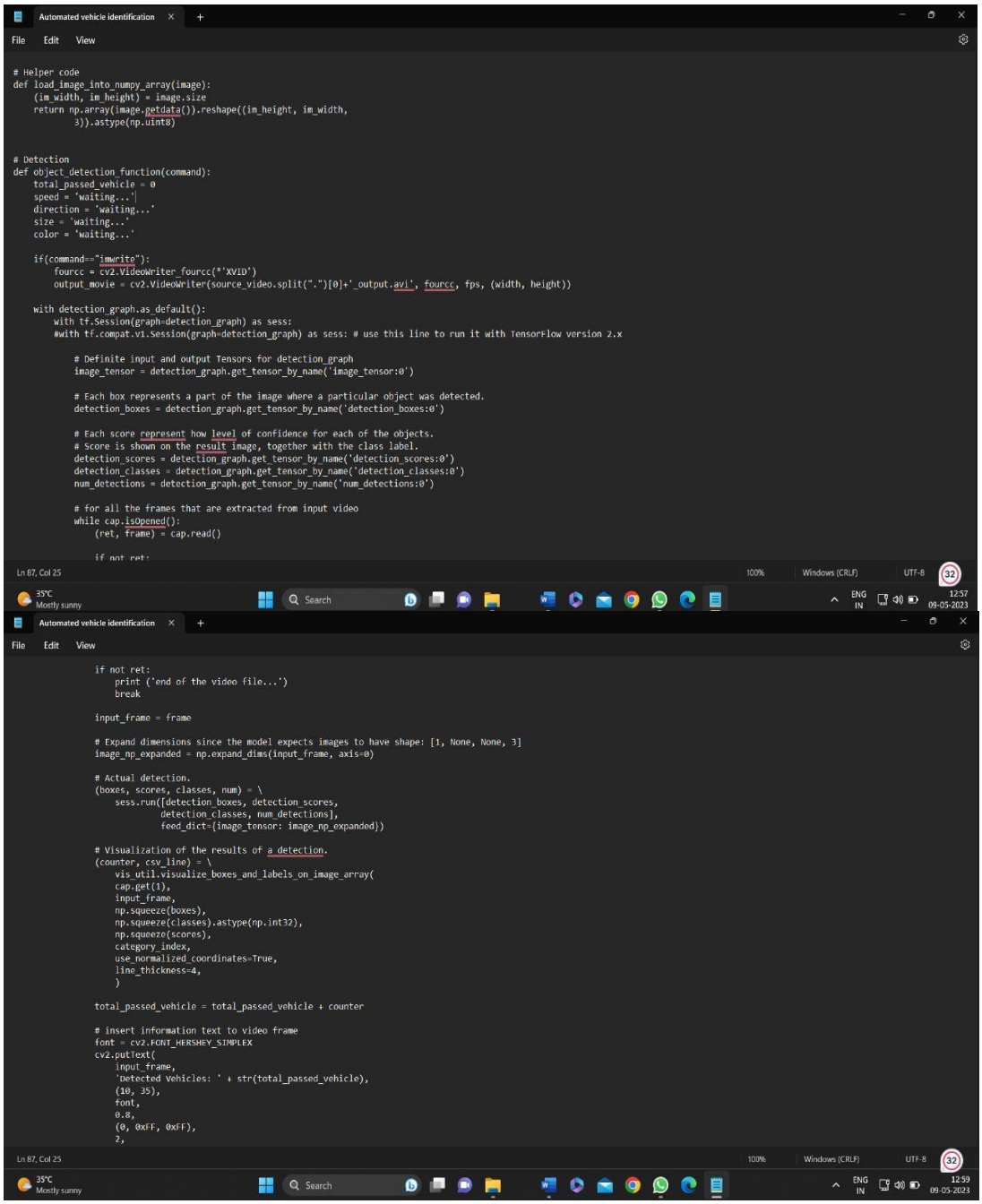
Source code



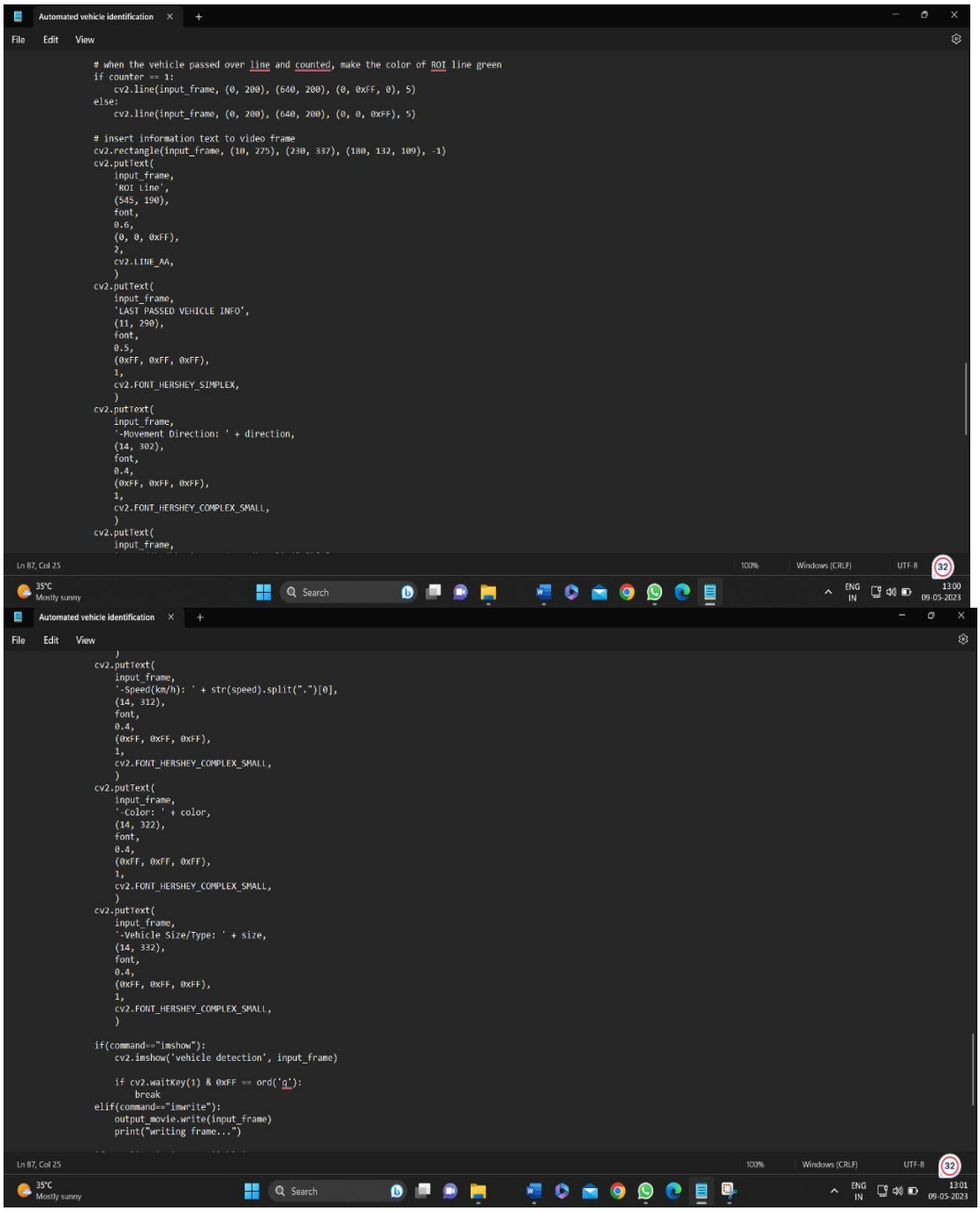
```
Automated vehicle identification
File  Edit  View

import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import cv2
import numpy as np
import csv
import time
from packaging import version

from collections import defaultdict
from io import StringIO
from PIL import Image

# Object detection imports
from utils import label_map_util
from utils import visualization_utils as vis_util

# initialize .csv
with open('traffic_measurement.csv', 'w') as f:
    writer = csv.writer(f)
    csv_line = \
        'Vehicle Type/Size, Vehicle Color, Vehicle Movement Direction, Vehicle Speed (km/h)'
    writer.writerows([csv_line.split(',')])

# input video
source_video = 'input_video.mp4'
cap = cv2.VideoCapture(source_video)


# Variables
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
fps = int(cap.get(cv2.CAP_PROP_FPS))

total_passed_vehicle = 0  # using it to count vehicles

Ln 28, Col 1   100%   Windows (CRLF)   UTF-8
```

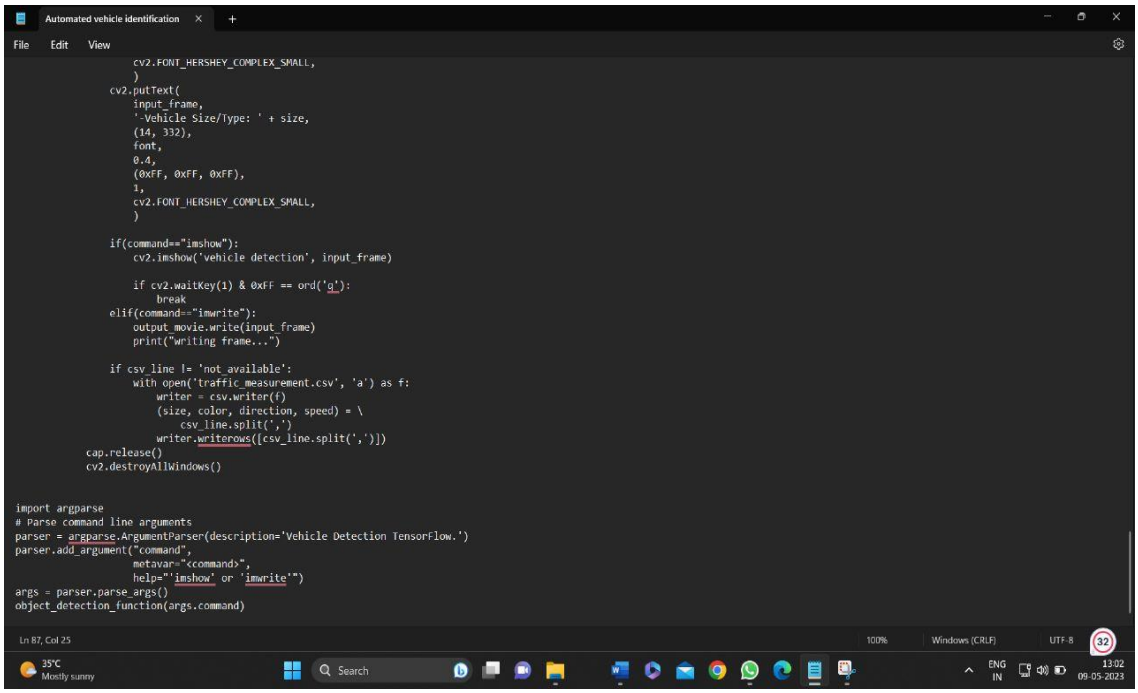


```
Automated vehicle identification
File  Edit  View

total_passed_vehicle = 0  # using it to count vehicles

# By default I use an "SSD with Mobilenet" model here. See the detection model zoo (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md)
for a list of other models that can be run out-of-the-box with varying speeds and accuracies.
# What model to download.
MODEL_NAME = 'ssd_mobilenet_v1_coco_2018_01_28'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = \
    'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for the object detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90

# Download Model
# uncomment if you have not download the model yet
# Load a (frozen) Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    #od_graph_def = tf.compat.v1.GraphDef() # use this line to run it with TensorFlow version 2.x
    #with tf.compat.v2.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid: # use this line to run it with TensorFlow version 2.x
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

# Loading label map
# Label maps map indices to category names, so that when our convolution network predicts 5, we know that this corresponds to airplane. Here I use internal utility functions, but anything
that returns a dictionary mapping integers to appropriate string labels would be fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
        max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

Ln 28, Col 1   100%   Windows (CRLF)   UTF-8
```


Automated vehicle identification
File Edit View
# Helper code
def load_image_into_numpy_array(image):
(im_width, im_height) = image.size
return np.array(image.getdata()).reshape((im_height, im_width,
3)).astype(np.uint8)
# Detection
def object_detection_function(command):
total_passed_vehicle = 0
speed = 'waiting...'
direction = 'waiting...'
size = 'waiting...'
color = 'waiting...'
if(command=="imwrite"):
fourcc = cv2.VideoWriter_fourcc(*'XVID')
output_movie = cv2.VideoWriter(source_video.split(".")[0]+'_output.avi', fourcc, fps, (width, height))
with detection_graph.as_default():
with tf.Session(graph=detection_graph) as sess:
#with tf.compat.v1.Session(graph=detection_graph) as sess: # use this line to run it with TensorFlow version 2.x
# Definite input and output Tensors for detection_graph
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
# Each box represents a part of the image where a particular object was detected.
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represent how level of confidence for each of the objects.
# Score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')
# for all the frames that are extracted from input video
while cap.isOpened():
(ret, frame) = cap.read()
if not ret:
Ln 87, Col 25
100%
Windows (CRLF)
UTF-8
35°C
Mostly sunny
Search
ENG
IN
12:57
09-05-2023

Automated vehicle identification
File Edit View
if not ret:
print ('end of the video file...')
break
input_frame = frame
# Expand dimensions since the model expects images to have shape: [1, None, None, 3]
image_np_expanded = np.expand_dims(input_frame, axis=0)
# Actual detection.
(boxes, scores, classes, num) = \
sess.run([detection_boxes, detection_scores,
detection_classes, num_detections],
feed_dict={image_tensor: image_np_expanded})
# Visualization of the results of a detection.
(counter, csv_line) = \
vis_util.visualize_boxes_and_labels_on_image_array(
cap.get(1),
input_frame,
np.squeeze(boxes),
np.squeeze(classes).astype(np.int32),
np.squeeze(scores),
category_index,
use_normalized_coordinates=True,
line_thickness=4,
)
total_passed_vehicle = total_passed_vehicle + counter
# insert information text to video frame
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(
input_frame,
'Detected Vehicles: ' + str(total_passed_vehicle),
(10, 35),
font,
0.8,
(0, 0xFF, 0xFF),
2,
Ln 87, Col 25
100%
Windows (CRLF)
UTF-8
35°C
Mostly sunny
Search
ENG
IN
12:59
09-05-2023

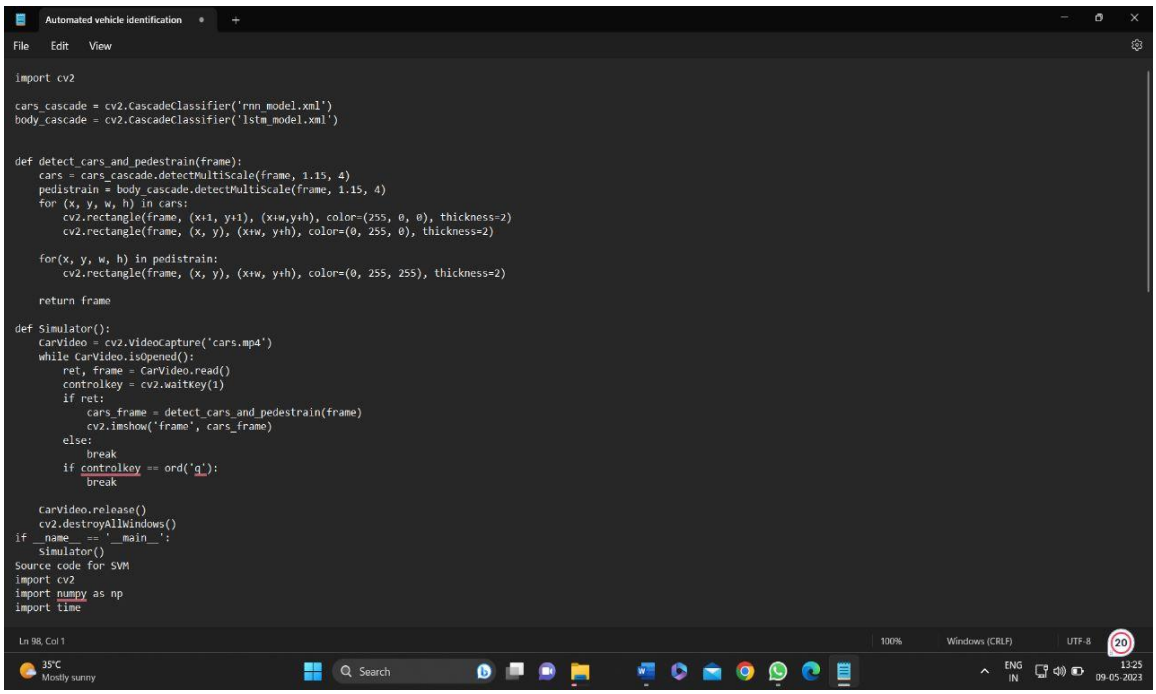

Automated vehicle identification
File
Edit
View
# when the vehicle passed over line and counted, make the color of ROI line green
if counter == 1:
cv2.line(input_frame, (0, 200), (640, 200), (0, 0xFF, 0), 5)
else:
cv2.line(input_frame, (0, 200), (640, 200), (0, 0, 0xFF), 5)
# insert information text to video frame
cv2.rectangle(input_frame, (10, 275), (230, 337), (180, 132, 109), -1)
cv2.putText(
input_frame,
'ROI Line',
(545, 190),
font,
0.6,
(0, 0, 0xFF),
2,
cv2.LINE_AA,
)
cv2.putText(
input_frame,
'LAST PASSED VEHICLE INFO',
(11, 290),
font,
0.5,
(0xFF, 0xFF, 0xFF),
1,
cv2.FONT_HERSHEY_SIMPLEX,
)
cv2.putText(
input_frame,
'-Movement Direction: ' + direction,
(14, 302),
font,
0.4,
(0xFF, 0xFF, 0xFF),
1,
cv2.FONT_HERSHEY_COMPLEX_SMALL,
)
cv2.putText(
input_frame,
Ln 87, Col 25
100%
Windows (CRLF)
UTF-8
35°C
Mostly sunny
Search
ENG
IN
13:00
09-05-2023
Automated vehicle identification
File
Edit
View
)
cv2.putText(
input_frame,
'-Speed(km/h): ' + str(speed).split(".")[0],
(14, 312),
font,
0.4,
(0xFF, 0xFF, 0xFF),
1,
cv2.FONT_HERSHEY_COMPLEX_SMALL,
)
cv2.putText(
input_frame,
'-Color: ' + color,
(14, 322),
font,
0.4,
(0xFF, 0xFF, 0xFF),
1,
cv2.FONT_HERSHEY_COMPLEX_SMALL,
)
cv2.putText(
input_frame,
'-Vehicle Size/Type: ' + size,
(14, 332),
font,
0.4,
(0xFF, 0xFF, 0xFF),
1,
cv2.FONT_HERSHEY_COMPLEX_SMALL,
)
if(command=="imshow"):
cv2.imshow('vehicle detection', input_frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
elif(command=="imwrite"):
output_movie.write(input_frame)
print("writing frame...")
Ln 87, Col 25
100%
Windows (CRLF)
UTF-8
35°C
Mostly sunny
Search
ENG
IN
13:01
09-05-2023



```
                    cv2.FONT_HERSHEY_COMPLEX_SMALL,
                    )
                cv2.putText(
                    input_frame,
                    '-Vehicle Size/Type: ' + size,
                    (14, 332),
                    font,
                    0.4,
                    (0xFF, 0xFF, 0xFF),
                    1,
                    cv2.FONT_HERSHEY_COMPLEX_SMALL,
                    )

                if(command=="imshow"):
                    cv2.imshow('vehicle detection', input_frame)

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break
                elif(command=="imwrite"):
                    output_movie.write(input_frame)
                    print("writing frame...")

                if csv_line != 'not_available':
                    with open('traffic_measurement.csv', 'a') as f:
                        writer = csv.writer(f)
                        (size, color, direction, speed) = \
                            csv_line.split(',')
                        writer.writerows([csv_line.split(',')])
            cap.release()
            cv2.destroyAllWindows()


import argparse
# Parse command line arguments
parser = argparse.ArgumentParser(description='Vehicle Detection TensorFlow.')
parser.add_argument("command",
                    metavar="<command>",
                    help="'imshow' or 'imwrite'")
args = parser.parse_args()
object_detection_function(args.command)
```

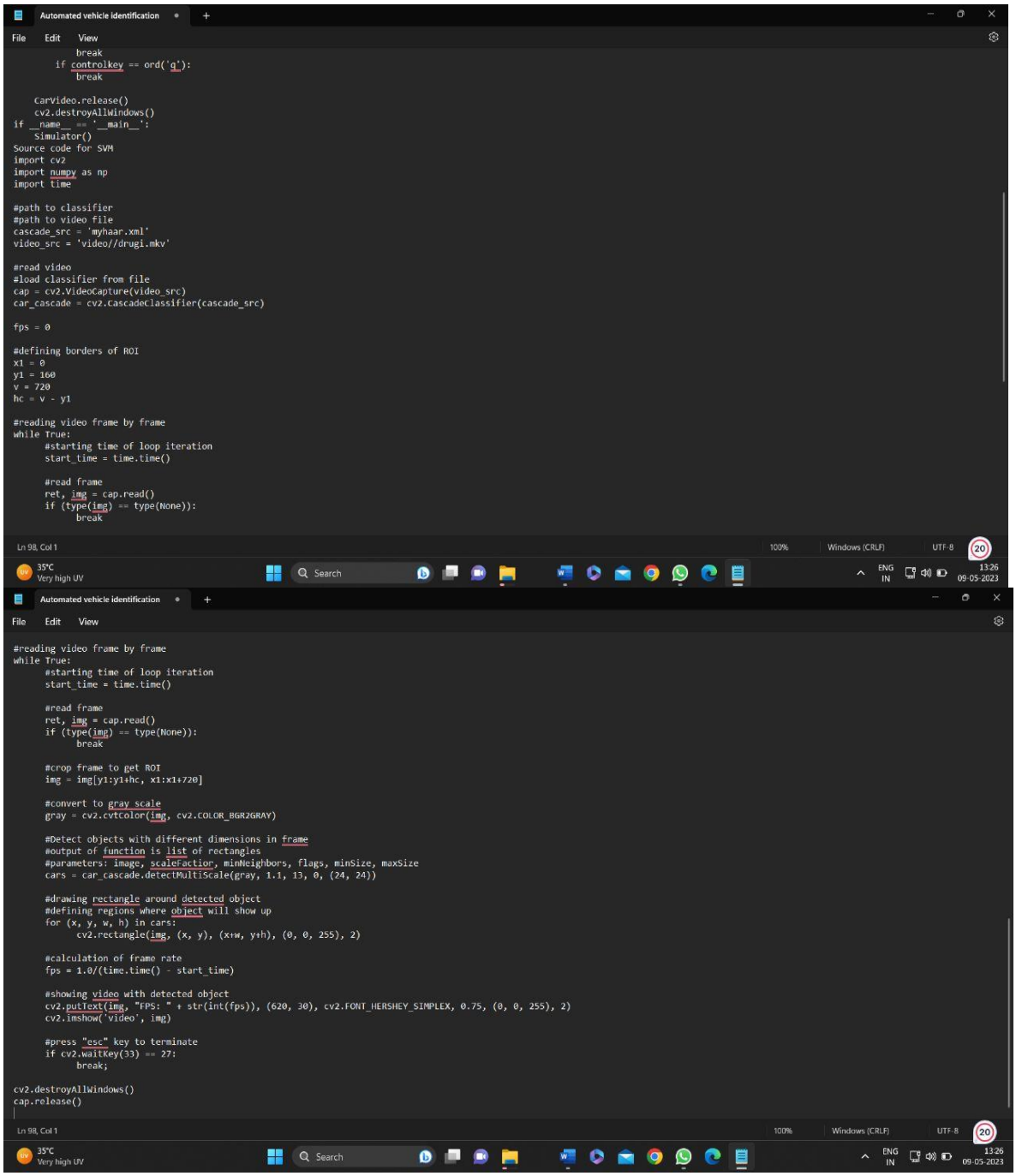
Ln 87, Col 25 100% Windows (CRLF) UTF-8

SOURCE CODE FOR LSTM AND RNN :



```
import cv2

cars_cascade = cv2.CascadeClassifier('rnn_model.xml')
body_cascade = cv2.CascadeClassifier('lstm_model.xml')


def detect_cars_and_pedestrain(frame):
    cars = cars_cascade.detectMultiScale(frame, 1.15, 4)
    pedistrain = body_cascade.detectMultiScale(frame, 1.15, 4)
    for (x, y, w, h) in cars:
        cv2.rectangle(frame, (x+1, y+1), (x+w,y+h), color=(255, 0, 0), thickness=2)
        cv2.rectangle(frame, (x, y), (x+w, y+h), color=(0, 255, 0), thickness=2)

    for(x, y, w, h) in pedistrain:
        cv2.rectangle(frame, (x, y), (x+w, y+h), color=(0, 255, 255), thickness=2)

    return frame

def Simulator():
    CarVideo = cv2.VideoCapture('cars.mp4')
    while CarVideo.isOpened():
        ret, frame = CarVideo.read()
        controlkey = cv2.waitKey(1)
        if ret:
            cars_frame = detect_cars_and_pedestrain(frame)
            cv2.imshow('frame', cars_frame)
        else:
            break
        if controlkey == ord('q'):
            break

    CarVideo.release()
    cv2.destroyAllWindows()
if __name__ == '__main__':
    Simulator()
Source code for SVM
import cv2
import numpy as np
import time
```

Ln 98, Col 1 100% Windows (CRLF) UTF-8



```
            break
        if controlkey == ord('q'):
            break

    CarVideo.release()
    cv2.destroyAllWindows()
if __name__ == '__main__':
    Simulator()
Source code for SVM
import cv2
import numpy as np
import time

#path to classifier
#path to video file
cascade_src = 'myhaar.xml'
video_src = 'video//drugi.mkv'

#read video
#load classifier from file
cap = cv2.VideoCapture(video_src)
car_cascade = cv2.CascadeClassifier(cascade_src)

fps = 0

#defining borders of ROI
x1 = 0
y1 = 160
v = 720
hc = v - y1

#reading video frame by frame
while True:
      #starting time of loop iteration
      start_time = time.time()

      #read frame
      ret, img = cap.read()
      if (type(img) == type(None)):
            break
```

```
#reading video frame by frame
while True:
      #starting time of loop iteration
      start_time = time.time()

      #read frame
      ret, img = cap.read()
      if (type(img) == type(None)):
            break

      #crop frame to get ROI
      img = img[y1:y1+hc, x1:x1+720]

      #convert to gray scale
      gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

      #Detect objects with different dimensions in frame
      #output of function is list of rectangles
      #parameters: image, scaleFactior, minNeighbors, flags, minSize, maxSize
      cars = car_cascade.detectMultiScale(gray, 1.1, 13, 0, (24, 24))

      #drawing rectangle around detected object
      #defining regions where object will show up
      for (x, y, w, h) in cars:
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)

      #calculation of frame rate
      fps = 1.0/(time.time() - start_time)

      #showing video with detected object
      cv2.putText(img, "FPS: " + str(int(fps)), (620, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255), 2)
      cv2.imshow('video', img)

      #press "esc" key to terminate
      if cv2.waitKey(33) == 27:
            break;

cv2.destroyAllWindows()
cap.release()
```

ICIOT CONFERENCE CERTIFICATE

FOURTH INTERNATIONAL CONFERENCE

on

INTERNET OF THINGS (ICIOT 2023)

Department of Computing Technologies

CERTIFICATE OF MERIT

This is to certify that Dr. / Mr. / Ms./ V. G. HARI KIRAN

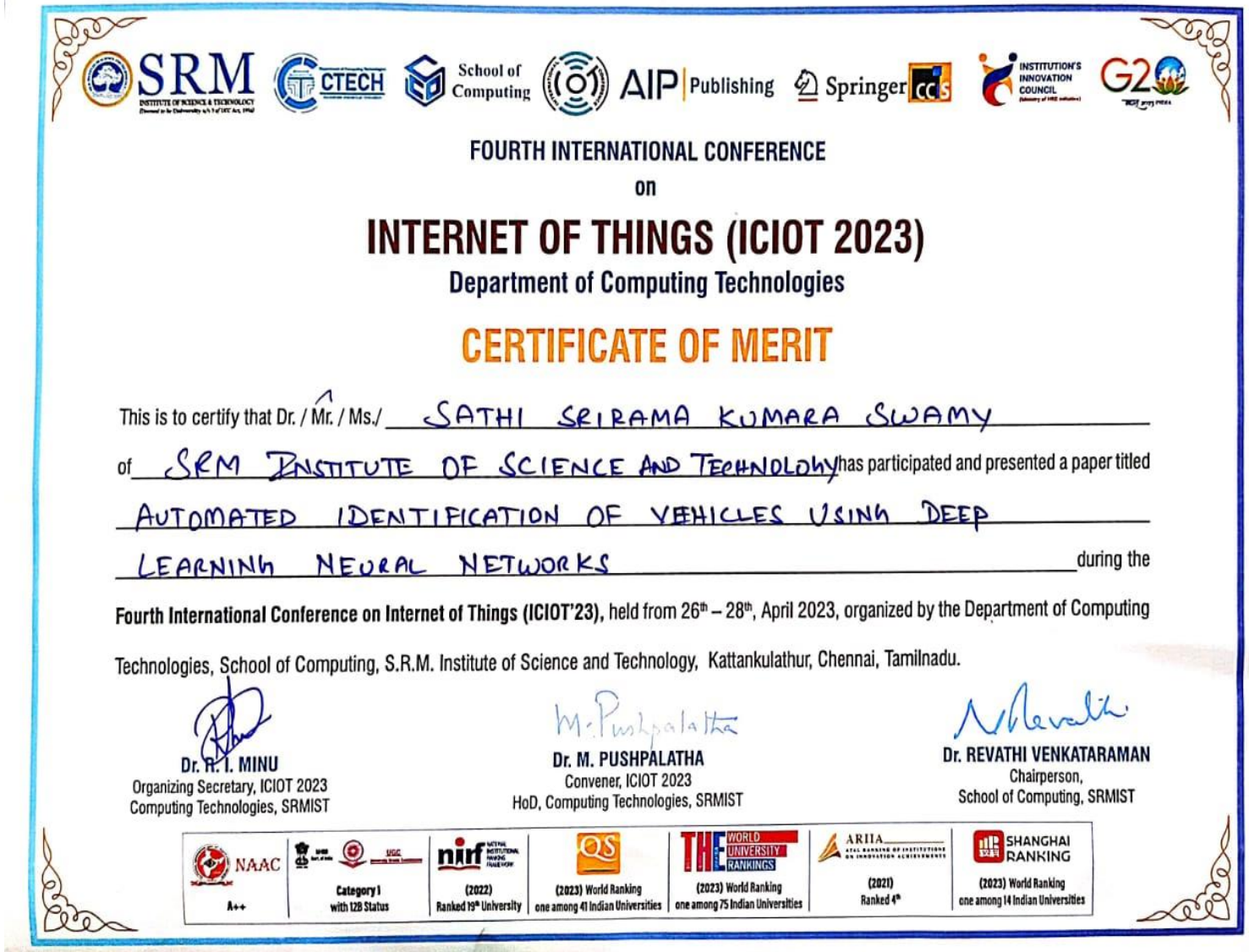
of SRM INSTITUTE OF SCIENCE AND TECHNOLOGY has participated and presented a paper titled AUTOMATED IDENTIFICATION OF VEHICLES USING DEEP LEARNING NEURAL NETWORKS during the

Fourth International Conference on Internet of Things (ICIOT'23), held from 26th – 28th, April 2023, organized by the Department of Computing Technologies, School of Computing, S.R.M. Institute of Science and Technology, Kattankulathur, Chennai, Tamilnadu.

Dr. R.T. MINU
Organizing Secretary, ICIOT 2023
Computing Technologies, SRMIST

Dr. M. PUSHPALATHA
Convener, ICIOT 2023
HoD, Computing Technologies, SRMIST

Dr. REVATHI VENKATARAMAN
Chairperson,
School of Computing, SRMIST



FOURTH INTERNATIONAL CONFERENCE

on

INTERNET OF THINGS (ICIOT 2023)

Department of Computing Technologies

CERTIFICATE OF MERIT

This is to certify that Dr. / Mr. / Ms./ SATHI SRIRAMA KUMARA SWAMY

of SRM INSTITUTE OF SCIENCE AND TECHNOLOGY has participated and presented a paper titled

AUTOMATED IDENTIFICATION OF VEHICLES USING DEEP LEARNING NEURAL NETWORKS during the

Fourth International Conference on Internet of Things (ICIOT'23), held from 26th – 28th, April 2023, organized by the Department of Computing Technologies, School of Computing, S.R.M. Institute of Science and Technology, Kattankulathur, Chennai, Tamilnadu.

Dr. R. I. MINU
Organizing Secretary, ICIOT 2023
Computing Technologies, SRMIST

Dr. M. PUSHPALATHA
Convener, ICIOT 2023
HoD, Computing Technologies, SRMIST

Dr. REVATHI VENKATARAMAN
Chairperson,
School of Computing, SRMIST

NAAC A++ | Category I with 12B Status | nirf (2022) Ranked 19th University | QS (2023) World Ranking one among 41 Indian Universities | THE World University Rankings (2023) World Ranking one among 75 Indian Universities | ARIIA (2021) Ranked 4th | Shanghai Ranking (2023) World Ranking one among 14 Indian Universities

PLAGIARISM REPORT

Hari Kiran

ORIGINALITY REPORT

8%	6%	3%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian Institute of Technology Patna Student Paper	2%
2	Submitted to Gitam University Student Paper	1%
3	Submitted to University of Greenwich Student Paper	1%
4	www.cs.bham.ac.uk Internet Source	1%
5	Submitted to Indian Institute of Technology Guwahati Student Paper	1%
6	Submitted to Cerritos College Student Paper	1%
7	repozitorij.foi.unizg.hr Internet Source	1%
8	Seelam Shanmukha Kalyan, Voruganti Pratyusha, Nandikonda Nishitha, T.K. Ramesh. "Vehicle Detection Using Image Processing",	1%

2020 IEEE International Conference for Innovation in Technology (INOCON), 2020

Publication

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		

FORMAT-I

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES

(To be attached in the dissertation/ project report)

1	Name of the Candidate **(IN BLOCK LETTERS)**	V.G.HARI KIRAN
2	Address of the Candidate	D.No.24-3-33/11-1, Padmavathi Nagar, Padmavathi Granite Shop backside, Mothukapalli, Hindupur.515201
3	Registration Number	RA1911003010840
4	Date of Birth	06 November 2001
5	Department	Computer Science and Engineering.
6	Faculty	Engineering and Technology, School of Computing Technologies.
7	Title of the Dissertation/Project	Automated Identification of Vehicles Using Deep Learning Neural Networks
8	Whether the above project /dissertation is done by	~~Individual~~ or group : (Strike whichever is not applicable) (a)If the project/ dissertation is done in group, then how many students together completed the project : 2 (Two) (b)Mention the Name & Register number of other candidate: SATHI SRIRAMA KUMARA SWAMY & RA1911003010827
9	Name and address of the Supervisor / Guide	Mrs. M. Vijayalakshmi Assistant Professor Department of Computer Science and Engineering SRM Institute of Science and Technology Kattankulatur - 603203. **Mail ID:** vijayalm@srmist.edu.in Mobile Number: 9884450223
10	Name and address of Co-Supervisor / Co- Guide (if any)	NIL **Mail ID: Mobile Number:**

11	Software Used	Turnitin
12	Date of Verification	11– May -2023
13	Plagiarism Details: (to attach the final report from the software)	

Chapter	Title of the Chapter	Percentage of similarity index (including self citation)	Percentage of similarity index (Excluding self citation)	% of plagiarism after excluding Quotes, Bibliography, etc.,
1	INTRODUCTION	1%	1%	1%
2	LITERATURE SURVEY	1%	1%	1%
3	PROPOSED METHODOLOGIES	1%	1%	1%
4	SYSTEM STUDY	1%	1%	1%
5	SYSTEM TESTING	4%	4%	3%
6	PROJECT DEMONSTRATION AND RESULTS	0%	0%	0%
7	CONCLUSION	0%	0%	1%
8	FUTURE ENHANCEMENTS	0%	0%	0%
9	APPENDIX-1	1%	1%	1%
10	APPENDIX-2	0%	0%	0%
	Appendices	4%	2%	2%

I / We declare that the above information have been verified and found true to the best of my / our knowledge.

S. Sriram Kumara Swamy

S Srirama Kumara Swamy
&
V G Hari Kiran
Signature of the Candidate

16/5/23

Mrs.M.Vijayalakshmi
Name & Signature of the Staff
(Who uses the plagiarism check software)

17/5/23

Mrs.MVijayalakshmi
Name & Signature of the Supervisor/ Guide

Name & Signature of the Co-Supervisor/ /Co- Guide (if any)

M. Pushpalatha

Dr.M.Pushpalatha
Name & Signature of the HOD