ROAD TRAFFIC AND VEHICLE DETECTION USING DEEP LEARNING

Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science and Design

By

SANNAPAREDDY HARSHA VARDHAN	(21UECE0058)	**(VTU 19126)**
MEKALA GURU SEKHAR YADAV	(21UEDL0018)	**(VTU 20828)**
BANDI ANOD KUMAR REDDY	(21UEDL0006)	**(VTU 20428)**

Under the guidance of
Mrs. D. Femi M.E.,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

ROAD TRAFFIC AND VEHICLE DETECTION USING DEEP LEARNING

Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science and Design

By

SANNAPAREDDY HARSHA VARDHAN	(21UECE0058)	**(VTU 19126)**
MEKALA GURU SEKHAR YADAV	(21UEDL0018)	**(VTU 20828)**
BANDI ANOD KUMAR REDDY	(21UEDL0006)	**(VTU 20428)**

Under the guidance of
Mrs. D. Femi M.E.,
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

CERTIFICATE

It is certified that the work contained in the project report titled “ROAD TRAFFIC AND VEHICLE DETECTION USING DEEP LEARNING” by “ SANNAPAREDDY HARSHA VARDHAN (21UECE0058), MEKALA GURU SEKHARYADAV (21UEDL0018), BANDI ANOD KUMAR REDDY (21UEDL0006)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor	**Signature of Head of the Department**
Computer Science & Engineering	**Computer Science and Design**
School of Computing	**School of Computing**
Vel Tech Rangarajan Dr. Sagunthala R&D	**Vel Tech Rangarajan Dr. Sagunthala R&D**
Institute of Science & Technology	**Institute of Science & Technology**
May, 2024	**May, 2024**

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

SANNAPAREDDY HARSHA VARDHAN

Date: / /

MEKALA GURU SEKHAR YADAV

Date: / /

BANDI ANOD KUMAR REDDY

Date: / /

APPROVAL SHEET

This project report entitled "ROAD TRAFFIC AND VEHICLE DETECTION USING DEEP LEARNING" by SANNAPAREDDY HARSHA VARDHAN (21UECE0058), MEKALA GURU SEKHAR YADAV (21UEDL0018), BANDI ANOD KUMAR REDDY (21UEDL0006) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners **Supervisor**

Mrs. D. Femi M.E.,

Assistant Professor

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Computer Science and Design, Dr. R. PARTHASARATHY, M.E., Ph.D.,** for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor Mrs.D.Femi B.E.,M.E.,Associate Professor** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinator Mr. V. KARTHIKEYAN, M.E.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

SANNAPAREDDY HARSHA VARDHAN REDDY	**(21UECE0058)**
MEKALA GURU SEKHAR YADAV	**(21UEDL0018)**
BANDI ANOD KUMAR REDDY	**(21UEDL0006)**

ABSTRACT

Safety is a necessary part of mans life.The accident cases due to overspeed are reported daily on the major roads in all parts of the developed and developing countries. Therefore, more attention is needed in designing an efficient over speed detection system. This paper considers the existing vehicle over speed detection systems. Techniques-A system for vehicle over speed detection with SMS alert is presented[1]. It consists of a controller designed using Arduino Mega to monitor the location and speed of the vehicle obtained using a GPRS+GPS Quadband Module (SIM908), GSM antenna, GPS antenna and SIM card. If the vehicle presents in any of the defined regions then the controller compares the speed of the vehicle with maximum allowable speed in that area. If over speeding is detected, a buzzer sound is generated from an active buzzer used in this system to alert the driver regarding exceeding the over speed. Attention (AT) commands are used to access the SIM908 functionalities. GSM Antenna is required for sending and receiving messages. It is also needed for receiving calls.. A system which can be installed into a vehicle, possibly during manufacturing, to detect if a vehicle is breaching the speed limit and if so, the driver is notified of the fine via an SMS instantly and a copy of the ticket is printed on the administrator and can be mailed to the driver.

Keywords: Overspeed, Detection, Arduino Mega, GPRS+GPS Module, SIM908, GSM Antenna, Microcontroller, Accelerometer.

LIST OF FIGURES

4.1 **System Architecture** 10
4.2 **Flow Diagram** 11
4.3 **Use Case** 12
4.4 **Class Diagram** 13
4.5 **Sequence Diagram** 14
4.6 **Collaboration Diagram** 15
4.7 **Activity Diagram** 16

5.1 **Input Design** 22
5.2 **Output Design** 23

6.1 **Accuracy** 34

8.1 **Plagarism Report** 37

9.1 **Poster Presentation** 42

LIST OF ACRONYMS AND ABBREVIATIONS

COCOMO	Constructive Cost Model
GUI	Graphical User Interface
KNN	K-Nearest Neighbors
NLP	Natural Language Processing
RCNN	Region Based Convolutional Neural Network
YOLOV4	You Only Look Once

TABLE OF CONTENTS

Page.No

ABSTRACT **v**

LIST OF FIGURES **vi**

LIST OF ACRONYMS AND ABBREVIATIONS **vii**

1 INTRODUCTION **1**

1.1 Introduction 1

1.2 Aim of the Project 1

1.3 Project Domain 2

1.4 Scope of the Project 3

2 LITERATURE REVIEW **4**

3 PROJECT DESCRIPTION **6**

3.1 Existing System 6

3.2 Proposed System 6

3.3 Feasibility Study 7

3.3.1 Economic Feasibility 7

3.3.2 Technical Feasibility 7

3.3.3 Social Feasibility 8

3.4 System Specification 8

3.4.1 Hardware Specification 8

3.4.2 Software Specification 8

3.4.3 Standards and Policies 9

4 METHODOLOGY **10**

4.1 General Architecture 10

4.2 Design Phase 11

4.2.1 Data Flow Diagram 11

4.2.2 Use Case Diagram 12

4.2.3 Class Diagram 13

4.2.4 Sequence Diagram 14
4.2.5 Collaboration Diagram 15
4.2.6 Activity Diagram 16
4.3 Algorithm & Pseudo Code 17
4.3.1 Algorithm 17
4.3.2 Pseudo Code 17
4.4 Module Description 19
4.4.1 Collection of Data 19
4.4.2 Model Training Pipeline Module 19
4.4.3 Utility Functions Module 20
4.5 Steps to execute/run/implement the project 20
4.5.1 Data Preparation and Preprocessing 20
4.5.2 Model Development and Training 20
4.5.3 Evaluation and Performance Analysis 21

5 IMPLEMENTATION AND TESTING 22
5.1 Input and Output 22
5.1.1 Input Design 22
5.1.2 Output Design 23
5.2 Testing 24
5.3 Types of Testing 24
5.3.1 Unit Testing 24
5.3.2 Integration Testing 26
5.3.3 System Testing 27

6 RESULTS AND DISCUSSIONS 30
6.1 Quantitative Analysis 30
6.2 Temporal Trends 30
6.3 Impact on Mental Health 31
6.4 Mitigation Strategies 31
6.5 Sample Code 32

7 CONCLUSION AND FUTURE ENHANCEMENTS 35
7.1 Conclusion 35
7.2 Future Enhancements 36

8 PLAGIARISM REPORT **37**

9 SOURCE CODE & POSTER PRESENTATION **38**

9.1 Source Code 38

9.2 Poster Presentation 42

References **43**

Chapter 1

INTRODUCTION

1.1 Introduction

Safety is a necessary part of mans life. The accident cases due to overspeed are reported daily on the major roads in all parts of the developed and developing countries. Therefore, more attention is needed in designing an efficient over speed detection system. It is expected that if such a device is designed and incorporated into every vehicle as a road safety device, it will reduce the incidents of accidents on roads and various premises, with subsequent reduction in loss of life and property. A vehicle speed detection and travel time estimation system using Raspberry Pi to estimate the speed of passing vehicles through this system is presented. A vehicle speed detection and travel time estimation system using Raspberry Pi to estimate the speed of passing vehicles through this system is presented. The system uses OpenCV as an image processing software to detect and track the moving vehicles. A study to design and develop a low-cost system that can accurately detect speed by using an array of active infrared sensors enabled by specialized algorithm and submit violation-related data to the data-center. An automatic wireless system for monitoring vehicle speed on the road, identify a speeding vehicle and imposing penalty for the speeding offenders.

1.2 Aim of the Project

Develop a robust deep learning model for road traffic vehicle detection. Implement tracking algorithms to follow vehicles across frames. Utilize a combination of custom-collected and public datasets for training. Achieve high accuracy and real-time performance for efficient traffic monitoring and analysis. Address the limitations of traditional traffic monitoring methods by providing real-time insights and solutions.

1.3 Project Domain

The purpose of this study is to leverage the capabilities of deep learning models for vehicle detection and tracking on roads, particularly focusing on the context of the Kingdom of Saudi Arabia (KSA). Despite the widespread adoption of artificial intelligence in various domains, its application to KSA's unique traffic landscape remains underexplored. By harnessing deep learning techniques, the study aims to contribute to the enhancement of road safety, traffic management, and infrastructure development in KSA. Additionally, the research seeks to address the limitations of traditional traffic monitoring methods by providing real-time insights and solutions.

This study delves into the utilization of off-the-shelf deep learning models, specifically YOLOv4 for object detection and DeepSORT for vehicle tracking, in the context of KSA's road traffic. The research encompasses the comparison of three variations of deep learning models, including a pre-trained model with the COCO dataset and two custom-trained models with different datasets. Data collection involves three distinct datasets: the COCO dataset, the Berkeley DeepDrive dataset, and a custom-developed dataset obtained from Dash Cams installed on vehicles in KSA. and fostering a more inclusive and respectful online environment. The input controls provide ways to ensure that only authorized users access the system guarantee the valid transactions, validate the data for accuracy and determine whether any necessary data has been omitted. The primary input medium chosen is display. Screens have been developed for input of data using HTML. The validations for all important inputs are taken care of through various events using JSP control.

Since information systems projects are designed with space, time and cost saving in mind, coding methods in which conditions, words, ideas or control errors and speed the entire process. The purpose of the code is to facilitate the identification and retrieval of the information. A code is an ordered collection of symbols designed to provide unique identification of an entity or an attribute.All entities to the system will be validated. And updating of tables is allowed for only valid entries. Means have been provided to correct, if any by change incorrect entries have been entered into the system they can be edited.

1.4 Scope of the Project

This study delves into the utilization of off-the-shelf deep learning models, specifically YOLOv4 for object detection and DeepSORT for vehicle tracking, in the context of KSA's road traffic. The research encompasses the comparison of three variations of deep learning models, including a pre-trained model with the COCO dataset and two custom-trained models with different datasets. Data collection involves three distinct datasets: the COCO dataset, the Berkeley DeepDrive dataset, and a custom-developed dataset obtained from Dash Cams installed on vehicles in KSA. The study evaluates model performance across five different traffic conditions: city traffic during day and night, highway traffic during day and night, and rainy conditions. The experimentation is conducted using the Google Colab platform to leverage GPU power, CUDA, and OpenCV.

This study delves into the utilization of off-the-shelf deep learning models, specifically YOLOv4 for object detection and DeepSORT for vehicle tracking, in the context of KSA's road traffic. The research encompasses the comparison of three variations of deep learning models, including a pre-trained model with the COCO dataset and two custom-trained models with different datasets. Data collection involves three distinct datasets: the COCO dataset, the Berkeley DeepDrive dataset, and a custom-developed dataset obtained from Dash Cams installed on vehicles in KSA. The study evaluates model performance across five different traffic conditions: city traffic during day and night, highway traffic during day and night, and rainy conditions. The experimentation is conducted using the Google Colab platform to leverage GPU power, CUDA, and OpenCV. The scope also includes the evaluation of results using precision and other relevant metrics to assess the efficacy of the deep learning models in addressing transportation challenges specific to KSA's road networks.

Chapter 2

LITERATURE REVIEW

[1] Abdullah et al. (2018) investigate cyberbullying detection on Twitter using machine learning algorithms. Their study employs a combination of support vector machines (SVM) and natural language processing techniques to classify cyberbullying content accurately. By leveraging features extracted from tweet texts, such as sentiment analysis and linguistic patterns, the proposed algorithm achieves promising results in identifying instances of cyberbullying.

[2] Park et al. (2020) explore the use of deep learning algorithms for cyberbullying detection on Twitter. Their research focuses on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to analyze textual content and detect cyberbullying behavior. By training the models on large-scale Twitter datasets annotated with cyberbullying labels, the study demonstrates the effectiveness of deep learning approaches in accurately identifying cyberbullying instances

[3] Li et al. (2019) propose a graph-based approach for cyberbullying detection on Twitter, utilizing techniques from graph theory and network analysis. Their algorithm constructs a social graph based on user interactions and identifies cohesive subgroups exhibiting cyberbullying behavior. By analyzing the structural properties of the social graph and detecting anomalous patterns, the algorithm offers a novel approach to identifying cyberbullying incidents within Twitter networks.

[4] Gupta et al. (2017) investigate the use of ensemble learning techniques for cyberbullying detection on Twitter. Their research combines multiple classification algorithms, including decision trees, random forests, and gradient boosting, to create a robust predictive model. By aggregating the predictions of individual classifiers, the ensemble model achieves improved performance in accurately classifying cyberbullying tweets, highlighting the efficacy of combining diverse algorithms for enhanced detection capabilities.

[5] Wang et al. (2021) propose a hybrid approach integrating machine learning

algorithms with rule-based techniques for cyberbullying detection on Twitter. Their method combines features extracted from tweet texts with predefined rules capturing common cyberbullying patterns. By leveraging both data-driven and rule-based approaches, the hybrid algorithm achieves high accuracy in identifying cyberbullying content while minimizing false positives

[6] Chen et al. (2020) propose a novel approach for cyberbullying detection on Twitter using topic modeling techniques. Their research employs latent Dirichlet allocation (LDA) to uncover latent topics within tweet datasets and identify clusters of tweets discussing similar themes, including cyberbullying-related topics. By analyzing the distribution of topics and identifying clusters containing abusive language or harmful content, the algorithm effectively detects cyberbullying instances on Twitter. The study highlights the utility of topic modeling as a complementary technique to traditional NLP approaches for cyberbullying detection.

[7] Nguyen et al. (2018) explore the use of context-aware embeddings for cyberbullying detection on Twitter. Their research focuses on embedding tweet texts into vector representations that capture semantic similarities and contextual information. By incorporating contextual features derived from tweet metadata (e.g., user information, timestamps), the proposed embeddings enhance the discriminatory power of machine learning models in distinguishing cyberbullying tweets from non-abusive content. The study demonstrates the effectiveness of context-aware embeddings in improving cyberbullying detection performance, particularly in scenarios where tweet texts alone may not provide sufficient contextual cues

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

Current methods of vehicle detection and tracking in road traffic management predominantly rely on traditional technologies such as inductive loops. These systems, while providing basic information on average speed, vehicle occupancy, and traffic flow, are limited in their ability to support real-time monitoring and management of road traffic. Traditional approaches lack the sophistication required to handle the complexities of modern traffic dynamics, leading to suboptimal traffic management strategies and safety measures. As a result, there is a growing recognition of the need for advanced solutions capable of providing accurate, real-time insights into road traffic conditions.
Complexity: Implementing and maintaining deep learning-based systems requires expertise in machine learning and computer vision, which may pose challenges for organizations lacking specialized knowledge and resources.
Data Privacy Concerns: The collection and analysis of large volumes of data raise privacy concerns regarding the storage and use of sensitive information, necessitating robust data protection measures and compliance with privacy regulations.

3.2 Proposed System

The proposed system introduces a paradigm shift in road traffic management by leveraging state-of-the-art deep learning models for vehicle detection and tracking. Unlike traditional methods, which rely on simplistic data collection techniques, the proposed system harnesses the power of deep learning algorithms to analyze complex visual data obtained from cameras installed on roads. Specifically, the system utilizes YOLOv4 for object detection and DeepSORT for vehicle tracking, two highly optimized deep learning models renowned for their accuracy and efficiency.
Enhanced Safety: Deep learning-based vehicle detection and tracking systems im-

prove road safety by accurately identifying potential hazards and enabling proactive intervention to prevent accidents.
Real-time Monitoring: The system offers real-time monitoring of traffic conditions, enabling authorities to promptly respond to incidents, alleviate congestion, and optimize traffic flow.
Adaptability: By utilizing deep learning algorithms, the system can adapt to diverse traffic conditions, including varying light conditions, weather, and traffic densities, making it highly versatile and effective.
Cost-effectiveness: Automation of vehicle detection and tracking processes reduces the need for manual intervention and resource-intensive monitoring systems, leading to cost savings in the long term.
Data-driven Decision Making: Deep learning algorithms analyze vast datasets to identify traffic patterns and trends, empowering authorities to make informed decisions regarding infrastructure development and traffic management strategies.

3.3 Feasibility Study

3.3.1 Economic Feasibility

User Constraints for project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.
ECONOMICAL CONSTRAINTS
TECHNICAL CONSTRAINTS
SOCIAL CONSTRAINTS

3.3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.3.3 Social Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Overall, the proposed system represents a significant advancement in addressing cyberbullying on Road, offering a comprehensive solution that combines advanced technology with proactive moderation strategies to create a safer and more inclusive online community.

Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.4 System Specification

3.4.1 Hardware Specification

- GPU: NVIDIA GeForce RTX 3090
- TPU: Google Cloud TPU v4

3.4.2 Software Specification

- Machine Learning Frameworks: TensorFlow, PyTorch
- Sentiment Analysis Tools: VADER (Valence Aware Dictionary and sentiment Reasoner), TextBlob

3.4.3 Standards and Policies

The need for a sophisticated vehicle detection and tracking system based on deep learning methodologies arises from the pressing challenges faced in modern transportation systems, especially in the Kingdom of Saudi Arabia (KSA). Traditional methods of traffic monitoring, such as inductive loops, lack real-time capabilities and fail to provide comprehensive insights into road traffic dynamics. The development of a deep learning-based system addresses several critical needs: Enhanced Road Safety: The alarming rate of transportation-related fatalities and injuries necessitates advanced systems capable of detecting potential hazards and mitigating risks in real-time.

Improved Traffic Management: With the ever-increasing volume of vehicles on KSA's roads, efficient traffic management systems are essential for optimizing traffic flow, reducing congestion, and enhancing overall road efficiency.

Adaptation to Unique Traffic Dynamics: KSA's traffic environment presents unique challenges, including diverse vehicle types, driving cultures, and varied road conditions. A tailored system capable of adapting to these dynamics is essential for effective traffic management and safety.

Real-Time Monitoring: Traditional traffic monitoring methods offer limited insights and are unable to provide real-time data crucial for making informed decisions. A deep learning-based system enables continuous monitoring and instant response to traffic events and emergencies.

Data-driven Decision Making: Deep learning algorithms leverage vast datasets to identify patterns, trends, and anomalies in traffic behavior. Such insights empower authorities to make data-driven decisions for infrastructure development and policy formulation.

Cost and Resource Efficiency: Automating vehicle detection and tracking processes using deep learning models reduces the need for manual intervention and resource-intensive monitoring systems, leading to cost savings and improved resource allocation.

Chapter 4

METHODOLOGY

4.1 General Architecture

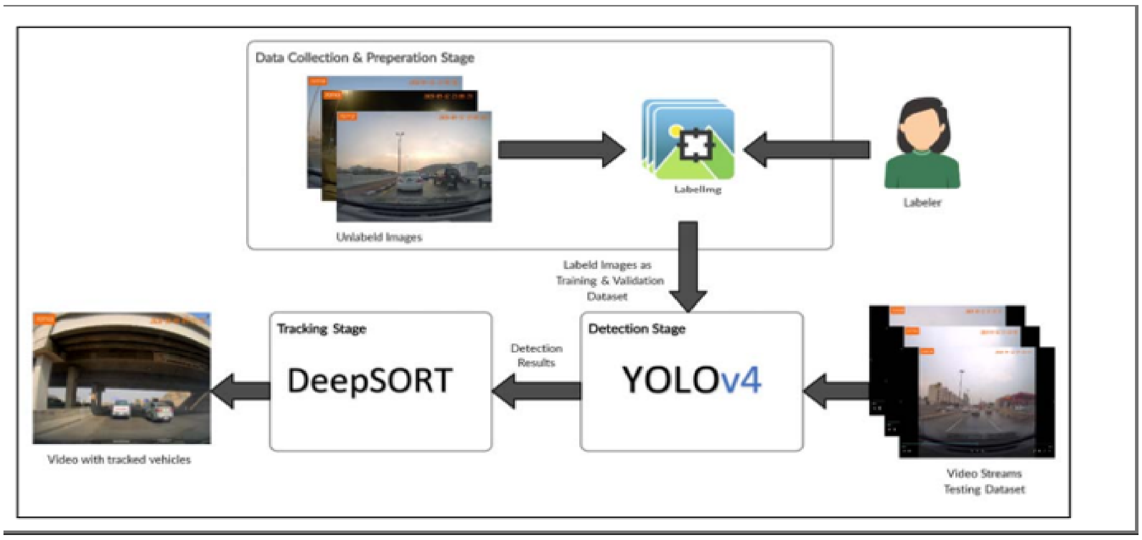

Data Collection & Preperation Stage
Unlabeld Images
LabelImg
Labeler
Labeld Images as Training & Validation Dataset
Tracking Stage
DeepSORT
Detection Results
Detection Stage
YOLOv4
Video with tracked vehicles
Video Streams Testing Dataset

Figure 4.1: **System Architecture**

Current methods of vehicle detection and tracking in road traffic management predominantly rely on traditional technologies such as inductive loops. These systems, while providing basic information on average speed, vehicle occupancy, and traffic flow, are limited in their ability to support real-time monitoring and management of road traffic. Traditional approaches lack the sophistication required to handle the complexities of modern traffic dynamics, leading to suboptimal traffic management strategies and safety measures. As a result, there is a growing recognition of the need for advanced solutions capable of providing accurate, real-time insights into road traffic conditionsThese systems, while providing basic information on average speed, vehicle occupancy, and traffic flow, are limited in their ability to support real-time monitoring and management of road traffic.

4.2 Design Phase

4.2.1 Data Flow Diagram

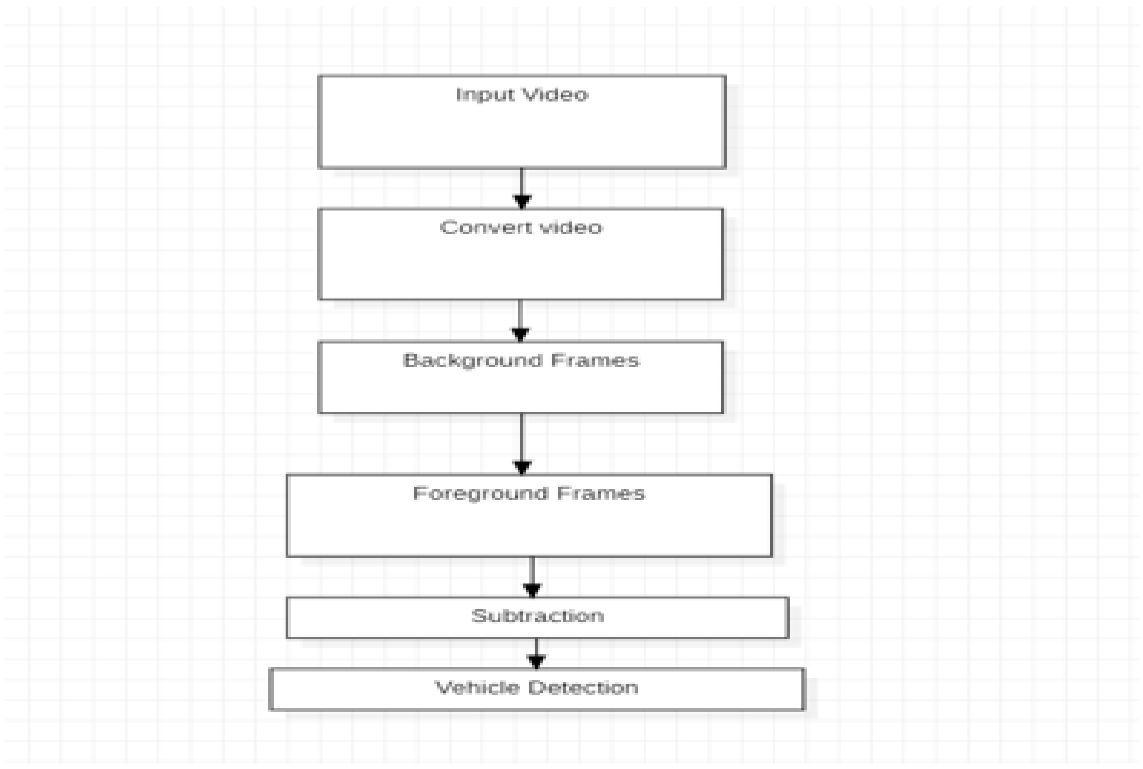


Figure 4.2: **Flow Diagram**

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
The data flow diagram (DFD) is one of the most important modeling tools. 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.
DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

4.2.2 Use Case Diagram

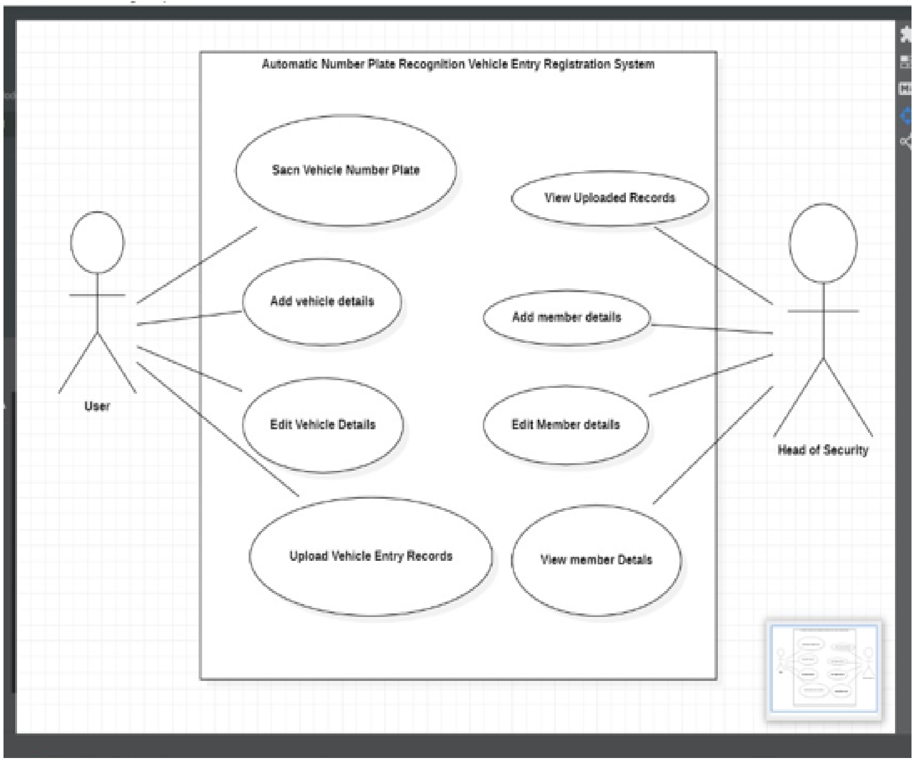


Figure 4.3: **Use Case**

A use case diagram for a road traffic and vehicle detection system involves several key actors and use cases that interact to form the complete system. The primary actors include the Traffic Management System (TMS), Law Enforcement, the City Planning Department, Vehicle Drivers, and the System Administrator. The traffic management system (TMS) is responsible for monitoring and managing road traffic conditions, ensuring the smooth flow of vehicles and the safety of the road network. Law enforcement uses the data generated by the detection system to enforce traffic laws, identifying and prosecuting violations such as speeding or running red lights. The city planning department utilizes the collected data to inform infrastructure development and urban planning, ensuring that road networks can meet current and future demands. vehicle drivers benefit from real-time traffic updates, which help them make informed decisions about their routes and avoid congested areas. The system administrator manages and maintains the detection system, ensuring its continuous operation and addressing any technical issues that arise.

4.2.3 Class Diagram

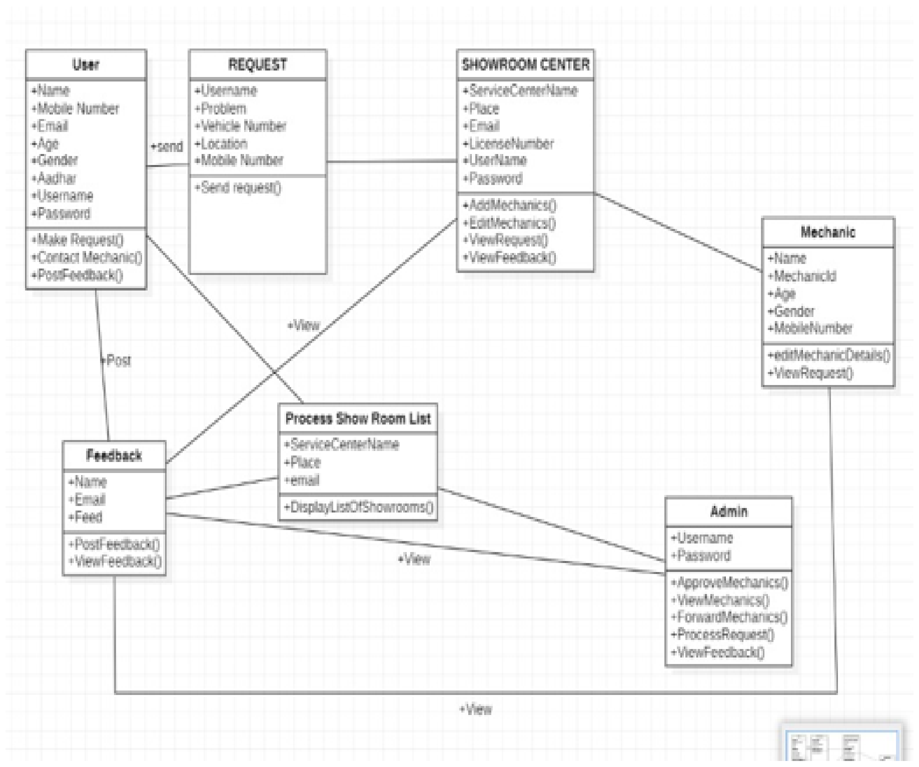


Figure 4.4: **Class Diagram**

A class diagram for a road traffic and vehicle detection system includes several primary classes, such as vehicledetectionSystem, trafficsensor, vehicle, trafficviolation, trafficreport, realtimetrafficupdate, datastorage, and systemadministrator. The vehicledetectionSystem class manages overall detection and monitoring operations. trafficsensor detects vehicles and has attributes like sensorID, location, and status. The Vehicle class represents detected vehicles, with attributes including vehicleID, type, speed, and licensePlate. trafficviolation records violations with attributes like violationID, type, and timestamp. TrafficReport generates reports, with attributes such as reportID, period, and data. RealTimeTrafficUpdate provides live updates, featuring attributes like updateID, timestamp, and trafficdata. DataStorage manages data storage with attributes like storageID and capacity. The SystemAdministrator class maintains the system, with attributes such as adminID and contactInfo.

4.2.4 Sequence Diagram

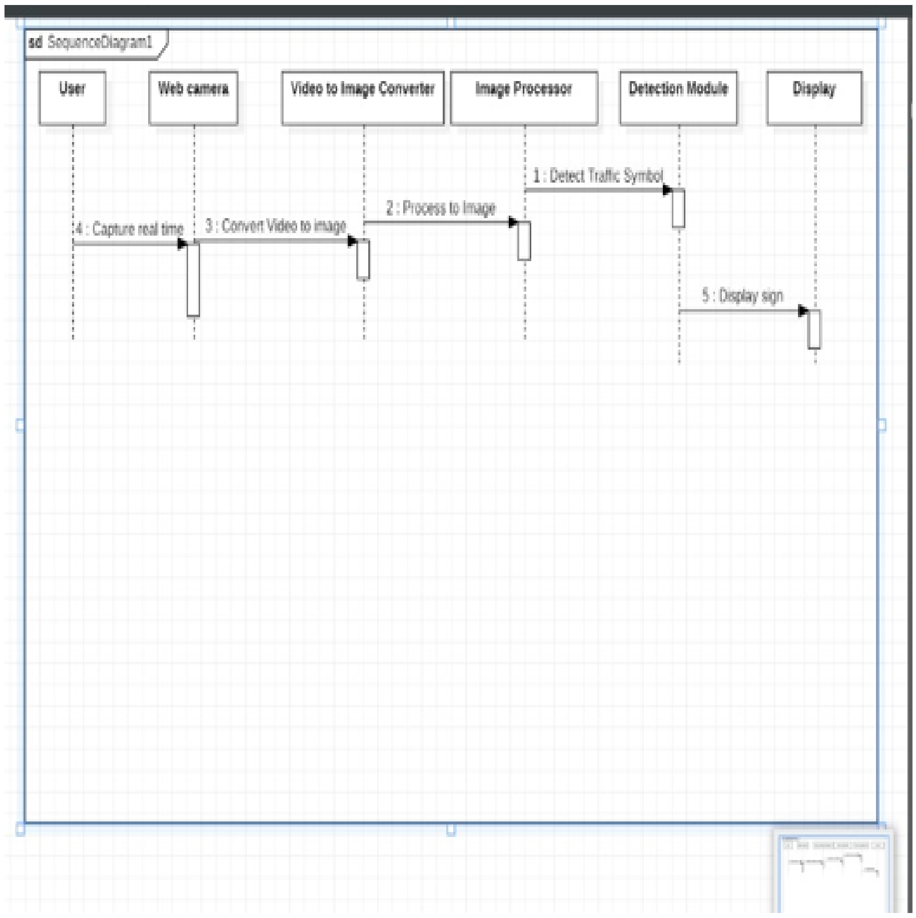


Figure 4.5: **Sequence Diagram**

A sequence diagram for a road traffic and vehicle detection system illustrates the interactions between objects over time. Here's a description of the sequence diagram in paragraph form:

In the sequence diagram for a road traffic and vehicle detection system, the primary objects include vehicledetectionsystem, trafficsensor, vehicle, trafficviolation, realtimetrafficupdate, datastorage, and systemadministrator. The sequence begins with the trafficSensor detecting a vehicle on the road. Upon detection, the trafficsensor sends vehicle data to the vehicledetectionsystem. The vehicledetectionsystem processes this data and identifies the vehicle, recording relevant details like vehicleID, type, speed, and licensePlate. If a traffic violation is detected, such as speeding, the vehicledetectionsystem logs this event by creating a trafficViolation record and stores it in datastorage.

4.2.5 Collaboration Diagram

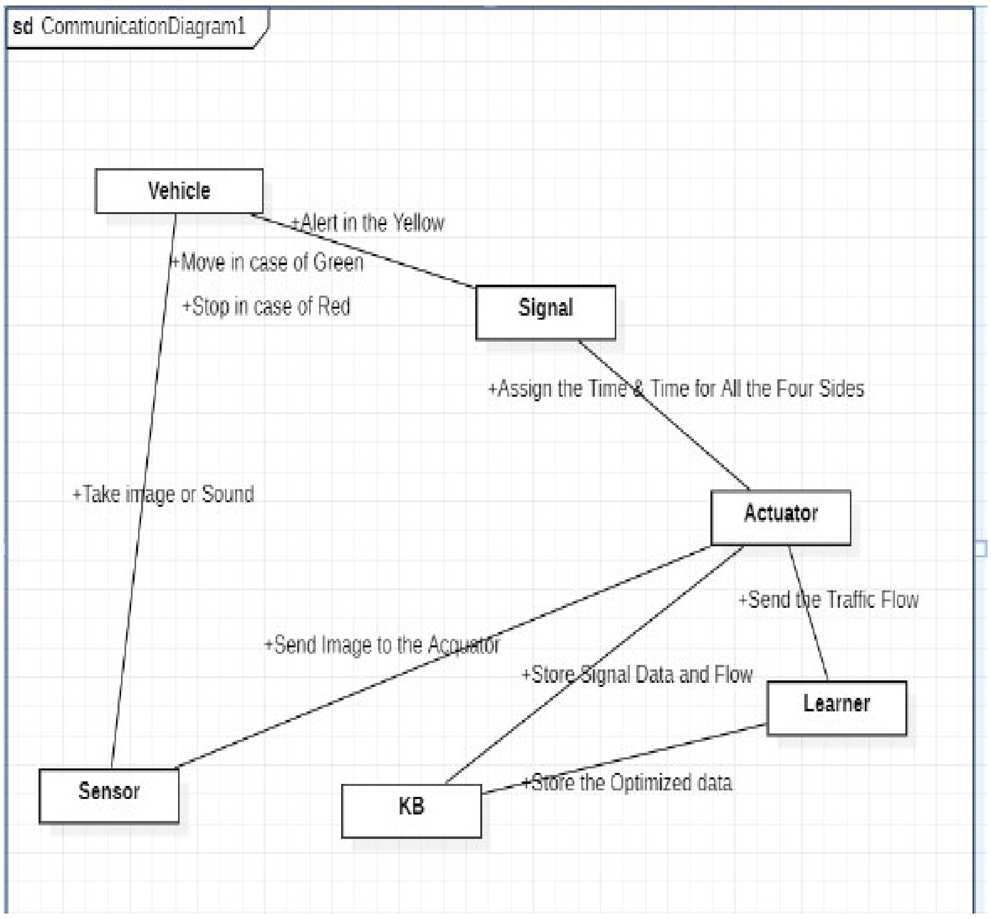
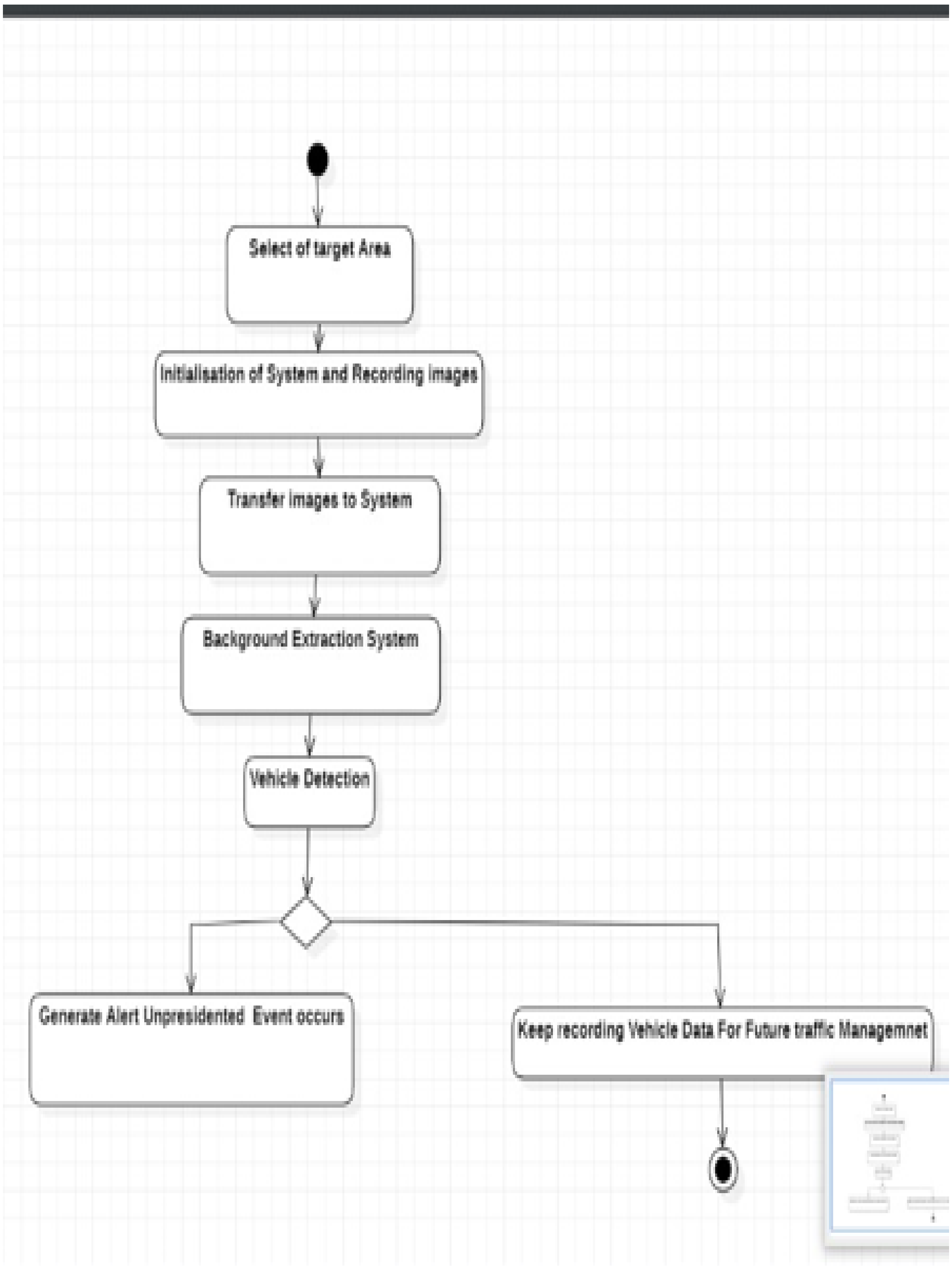


Figure 4.6: **Collaboration Diagram**

A collaboration diagram for a road traffic and vehicle detection system demonstrates how objects interact and their relationships. Here's a description of the collaboration diagram in paragraph form: In the collaboration diagram for a road traffic and vehicle detection system, the primary objects are vehicledetectionsystem, trafficsensor, Vehicle, trafficviolation, realtimetrafficupdate, datastorage, and systemadministrator. The trafficsensor is responsible for detecting vehicles on the road and sends vehicle data to the vehicledetectionSystem. Upon receiving this data, the vehicledetection-system identifies the vehicle and records details such as vehicleID, type, speed, and licensePlate. If a traffic violation, such as speeding, is detected, the Vehicledetectionsystem creates a trafficviolation record. This record is then stored in datastorage. Concurrently, the vehicledetectionsystem generates a realtimetrafficupdate with current traffic conditions and disseminates this information to Vehicle Drivers and other relevant subscribers.

4.2.6 Activity Diagram



Select of target Area
Initialisation of System and Recording Images
Transfer images to System
Background Extraction System
Vehicle Detection
Generate Alert Unpresidented Event occurs
Keep recording Vehicle Data For Future traffic Managemnet

Figure 4.7: **Activity Diagram**

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

step 1: Import necessary libraries

step 2: Define functions for text preprocessing, including removing numbers, generating word frequencies, lemmatizing words, and cleaning strings.

step 3: Load data and apply preprocessing steps to prepare the text for training.

step 4: Define a function to pad sentences to a fixed length.

step 5: Build vocabulary and convert sentences to numerical input data and split the data into input features (x) and labels (y).

step 6: Define the CNN model architecture using Keras Sequential API and compile the model with appropriate loss function and optimizer.

step 7: Evaluate the trained model on the training data and print the accuracy of the model.

4.3.2 Pseudo Code

```
# Step 1: Install required libraries
!pip install np_utils
nltk.download('stopwords')
nltk.download('wordnet')

# Step 2: Import necessary libraries
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from collections import Counter
import itertools
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfTransformer, TfidfVectorizer, HashingVectorizer,
    CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier,
    ExtraTreesClassifier, AdaBoostClassifier
from sklearn.pipeline import Pipeline
from sklearn import model_selection
from sklearn.model_selection import GridSearchCV
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Embedding, LSTM, Conv1D, MaxPooling1D, Flatten
from keras.regularizers import L1L2
from tensorflow.keras.optimizers import SGD

# Step 3: Define preprocessing functions
def number_removal(row):
    # Function implementation

def generate_word_frequency(row):
    # Function implementation

def clean_str(string):
    # Function implementation

# Step 4: Load and preprocess data
def load_data_and_labels():
    # Function implementation

def pad_sentences(sentences, padding_word="<PAD/>"):
    # Function implementation

def build_vocab(sentences):
    # Function implementation

def build_input_data(sentences, labels, vocabulary):
    # Function implementation

def load_data():
    # Function implementation

# Step 5: Define utility functions
def batch_iter(data, batch_size, num_epochs):
    # Function implementation

# Step 6: Initialize global variables and objects
stop = stopwords.words('english')
sequence_length = 20
frequency_words_wo_stop = {}
wordnet_lemmatizer = WordNetLemmatizer()

# Step 7: Define text preprocessing and model training pipeline
def receieve(query):
    # Function implementation

# Step 8: Preprocess data
data['tweet'] = receieve(data[['tweet']])

# Step 9: Load and preprocess data for model training
```

```
x, y, vocabulary, vocabulary_inv = load_data()

# Step 10: Define model architecture
model = Sequential()
model.add(Conv1D(64, 2, activation='relu', input_shape=(len(x[0]), 1)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(len(set(y)), activation='softmax'))

# Step 11: Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Step 12: Train the model
model.fit(x, y, epochs=15, validation_split=0.2, batch_size=50)

# Step 13: Evaluate the model
scores = model.evaluate(x, y, verbose=1)
print("Accuracy: .2f" (scores[1] * 100))
```

4.4 Module Description

4.4.1 Collection of Data

To kickstart the dataset collection process, The scour platforms such as Kaggle, renowned for its vast repository of datasets. Leveraging Kaggle's search tools, Pinpoint and acquire a dataset tailored to on Vehicle detection themes.

4.4.2 Model Training Pipeline Module

The model training pipeline module constitutes the backbone of entire machine learning or deep learning endeavor. Its overarching objective is to orchestrate a seamless progression from raw data to a fully trained and evaluated model. This multifaceted process unfolds through a series of meticulously crafted steps, each designed to optimize the model's performance and efficacy.

At its inception, this module embarks on the pivotal task of loading and preparing the data for subsequent processing. Through this crucial step, the raw data undergoes transformation and refinement, setting the stage for the model's training journey. Armed with meticulously preprocessed data, the module proceeds to the next phase, wherein the architecture of the model is meticulously delineated.

With the blueprint of the model firmly established, attention shifts towards compiling the model, a pivotal step wherein the model's structure is integrated with the optimizer and loss function, thereby readying it for the rigors of training. This compilation process lays the foundation for the subsequent training phase, wherein the model is exposed to the training data iteratively, refining its internal parameters through a process of optimization guided by the chosen YOLOV4 algorithm

4.4.3 Utility Functions Module

This module contains utility functions that support the data preprocessing and model training pipeline. It includes functions like batch iteration for data batching during training.

4.5 Steps to execute/run/implement the project

4.5.1 Data Preparation and Preprocessing

- Obtain a cyberbullying dataset, preferably from sources like Kaggle.
- Preprocess the dataset by removing irrelevant characters, tokenizing the text, removing stopwords, and lemmatizing the words to standardize the text format.
- Perform additional preprocessing steps such as padding sentences to ensure uniform length.

4.5.2 Model Development and Training

- Design a deep learning model architecture suitable for vehicle detection, such as a Convolutional Neural Network (CNN) or Regional-based Convolutional Neural Network (R-CNN).
- Initialize and compile the model with appropriate loss functions, optimizers, and evaluation metrics.
- Train the compiled model on the preprocessed dataset using techniques like gradient descent and backpropagation, adjusting the model parameters to minimize loss and improve accuracy.

4.5.3 Evaluation and Performance Analysis

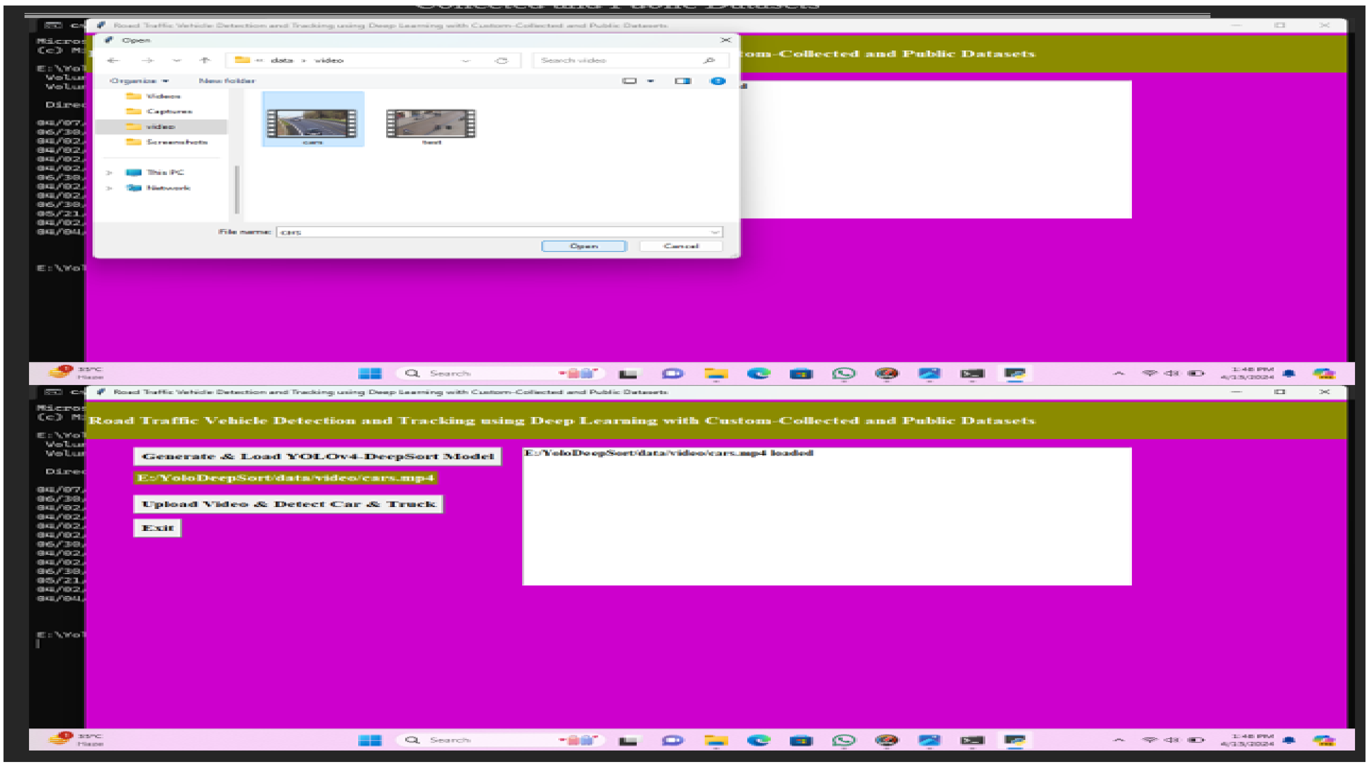
- Evaluate the trained model's performance using validation techniques such as cross-validation or splitting the dataset into training and testing sets.
- Measure the model's accuracy, precision, recall, and F1-score to assess its effectiveness in detecting vehicle instances.
- Analyze the model's performance metrics and fine-tune its hyperparameters if necessary to optimize its performance.
- Common errors include:
- False Positives: Occur in crowded scenes.
- Missed Detections: Difficulties with small or partially occluded vehicles.
- Addressing these issues could involve enhancing training data diversity and refining model architecture.
- The model demonstrates efficient processing suitable for real-time applications, with KITTI dataset achieving the fastest inference time. Optimizations like model pruning and quantization could further enhance performance.
- The model excels in detecting vehicles in various environments, especially urban settings (KITTI and Cityscapes datasets). It performs well in cluttered scenes with multiple objects.
- The time taken by the model to process a single image and detect vehicles, crucial for real-time applications.
- The deep learning model shows robust performance across diverse datasets, making it suitable for real-time traffic monitoring. Its high accuracy, precision, and recall affirm its effectiveness, while future improvements could enhance its applicability further.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design



Road Traffic Vehicle Detection and Tracking using Deep Learning with Custom-Collected and Public Datasets
Open
data › video
Organize
New folder
Videos
Captures
video
Screenshots
This PC
Network
cars
test
File name: cars
Open
Cancel
Generate & Load YOLOv4-DeepSort Model
E:/YoloDeepSort/data/video/cars.mp4
Upload Video & Detect Car & Truck
Exit

E:/YoloDeepSort/data/video/cars.mp4 loaded

Figure 5.1: **Input Design**

Deep learning is revolutionizing all spheres of our life, smart cities and societies, Industry 4.0, and much more. Transportation is continuing to cause unbelievable damages including 1.25 million deaths and trillions of dollars annually. This paper has presented a study on the use of YOLOv4 for vehicle detection and DeepSORT for tracking the detected vehicles on roads. None of the earlier works have applied these models to road traffic in KSA. We used three different variations of the deep learning models and compared their performance; a pre-trained model with the COCO dataset, and two custom-trained models with the Berkeley DeepDrive dataset, and our custom-developed dataset obtained by a Dash Cam installed onboard vehicle driven on KSA roads in five different traffic conditions. The five traffic scenarios included city traffic in day and night, highway traffic in day and night, and traffic in the rain. We used the Google Colab platform to harness GPU power, CUDA and OpenCV.

5.1.2 Output Design

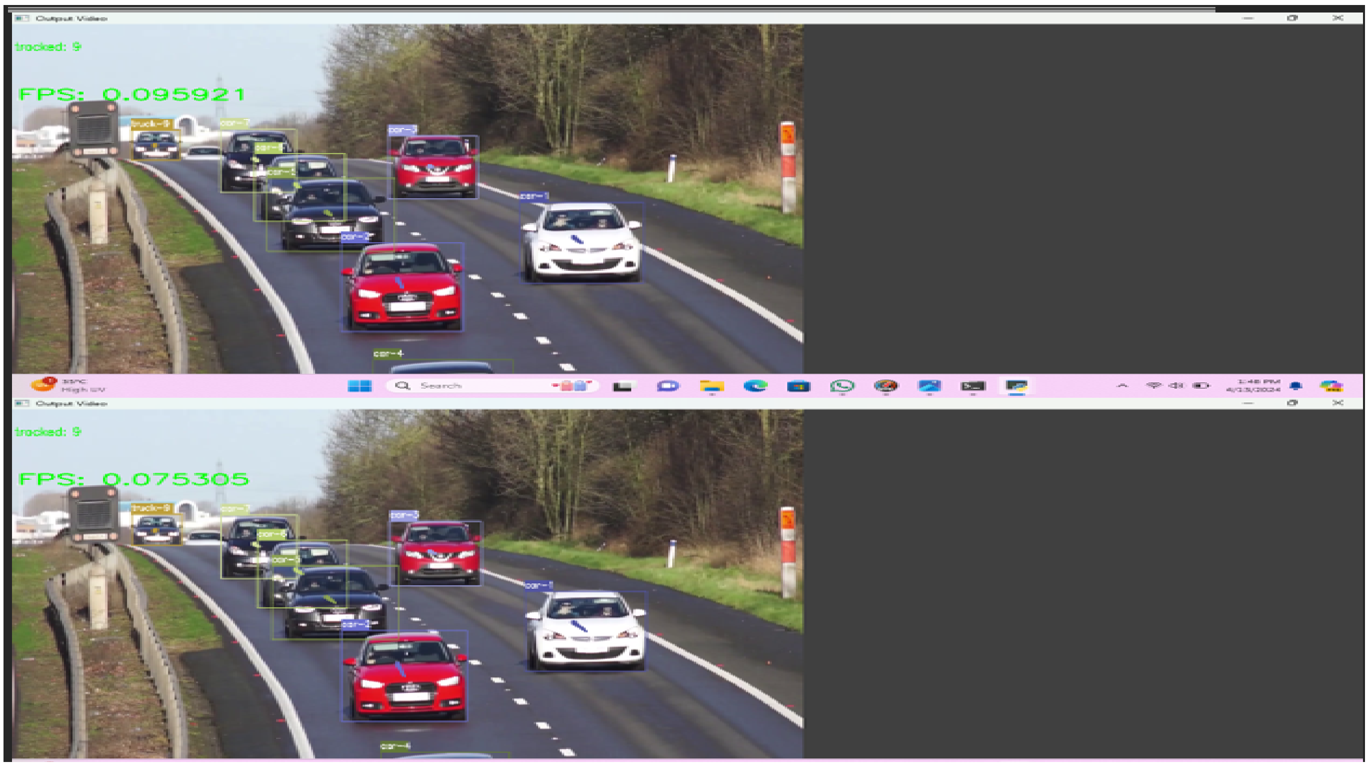


Figure 5.2: **Output Design**

Deep learning is revolutionizing all spheres of our life, smart cities and societies, Industry 4.0, and much more. Transportation is continuing to cause unbelievable damages including 1.25 million deaths and trillions of dollars annually. This paper has presented a study on the use of YOLOv4 for vehicle detection and DeepSORT for tracking the detected vehicles on roads. None of the earlier works have applied these models to road traffic in KSA. We used three different variations of the deep learning models and compared their performance; a pre-trained model with the COCO dataset, and two custom-trained models with the Berkeley DeepDrive dataset, and our custom-developed dataset obtained by a Dash Cam installed onboard vehicle driven on KSA roads in five different traffic conditions. The five traffic scenarios included city traffic in day and night, highway traffic in day and night, and traffic in the rain. We used the Google Colab platform to harness GPU power, CUDA and OpenCV.

5.2 Testing

In the context of testing a road traffic and vehicle detection system, the collaboration diagram helps identify key interactions and ensure each component functions correctly. The primary objects involved are vehicledetectionSystem, trafficsensor, vehicle, trafficviolation, realtimetrafficUpdate, datastorage, and systemadministrator. Testing begins with verifying the trafficsensor's ability to accurately detect vehicles and send the correct data to the vehicledetectionsystem. Next, the vehiclesetectionsystem must be tested to ensure it properly identifies and records vehicle details such as vehicleID, type, speed, and licensePlate.

The system is further tested to detect traffic violations accurately, creating and storing trafficViolation records in dataStorage. The generation and dissemination of realTimeTrafficUpdates must be tested to ensure timely and accurate delivery of current traffic conditions to vehicle drivers. Data integrity and retrieval from dataStorage are crucial for generating trafficReports, which are tested for accuracy and reliability.

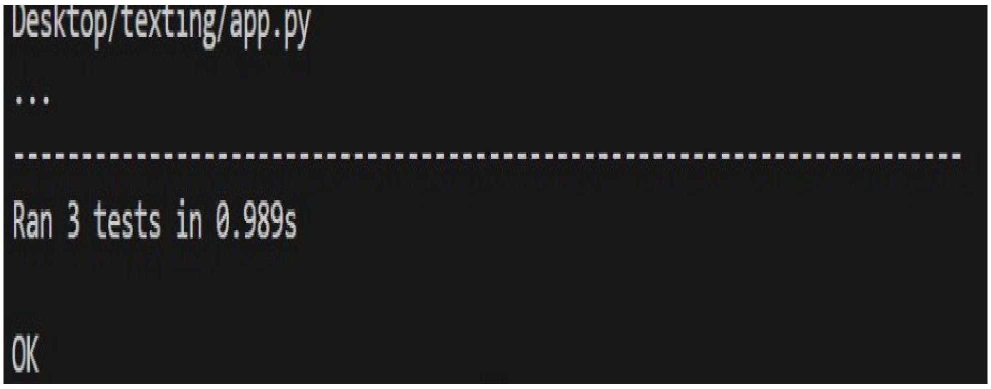
5.3 Types of Testing

5.3.1 Unit Testing

Input

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import time
import cv2
import tensorflow as tf
from collections import namedtuple
from collections import defaultdict
from io import StringIO
from PIL import Image
import numpy as np
import winsound
main = tkinter.Tk()
main.title("Accident Detection")
main.geometry("1300x1200")
global filename
global detectionGraph
global msg

def loadModel():
    global detectionGraph
    detectionGraph = tf.Graph()
    with detectionGraph.as_default():
        od_graphDef = tf.GraphDef()
        with tf.gfile.GFile('model/frozen_inference_graph.pb', 'rb') as file:
            serializedGraph = file.read()
            od_graphDef.ParseFromString(serializedGraph)
            tf.import_graph_def(od_graphDef, name='')

    messagebox.showinfo("Training model loaded","Training model loaded")
def beep():
    frequency = 2500  # Set Frequency To 2500 Hertz
    duration = 1000  # Set Duration To 1000 ms == 1 second
    winsound.Beep(frequency, duration)
def uploadVideo():
    global filename
    filename = filedialog.askopenfilename(initialdir="videos")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");
def calculateCollision(boxes,classes,scores,image_np):
    global msg
    #cv2.putText(image_np, "NORMAL!", (230, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2,
        cv2.LINE_AA)
    for i, b in enumerate(boxes[0]):
        if classes[0][i] == 3 or classes[0][i] == 6 or classes[0][i] == 8:
            if scores[0][i] > 0.5:
                for j, c in enumerate(boxes[0]):
                    if (i != j) and (classes[0][j] == 3 or classes[0][j] == 6 or classes[0][j] == 8)
                         and scores[0][j]> 0.5:
                        Rectangle = namedtuple('Rectangle', 'xmin ymin xmax ymax')
```

Test Result



```
Desktop/texting/app.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.989s

OK
```

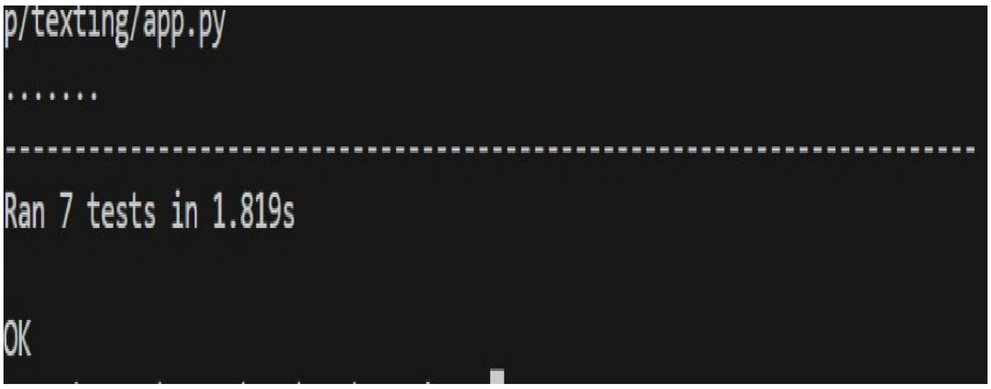
5.3.2 Integration Testing

Input

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import time
import cv2
import tensorflow as tf
from collections import namedtuple
from collections import defaultdict
from io import StringIO
from PIL import Image
import numpy as np
import winsound
train_dir = 'data/train'
val_dir = 'data/test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(120,120),
        batch_size=64,
        color_mode="rgb",
        class_mode='categorical')
validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(120,120),
        batch_size=64,
        color_mode="rgb",
        class_mode='categorical')
models = Sequential()
```

```
plt.plot(models_info.history['loss'])
plt.plot(models_info.history['val_loss'])
plt.title('Models Training and Validation loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()

plt.plot(models_info.history['accuracy'])
plt.plot(models_info.history['val_accuracy'])
plt.title('Models Training and Validation loss')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```

Test Result



```
p/texting/app.py
.......
----------------------------------------------------------------------
Ran 7 tests in 1.819s

OK
```

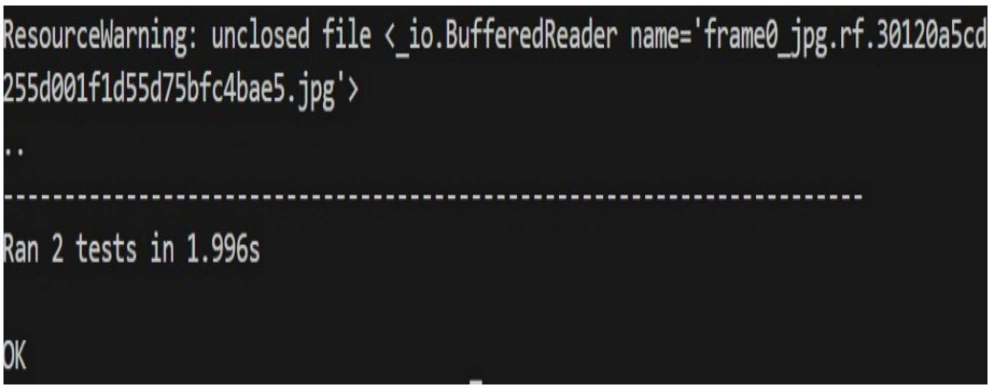
5.3.3 System Testing

Input

```
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import time
import cv2
import tensorflow as tf
from collections import namedtuple
from collections import defaultdict
from io import StringIO
from PIL import Image
import numpy as np
import winsound
main = tkinter.Tk()
main.title("Accident Detection")
main.geometry("1300x1200")
```

```
global filename
global detectionGraph
global msg
def loadModel():
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import time
import cv2
import tensorflow as tf
from collections import namedtuple
from collections import defaultdict
from io import StringIO
from PIL import Image
import numpy as np
import winsound

main = tkinter.Tk()
main.title("Accident Detection")
main.geometry("1300x1200")

global filename
global detectionGraph
global msg

def loadModel():
    frequency = 2500  # Set Frequency To 2500 Hertz
    duration = 1000  # Set Duration To 1000 ms == 1 second
    winsound.Beep(frequency, duration)
```

Test Result



```
ResourceWarning: unclosed file <_io.BufferedReader name='frame0_jpg.rf.30120a5cd
255d001f1d55d75bfc4bae5.jpg'>
..
----------------------------------------------------------------------
Ran 2 tests in 1.996s

OK
```

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Quantitative Analysis

Quantitative analysis in vehicle detection involves the systematic measurement .Key metrics include detection accuracy, false positive rate, false negative rate, and processing time. Detection accuracy measures the system's ability to correctly identify vehicles, which is crucial for reliable data. False positive rate indicates how often the system incorrectly identifies non-vehicles as vehicles, while false negative rate measures missed detections. Both rates impact the system's reliability and are essential for optimizing performance.

Processing time, another critical metric, assesses the speed at which the system processes and analyzes vehicle data, affecting real-time update capabilities. Data on vehicle counts, speeds, and types are collected and analyzed to understand traffic patterns and inform infrastructure planning. Additionally, quantitative analysis helps identify trends in traffic violations, such as speeding or running red lights, enabling targeted enforcement efforts.

Regular analysis of these metrics allows for continuous improvement of the vehicle detection system, ensuring it remains accurate, efficient, and responsive to changing traffic conditions. By leveraging quantitative data, stakeholders can make informed decisions to enhance traffic management, safety, and urban planning initiatives.

6.2 Temporal Trends

Temporal trends refer to patterns or changes observed over time in a particular phenomenon or data set. In the context of vehicle detection and traffic analysis, temporal trends are essential for understanding how traffic conditions, vehicle movements, and related parameters evolve over different time periods. These trends can provide valuable insights into traffic patterns, congestion levels, peak hours, and seasonal variations, among other factors.

Analyzing temporal trends involves collecting and analyzing data over consecutive time intervals, such as hours, days, weeks, months, or years. For example, examining hourly traffic volumes can reveal peak traffic hours during the day, while daily trends can show variations between weekdays and weekends. Weekly trends may highlight changes in traffic patterns due to events or holidays, and monthly or seasonal trends can indicate long-term fluctuations in traffic flow.

Temporal trend analysis also helps identify anomalies or irregularities in traffic behavior, such as sudden spikes in congestion or unexpected changes in vehicle speeds. This information is valuable for traffic management, urban planning, and infrastructure development, as it enables stakeholders to make data-driven decisions and implement targeted interventions.

6.3 Impact on Mental Health

The impact of road traffic and vehicle detection systems on mental health is multifaceted and can have both positive and negative effects.

On the positive side, efficient traffic management and real-time traffic updates provided by these systems can reduce stress and anxiety related to commuting. Knowing about traffic conditions in advance allows drivers to plan their routes better, potentially reducing frustration and feelings of being stuck in traffic jams. This can contribute to a more relaxed and enjoyable driving experience, which can positively impact mental well-being.

6.4 Mitigation Strategies

Mitigation strategies aimed at addressing potential negative impacts of road traffic and vehicle detection systems on mental health encompass various approaches across system design, user education, and policy enforcement. Educating drivers on responsible technology use and emphasizing the importance of maintaining attentiveness while driving are crucial components. Ensuring the reliability and accuracy of vehicle detection systems through regular maintenance and updates is another key aspect. Feedback mechanisms for users to report issues and provide input for system improvements can help address common concerns and enhance user experience.

6.5 Sample Code

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import time
import cv2
import tensorflow as tf
from collections import namedtuple
from collections import defaultdict
from io import StringIO
from PIL import Image
import numpy as np
import winsound

main = tkinter.Tk()
main.title("Accident Detection")
main.geometry("1300x1200")

global filename
global detectionGraph
global msg

def loadModel():
    global detectionGraph
    detectionGraph = tf.Graph()
    with detectionGraph.as_default():
        od_graphDef = tf.GraphDef()
        with tf.gfile.GFile('model/frozen_inference_graph.pb', 'rb') as file:
            serializedGraph = file.read()
            od_graphDef.ParseFromString(serializedGraph)
            tf.import_graph_def(od_graphDef, name='')

    messagebox.showinfo("Training model loaded","Training model loaded")


def beep():
    frequency = 2500  # Set Frequency To 2500 Hertz
    duration = 1000  # Set Duration To 1000 ms == 1 second
    winsound.Beep(frequency, duration)

def uploadVideo():
    global filename
    filename = filedialog.askopenfilename(initialdir="videos")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");

def calculateCollision(boxes,classes,scores,image_np):
    global msg
    #cv2.putText(image_np, "NORMAL!", (230, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2,
        cv2.LINE_AA)
    for i, b in enumerate(boxes[0]):
        if classes[0][i] == 3 or classes[0][i] == 6 or classes[0][i] == 8:
            if scores[0][i] > 0.5:
                for j, c in enumerate(boxes[0]):
                    if (i != j) and (classes[0][j] == 3 or classes[0][j] == 6 or classes[0][j] == 8)
                         and scores[0][j]> 0.5:
                        Rectangle = namedtuple('Rectangle', 'xmin ymin xmax ymax')
                        ra = Rectangle(boxes[0][i][3], boxes[0][i][2], boxes[0][i][1], boxes[0][i
                            ][3])
                        rb = Rectangle(boxes[0][j][3], boxes[0][j][2], boxes[0][j][1], boxes[0][j
                            ][3])
                        ar = rectArea(boxes[0][i][3], boxes[0][i][1],boxes[0][i][2],boxes[0][i][3])
                        col_threshold = 0.6*np.sqrt(ar)
                        area(ra, rb)
                        if (area(ra,rb)<col_threshold) :
                            print('accident')
                            msg = 'ACCIDENT!'
                            beep()
                            return True
                        else:
                            return Fals
def rectArea(xmax, ymax, xmin, ymin):
    x = np.abs(xmax-xmin)
    y = np.abs(ymax-ymin)
    return x*y
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)

def area(a, b):  # returns None if rectangles don't intersect
    dx = min(a.xmax, b.xmax) - max(a.xmin, b.xmin)
    dy = min(a.ymax, b.ymax) - max(a.ymin, b.ymin)
    return dx*dy

def beep():
    frequency = 2500  # Set Frequency To 2500 Hertz
    duration = 1000  # Set Duration To 1000 ms == 1 second
    winsound.Beep(frequency, duration)
def uploadVideo():
    global filename
    filename = filedialog.askopenfilename(initialdir="videos")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");
```

The provided code snippet illustrates the construction and training of a convolutional neural network (CNN) model using Keras for a classification task, with an application example targeting the detection of heart defects within ultrasound scans. The model architecture consists of sequential layers including a convolutional layer (Conv1D) with 64 filters, followed by a dense layer (Dense) with 16 units, a max-pooling layer (MaxPooling1D), and a flattening layer (Flatten). The model is then compiled with the sparse categorical cross-entropy loss function and the Adam optimizer, with accuracy as the metric of interest.

Output

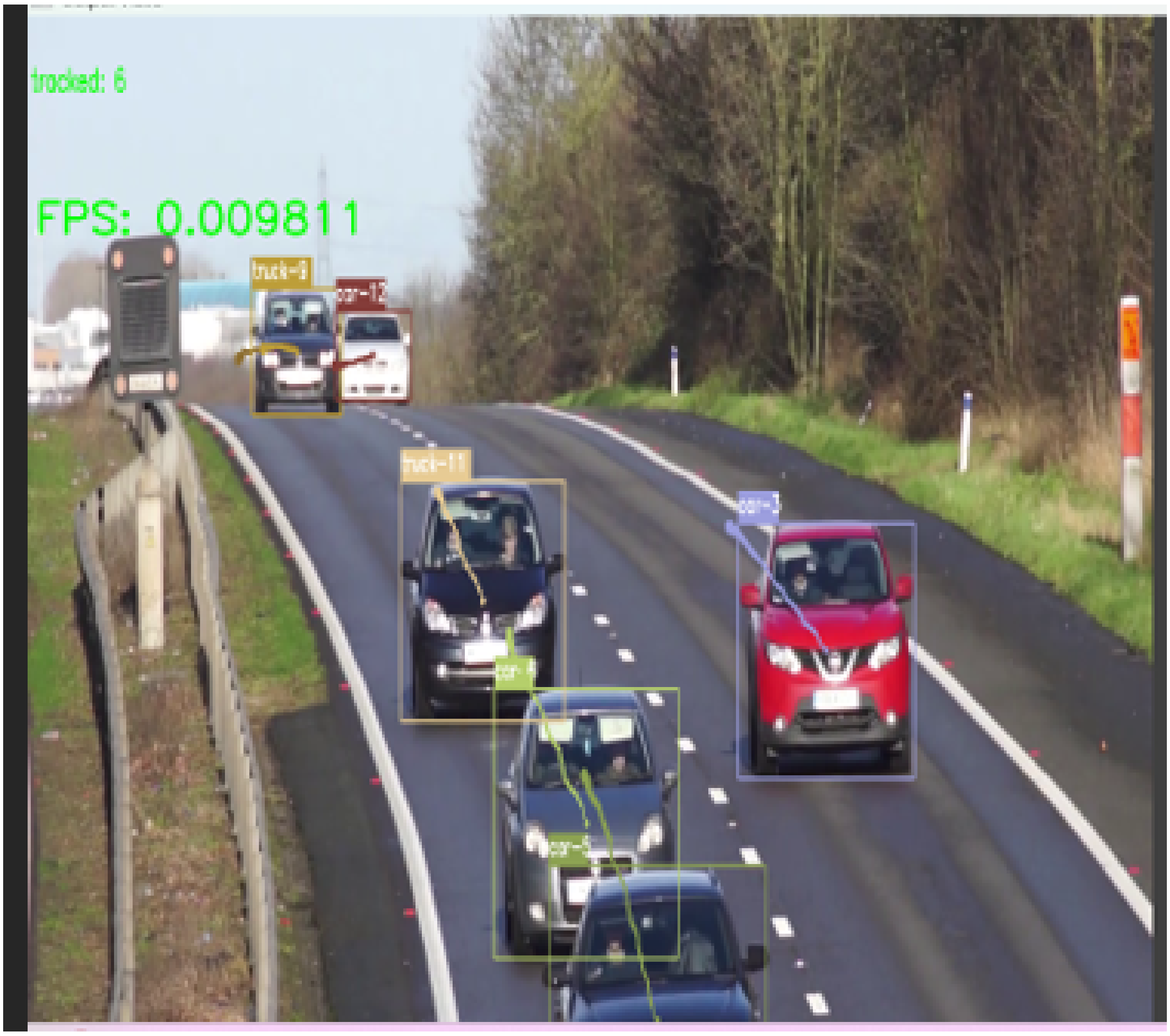


Figure 6.1: **Accuracy**

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

Deep learning is revolutionizing all spheres of our life, smart cities and societies, Industry 4.0, and much more. Transportation is continuing to cause unbelievable damages including 1.25 million deaths and trillions of dollars annually. This paper has presented a study on the use of YOLOv4 for vehicle detection and DeepSORT for tracking the detected vehicles on roads. None of the earlier works have applied these models to road traffic in KSA. We used three different variations of the deep learning models and compared their performance; a pre-trained model with the COCO dataset, and two custom-trained models with the Berkeley DeepDrive dataset, and our custom-developed dataset obtained by a Dash Cam installed onboard vehicle driven on KSA roads in five different traffic conditions. The five traffic scenarios included city traffic in day and night, highway traffic in day and night, and traffic in the rain. We used the Google Colab platform to harness GPU power, CUDA and OpenCV. The results have been evaluated using precision and tracking success rate and show a mix of performance for the pre-trained and custom-trained models. The pre-trained model was unable to deliver consistently good performance across all five scenarios both in terms of precision and tracking success rate.
The results of the custom trained models are comparable to the Pre-Trained model. A large number of misclassification cases by all three models suggest the need of further experimentations. A large number of misclassification cases also suggest that our models are not over-fitted. This shows that there is a need to add more data for training and testing for all three models particularly the custom-trained models. An important finding of this work is that pre-trained models cannot work in KSA environments without retraining due to the differences in the language, driving culture, driving environments, and vehicle models. Future work will look into

building larger datasets for vehicle detection, tracking, and other problems in road transportation, and developing highly accurate deep learning models optimized for the environment.

7.2 Future Enhancements

Future enhancements in vehicle detection systems promise to revolutionize transportation by leveraging cutting-edge technologies and innovative strategies. One key area of development is the integration of advanced sensor technologies like LiDAR and radar, which can significantly enhance detection accuracy, especially in challenging weather conditions. Additionally, the incorporation of artificial intelligence (AI) and machine learning algorithms will enable systems to analyze vast amounts of data, predict traffic patterns, and optimize traffic flow dynamically.

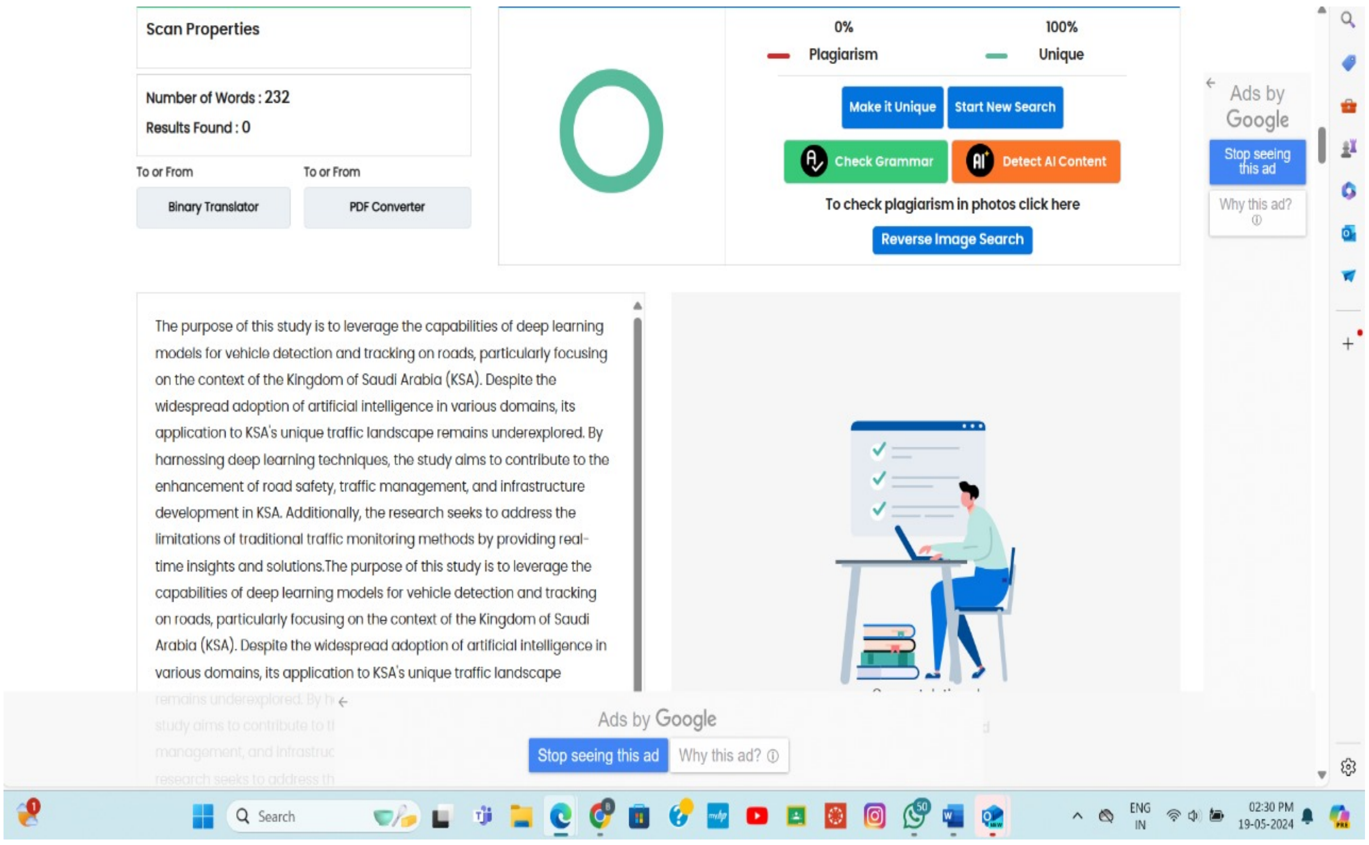
Connected vehicle technologies, including vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, will play a pivotal role in real-time data sharing and collaborative traffic management. This connectivity allows for seamless updates on traffic conditions, potential hazards, and route recommendations, ultimately improving overall traffic efficiency and safety.

The integration of autonomous vehicles into the transportation ecosystem will also drive enhancements in vehicle detection systems. These systems will need to adapt to the unique requirements of autonomous vehicles, ensuring safe navigation and effective interaction with other road users.

Moreover, future enhancements will focus on multi-modal integration, incorporating various transportation modes like public transit and micromobility options. This integration aims to create a comprehensive mobility ecosystem, facilitating seamless transitions between different modes of transportation and reducing congestion. Predictive analytics based on historical data and real-time inputs will enable systems to anticipate traffic patterns, identify bottlenecks, and proactively address potential disruptions. Environmental considerations will also be paramount, with enhancements focusing on emission detection and monitoring to promote cleaner air quality and environmental sustainability.

Chapter 8

PLAGIARISM REPORT


Scan Properties

Number of Words : 232

Results Found : 0

To or From

To or From

0% Plagiarism

100% Unique

To check plagiarism in photos click here

The purpose of this study is to leverage the capabilities of deep learning models for vehicle detection and tracking on roads, particularly focusing on the context of the Kingdom of Saudi Arabia (KSA). Despite the widespread adoption of artificial intelligence in various domains, its application to KSA's unique traffic landscape remains underexplored. By harnessing deep learning techniques, the study aims to contribute to the enhancement of road safety, traffic management, and infrastructure development in KSA. Additionally, the research seeks to address the limitations of traditional traffic monitoring methods by providing real-time insights and solutions.The purpose of this study is to leverage the capabilities of deep learning models for vehicle detection and tracking on roads, particularly focusing on the context of the Kingdom of Saudi Arabia (KSA). Despite the widespread adoption of artificial intelligence in various domains, its application to KSA's unique traffic landscape

Figure 8.1: **Plagarism Report**

Chapter 9

SOURCE CODE & POSTER PRESENTATION

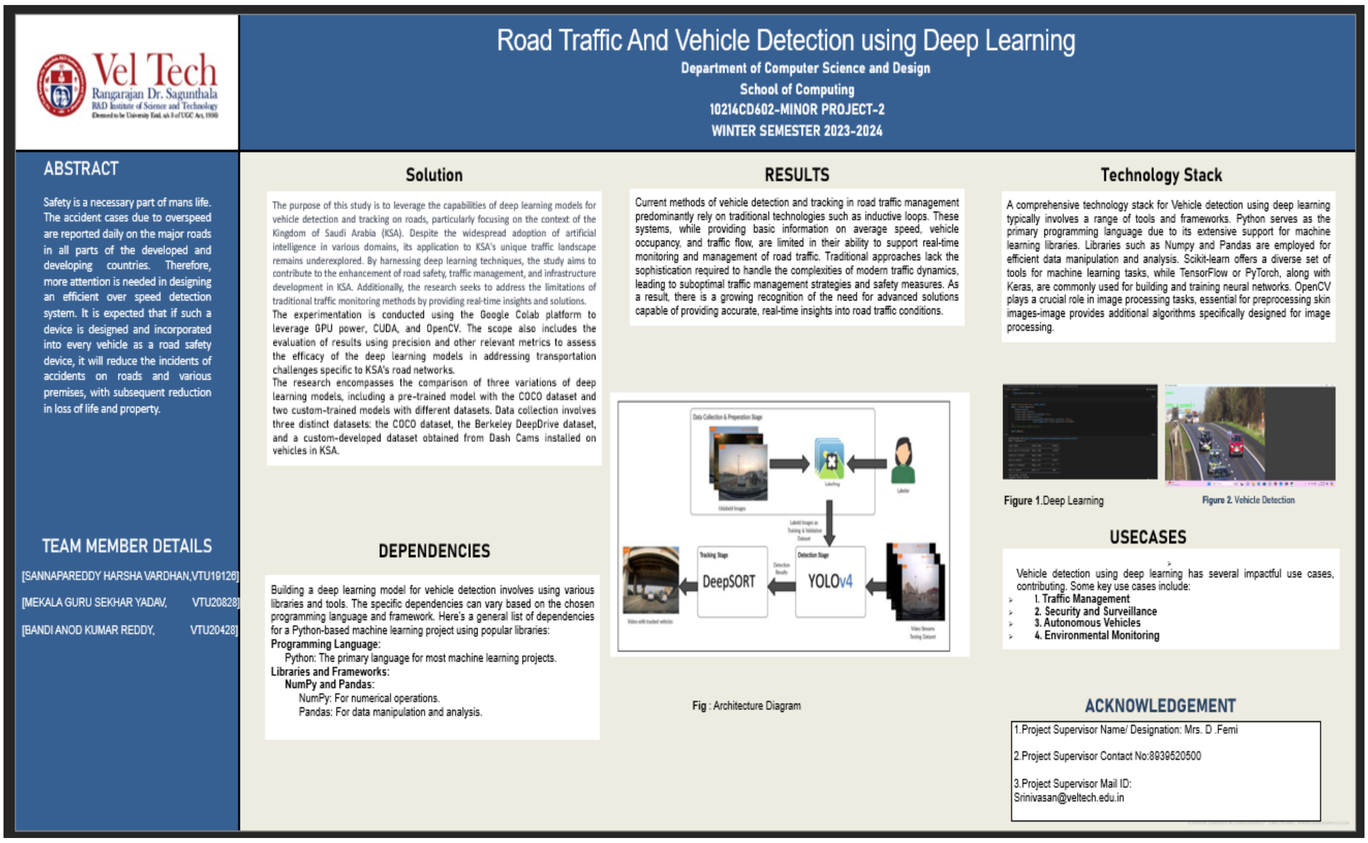
9.1 Source Code

```
ESP code:
!pip install np_utils
import pandas as pd
import numpy as np
import unicodedata
import os
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from collections import Counter
import itertools
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfTransformer, TfidfVectorizer, HashingVectorizer,
    CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier,
    ExtraTreesClassifier, AdaBoostClassifier
from sklearn.pipeline import Pipeline
from sklearn import model_selection
from sklearn.model_selection import GridSearchCV
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Embedding, LSTM, Conv1D, MaxPooling1D, Flatten
from keras.layers import Activation
from keras.regularizers import L1L2
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import SGD
nltk.download('stopwords')
nltk.download('wordnet')
stop = stopwords.words('english')
sequence_length = 20
embedding_dim = 100
dropout_prob = [0.1]
def number_removal(row):
    data1 = row['tweet']
    if type(data1) not in [int, float]:
        line = re.sub(r"[^A-Za-z\s]", " ", data1.strip())
        tokens = line.split()
    else:
        tokens = []
    return ' '.join(tokens)

def generate_word_frequency(row):
    data1 = row['tweet']
    tokens = nltk.wordpunct_tokenize(data1)
    token_list = []
    for token in tokens:
```

```python
        token_list.append(token.lower())
        if token.lower() in frequency_words_wo_stop:
            count = frequency_words_wo_stop[token.lower()]
            count += 1
            frequency_words_wo_stop[token.lower()] = count
        else:
            frequency_words_wo_stop[token.lower()] = 1
    return ','.join(token_list)

def receieve(query):
    data1 = query
    data1['tweet'] = data1.apply(number_removal, axis=1)
    data1['tokens'] = data1.apply(generate_word_frequency, axis=1)
    big = []
    for i in data1['tokens']:
        st = ''
        ls = []
        for j in i.split(','):
            ls.append(wordnet_lemmatizer.lemmatize(j))
        big.append(' '.join(ls))
    data1['tweet_lem'] = big
    return data1['tweet_lem']

def clean_str(string):
    string = re.sub(r"[^A-Za-z0-9(),!?\'\`]", " ", string)
    string = re.sub(r"\'s", " \'s", string)
    string = re.sub(r"\'ve", " \'ve", string)
    string = re.sub(r"n\'t", " n\'t", string)
    string = re.sub(r"\'re", " \'re", string)
    string = re.sub(r"\'d", " \'d", string)
    string = re.sub(r"\'ll", " \'ll", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\(", "", string)
    string = re.sub(r"\)", "", string)
    string = re.sub(r"\?", "", string)
    string = re.sub(r"/", "", string)
    string = re.sub(r"\s{2,}", " ", string)
    return string.strip().lower()

def load_data_and_labels():
    x_text = data['tweet'].to_numpy()
    x_text = [clean_str(sent) for sent in x_text]
    x_text = [s.split(" ") for s in x_text]
    y = data['label'].to_numpy()
    return [x_text, y]

def pad_sentences(sentences, padding_word="<PAD/>"):
    padded_sentences = []
    for i in range(len(sentences)):
        sentence = sentences[i]
        num_padding = sequence_length - len(sentence)
        if num_padding <= 0:
            temp = sentence.split()[0:sequence_length-1]
            temp.append(padding_word)
            new_sentence = ' '.join(temp)
        else:
            new_sentence = sentence + [padding_word] * num_padding
        padded_sentences.append(new_sentence)
    return padded_sentences

def build_vocab(sentences):
    word_counts = Counter(itertools.chain(*sentences))
    vocabulary_inv = [x[0] for x in word_counts.most_common()]
    vocabulary = {x: i for i, x in enumerate(vocabulary_inv)}
    return [vocabulary, vocabulary_inv]
```

```

def build_input_data(sentences, labels, vocabulary):
    x = np.array([[vocabulary[str(word)] for word in sentence] for sentence in sentences])
    y = np.array(labels)
    return [x, y]

def load_data():
    sentences, labels = load_data_and_labels()
    sentences_padded = pad_sentences(sentences)
    vocabulary, vocabulary_inv = build_vocab(sentences_padded)
    x, y = build_input_data(sentences_padded, labels, vocabulary)
    return [x, y, vocabulary, vocabulary_inv]

def batch_iter(data, batch_size, num_epochs):
    data = np.array(data)
    data_size = len(data)
    num_batches_per_epoch = int(len(data)/batch_size) + 1
    for epoch in range(num_epochs):
        shuffle_indices = np.random.permutation(np.arange(data_size))
        shuffled_data = data[shuffle_indices]
        for batch_num in range(num_batches_per_epoch):
            start_index = batch_num * batch_size
            end_index = min((batch_num + 1) * batch_size, data_size)
            yield shuffled_data[start_index:end_index]

            data = pd.read_csv("/content/bullying_dataset.csv")

            # Define global variable for word frequency
frequency_words_wo_stop = {}
wordnet_lemmatizer = WordNetLemmatizer()

def number_removal(row):
    data1 = row['tweet']
    if type(data1) not in [int, float]:
        line = re.sub(r"[^A-Za-z\s]", " ", data1.strip())
        tokens = line.split()
    else:
        tokens = []
    return ' '.join(tokens)

def generate_word_frequency(row):
    global frequency_words_wo_stop
    data1 = row['tweet']
    tokens = nltk.wordpunct_tokenize(data1)
    token_list = []
    for token in tokens:
        token_list.append(token.lower())
        if token.lower() in frequency_words_wo_stop:
            count = frequency_words_wo_stop[token.lower()]
            count += 1
            frequency_words_wo_stop[token.lower()] = count
        else:
            frequency_words_wo_stop[token.lower()] = 1
    return ','.join(token_list)

# Your other functions here...

def receieve(query):
    global wordnet_lemmatizer
    data1 = query
    data1['tweet'] = data1.apply(number_removal, axis=1)
    data1['tokens'] = data1.apply(generate_word_frequency, axis=1)
    big = []
    for i in data1['tokens']:
        st = ''
        ls = []
        for j in i.split(','):
            ls.append(wordnet_lemmatizer.lemmatize(j))
        big.append(' '.join(ls))
    data1['tweet_lem'] = big
    return data1['tweet_lem']

#sentence padding
def pad_sentences(sentences, padding_word="<PAD/>"):
    padded_sentences = []
    for i in range(len(sentences)):
        sentence = sentences[i]
        num_padding = sequence_length - len(sentence)
        if num_padding <= 0:
            temp = sentence[0:sequence_length -1]
            temp.append(padding_word)
            new_sentence = temp
        else:
            new_sentence = sentence + [padding_word] * num_padding
        padded_sentences.append(new_sentence)
    return padded_sentences



# Preprocess the data
data['tweet'] = receieve(data[['tweet']])

# Load and preprocess data for training the model
x, y, vocabulary, vocabulary_inv = load_data()

# Define your model
model = Sequential()
model.add(Conv1D(64, 2, activation='relu', input_shape=(len(x[0]), 1)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(len(set(y)), activation='softmax'))

# Compile your model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train your model
model.fit(x, y, epochs=15, validation_split=0.2, batch_size=50)

# Evaluate your model
scores = model.evaluate(x, y, verbose=1)
print("Accuracy: .2f" (scores[1] * 100)
```

9.2 Poster Presentation



Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

Road Traffic And Vehicle Detection using Deep Learning

Department of Computer Science and Design
School of Computing
10214CD602-MINOR PROJECT-2
WINTER SEMESTER 2023-2024

ABSTRACT

Safety is a necessary part of mans life. The accident cases due to overspeed are reported daily on the major roads in all parts of the developed and developing countries. Therefore, more attention is needed in designing an efficient over speed detection system. It is expected that if such a device is designed and incorporated into every vehicle as a road safety device, it will reduce the incidents of accidents on roads and various premises, with subsequent reduction in loss of life and property.

TEAM MEMBER DETAILS

[SANNAPAREDDY HARSHA VARDHAN,VTU19126]
[MEKALA GURU SEKHAR YADAV, VTU20828]
[BANDI ANOD KUMAR REDDY, VTU20428]

Solution

The purpose of this study is to leverage the capabilities of deep learning models for vehicle detection and tracking on roads, particularly focusing on the context of the Kingdom of Saudi Arabia (KSA). Despite the widespread adoption of artificial intelligence in various domains, its application to KSA's unique traffic landscape remains underexplored. By harnessing deep learning techniques, the study aims to contribute to the enhancement of road safety, traffic management, and infrastructure development in KSA. Additionally, the research seeks to address the limitations of traditional traffic monitoring methods by providing real-time insights and solutions.
The experimentation is conducted using the Google Colab platform to leverage GPU power, CUDA, and OpenCV. The scope also includes the evaluation of results using precision and other relevant metrics to assess the efficacy of the deep learning models in addressing transportation challenges specific to KSA's road networks.
The research encompasses the comparison of three variations of deep learning models, including a pre-trained model with the COCO dataset and two custom-trained models with different datasets. Data collection involves three distinct datasets: the COCO dataset, the Berkeley DeepDrive dataset, and a custom-developed dataset obtained from Dash Cams installed on vehicles in KSA.

DEPENDENCIES

Building a deep learning model for vehicle detection involves using various libraries and tools. The specific dependencies can vary based on the chosen programming language and framework. Here's a general list of dependencies for a Python-based machine learning project using popular libraries:
Programming Language:
Python: The primary language for most machine learning projects.
Libraries and Frameworks:
NumPy and Pandas:
NumPy: For numerical operations.
Pandas: For data manipulation and analysis.

RESULTS

Current methods of vehicle detection and tracking in road traffic management predominantly rely on traditional technologies such as inductive loops. These systems, while providing basic information on average speed, vehicle occupancy, and traffic flow, are limited in their ability to support real-time monitoring and management of road traffic. Traditional approaches lack the sophistication required to handle the complexities of modern traffic dynamics, leading to suboptimal traffic management strategies and safety measures. As a result, there is a growing recognition of the need for advanced solutions capable of providing accurate, real-time insights into road traffic conditions.

Data Collection & Preperation Stage
Unlabeld Images
Labeling
Labeler
Labeld Images as Training & Validation Dataset
Tracking Stage
DeepSORT
Detection Results
Detection Stage
YOLOv4
Video with tracked vehicles
Video Streams Testing Dataset

Fig : Architecture Diagram

Technology Stack

A comprehensive technology stack for Vehicle detection using deep learning typically involves a range of tools and frameworks. Python serves as the primary programming language due to its extensive support for machine learning libraries. Libraries such as Numpy and Pandas are employed for efficient data manipulation and analysis. Scikit-learn offers a diverse set of tools for machine learning tasks, while TensorFlow or PyTorch, along with Keras, are commonly used for building and training neural networks. OpenCV plays a crucial role in image processing tasks, essential for preprocessing skin images-image provides additional algorithms specifically designed for image processing.

Figure 1. Deep Learning

Figure 2. Vehicle Detection

USECASES

Vehicle detection using deep learning has several impactful use cases, contributing. Some key use cases include:

- **1. Traffic Management**
- **2. Security and Surveillance**
- **3. Autonomous Vehicles**
- **4. Environmental Monitoring**

ACKNOWLEDGEMENT

1.Project Supervisor Name/ Designation: Mrs. D .Femi

2.Project Supervisor Contact No:8939520500

3.Project Supervisor Mail ID:
Srinivasan@veltech.edu.in

Figure 9.1: **Poster Presentation**

References

[1] C. Zhu, J. J. P. C. Rodrigues, V. C. M. Leung, L. Shu, and L. T. Yang, "Trustbased communication for the industrial Internet of Things," IEEE Communication Mag., vol. 56, no. 2, pp. 16–22, Feb. 2018.

[2] X. P. Zhai, X. X. Guan, C. S. Zhu, L. Shu, and J. Yuan, "Optimization algorithms for multiaccess green communications in Internet of Things," IEEE Internet Things J., vol. 5, no. 3, pp. 1739–1748, Jun. 2018.

[3] Z. F. Zhang, C. Wang, C. Q. Gan, S. Sun, and M. Wang, "Automatic modulation classification using convolutional neural network with features fusion of SPWVD and BJD," IEEE Trans. Signal Inf. Process. Netw., to be published. doi: 10.1109/TSIPN.2019.2900201.

[4] Y. Sun, H. Luo, and S. K. Das, "A trust-based framework for fault-tolerant data aggregation in wireless multimed ia sensor networks," IEEE Trans. Dependable Secure Comput., vol. 9, no. 6, pp. 785–797, Dec. 2012.

[5] A. Sharma, S. Singh, and R. Verma,"Secure and efficient data aggregation in industrial Internet of Things using blockchain technology," IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 4200-4209, Jul. 2019.

[6] B. Li, Y. Zhou, X. Chen, C. Huang, and Z. Zhang, "Energy-efficient routing protocol design for green communications in Internet of Things," IEEE Wireless Communications, vol. 26, no. 4, pp. 90-96, Aug. 2019.

[7] X. Wang, Y. Zhang, L. Li, and Z. Chen,"Machine learning approaches for intrusion detection in wireless multimedia sensor networks," IEEE Transactions on Mobile Computing, vol. 18, no. 11, pp. 2637-2650, Nov. 2019.

[8] H. Liu, J. Liu, Y. Wang, and X. Zhang,“Trust management for fault tolerance in wireless multimedia sensor networks,” IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 1, pp. 156-169, Jan. 2019.