**SPECIMEN**

**ROAD ACCIDENT AUTOMATED RECOVERY PROCESS**

**A PROJECT REPORT**

***Submitted by***

## ASHWINI. K [REGISTER NO: 211417104026] BHAVANA. P [REGISTER NO: 211417104037]

***in partial fulfillment for the award of the degree of***

## BACHELOR OF ENGINEERING

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report **“ROAD ACCIDENT AUTOMATED RECOVERY PROCESS”** is the bonafide work of **‘‘ASHWINI. K [Register No: 211417104026] BHAVANA.P [Register No: 211417104037]’’**who carried out the project work under my supervision

**SIGNATURE SIGNATURE**

**Dr.S.MURUGAVALLI,M.E.,Ph.D., Mrs.S.T.SANTHANALAKSHMI.M.TECH HEAD OF THE DEPARTMENT SUPERVISOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, NAZARATHPETTAI

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR, M.E., Ph.D** and Tmt**. SARANYASREE SAKTHIKUMAR B.E., M.B.A.,** for

providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project guide Mrs.S.T.SANTHANALAKSHMI M.Tech (IT),** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**ASHWINI.K BHAVANA.P**

# ABSTRACT

Road accident are one of the major causes of human death. In this project we have investigated a practical and novel method of road accident automated recovery process which can real-time monitor and also can improve the rules and regulations for the car drive Automated analysis is used to find Car accident detection. The image processing and machine learning techniques are employed in surveillance system of power substation. The video surveillance system automatically detecting the car number with sending mail to certain authorities and taking care of first aids. If the car number are found with the accident detection ,his/her license plate(LP) number is recognized to initiate further actions such as deduction of penalty amount from one’s account linked with the vehicle license and Aadhar number by the traffic police and the legal authority. It is a whole system which contains proper flow for the detection of road accident

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TABLE OF CONTENT** |  |  |
|  |  |  |  |
| **CHAPTER NO** | **TITLE** |  | **PAGE NO** |
|  | **ABSTRACT** |  | iii |
|  | **LIST OF TABLES** |  | v |
|  | **LIST OF FIGURES** |  | vi |
|  | **LIST OF SYMBOLS,ABBREVIATIONS** |  | vi |
| 1 | **INTRODUCTION** |  |  |
|  | 1.1 OVERVIEW |  | 1 |
|  | 1.2 PROBLEM DEFINITION |  | 2 |
| 2 | **LITERATURE SURVEY** |  | 4 |
| 3 | **SYSTEM ANALYSIS** |  |  |
|  | 3.1 Existing System |  | 5 |
|  | 3.2 Proposed system |  | 7 |
|  | 3.3 Technology Stack |  | 8 |
| 4 | **SYSTEM DESIGN** |  |  |
|  | 4.1 ER diagram |  | 9 |
|  | 4.2 Data dictionary |  | 10 |
| 5 | **SYSTEM ARCHITECTURE** |  |  |
|  | 5.1 Architecture Overview |  | 13 |
|  | 5.2 Description of the Modules |  | 14 |
| 6 | **SYSTEM IMPLEMENTATION** |  |  |
|  | 6.1 Client -side coding |  | 17 |
|  | 6.2 Server -side coding |  | 20 |
| 7 | **SYSTEM TESTING** |  |  |
|  | 7.1 Test case report |  | 39 |
|  | 7.2 Performance analysis |  | 40 |
| 8 | **CONCLUSION** |  |  |
|  | Conclusion and Future Enhancement |  | 41 |
| 9 | **APPENDICES** |  |  |
|  | 9.1 Sample screenshot |  | 42 |
| 10 | **REFERENCES** |  | 45 |
|  |  |  |  |
|  |  |  |  |

# LIST OF TABLES

|  |  |
| --- | --- |
| **1** | Data dictionary |

# LIST OF FIGURES

|  |  |
| --- | --- |
| **1** | Traffic congestion |
| **2** | Car accident |
| **3** | Monitoring cctv camera |
| **4** | Road safety work |
| **5** | Safety first |
| **6** | Weather condition |
| **7** | Cctv camera |
| **8** | Number plate |
| **9** | Signal cctv camera |
| **10** | Windows xp professionals |
| **11** | Er diagram |
| **12** | Use case diagram |
| **13** | Sequence diagram |
| **14** | Class diagram |
| **15** | System architecture |
| **16** | Cctv camera |
| **17** | Optical character recognition |
| **18** | INRIX GC weight on map |

|  |  |
| --- | --- |
| **19** | Loop GC weight on map |
| **20** | Loop/INRIX bar diagram |
| **21** | Output in frames |
| **22** | Output in frames |
| **23** | Detecting speed of the car |
| **24** | Detected car speed |
| **25** | Detected of accidented car |
| **26** | Detection of number plate |

# LIST OF SYMBOLS, ABBREVIATION

|  |  |
| --- | --- |
| LP | License plate |
| ITS | Intelligent Transportation System |
| KNN | K-Nearest Neighbor |
| ANN | Artificial Neural Network |
| TSA | Time Series Analysis |
| SVR | Support Vector Regression |
| EI | Emergent Intelligence |
| YOLO | You Only Look Once |
| VRI | Video Relay Interpreting |
| OCR | Optical Character Recognition |

# INTRODUCTION

## OVERVIEW

Accident prediction is one of the key problems in traffic control and guidance system as well as the important functions of intelligent transportation system (ITS). The fast expansion in machine learning new methods and in the appearance of new data sources makes it possible to evaluate and forecast accident conditions in smart cities more quickly and accurately. Accident estimation and prediction system has the ability to reach destination immediately and deaths taking place due to road accident. In this paper, the existing accident prediction methods for smart cities are provided in detail and the problems and challenges of the prediction models are analyzed in depth. Based on the analysis of the existing short-term accident detection flow forecasting methods, the possible development trend of short-term accident detections approaches in the future is pointed out.



### Figure 1

## PROBLEM DEFINITION

A metropolitan area is densely populated urban core area and is surrounded by less populated territories, sharing industry, housing, and infrastructure. It consists of urban areas, satellite cities, rural areas and towns, and these are socio- economically tied to the urban core, and are typically measured by commuting patterns of commuters. The rapid development, urbanization, national and international trades in a metropolitan area leads to increase in the accidents, demands, freight flows and frequency of vehicles. The increase of these factors results in growing of frequency of traffic jams, accidents and fatalities and they lead to significant recurring delay in a metropolitan area. Most of the time these factors depend on the type of place and periods of time. Therefore, in a metropolitan area accident is an unbearable event for commuters and wastes their precious time and burns money in the form of fuel hence, in near future most of the metropolitan areas will become less attractive to business and new residents.



### Figure 2

The growing metropolitan areas are facing challenges, such as accidents, analysis and sharing, traffic density and travel time predictions on routes, to improve the accident detection these challenges heavily impact on daily routine activities and economic losses.

Some of these are achieved by using prediction models and they provide more accurate and realistic accident information than the more recently estimated information.

Short term methods, such as k-nearest neighbour (KNN), artificial neural network (ANN), time series analysis (TSA) and support vector regression (SVR) data mining and deep learning techniques are used for Accident analysis, prediction and management. These methods may suffer from several drawbacks, including adaptability during dynamic changes in road conditions and new structures, low computing speed and more processing time.

These drawbacks have motivated us to re-examine the analysis, sharing, monitoring, prediction and management of resources, traffic and travel time using emergent intelligence (EI) technique in a metropolitan area. Because the EI technique adapts to dynamic behaviour in distributed environments and is the best choice for dynamic accident recovery management system, which also improves the traffic efficiency, reduces waiting time and under-utilization of resources (like number of vehicles, amount of fuel, parking space, etc.).



### Figure 3

In this paper, we propose an intelligent road accident automated detection using prediction information in a metropolitan area.

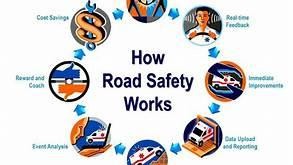
# LITERATURE SURVEY

1. **Salaja silas et all proposed a system ‘’ Performance Analysis of Edge Detection Algorithms for Object Detection in Accident Images’’ in 2019.** This system provides an Edge detection is used in identifying various objects such as vehicles and victims by outlining the sudden change of pixel intensity from the given accidental images. Algorithms are evaluated in terms of Peak Signal-to-Noise Ratio (PSNR) and Mean Square Error (MSE) which delays the output
2. **Amrutha Madhusan et all proposed a system ‘’ A Survey on Road Accident Detection and Reporting’’ in 2019.** This system has Two types of modules are used in this system, they are GPS and GSM where both are interfaced using serial communication with Micro electric sensor and vibrating sensor is used. If the interface is misled, both GPS and GSM gets affected. High cost and also need more number of sensors
3. **Pranav Chitale et all proposed a system ‘’ Smart Accident Recognition and Alerting System for Edge Devices’’ in 2020**.this system has End to end deep learning solution to automate accident recognition and send real-time alerts to emergency services using the intel’s openVINO . For such pretrained models with your costume data is not always easy, as not everything is well documented around re-training/fine-tunning for all provided models
4. **Earnest Paul ljjina et all proposed a system ‘’Computer Vision-Based Accident Detection in Traffic Surveillance’ in 2019**.this system has a neoteric framework for detection of road accident by framework capitalizes on mask R-CNN for accurate object detection followed by an efficient centroid based object tracking algorithm for surveillance footage . Mask R-CNN is appropriate to use classification confidence to measure the mask quality since it only serves for distinguishing the semantic categories of proposals and also not aware of the actual quality
5. **C. Vishnu et all proposed a system ‘’ Detection of Motorcyclists without Helmet in Videos using Convolutional Neural Network’’ in 2017**.this system has a framework for automatic detection of motorcyclists driving without helmets in surveillance video .high cost and cannot detect automatically.

# SYSTEM ANALYSIS

## EXISTING SYSTEM

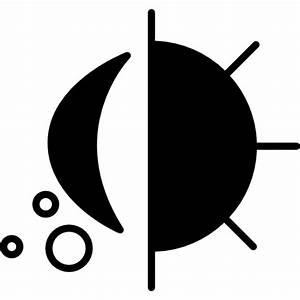
In the present existing system, the CCTV camera captures video and stores it in the memory of the computer. For references the user can view the video but it is a tedious process. IN the CCTV system ,I need a dedicated PC for this module that is eliminated in the raspberry pi module.Secured & gated entrances, such as military bases or permit-based , will automatically compare each approaching vehicle's license plate to a database of authorized personnel, and only open the gates if that license plate number is on the list.Parking enforcement officials can record cars as they enter and park, giving them a video to refer to if cars go over the time limit or attempt in and out parking.Police and other law officials can review local traffic videos to search for the license plates of stolen vehicles or suspected cars in other crimes



### Figure 4 Figure 5

**DRAWBACKS:**

* Weather - Whether they're monitoring intersections or looking out for traffic jams, traffic cameras are subject to damage caused by weather. Heat, wind, rain, snow and ice can all damage or ruin a traffic security camera.
* Accidents - Since they're placed on busy roads and intersections, there is also a chance that accidents could damage traffic cameras
* The angle of the camera is extremely important, because a camera that's installed too high cannot see the license plate
* The distance from the camera to the car must also be considered - while a camera may be able to zoom in quite a distance, you want to minimize the distance between camera and car
* Lighting is also important - whether you install lights or use the camera's built-in IR, you will need additional lighting at night
* The speed of oncoming traffic must be low enough that the camera has enough time to focus on the license plate, often times no more than 35 mph depending on the angle, distance, and lighting



### Figure 6 Figure 7

## PROPOSED SYSTEM:

In this paper we can improve the rules and regulations for the car drive Automated analysis is used to find Car accident detection. This article presents an intelligent video surveillance system for automatically detecting the Car Number with sending mail. If the car number are found with accident detection, his/her license plate (LP) number is recognized to initiate further actions such as deduction of penalty amount from one's account linked with the vehicle license and Aadhar Number (Applicable to Indian Scenario) by the traffic police and the legal authority



### Figure 9

Figure8

## TECHNOLOGY STACK:

* + .net
  + Windows xp professional

**HARDWARE REQUIREMENTS**

* + - Processor - Pentium IV 2.4GHZ.
    - RAM - 4GB
    - ROM - 1TB
    - Cache memory - 512KB
    - CCTV camera - ADVANCED

**SOFTWARE REQUIREMENTS**

* + - Operating System - Windows XP Professionals



### Figure 10

# SYSTEM DESIGN:

## ER-DIAGRAM:

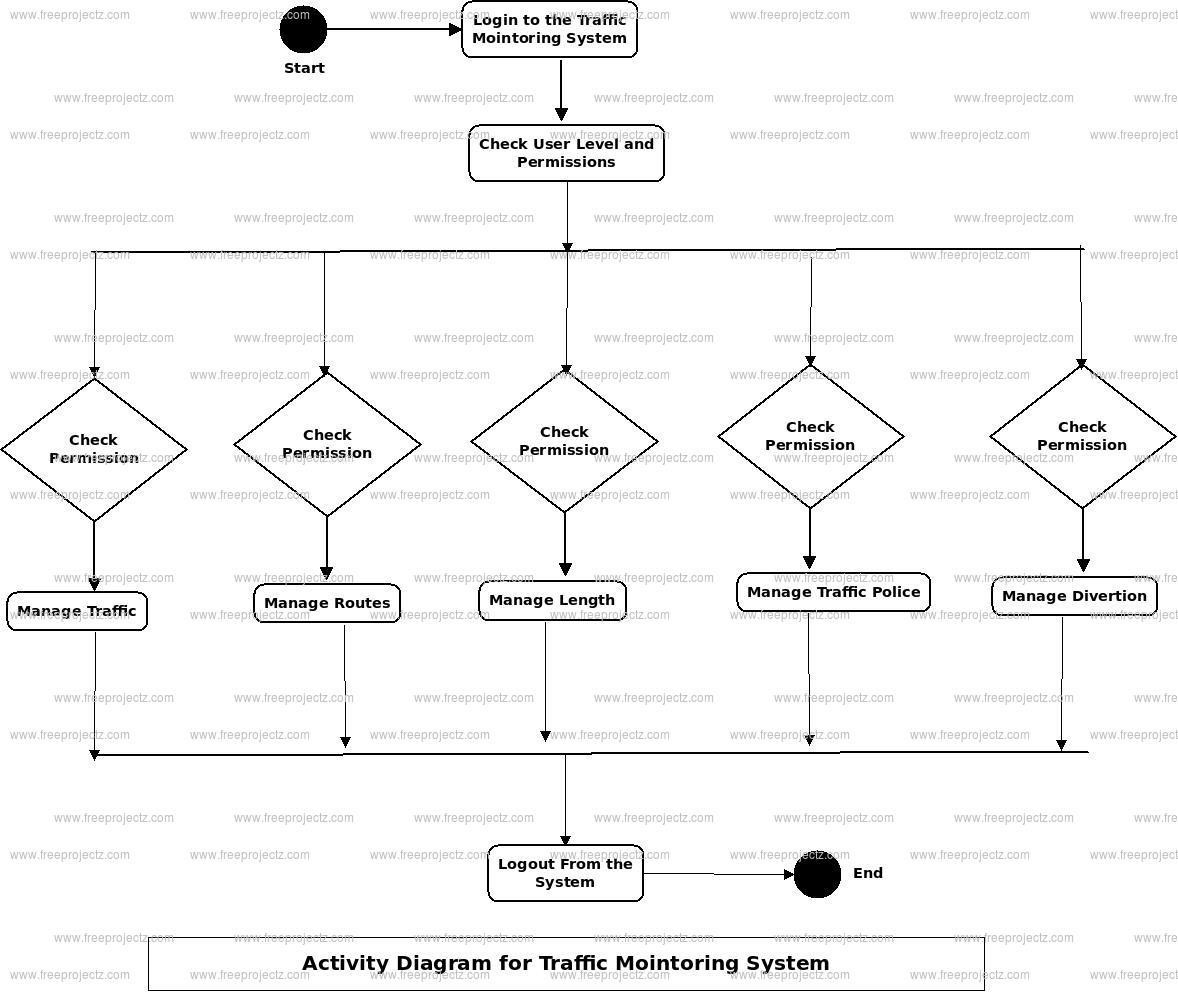


Figure 11

## DATA DICTIONARY:

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** | **EXAMPLE** |
| Accident status | string | to determine whether the accident occurs | none |
| Noise sensor | float | Sensor measure decibels(loudness) | 80-100db |
| License plate | float | Extract number through video | TN14E2344 |
| location \_status | string | location of the accident | GPS |

**Table 1**

**UML DIAGRAMS:**

**USE CASE DIAGRAM**:

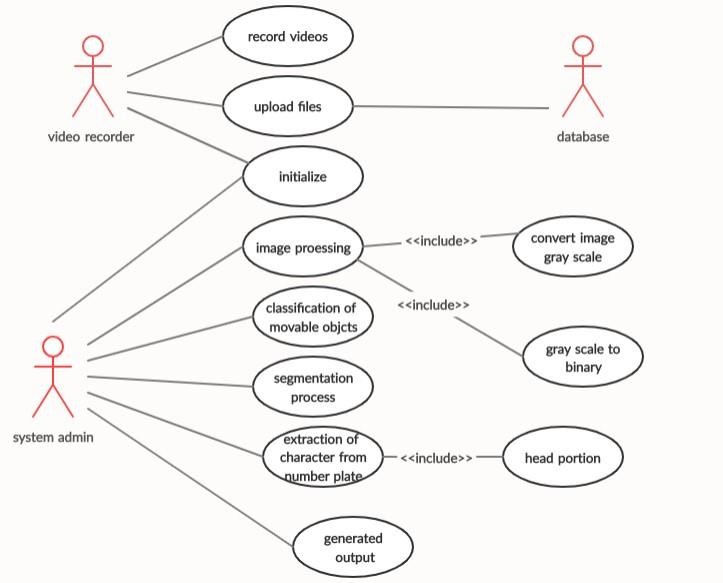
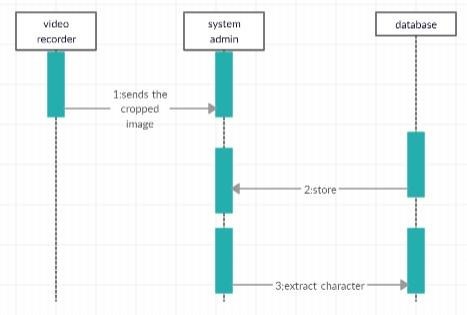


Figure 12

**SEQUENCE DIAGRAM**:



**CLASS DIAGRAM**:

Figure 13

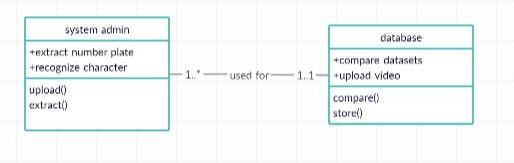
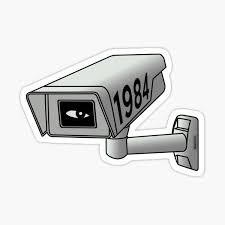


Figure 14

# SYSTEM ARCHITECTURE

## OVERVIEW:



Cctv camera



Centralized station

Accident spot

Figure 15

## DESCRIPTION OF MODULES

**List of Module**

1. **Classification of movable objects**
2. **Segmentation**
3. **Recognition of characters**
   1. **Classification Of Movable Objects**

* The surveillance camera that capture the movable objects from the traffic road. And it uploads to the database which is already created to store the video files for the further uses based on the needs of the system admin.
* It uploads the video from the camera and stores the video in the database.
* Admin uses the multiple functions to be done in the class



### Figure 16

* 1. **Segmentation**

Segmentation means to divide the frames into parts, or segments, which are definable, accessible, actionable, and profitable and have a growth potential. In other words, a company would find it impossible to target the entire market, because of time, cost and effort restrictions. Once the person-Car pair is obtained, the person images is given as input to accident detection model. While testing the accident detection model, some false detections were observed. So, the person image was cropped to get only top- fourth portion of image. They detected accident as recognition technology that uses YOLOV3. As the rider Car Number and accident detection in case 1, bounding box is created, no further processing is necessary. Since in case 2, rider is not Car Number and accident detection, no bounding box is created.

## 3 .RECOGNITION OF CHARACTERS

Automatic number-plate recognition is a technology that uses optical character recognition on images to read vehicle registration plates to create vehicle location data Video Relay Interpreting(VRI) is used by police forces around the world for law enforcement purposes, including to check if a vehicle is registered or licensed. Video Relay Interpreting (VRI) can be used to store the images captured by the cameras as well as the text from the license plate, with some configurable to store a photograph of the driver.

Systems commonly use infrared lighting to allow the camera to take the picture at any time of day or night**.** Then optical character recognition (OCR) to extract the alphanumeric of the license plate.



### Figure 17

# SYSTEM IMPLEMENTATION

## CLIENT SIDE CODING:

@autho

r: Zhiyon g

"""

import torch.utils.data as utils import torch

import numpy as np import pandas as pd from Models import \*

from Train\_Validate import \*

def PrepareDataset(speed\_matrix, BATCH\_SIZE = 40, seq\_len = 10, pred\_len = 1, train\_propotion = 0.7, valid\_propotion = 0.2):

""" Prepare training and testing datasets and dataloaders.

Convert speed/volume/occupancy matrix to training and testing dataset.

The vertical axis of speed\_matrix is the time axis and the horizontal axis

is the spatial axis.

Args:

speed\_matrix: a Matrix containing spatial-temporal speed data for a network

seq\_len: length of input sequence pred\_len: length of predicted sequence

Returns:

Training dataloader Testing dataloader

"""

time\_len = speed\_matrix.shape[0]

max\_speed = speed\_matrix.max().max() speed\_matrix = speed\_matrix / max\_speed

speed\_sequences, speed\_labels = [], []

for i in range(time\_len - seq\_len - pred\_len): speed\_sequences.append(speed\_matrix.iloc[i:i+seq\_len].values)

speed\_labels.append(speed\_matrix.iloc[i+seq\_len:i+seq\_len+pred

\_len].values)

speed\_sequences, speed\_labels = np.asarray(speed\_sequences), np.asarray(speed\_labels)

# shuffle and split the dataset to training and testing datasets sample\_size = speed\_sequences.shape[0]

index = np.arange(sample\_size, dtype = int) np.random.shuffle(index)

train\_index = int(np.floor(sample\_size \* train\_propotion)) valid\_index = int(np.floor(sample\_size \* ( train\_propotion +

valid\_propotion)))

train\_data, train\_label = speed\_sequences[:train\_index], speed\_labels[:train\_index]

valid\_data, valid\_label = speed\_sequences[train\_index:valid\_index], speed\_labels[train\_index:valid\_index]

test\_data, test\_label = speed\_sequences[valid\_index:], speed\_labels[valid\_index:]

train\_data, train\_label = torch.Tensor(train\_data), torch.Tensor(train\_label)

valid\_data, valid\_label = torch.Tensor(valid\_data), torch.Tensor(valid\_label)

test\_data, test\_label = torch.Tensor(test\_data), torch.Tensor(test\_label)

train\_dataset = utils.TensorDataset(train\_data, train\_label) valid\_dataset = utils.TensorDataset(valid\_data, valid\_label) test\_dataset = utils.TensorDataset(test\_data, test\_label)

train\_dataloader = utils.DataLoader(train\_dataset, batch\_size = BATCH\_SIZE, shuffle=True, drop\_last = True)

valid\_dataloader = utils.DataLoader(valid\_dataset, batch\_size = BATCH\_SIZE, shuffle=True, drop\_last = True)

test\_dataloader = utils.DataLoader(test\_dataset, batch\_size = BATCH\_SIZE, shuffle=True, drop\_last = True)

return train\_dataloader, valid\_dataloader, test\_dataloader, max\_speed

if name == " main ": data = 'inrix'

# data = 'loop'

if data == 'inrix': speed\_matrix =

pd.read\_pickle('../Data/inrix\_seattle\_speed\_matrix\_2012') A = np.load('../Data/INRIX\_Seattle\_2012\_A.npy') FFR\_5min =

np.load('../Data/INRIX\_Seattle\_2012\_reachability\_free\_flow\_5mi n.npy')

FFR\_10min = np.load('../Data/INRIX\_Seattle\_2012\_reachability\_free\_flow\_10m in.npy')

FFR\_15min = np.load('../Data/INRIX\_Seattle\_2012\_reachability\_free\_flow\_15m in.npy')

FFR\_20min = np.load('../Data/INRIX\_Seattle\_2012\_reachability\_free\_flow\_20m in.npy')

FFR\_25min = np.load('../Data/INRIX\_Seattle\_2012\_reachability\_free\_flow\_25m in.npy')

FFR = [FFR\_5min, FFR\_10min, FFR\_15min, FFR\_20min, FFR\_25min]

elif data == 'loop':

speed\_matrix = pd.read\_pickle('../Data/speed\_matrix\_2015') A = np.load('../Data/Loop\_Seattle\_2015\_A.npy')

FFR\_5min = np.load('../Data/Loop\_Seattle\_2015\_reachability\_free\_flow\_5min. npy')

FFR\_10min = np.load('../Data/Loop\_Seattle\_2015\_reachability\_free\_flow\_10mi n.npy')

FFR\_15min = np.load('../Data/Loop\_Seattle\_2015\_reachability\_free\_flow\_15mi n.npy')

FFR\_20min = np.load('../Data/Loop\_Seattle\_2015\_reachability\_free\_flow\_20mi n.npy')

FFR\_25min = np.load('../Data/Loop\_Seattle\_2015\_reachability\_free\_flow\_25mi n.npy')

FFR = [FFR\_5min, FFR\_10min, FFR\_15min, FFR\_20min, FFR\_25min]

#

# train\_dataloader, valid\_dataloader, test\_dataloader, max\_speed

= PrepareDataset(speed\_matrix) #

rnn, rnn\_loss = TrainRNN(train\_dataloader, valid\_dataloader, num\_epochs = 1)

### rnn\_loss = [losses\_train, losses\_interval\_train, losses\_valid, losses\_interval\_valid]

rnn\_test = TestRNN(rnn, test\_dataloader, max\_speed ) ### rnn\_test = [losses\_l1, losses\_mse, mean\_l1, std\_l1]

lstm, lstm\_loss = TrainLSTM(train\_dataloader, valid\_dataloader, num\_epochs = 1)

lstm\_test = TestLSTM(lstm, test\_dataloader, max\_speed )

gclstm, gclstm\_loss = TrainGraphConvolutionalLSTM(train\_dataloader, valid\_dataloader, A, FFR, K=3, back\_length = 2, num\_epochs = 1, Clamp\_A = True)

gclstm\_test = TestGraphConvolutionalLSTM(gclstm, test\_dataloader, max\_speed)

gclstm\_proposed, gclstm\_proposed\_loss = TrainGraphConvolutionalLSTM\_Proposed(train\_dataloader, valid\_dataloader, A, FFR, K=3, back\_length = 2, num\_epochs = 1, Clamp\_A = True, lambda\_Aweight = 0.01, lambda\_fea = 0.01)

gclstm\_proposed\_test = TestGraphConvolutionalLSTM(gclstm\_proposed, test\_dataloader, max\_speed)

## SERVER SIDE CODING:

@author: Zhiyong

"""

import torch

import numpy as np

from torch.autograd import Variable import time

from Models import \*

def TrainRNN(train\_dataloader, valid\_dataloader, num\_epochs = 3):

inputs, labels = next(iter(train\_dataloader)) [batch\_size, step\_size, fea\_size] = inputs.size() input\_dim = fea\_size

hidden\_dim = fea\_size output\_dim = fea\_size

rnn = RNN(input\_dim, hidden\_dim, output\_dim) rnn.cuda()

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.L1Loss()

learning\_rate = 1e-5

optimizer = torch.optim.RMSprop(rnn.parameters(), lr = learning\_rate)

use\_gpu = torch.cuda.is\_available() interval = 100

losses\_train = []

losses\_interval\_train = []

losses\_valid = [] losses\_interval\_valid = []

cur\_time = time.time() pre\_time = time.time()

for epoch in range(num\_epochs):

print('Epoch {}/{}'.format(epoch, num\_epochs - 1)) print('-' \* 10)

trained\_number = 0

valid\_dataloader\_iter = iter(valid\_dataloader)

for data in train\_dataloader: inputs, labels = data

if inputs.shape[0] != batch\_size: continue

if use\_gpu:

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

else:

inputs, labels = Variable(inputs), Variable(labels) rnn.zero\_grad()

# rnn.loop()

hidden = rnn.initHidden(batch\_size)

outputs = None for i in range(10):

outputs, hidden = rnn(torch.squeeze(inputs[:,i:i+1,:]),

hidden)

#######

loss\_train = loss\_MSE(outputs, labels)

losses\_train.append(loss\_train.data) optimizer.zero\_grad() loss\_train.backward() optimizer.step()

# validation try:

inputs\_val, labels\_val = next(valid\_dataloader\_iter) except StopIteration:

valid\_dataloader\_iter = iter(valid\_dataloader) inputs\_val, labels\_val = next(valid\_dataloader\_iter)

if use\_gpu:

inputs\_val, labels\_val = Variable(inputs\_val.cuda()), Variable(labels\_val.cuda())

else:

inputs\_val, labels\_val = Variable(inputs\_val), Variable(labels\_val)

hidden = rnn.initHidden(batch\_size)

outputs = None for i in range(10):

outputs, hidden = rnn(torch.squeeze(inputs\_val[:,i:i+1,:]), hidden)

loss\_valid = loss\_MSE(outputs, labels\_val) losses\_valid.append(loss\_valid.data)

# output trained\_number += 1

if trained\_number % interval == 0: cur\_time = time.time()

loss\_interval\_train = np.around(sum(losses\_train[- interval:]).cpu().numpy()[0]/interval, decimals=8)

losses\_interval\_train.append(loss\_interval\_train)

loss\_interval\_valid = np.around(sum(losses\_valid[- interval:]).cpu().numpy()[0]/interval, decimals=8)

losses\_interval\_valid.append(loss\_interval\_valid) print('Iteration #: {}, train\_loss: {}, valid\_loss: {},

time: {}'.format(\ trained\_number \* batch\_size, \ loss\_interval\_train,\ loss\_interval\_valid,\

np.around([cur\_time - pre\_time], decimals=8) ) ) pre\_time = cur\_time

return rnn, [losses\_train, losses\_interval\_train, losses\_valid, losses\_interval\_valid]

def TrainLSTM(train\_dataloader, valid\_dataloader, num\_epochs

= 3):

inputs, labels = next(iter(train\_dataloader)) [batch\_size, step\_size, fea\_size] = inputs.size() input\_dim = fea\_size

hidden\_dim = fea\_size output\_dim = fea\_size

lstm = LSTM(input\_dim, hidden\_dim, output\_dim) lstm.cuda()

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.L1Loss()

learning\_rate = 1e-5

optimizer = torch.optim.RMSprop(lstm.parameters(), lr = learning\_rate)

use\_gpu = torch.cuda.is\_available()

interval = 100 losses\_train = [] losses\_interval\_train = [] losses\_valid = [] losses\_interval\_valid = []

cur\_time = time.time() pre\_time = time.time()

for epoch in range(num\_epochs):

print('Epoch {}/{}'.format(epoch, num\_epochs - 1)) print('-' \* 10)

trained\_number = 0

valid\_dataloader\_iter = iter(valid\_dataloader)

for data in train\_dataloader: inputs, labels = data

if inputs.shape[0] != batch\_size: continue

if use\_gpu:

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

else:

inputs, labels = Variable(inputs), Variable(labels) lstm.zero\_grad()

Hidden\_State, Cell\_State = lstm.loop(inputs)

loss\_train = loss\_MSE(Hidden\_State, labels)

losses\_train.append(loss\_train.data) optimizer.zero\_grad() loss\_train.backward() optimizer.step()

# validation try:

inputs\_val, labels\_val = next(valid\_dataloader\_iter) except StopIteration:

valid\_dataloader\_iter = iter(valid\_dataloader) inputs\_val, labels\_val = next(valid\_dataloader\_iter)

if use\_gpu:

inputs\_val, labels\_val = Variable(inputs\_val.cuda()), Variable(labels\_val.cuda())

else:

inputs\_val, labels\_val = Variable(inputs\_val), Variable(labels\_val)

Hidden\_State, Cell\_State = lstm.loop(inputs\_val)

loss\_valid = loss\_MSE(Hidden\_State, labels\_val) losses\_valid.append(loss\_valid.data)

# output trained\_number += 1

if trained\_number % interval == 0: cur\_time = time.time()

loss\_interval\_train = np.around(sum(losses\_train[- interval:]).cpu().numpy()[0]/interval, decimals=8)

losses\_interval\_train.append(loss\_interval\_train) loss\_interval\_valid = np.around(sum(losses\_valid[-

interval:]).cpu().numpy()[0]/interval, decimals=8) losses\_interval\_valid.append(loss\_interval\_valid) print('Iteration #: {}, train\_loss: {}, valid\_loss: {},

time: {}'.format(\

trained\_number \* batch\_size, \ loss\_interval\_train,\ loss\_interval\_valid,\

np.around([cur\_time - pre\_time], decimals=8) ) ) pre\_time = cur\_time

return lstm, [losses\_train, losses\_interval\_train, losses\_valid, losses\_interval\_valid]

def TrainGraphConvolutionalLSTM(train\_dataloader, valid\_dataloader, A, FFR, K, back\_length = 3, num\_epochs = 3, Clamp\_A=False):

inputs, labels = next(iter(train\_dataloader)) [batch\_size, step\_size, fea\_size] = inputs.size() input\_dim = fea\_size

hidden\_dim = fea\_size output\_dim = fea\_size

gclstm = GraphConvolutionalLSTM(K, torch.Tensor(A), FFR[back\_length], A.shape[0], Clamp\_A=Clamp\_A)

gclstm.cuda()

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.L1Loss()

learning\_rate = 1e-5

optimizer = torch.optim.RMSprop(gclstm.parameters(), lr = learning\_rate)

use\_gpu = torch.cuda.is\_available() interval = 100

losses\_train = []

losses\_interval\_train = [] losses\_valid = [] losses\_interval\_valid = []

cur\_time = time.time() pre\_time = time.time()

for epoch in range(num\_epochs):

print('Epoch {}/{}'.format(epoch, num\_epochs - 1)) print('-' \* 10)

trained\_number = 0

# validation data loader iterator init valid\_dataloader\_iter = iter(valid\_dataloader)

for data in train\_dataloader: inputs, labels = data

if inputs.shape[0] != batch\_size: continue

if use\_gpu:

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

else:

inputs, labels = Variable(inputs), Variable(labels) gclstm.zero\_grad()

Hidden\_State, Cell\_State = gclstm.loop(inputs) loss\_train = loss\_MSE(Hidden\_State, labels) optimizer.zero\_grad()

loss\_train.backward()

optimizer.step()

losses\_train.append(loss\_train.data)

# validation try:

inputs\_val, labels\_val = next(valid\_dataloader\_iter) except StopIteration:

valid\_dataloader\_iter = iter(valid\_dataloader) inputs\_val, labels\_val = next(valid\_dataloader\_iter)

if use\_gpu:

inputs\_val, labels\_val = Variable(inputs\_val.cuda()), Variable(labels\_val.cuda())

else:

inputs\_val, labels\_val = Variable(inputs\_val), Variable(labels\_val)

Hidden\_State, Cell\_State = gclstm.loop(inputs\_val) loss\_valid = loss\_MSE(Hidden\_State, labels) losses\_valid.append(loss\_valid.data)

# output trained\_number += 1

if trained\_number % interval == 0: cur\_time = time.time()

loss\_interval\_train = np.around(sum(losses\_train[- interval:]).cpu().numpy()[0]/interval, decimals=8)

losses\_interval\_train.append(loss\_interval\_train) loss\_interval\_valid = np.around(sum(losses\_valid[-

interval:]).cpu().numpy()[0]/interval, decimals=8) losses\_interval\_valid.append(loss\_interval\_valid) print('Iteration #: {}, train\_loss: {}, valid\_loss: {},

time: {}'.format(\ trained\_number \* batch\_size, \ loss\_interval\_train,\ loss\_interval\_valid,\

np.around([cur\_time - pre\_time], decimals=8) ) )

pre\_time = cur\_time

return gclstm, [losses\_train, losses\_interval\_train, losses\_valid, losses\_interval\_valid]

def TrainGraphConvolutionalLSTM\_Proposed(train\_dataloader, valid\_dataloader, A, FFR, K, back\_length = 3, num\_epochs = 3, Clamp\_A=False, lambda\_Aweight = 0.01, lambda\_fea = 0.01):

inputs, labels = next(iter(train\_dataloader)) [batch\_size, step\_size, fea\_size] = inputs.size() input\_dim = fea\_size

hidden\_dim = fea\_size output\_dim = fea\_size

gclstm = GraphConvolutionalLSTM(K, torch.Tensor(A), FFR[back\_length-1], A.shape[0], Clamp\_A=Clamp\_A)

gclstm.cuda()

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.L1Loss()

learning\_rate = 1e-5

optimizer = torch.optim.RMSprop(gclstm.parameters(), lr = learning\_rate)

use\_gpu = torch.cuda.is\_available() interval = 100

losses\_train = []

losses\_interval\_train = [] losses\_valid = [] losses\_interval\_valid = []

cur\_time = time.time() pre\_time = time.time()

for epoch in range(num\_epochs):

print('Epoch {}/{}'.format(epoch, num\_epochs - 1)) print('-' \* 10)

trained\_number = 0

# validation data loader iterator init valid\_dataloader\_iter = iter(valid\_dataloader)

for data in train\_dataloader: inputs, labels = data

if inputs.shape[0] != batch\_size: continue

if use\_gpu:

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

else:

inputs, labels = Variable(inputs), Variable(labels) gclstm.zero\_grad()

batch\_size = inputs.size(0) time\_step = inputs.size(1)

Hidden\_State, Cell\_State = gclstm.initHidden(batch\_size) previous\_grads = []

# Proposed Real Time Branching Learning Method for i in range(time\_step):

Hidden\_State, Cell\_State, gc = gclstm.forward(torch.squeeze(inputs[:,i:i+1,:]), Hidden\_State, Cell\_State)

gclstm.zero\_grad() if i != time\_step - 1:

label\_loss = loss\_MSE(Hidden\_State, torch.squeeze(inputs[:,i+1:i+2,:]))

else:

label\_loss = loss\_MSE(Hidden\_State, labels)

# Graph Convolution Weight Regularization weight\_loss = 0

for idx in range(K):

gc\_i\_weight = gclstm.gc\_list[idx].weight A\_i = gclstm.A\_list[idx]

weight\_loss+=loss\_L1(torch.mul(gc\_i\_weight, Variable(A\_i).cuda()), target=torch.zeros\_like(gc\_i\_weight))

# Graph Convolution Features Regularization gc\_loss = 0

gc\_features = torch.chunk(gc, K, 1) for idx in range(K-1):

gc\_i = gc\_features[idx] gc\_i1 = gc\_features[idx+1] gc\_loss = gc\_loss +

loss\_MSE(Variable(gc\_i.data).cuda(), Variable(gc\_i1.data).cuda())

loss = label\_loss + weight\_loss \* lambda\_Aweight + gc\_loss \* lambda\_Aweight

optimizer.zero\_grad() loss.backward()

curr\_grad = [x.grad.data for x in list(gclstm.parameters())]

if len(previous\_grads) != 0: # not null previous\_grads\_sum = previous\_grads[0] # sum of

gradients in previous steps

for idx in range(1, len(previous\_grads)): pre\_grad = previous\_grads[idx] previous\_grads\_sum += pre\_grad

# add previous gradients to current step only for the LSTM weights and bias, not for GC weights

idx = 0

for x in list(gclstm.parameters()):

if idx >= K: # not add grads on GC1, GC2, GC3... x.grad.data += previous\_grads\_sum[idx]

idx+=1

# only store fixed steps of previous gradients ( length = back\_length)

if len(previous\_grads) == back\_length: previous\_grads.pop(0) previous\_grads.append(curr\_grad)

else:

previous\_grads.append(curr\_grad) optimizer.step()

Hidden\_State, Cell\_State = gclstm.reinitHidden(batch\_size, Hidden\_State.data, Cell\_State.data)

loss\_train = loss\_MSE(Hidden\_State, labels) losses\_train.append(loss\_train.data)

# validation try:

inputs\_val, labels\_val = next(valid\_dataloader\_iter) except StopIteration:

valid\_dataloader\_iter = iter(valid\_dataloader) inputs\_val, labels\_val = next(valid\_dataloader\_iter)

if use\_gpu:

inputs\_val, labels\_val = Variable(inputs\_val.cuda()), Variable(labels\_val.cuda())

else:

inputs\_val, labels\_val = Variable(inputs\_val), Variable(labels\_val)

Hidden\_State, Cell\_State = gclstm.loop(inputs\_val) loss\_valid = loss\_MSE(Hidden\_State, labels) losses\_valid.append(loss\_valid.data)

# output trained\_number += 1

if trained\_number % interval == 0: cur\_time = time.time()

loss\_interval\_train = np.around(sum(losses\_train[- interval:]).cpu().numpy()[0]/interval, decimals=8)

losses\_interval\_train.append(loss\_interval\_train) loss\_interval\_valid = np.around(sum(losses\_valid[-

interval:]).cpu().numpy()[0]/interval, decimals=8) losses\_interval\_valid.append(loss\_interval\_valid) print('Iteration #: {}, train\_loss: {}, valid\_loss: {},

time: {}'.format(\ trained\_number \* batch\_size, \ loss\_interval\_train,\ loss\_interval\_valid,\

np.around([cur\_time - pre\_time], decimals=8) ) ) pre\_time = cur\_time

return gclstm, [losses\_train, losses\_interval\_train, losses\_valid, losses\_interval\_valid]

def TestRNN(rnn, test\_dataloader, max\_speed):

inputs, labels = next(iter(test\_dataloader)) [batch\_size, step\_size, fea\_size] = inputs.size()

cur\_time = time.time()

pre\_time = time.time()

use\_gpu = torch.cuda.is\_available()

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.MSELoss()

tested\_batch = 0

losses\_mse = [] losses\_l1 = []

for data in test\_dataloader: inputs, labels = data

if inputs.shape[0] != batch\_size: continue

if use\_gpu:

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

else:

inputs, labels = Variable(inputs), Variable(labels)

# rnn.loop()

hidden = rnn.initHidden(batch\_size)

outputs = None for i in range(10):

outputs, hidden = rnn(torch.squeeze(inputs[:,i:i+1,:]), hidden)

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.L1Loss() loss\_mse = loss\_MSE(outputs, labels) loss\_l1 = loss\_L1(outputs, labels)

losses\_mse.append(loss\_mse.data) losses\_l1.append(loss\_l1.data)

tested\_batch += 1

if tested\_batch % 1000 == 0: cur\_time = time.time()

print('Tested #: {}, loss\_l1: {}, loss\_mse: {}, time:

{}'.format( \

tested\_batch \* batch\_size, \ np.around([loss\_l1.data[0]], decimals=8), \ np.around([loss\_mse.data[0]], decimals=8), \ np.around([cur\_time - pre\_time], decimals=8) ) )

pre\_time = cur\_time losses\_l1 = np.array(losses\_l1) losses\_mse = np.array(losses\_mse)

mean\_l1 = np.mean(losses\_l1) \* max\_speed std\_l1 = np.std(losses\_l1) \* max\_speed

print('Tested: L1\_mean: {}, L1\_std : {}'.format(mean\_l1, std\_l1))

return [losses\_l1, losses\_mse, mean\_l1, std\_l1]

def TestLSTM(lstm, test\_dataloader, max\_speed):

inputs, labels = next(iter(test\_dataloader)) [batch\_size, step\_size, fea\_size] = inputs.size()

cur\_time = time.time() pre\_time = time.time()

use\_gpu = torch.cuda.is\_available()

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.MSELoss()

tested\_batch = 0

losses\_mse = [] losses\_l1 = []

for data in test\_dataloader:

inputs, labels = data

if inputs.shape[0] != batch\_size: continue

if use\_gpu:

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

else:

inputs, labels = Variable(inputs), Variable(labels)

Hidden\_State, Cell\_State = lstm.loop(inputs) loss\_MSE = torch.nn.MSELoss()

loss\_L1 = torch.nn.L1Loss()

loss\_mse = loss\_MSE(Hidden\_State, labels) loss\_l1 = loss\_L1(Hidden\_State, labels)

losses\_mse.append(loss\_mse.data) losses\_l1.append(loss\_l1.data)

tested\_batch += 1

if tested\_batch % 1000 == 0: cur\_time = time.time()

print('Tested #: {}, loss\_l1: {}, loss\_mse: {}, time:

{}'.format( \

tested\_batch \* batch\_size, \ np.around([loss\_l1.data[0]], decimals=8), \ np.around([loss\_mse.data[0]], decimals=8), \ np.around([cur\_time - pre\_time], decimals=8) ) )

pre\_time = cur\_time losses\_l1 = np.array(losses\_l1) losses\_mse = np.array(losses\_mse)

mean\_l1 = np.mean(losses\_l1) \* max\_speed std\_l1 = np.std(losses\_l1) \* max\_speed

print('Tested: L1\_mean: {}, L1\_std : {}'.format(mean\_l1, std\_l1))

return [losses\_l1, losses\_mse, mean\_l1, std\_l1]

def TestGraphConvolutionalLSTM(gclstm, test\_dataloader, max\_speed):

inputs, labels = next(iter(test\_dataloader)) [batch\_size, step\_size, fea\_size] = inputs.size()

cur\_time = time.time() pre\_time = time.time()

use\_gpu = torch.cuda.is\_available()

loss\_MSE = torch.nn.MSELoss() loss\_L1 = torch.nn.L1Loss()

tested\_batch = 0

losses\_mse = [] losses\_l1 = []

for data in test\_dataloader: inputs, labels = data

if inputs.shape[0] != batch\_size: continue

if use\_gpu:

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

else:

inputs, labels = Variable(inputs), Variable(labels)

Hidden\_State, Cell\_State = gclstm.loop(inputs) loss\_MSE = torch.nn.MSELoss()

loss\_L1 = torch.nn.L1Loss()

loss\_mse = loss\_MSE(Hidden\_State, labels) loss\_l1 = loss\_L1(Hidden\_State, labels)

losses\_mse.append(loss\_mse.data) losses\_l1.append(loss\_l1.data)

tested\_batch += 1

if tested\_batch % 1000 == 0: cur\_time = time.time()

print('Tested #: {}, loss\_l1: {}, loss\_mse: {}, time:

{}'.format( \

tested\_batch \* batch\_size, \ np.around([loss\_l1.data[0]], decimals=8), \ np.around([loss\_mse.data[0]], decimals=8), \ np.around([cur\_time - pre\_time], decimals=8) ) )

pre\_time = cur\_time losses\_l1 = np.array(losses\_l1) losses\_mse = np.array(losses\_mse)

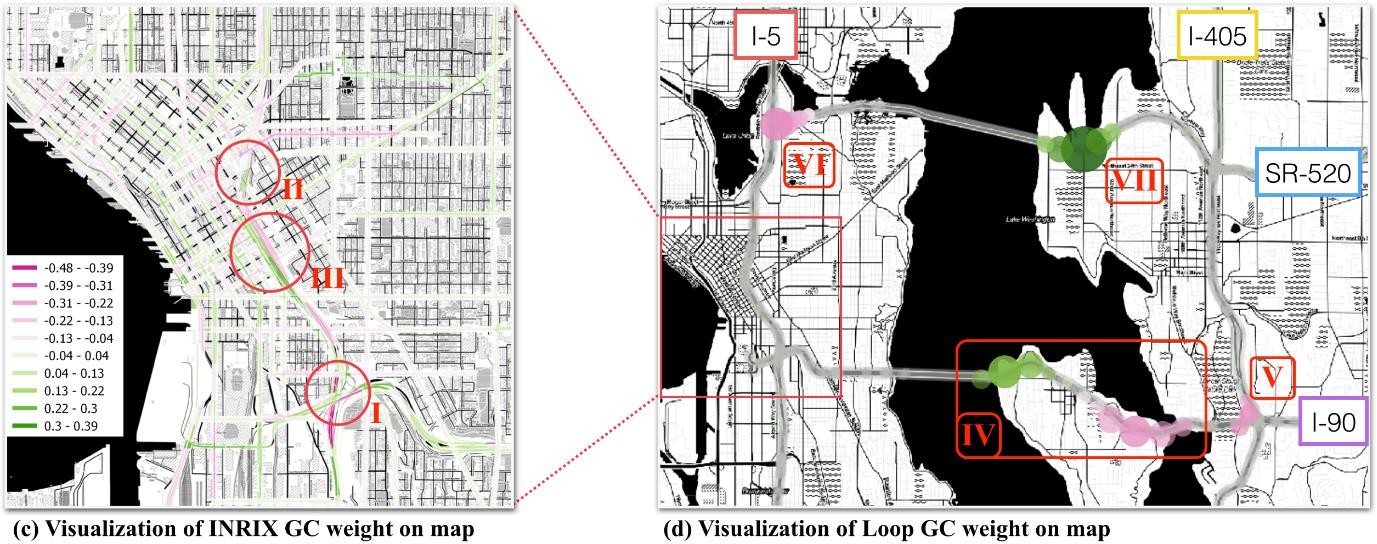
mean\_l1 = np.mean(losses\_l1) \* max\_speed std\_l1 = np.std(losses\_l1) \* max\_speed

print('Tested: L1\_mean: {}, L1\_std : {}'.format(mean\_l1, std\_l1))

return [losses\_l1, losses\_mse, mean\_l1, std\_l1]

# SYSTEM TESTING

## TEST CASE REPORT



### Figure 18 Figure 19

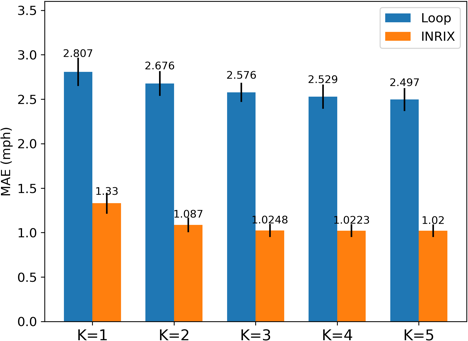
 **PERFORMANCE ANALYSIS**

Figure 20

# CONCLUSION

## CONCLUSION AND FUTURE ENHANCEMENT

This methods are efficient to detect accidents using both hardware and software methods which provide good results. Most of the discussed methods also provide the driver with the option of turning of the alarm in cases where the accident is not serious or false detections of an accident. Previous methods are either mostly dependent on some hardware like sensors that have to be present in the car or require a smart phone to be present within the car. While the use of such hardware can prove to be a more cost-efficient approach it has the drawback of being destroyed in the accident and hence giving spurious or no readings at all. Hence, an approach that does not depend on any hardware device or sensor that is associated with the car is required for the detection of traffic accidents.

# APPENDICES:

## SAMPLE SCREENSHOTS:

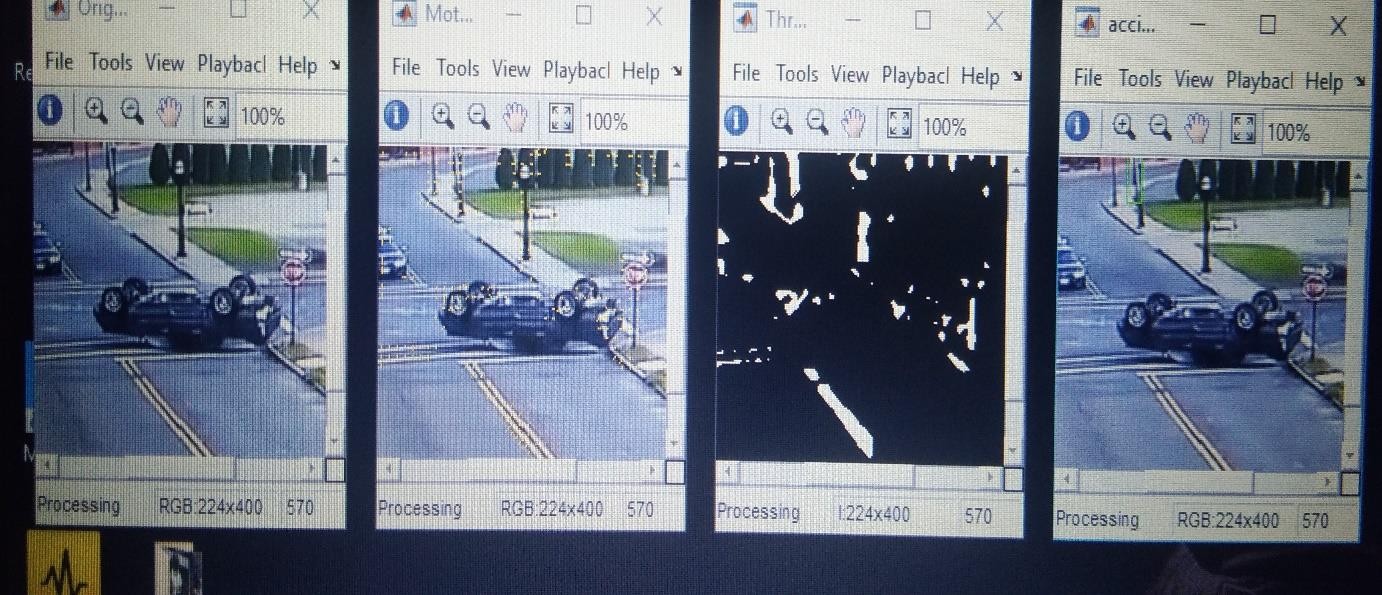


Figure 21

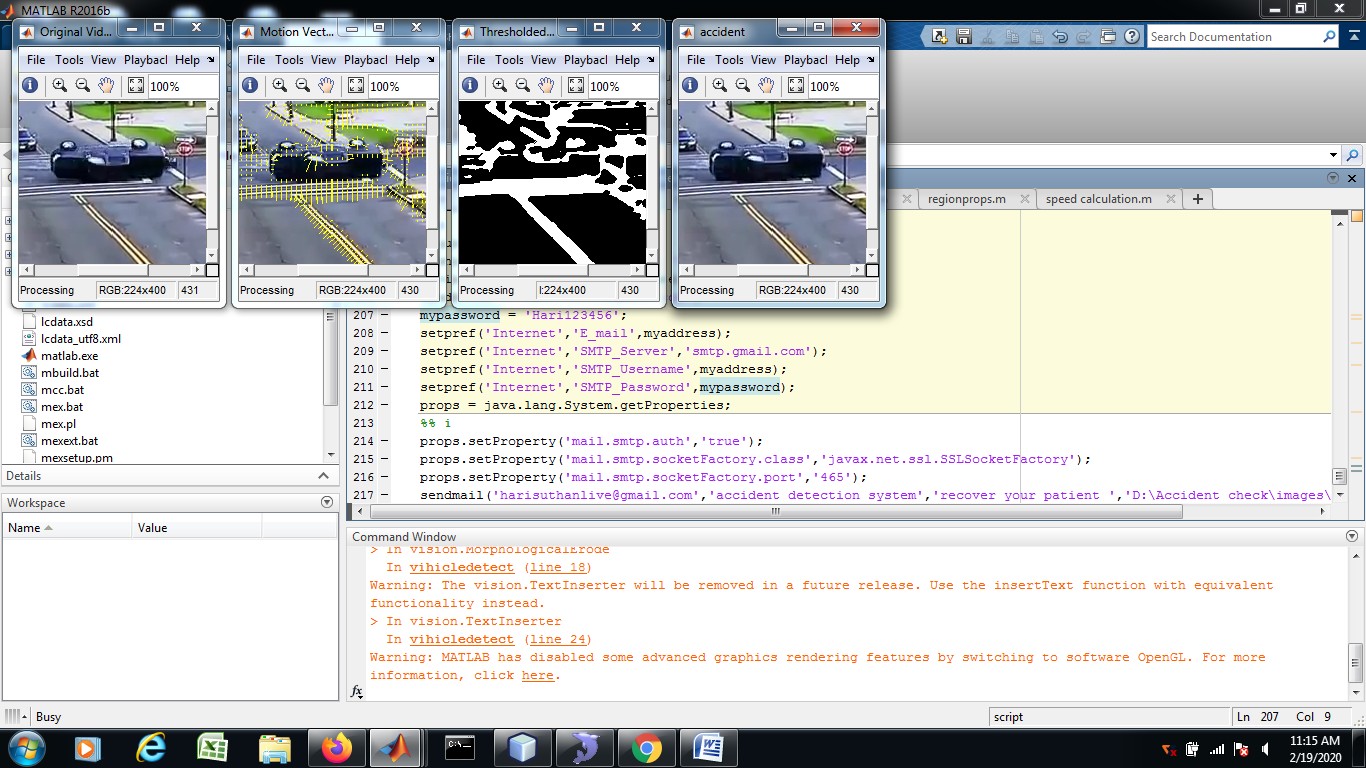


Figure 22

**SAMPLE SCREENSHOTS -CONT**

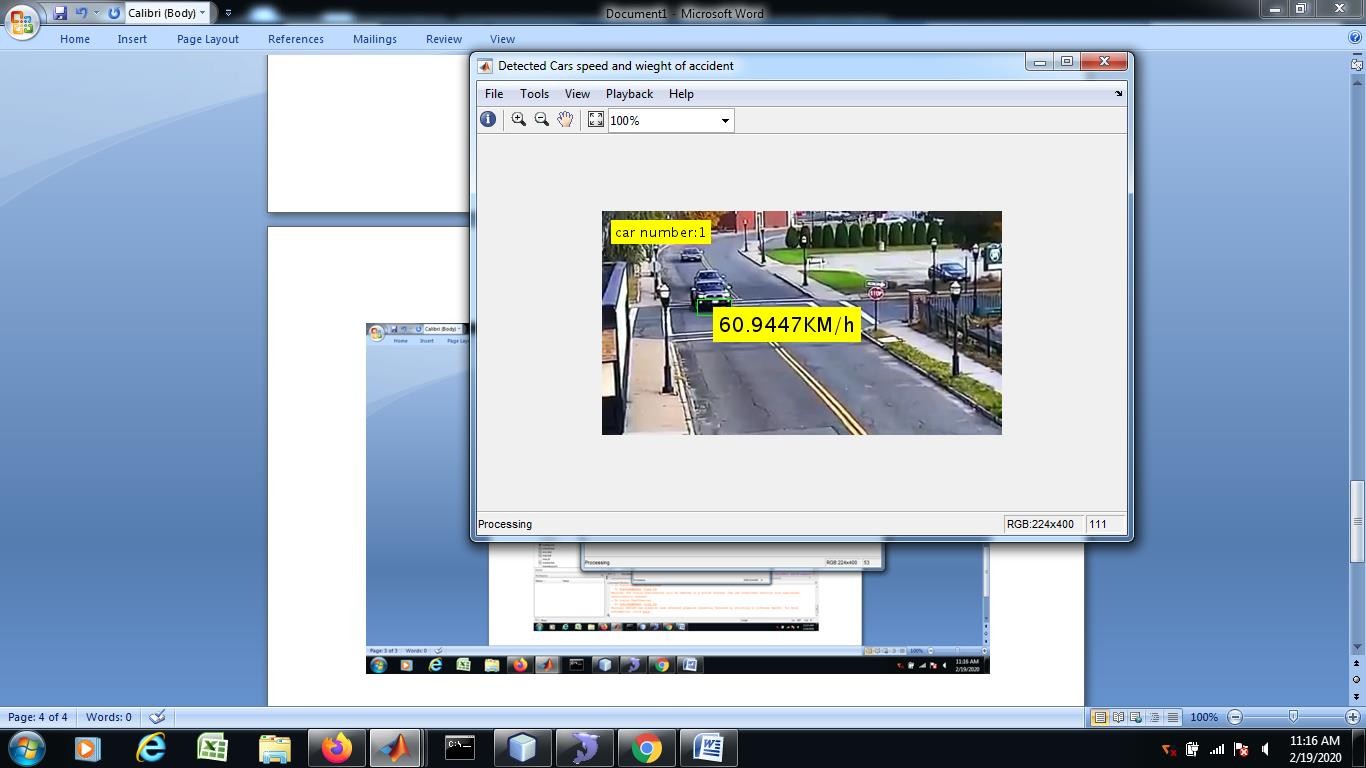
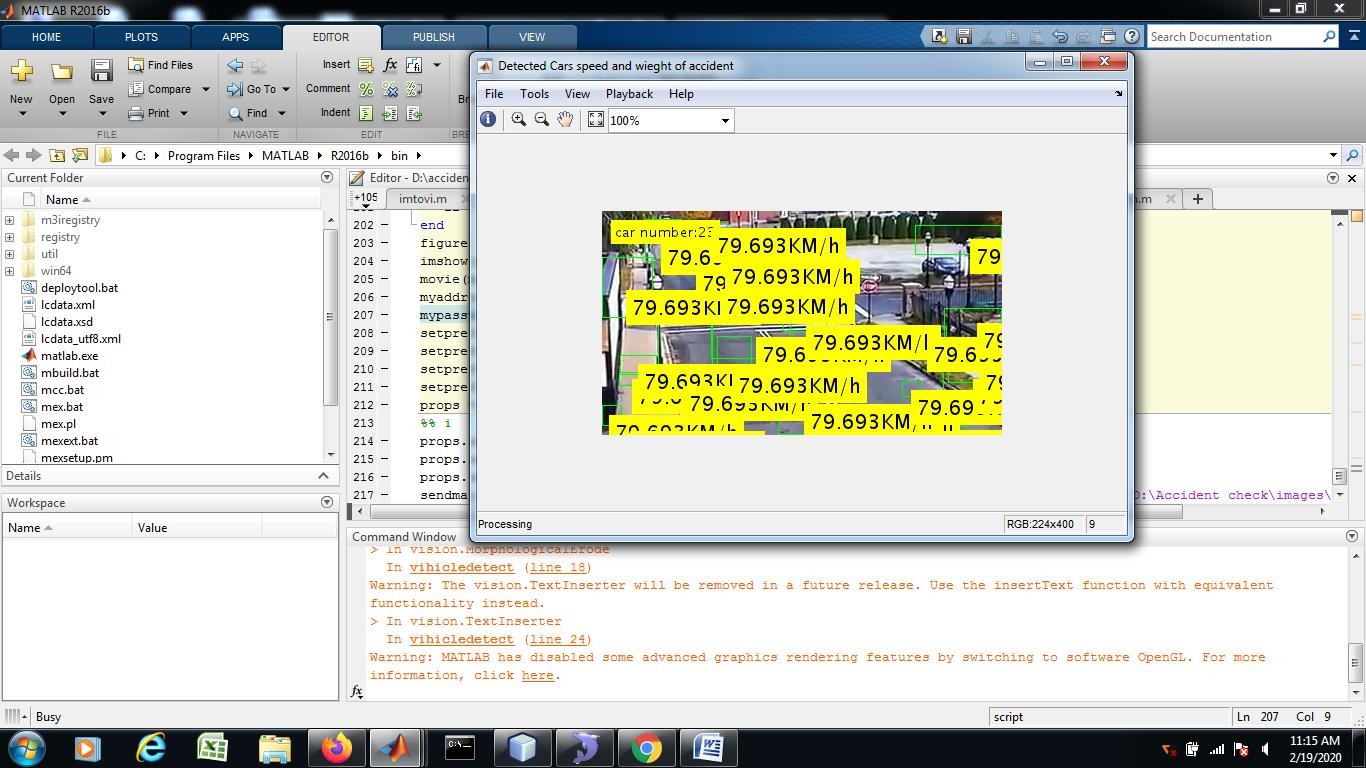


Figure 23



**SAMPLE SCREENSHOT-CONT**

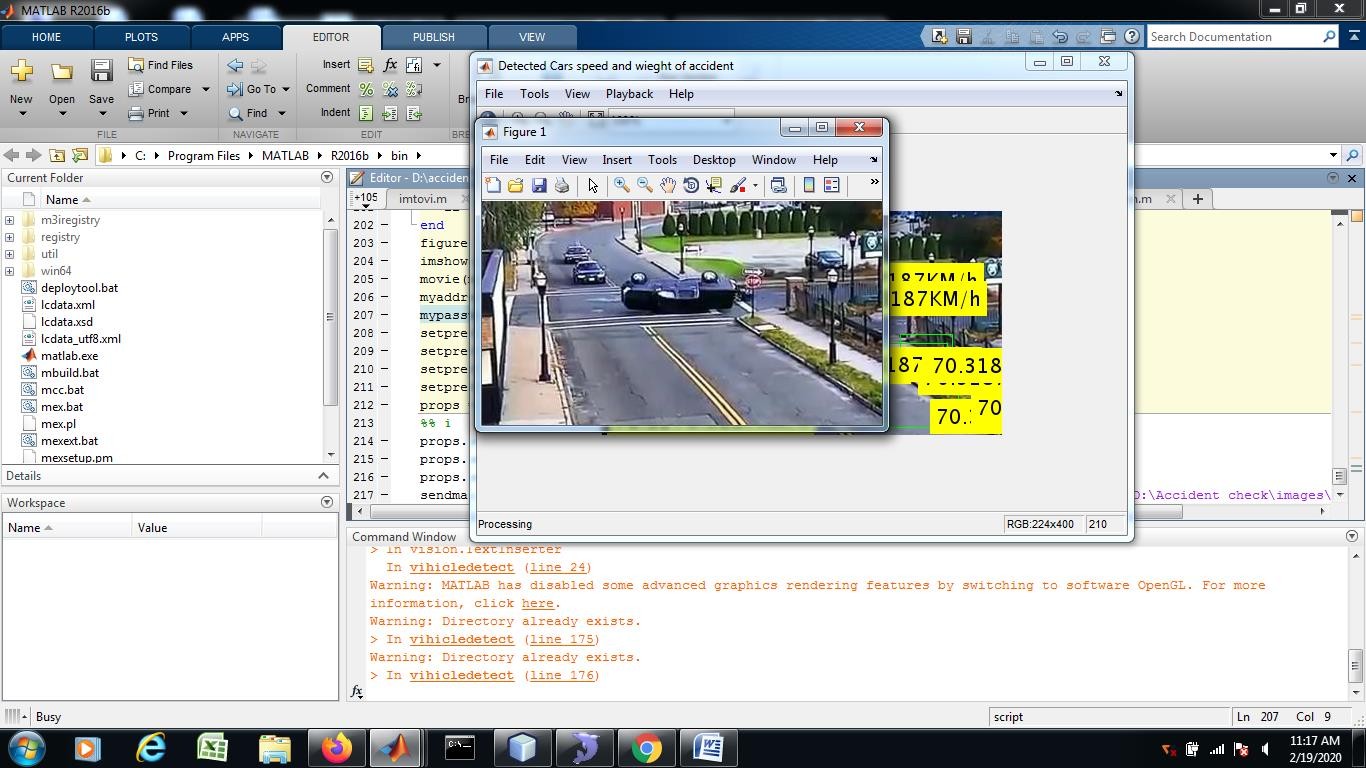
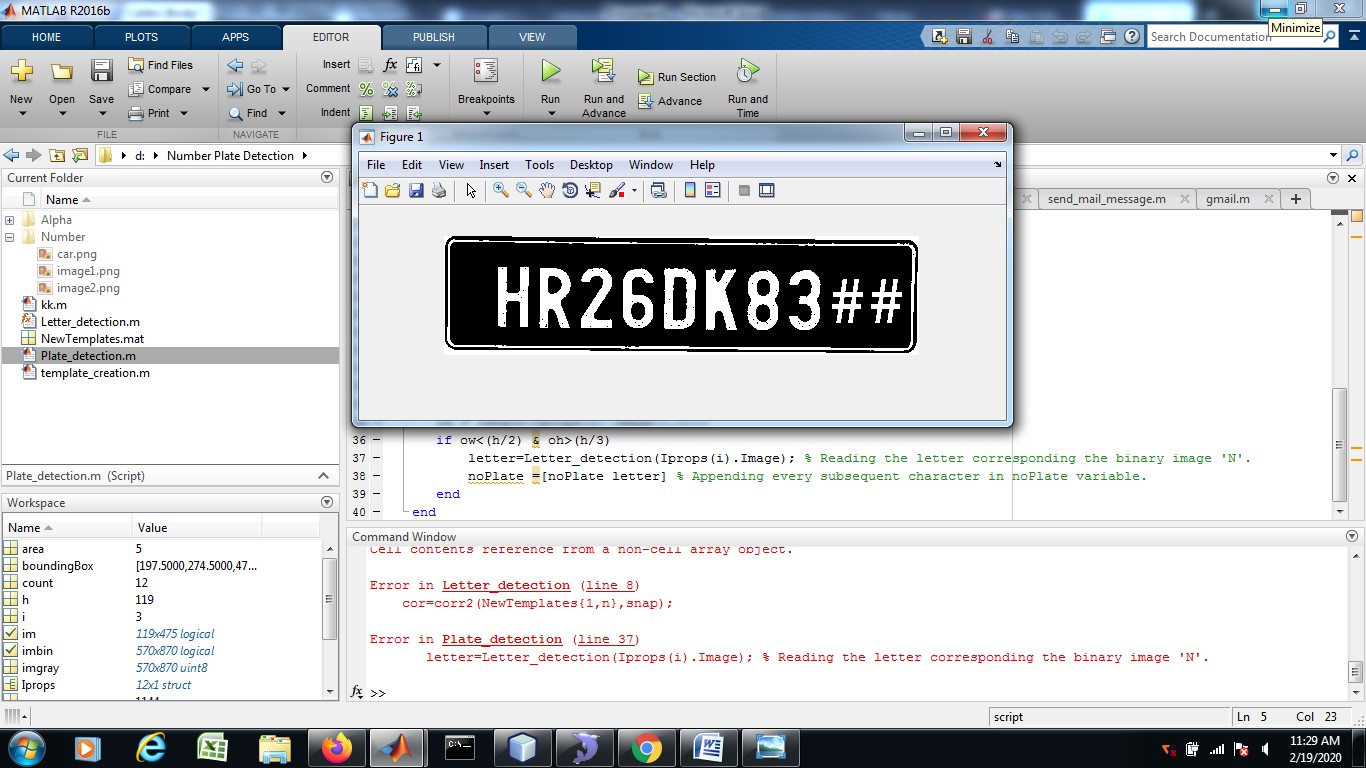


Figure 25



# REFERENCES:

* + "Global status report on road safety 2015", World Health Organization, 2016. [Online]. Available: <http://www.who.int/violence_injury_preve> ntion/road\_safety\_status/2015/en/. (Accessed: 22- Mar- 2016)
  + B. Prachi, D. Kasturi and C. Priyanka, "Intelligent Accident-Detection And Ambulance- Rescue System", International Journal Of Scientific & Technology Research, Vol. 3, No. 6, 2016.
  + M. Ruikar, "National statistics of road traffic accidents in India", Journal of Orthopedics, Traumatology and Rehabilitation, vol. 6, no. 1, p. 1, 2013.
  + T. Rahman, "Road Accidents in Bangladesh: An Alarming Issue", The World Bank, 2012. [Online]
  + M. M. L. Elahi, R. Yasir, M. A. Syrus, M. S. Q. Z. Nine, I. Hossain and N. Ahmed, "Computer vision based road traffic accident and anomaly detection in the context of Bangladesh," 2014 International Conference on Informatics, Electronics & Vision (ICIEV), Dhaka, 2014, pp. 1-6
  + P. A. Nandurge and N. V. Dharwadkar, "Analyzing road accident data using machine learning paradigms," 2017 International Conference on I- SMAC (IoT in Social, Mobile, Analytics, and Cloud) (I-SMAC), Palladam, 2017, pp. 604-610
  + K. M. Habibullah, A. Alam, S. Saha, A. Amin and A. K. Das, " A Driver- Centric Carpooling: Optimal Route-Finding Model using Heuristic Multi– Objective Search," 2019 4th International Conference on Computer and Communication Systems (ICCCS), Singapore, 2019
  + M. S. Satu, S. Ahamed, F. Hossain, T. Akter and D. M. Farid, "Mining traffic accident data of N5 national highway in Bangladesh employing decision trees," 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, 2017, pp. 722-725.
  + S. Kumar and D. Toshniwal, "A data mining approach to characterize road accident locations," Journal of Modern Transportation(2016), vol. 24, issue no. 1, pp. 62-72
  + Beshah, Tibebe, and Shawndra Hill, “Mining Road Traffic Accident Data to Improve Safety: Role of Road-Related Factors on Accident Severity in Ethiopia,” AAAI Spring Symposium: Artificial Intelligence for Development (2010).
  + A. Esmaeili, M. Khalili and A. Pakgohar, "Determining the road defects impact on accident severity; based on vehicle situation after accident, an approach of logistic regression," 2012 International
  + Conference on Statistics in Science, Business and Engineering (ICSSBE), Langkawi, 2012, pp. 1-4.
  + I. Rish, "An Empirical Study of the Naïve Bayes Classifier," IJCAI 2001 Work Empir Methods Artif Intell (2001), vol