Spring 2018

Real time traffic congestion detection using images

Revanth Ayala Somayajula

Iowa State University, revanth@iastate.edu

Follow this and additional works at: https://lib.dr.iastate.edu/creativecomponents

Part of the Other Computer Engineering Commons

Recommended Citation
Ayala Somayajula, Revanth, "Real time traffic congestion detection using images" (2018). Creative Components. 2.
https://lib.dr.iastate.edu/creativecomponents/2

This Creative Component is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Creative Components by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Real time traffic congestion detection using images

by

Revanth Ayala Somayajula

A Creative Component submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Anuj Sharma, Co-major Professor
Samik Basu, Co-major Professor
Soumik Sarkar

Iowa State University
Ames, Iowa
2018

Copyright © Revanth Ayala Somayajula, 2018. All rights reserved.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>vi</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vii</td>
</tr>
<tr>
<td>CHAPTER 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 MOTIVATION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2. BACKGROUND</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER 3. METHODOLOGIES</td>
<td>4</td>
</tr>
<tr>
<td>3.1 DESIGN</td>
<td>4</td>
</tr>
<tr>
<td>3.1.1 Image Module</td>
<td>4</td>
</tr>
<tr>
<td>3.1.2 Video Module</td>
<td>5</td>
</tr>
<tr>
<td>3.1.3 Open Source Libraries Used</td>
<td>6</td>
</tr>
<tr>
<td>3.2 IMPLEMENTATION</td>
<td>7</td>
</tr>
<tr>
<td>3.2.1 Stack Used</td>
<td>7</td>
</tr>
<tr>
<td>3.2.2 Image Module</td>
<td>7</td>
</tr>
<tr>
<td>3.2.3 Video Module</td>
<td>11</td>
</tr>
<tr>
<td>3.3 INFRASTRUCTURE</td>
<td>13</td>
</tr>
<tr>
<td>3.3.1 Implementation</td>
<td>13</td>
</tr>
<tr>
<td>3.3.2 Maintenance</td>
<td>14</td>
</tr>
<tr>
<td>CHAPTER 4. RESULTS</td>
<td>15</td>
</tr>
<tr>
<td>4.1 Accuracy</td>
<td>15</td>
</tr>
<tr>
<td>Table</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Accuracy Test</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Model Accuracy</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.1</td>
<td>Overall design of the system</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>ER diagram of the database</td>
<td>10</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Machines being used on Azure</td>
<td>13</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to Dr. Anuj Sharma for his guidance, patience and support throughout the course of this research and the writing of this report. I would also like to extend my gratitude to Dr. Samik Basu for his valuable suggestions and Dr. Soumik Sarkar for his help and support.

In addition, I would also like to thank Pranamesh Chakraborty for his help during the course of this project.
ABSTRACT

There is an increasing demand to utilize modern technology in the field of transportation to help decrease congestion on roads so that proper measures can be pursued to facilitate lower travel times and an effective utilization of the transportation network. This project aims to develop a solution for real time detection of traffic congestion on a road. The solution captures images from the live feed of traffic cameras situated at various locations and runs a deep learning algorithm to detect whether an image shows traffic congestion. Using a set of these images and a persistence check, the application identifies the time congestion started to take place and the time it has ended. For the purposes of analysis, the solution also records videos of the detected incidents.
CHAPTER 1. INTRODUCTION

1.1 MOTIVATION

Traffic congestion is an issue worldwide. There are multiple factors that contribute for the congestion of traffic. Some of those factors are heavy volume of vehicles, inadequate green time, too many pedestrians etc... Some of the problems that arise due to traffic congestion are delays, more fuel consumption leading to more pollution, emergency vehicles taking longer time etc... (6). In 2015, according to the analytics firm Inrix, a driver spent approximately 81 hours a year stuck in Traffic in Los Angles and around 75 hours a year in San Francisco and Washington, DC (7).

It is common knowledge that congestion occurs primarily during the morning time when people are going to work and again during the evening time when people are trying to get home. This is down to the fact that the volume of traffic is very high during these hours. This is known as recurring congestion. If we can collect enough data to help us identify when congestion takes place, using this information we can observe if this is a recurring congestion or a non-recurring congestion. This will aid the authorities in identifying the cause in case of recurring congestion and aid them in taking the appropriate measures, so it can be avoided.

This is the long-term view. However, if the authorities can identify congestion in real time, they can use methods such as Dynamic Message boards to re-route the traffic and thus reduce the volume of traffic on a congested road and clear out the congestion. This also helps in better utilization of the traffic network as other routes will be used effectively.

This application caters to both the needs. The primary aim of this work is to use images captured from the live feed of traffic camera and identify if an image is congested or not. However, a single image cannot help detect whether congestion is taking place or not. Hence, a set of consecutive images from a single camera are used to identify if congestion is taking place or not. This is known as persistence check.
In this report, firstly a brief background of the existing work regarding real time detection of congestion is provided followed by design and the implementation of this project. It then provides an insight to the tests conducted and their results and finally concludes by talking about how this project can be further improved and how it can be integrated with other works.
CHAPTER 2. BACKGROUND

In this section, we discuss the already existing work that is being done in detecting traffic congestion using various other methods and analyze how our methodology is different from what is already being done.

In a study by Li et al. [3], a human computer interaction vehicle area detection was put forward to speed up processing and select the interesting area. Then using the texture difference between a congested image and an unobstructed image, they propose a vehicle density estimation. They estimate the threshold vehicle density using a variety of methods and compare the seen value with the threshold to determine traffic congestion. One of the key contributions of this study proposal of vehicle density and an accuracy of $\approx 99\%$.

In another study by Mandal et al. [2] propose a system that uses Radio Frequency Identification, an emerging technology. They use RFID to count the number of vehicles on the road. In this paper they propose a system that effectively detects vehicle congestion and subsequently manages it efficiently to ensure the smooth flow of traffic using the Active RFID devices.

According to the study by Nidhal et al.[1], the authors developed a system that counts the vehicles on road by detecting and paring the vehicles backlights from a real time captured image. This ensures that the system takes shorter time and can be implemented at a highway level.

We can observe that papers mentioned above try to detect traffic congestion using various different methods and try to leverage the other technologies. Our project is different from the others in a way that we are using images captured from camera and also with the advancements in Machine Learning over the past few years, many new technologies and libraries have been developed that can classify an image and detect objects much more accurately and at a much faster pace than proposed in [3].
CHAPTER 3. METHODOLOGIES

3.1 DESIGN

The application comprises of 2 main modules: 1. Image module 2. Video module

The Figure 3.1 shows the overall design of the application.

![Figure 3.1 Overall design of the system](image)

3.1.1 Image Module

The Image module has 2 sub modules, Image Capture and Image Prediction modules.
3.1.1.1 Image Capture Module

The primary purpose of the Image capture module is to continuously capture images from the live feed of a camera at particular intervals of time and store those images on to a disk as jpg file. While storing those images on to the disk, it is important that the file name follow a particular format so that metadata such as camera name and the time the image was taken can be deduced from the file name.

3.1.1.2 Image Prediction Module

The Image Prediction module is the module that predicts whether there is congestion on an image or not. It takes as inputs the images captured and stored on disk by the Image Capture Module, runs a deep learning algorithm that predicts the probability of congestion and no-congestion on an image. The algorithm outputs 2 values for each image. One is the probability of congestion in the image and the other is the probability of no-congestion in the image.

Depending on the probabilities of congestion and no-congestion, each image is classified into 2 classes. If the probability of congestion is more than the probability of no-congestion, then it is classified as congested image or else it is classified as non-congested image. All the values, the camera from which the image was taken, the time at which the image was taken both of which can be gathered by the image file name and the values of congestion probability, no-congestion probability and the final Class prediction are stored into a database.

Once a set of images and their respective values are stored on to a database, the database then has a stored procedure known as Persistence Check. This Persistence Check uses the values stored in the database for a particular camera to predict if any incidents have occurred for that particular camera and stores the incident details such as camera, time incident predicted to have started and time incident predicted to have ended.

3.1.2 Video Module

The Video module has 2 sub modules, Video Capture and Video Upload modules.
3.1.2.1 Video Capture Module

The aim of the video module is to capture the live feed of a camera in chunks of 5 minute videos and store those videos on to a disk. As with the Image Capture Module, even with this module it is important that the video file name follow a proper format so that metadata such as the time video has started and the camera to which the video belongs can be easily obtained from the file name.

3.1.2.2 Video Upload Module

Video Upload module is the module that takes in as inputs the videos stored on the disk by the Video Capture module and using the incidents predicted by the persistence check in the database, output all the videos related to the incidents. These relevant videos are then stored into a mongo database, along with the incident details such as the incident ID to help us identify the all the videos related to a particular incident.

3.1.3 Open Source Libraries Used

3.1.3.1 ffmpeg

ffmpeg is a very fast audio and video converter that can capture from a live audio/video stream. It can convert between arbitrary sample rates and resize the video on the fly with a high quality polyphase filter. ffmpeg can read from an arbitrary number of input files which can be regular files, streams, pipes etc... specified by the option -i and write to an arbitrary number of output files which are specified by a plain output url.

3.1.3.2 keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with
image and text data easier. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, on the Java Virtual Machine or on clusters of Graphic Processing Units.

3.2 IMPLEMENTATION

3.2.1 Stack Used

- Python
- Bash
- Java
- MySQL
- mongoDB

3.2.2 Image Module

The Image module has 2 sub modules, Image Capture and Image Prediction modules.

3.2.2.1 Image Capture Module

As mentioned earlier, the Image capture module is used to continuously capture images from the live feed of a camera at particular intervals of time and store those images on to a disk. The module ensures that when images are saved to disk, they are stored in such a way that the absolute path to the image file gives us all the metadata we need to store in the database. This way we can extract all the metadata we need from the path and need not scan or process the image to extract the relevant information from the image. The metadata that we need for an image file are the camera that the image was taken from and the time that this image was taken at. Hence, the absolute path to the image has the following format:

<Particular_Location>/Camera_The_Image_Was_Taken_From/<Date_&_Time_The_Image_Was_Taken>.jpg
The module also verifies whether the directory exists or not. If it does not, creates that directory and save the images to the directory. The module uses a library called ffmpeg to continuously capture an image from an IP address and converts into image JPG image and save it to the disk. To achieve this, the application inputs a file called camera.txt. This file contains camera details in a particular format. Each line has details about one camera separated by “”. The application first inputs the above-mentioned text file, gets each line in the file, splits the line into its respective values using the delimiter and gets the name of the camera and the IP address of the camera.

For each camera and its corresponding IP address, the module creates a command that captures the image from that IP address at every 20 second interval and runs the command in the background and saves the output to a log file. The command looks something like this:

```bash
ffmpeg -i <ip_address> -vf fps=1/20 -strftime 1 <Location_on_disk>/ %Y-%m-%d-%H-%M-%S.jpg > ~/logs/<camera_name>.txt 2>&1 &
```

The `%Y-%m-%d-%H-%M-%S.jpg` makes sure that the name of the file is contains the date and time the image is being saved which is the same as the time the image is being captured. The `> ~/logs/<camera_name>.txt 2>&1 &` makes sure that the command is running in the background and that the output of the saved is saved to a log file. `fps = 1/20` ensures that an image is captured every 20 seconds. This module once kicked off will run until explicitly shutdown. Hence, it will keep capturing images endlessly at every 20 second intervals for all the cameras mentioned in the camera.txt file and save them to the disk.

### 3.2.2.2 Image Prediction Module

Once the first set of images are saved to the disk, the Image Prediction module will start to run. This module takes as input a location on disk and gets all the images in that folder, its subfolders and so on. This way it collects all the images stored on the disk for various cameras. Once this module has the images, using the scipy module, it resizes the images to a size of 266 x 155 pixels and creates an np array for these images.
The deep learning model uses a 4-layer deep convoluted Neural Network. The deep learning model is stored in 2 files. The model is stored in a file called model.json and the weights are stored in a file called models_weights.h5. The module then loads this json file, the weights file and creates a model using the Keras Library. The model then takes in as input the np array of the images and predicts the probability of congestion and no-congestion for each image and outputs an array where each element contains 2 probabilities, one for congestion and one for no-congestion. The order of the array is the same as the order of the np-array of images making it easier for us to correlate.

The np array of the images and the probabilities are then send to the Database Module. The database Module, for each image, looks at the probability of congestion and no-congestion. If the probability of congestion is more than the probability of no-congestion, then that images is classified as congested image. If not, it is classified as non-congested image. Then from the np array of the images, extracts all the necessary information that we need to store in the database such as the camera name, the date and time the image was taken. Then from the probabilities array gets the congestion and no-congestion, compare the values and determine the final class. All these values are then stored on to the database.

The database that we are using is MySQL. The database basically consists of 2 tables. One is the images table and the other is the detected_congestion table. The ER diagram for all the tables in the database is given in Figure 3.2 below.

### 3.2.2.3 Persistence Check

The images table is the table where the values from the Database Module are saved. Once the values from the Database Module are stored into the database, then a stored procedure called persistence check is called for each camera to determine all the incidents that have taken place. The flow chart below give a picture of how the check works.

- Step 1: Initialize 2 variables ‘r’ and ‘g’
- Step 2: Get all the records in the images table for a particular camera where date of the record is in the last one hour and order them according to the time the images were taken
• Step 3: Store the finalClass of the previous record in variable g and check if current record has the same finalClass as ‘g’

• Step 4: If so, do not increment the value of ‘r’. If not, increment the value of. By doing so, all the consecutive images that have the same class will have the same value of ‘r’

• Step 5: Group all the values that have the same ‘r’ and ‘g’ values into one record and have another column that has the count of all the records in this group and also the minimum time and maximum time

• Step 6: Select the records where the ‘g’ value is ‘1’ and count greater than 3

3.2.2.4 Model

Deep Convolutional Neural Networks are a state of the art technique for object detection and image classification. To develop this model, a traditional convolutional neural network architecture consisting of convolution and pooling layers was used. The images taken from the cameras of
various sizes majority of them being \(800 \times 450\). The images were then resized to \(266 \times 150\) to prevent memory allocation errors during training of the model. Next, these images were fed into the model as two consecutive convolution layers \(32 \times 3 \times 3\) in size (i.e., kernel size \(3 \times 3\)) followed by a max pooling layer \(2 \times 2\) in size. This was followed by two additional convolution layers \(64\) in size and then again max pooling with a \(2 \times 2\) filter. Each max pooling layer was followed by dropout with a probability of 0.25 to prevent overfitting. Finally, two fully connected layers (dense) were used, the first one with \(512\) neurons and the final one with two neurons corresponding to the binary classes (congestion and no-congestion). A batch size of 32 was used throughout the model and Leaky-ReLU was used as an activation function. This model was developed by Pranamesh Chakraborty.

### 3.2.3 Video Module

The Video module has 2 sub modules, Video Capture and Video Upload modules.

#### 3.2.3.1 Video Capture Module

The primary purpose of the Video capture module is to continuously record videos in chunks of 5 minutes from the live feed of a camera and store those images on to a disk. Just like the Image Capture Module, this module also ensures that if the directory does not exist, it creates that directory and save the videos to that directory. The module makes sure that when Videos are saved to disk, they are stored in such a way that the absolute path to the video file gives us all the metadata we need to store in the database. This way we can extract all the metadata we need from the path and need not process the Video to extract the relevant information from the video. The metadata that we need for an Video file are the camera that the Video is being taken from and the time that this video was started recording. Hence, the absolute path to the image has the following format:

\[
<\text{Particular\_Location}>/<\text{Camera\_The\_Video\_Was\_Taken\_From}>/<\text{Date\_&\_Time\_The\_Video\_Started}>.jpg
\]
The module uses a library called ffmpeg to continuously capture a video of 15 minutes from an IP address and converts into image flv video and save it to the disk. Using the camera.txt, the module gets the names of the camera and their corresponding IP Addresses. So, for each camera and its IP address, the module creates a command that records the video from the IP address in chunks of 5 minutes and runs the command in the background and saves the output to a log file. The command looks something like this:

```
ffmpeg -i <ip_address> -c copy -t 00:15:00 -f segment -segment_list out.list -reset_timestamps 1 -segment_time 300 -segment_atclocktime 1 -strftime 1 -time_base 1/30 <Location_on_disk>/%m-%d-%y-%H-%M-%S.mkv </dev/null > /dev/null 2>&1 &
```

The `-i <ip_address>` tells that the input is the ip address. The `<Location_on_disk>/%m-%d-%y-%H-%M-%S.mkv` ensures that when saving the video on to the disk the name of the video file is the same as the time when it started recording. `-segment_time 300` says that the video file should be 5 minutes long. The `reset_timestamps 1` makes sure that video recording starts at the next 0th second.

### 3.2.3.2 Video Upload Module

This module is implemented as a cronjob that runs once every day. This module scans all the videos stored on the disk by the Video Capture module and ensures that we are storing only the videos that are related to the incidents that have been identified by the persistence check procedure in the database.

This module, for each camera and for each incident looks at the incident start time and end time. Then using the names of the video files, determines the videos that are relevant to the incident start time and end time and uploads only those videos the mongoDB. Then all the videos are discarded. When uploading to the mongoDB, the relevant metadata such as the incident id that these videos are related to are also inserted so that the we know which incident these videos are related to.
3.3 INFRASTRUCTURE

3.3.1 Implementation

It is important that the systems running the various modules have sufficient computing power as the application is continuously captures images and videos indefinitely and also as the prediction module runs a 4-layer convoluted neural network. Hence, to achieve this we are currently using Azure cloud services. This image represents the currently deployment scenario on Microsoft Azure.

We currently have 4 D16s v3 machines on Azure. One of machines runs the MySQL database and one of the machine runs the mongoDB. Each machine has 16 vcpus and a RAM of 64GB. Among the other 2 machines, one machine runs the Image Module, both the Image Capture Module and the Image Prediction module. The other machine runs the Video Module, both the video capture and video upload modules. However, at this moment we have more than 300 cameras and these machine does not have enough computing power to handle capturing images and videos from 300 cameras and store them. Hence for every 50 cameras we use 2 machines, One machine that runs the Image Module for those set of cameras and another machine that runs the Video Module depicted in Figure 3.3 below.

![Figure 3.3 Machines being used on Azure](image-url)
All these machines will capture the images, process them individually but will send the the information on to the same database. This way we ensure consistency. As with the Video Module, they capture the videos for all the 50 cameras and at the end of the run the Video Upload Module uploads the videos to mongoDB, clears up all the videos and records the videos for the next day.

### 3.3.2 Maintenance

Since both the modules have programs that are running 24x7 every day it is important that we need have a system in place that identifies if any of the programs have shut down and tries to restart the program. If it can’t restart, then it should identify the administrator that there is a problem with the system.

This application also implements such a system. The application has a bash script in place that runs as a cronjob every 5 minutes. The script uses the Linux bash command “ps -ef” and tries to see if all the Image Capture, Image Prediction and Video Capture modules are running or not. If they are, it does nothing. If they aren’t, then it identifies the programs that are not running and tries to restart those programs. If it can’t restart them, then it sends an email to administrator to let him know that the program has stopped and that the script was not able to restart program. The same is described below.

- Run ps -ef and check if any of the modules are not running
- For each module not running
  - Try to restart the module. If not restarted, send an email.
CHAPTER 4. RESULTS

This section describes the various tests that we have performed and shows the results of those tests.

4.1 Accuracy

This test was done to test the accuracy of our system. To perform this test, we have run the system for 3 days and detected incidents. Then, used data set provided by the Iowa DOT which provides the correct set of incidents to determine the accuracy of the system.

To calculate the accuracy of the system, we need to know the number of true positives. True positives are the incidents that have been detected by the system and also present in the list of incidents sent by the Iowa DOT. Using true positives, we can calculate the accuracy as

\[ \text{Accuracy} = \frac{\text{True Positives}}{\text{Total}} \]

The data corresponding to this experiment can be seen in the Table 4.1 below.

<table>
<thead>
<tr>
<th>Day</th>
<th>System Detected Incidents</th>
<th>Actual Incidents</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>11</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>9</td>
<td>44</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>13</td>
<td>38</td>
</tr>
</tbody>
</table>

Tests were also conducted to calculate the accuracy of the model. Parameters such as Precision, Recall and accuracy were calculated and the table shows the results of those tests [8].

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision %</th>
<th>Recall %</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCNN</td>
<td>86.9</td>
<td>93.9</td>
<td>90.2</td>
</tr>
</tbody>
</table>
CHAPTER 5. FUTURE SCOPE

This application has a lot to offer. Looking towards the future, with vast advancements in Machine Learning, this application can be further enhanced so that incident detection is much more accurate. Since this is purely a backend application, it can be integrated with various other applications to create more accurate, advanced and complex systems.

5.1 Model & Persistence Check

From the results, it can be seen that this application can be further improved to provide more accuracy. As the model uses a Convoluted Neural Network, effort can be made to make the model perform better especially on images after sunset. It can also been seen that the persistence check is not the best way for us to detect an incident. Hence, further work can be done to develop a method or an algorithm that can be used to better detect incidents.

5.2 Integration with other applications

As mentioned earlier, this is purely a back-end application. There is no frontend that allows the user to view the incident videos and take notes of how the incident has occurred so that improvements can be made to ensure that such incidents do not happen again. At the Institute of Transportation, there is a Front-End application developed that allows a user to view all the videos pertaining to a particular incident and take notes. If our application can upload the videos in such a way that the frontend can use these videos, it would be really helpful to view and take note. In addition to that, it would automate the process of identifying incidents, getting the relevant videos and taking notes from the videos as to why the incident has happened so that further preventive action can be taken.
CHAPTER 6. CONCLUSION

The main aim of this project was to create a real time system that captures a set of images from cameras, process those images, conclude whether there is congestion on that road and if there is then try to identify when the congestion has started and when it has ended. The application developed not only creates such a system but also ensures that the system can be maintained easily and handles errors. We primarily focused on how to modularize the system in such a way that modules can be easily reused and also make sure that each module can be used in cohesion with other systems and applications. We have also ensured that to make the system run in real time, tests were run to determine the maximum time of delay tolerable and ensured that the proper infrastructure was in place to make sure that from the time taken from the moment an image is captured to the time it is saved to the database is not more than the threshold value determined previously.

In our analysis of the system, we have found that system accurately detects approximately 45% of the actual incidents. With advancements in the field of the Machine learning, the accuracy of the system can be further increased and can be used to make more accurate conclusions. With the various others systems in place that use data streams such as Inrix, Wavetronix to detect incidents, this application in cohesion with such other systems can help us better detect incidents happening in real time and take appropriate measures.
REFERENCES


