Traffic Light Management System Using Image Processing

Gauri Rao[1], Bhavya Divecha[2], Jenish Joshi[3], Anishk Jaiswal[4]

[1]Assistant Professor, Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India

[2, 3, 4]Student, Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India

Abstract - These days, traffic congestion is a major issue. Urban areas are the ones most impacted by it, despite the fact that it appears to be present practically everywhere. Given that it is always increasing, it is essential to be aware of the road traffic density in real time for better signal control and effective traffic management. Different factors, such as inadequate capacity, unregulated demand, long wait periods, etc., can contribute to traffic congestion. Although insufficient capacity and unchecked demand are somewhat related, each light's delay is hard-coded and unrelated to traffic. In order to effectively meet this growing demand, traffic control needs to be optimised and simulated. For traveller information, ramp metering, and real-time updates, image processing and surveillance systems have been extensively used in traffic management in recent years. Image processing can also be used to estimate traffic density.

This project demonstrates how to calculate the real-time traffic density using image processing using live images from the cameras at traffic intersections. It also focuses on the algorithm for adjusting traffic signals in accordance with the number of vehicles on the road, with the goal of minimising traffic congestion and subsequently the frequency of accidents. People will be able to travel safely as a result, and both fuel usage and waiting times will be decreased. Additionally, it will deliver considerable data that will support study and planning of next road projects. With the goal of reducing traffic congestion and promoting free flow of traffic, additional traffic lights may be synchronised with one another. The technology doesn't use electronic sensors buried in the pavement to find the automobiles; instead, it uses photographs.

There will be a camera set up next to the traffic light. It will record video clips. A better method to manage the traffic light's state change is image processing. It demonstrates the ability to lessen traffic congestion and prevents time from being lost due to a green signal on an empty road. Because it makes use of actual traffic photos, it is also more accurate in calculating the presence of vehicles. It performs far better than systems that rely on the metal content of the automobiles since it can envision the practicality. Additionally, it gives emergency vehicles like ambulances priority.

I. INTRODUCTION

Common, middle-class consumers have been able to employ tiny, specialised applications to improve their quality of life during the past ten years thanks to the adoption and use of technologies like Mobility, Cloud, and Social Platforms. The usage of technology has significantly transformed the way we live, play, and work, whether it is as simple as paying your energy bills via mobile banking or purchasing your favourite movie ticket by simply clicking a few buttons. Despite the fact that we have been talking about smart cities and communities for some time, let's take a closer look at how the information and data that are currently at our disposal can be leveraged to build some truly smart services that, in a genuine sense, improve our quality of life.

We'll examine a significant case that affects us virtually every day: traffic management. Real-time analysis and the use of technology can truly result in efficient traffic management. Poor traffic prioritisation is a typical cause of traffic congestion; in such cases, certain lanes experience lighter traffic than others. Traffic congestion is escalating exponentially.

Take Chandigarh, one of India's Union Territories, as our example. India's highest car density is found in Chandigarh. More than 45,000 automobiles were registered in Chandigarh this year, bringing the total number of vehicles on the road there to more than 8 lakhs, according to Chandigarh Transport Undertaking. Although the number of automobiles is rising quickly, the city's infrastructure cannot keep up with this expansion.

Rush-hour traffic congestion is now commonplace, particularly in internal sections where stuck vehicles can be seen in lengthy lines. As a result, we have attempted to address the issue with the aid of our initiative, with the goal of reducing vehicular congestion. The final outcome of this was the implementation of a feedback system in the functioning of the traffic lights where the volume of traffic would also be taken into account while making choices. This was achieved through the use of image processing, which may be obtained through security cameras.

MOTIVATION

The primary driving force behind this initiative is a significant issue that practically every individual who used to travel by personal automobile in cities like Delhi encounters. For a time now, Delhi's traffic has been a major issue, and it is now becoming increasingly aggravating. If someone needs to be somewhere on time, they must depart far earlier than they would have if there was less traffic. Early in the morning and late at night, when most people commute to work, the issue only grows worse. All people must leave for work early and get home later than usual, which has an impact on how they behave.
Now, imagine what occurs when traffic becomes stalled due to extreme congestion. A traffic police officer arrives to help. He now switches off the static traffic signal and moves the traffic lanes in accordance with the volume of traffic in each lane.
Therefore, the likelihood of such traffic congestion will be reduced in the first place, sparing commuters from such traffic delays, if we can integrate a comparable thing directly into such traffic light systems.
According to a review directed by IIT Madras, Congestion forces stunning costs on the economy that nobody pays for. An expected yearly blockage cost of Rs 54,000 crore in 2013 which is 12.5 percent higher than Delhi's absolute yearly spending plan for the year 2017-18. The BCG report guaranteed that the 4 most clogged urban areas in India, alone expense around 1.5 lakh crore to India because of clog.

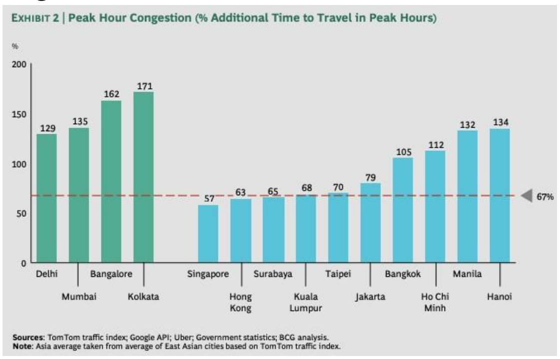


Fig. 1: Traffic Congestion in different cities

According to the BCG report ready for Uber, the vehicle interest in India is expanding at an exceptionally high speed, thus on the off chance that nothing will be finished to deal with the traffic, the clog will be practically excruciating and a ton of misfortune will be caused by the blockage.

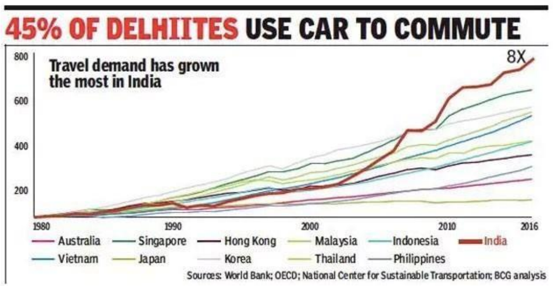


Fig. 2: Increasing growth of travel demand in India

PROBLEM STATEMENT

To foster a Traffic Signal Management framework that in light of the thickness of traffic on every path will progressively distribute time to every path as opposed to dispensing static opportunity to every path.
The framework will likewise focus on the entry of crisis vehicles, for example, ambulances with the goal that there isn't any postponement for the ones in crises because of traffic.

FUNCTIONAL REQUIREMENTS

Input: The input to the system would be the image captured by the each of the cameras installed at each traffic light.
Output: The output of the system would be the dynamic time allocated to each lane depending upon the density of the traffic on each lane which is calculated as per the count of the vehicles. Its also depends on the system which lane is given a green signal first depending upon the presence of any emergency vehicle.

2. LITERATURE SURVEY & TECHNOLOGIES TO BE USED

PYTHON

Python is a high-level syntax sensitive language. Python is free and open-source programming language. Python is extensively used with work related to computer vision and machine learning. Due to its ease of coding, it saves the programmers a lot of time to think about the code. A large number of libraries based on machine learning, deep learning, computer vision is available on open- source platform because of the compatibility and the ease of coding provided by it and this is the reason why programmer choose python for this particular work. Additionally, python has its own libraries like numpy and scipy which makes data representation and computation easier. Python is also used because it is not only limited to machine learning and related fields but also allows us to make a complete project by aiding us with GUI development, frontend development, etc.

MACHINE LEARNING

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data. The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension; as is the case in data mining applications, machine learning uses that data to detect patterns in data and adjust program actions accordingly. Machine learning algorithms are often categorized as being supervised or unsupervised.

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

NEURAL NETWORKS

Artificial neural networks (ANNs) are a computational model used in machine learning, computer science and other research disciplines, which is based on a large collection of connected simple units called artificial neurons, loosely analogous to axons in a biological brain. Connections between neurons carry an activation signal of varying strength. If the combined incoming signals are strong enough, the neuron becomes activated and the signal travels to other neurons connected to it. Such systems can be trained from examples, rather than explicitly programmed, and excel in areas where the solution or feature detection is difficult to express in a traditional computer program. Like other machine learning methods, neural networks have been used to solve a wide variety of tasks, like computer vision and speech recognition, that are difficult to solve using ordinary rule-based programming.

Typically, neurons are connected in layers, and signals travel from the first (input), to the last (output) layer. Modern neural network projects typically have a few thousand to a few million neural units and millions of connections; their computing power is similar to a worm brain, several orders of magnitude simpler than a human brain. The signals and state of artificial neurons are real numbers, typically between 0 and 1.

There may be a threshold function or limiting function on each connection and on the unit itself, such that the signal must surpass the limit before propagating. Back propagation is the use of forward stimulation to modify connection weights, and is sometimes done to train the network using known correct outputs

Neural Networks are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other.

The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

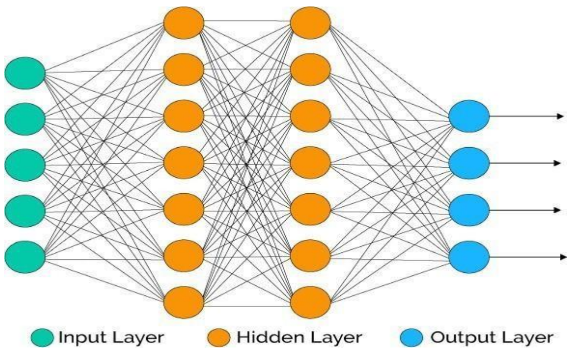


Fig.3: Example of ANN

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple Neural Network.

CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNN’s have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars.

RELU

ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by:

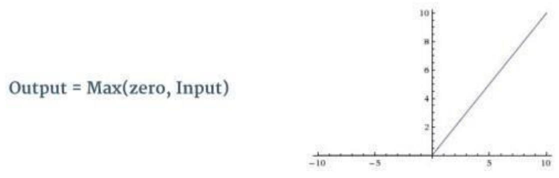


Fig. 4: RELU Function

ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

TENSORFLOW

TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

TRANSFER LEARNING

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems

Transfer learning is related to problems such as multi-task learning and concept drift and is not exclusively an area of study for deep learning.

Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained.

Transfer learning only works in deep learning if the model features learned from the first task are general.

Two common approaches to use transfer learning are as follows:

1.) Develop Model Approach
2.) Pre-trained Model Approach

OBJECT DETECTION

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e., the centre) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

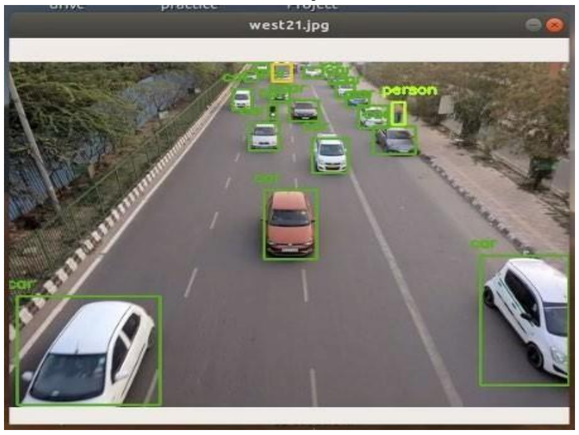


Fig. 5: Object detection used on vehicles on road

KERAS

Keras is an open-source neural network library written in Python. It is capable of running on top Tensorflow or Theano. Designed to enable fast experimentation with deep neural networks, Keras focuses on being minimal, modular and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. The library 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.

OPENCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OVERALL DESCRIPTION

This section contains general information about the specific requirements of the product, which will be developed shortly. Although we do not describe all requirements in detail, this section describes the factors that affect the final product.

The product takes 4 images as input, one for each direction, which are processed by the vehicle-trained object detection model. This model provides the number of vehicles on each street. It also provides the system with the presence of emergency vehicles on each road, if any.

The system then gives a green signal to each lane where emergency vehicles are located, and then moves the remaining lanes in cyclic order. It allocates the time to each lane depending on the vehicle density in each lane, solving the congestion problem and saving time wasted unnecessarily at a static traffic light when there is no traffic on some lanes.

The main component of the system is the model used to predict the vehicle density on each road. It is the backbone of the whole project, because if the density is not calculated correctly, it will lead to the failure of the whole project. Therefore, the model is extensively trained on the vehicles for a good calculation.

The traffic light allocates the time depending on the number and type of vehicles, for two-wheelers the time per vehicle is 1 second, for cars 2 seconds and for heavy vehicles such as buses and trucks 3 seconds.

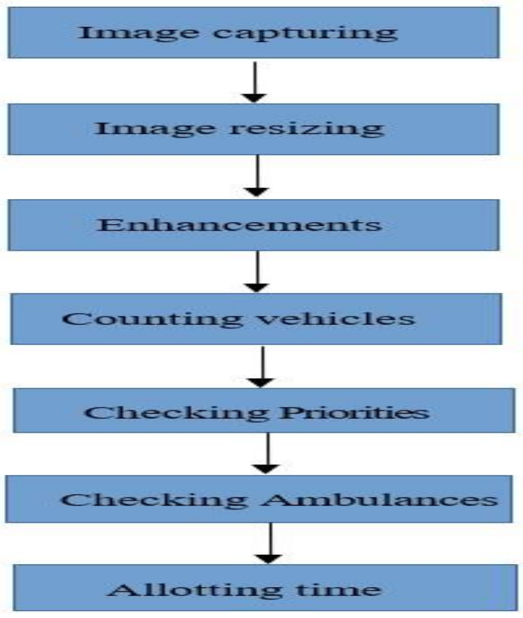


Fig. 6: Overall functioning of the mode

PRODUCT PERSPECTIVE & FUNCTIONS

This software product is eventually intended to automate the process of traffic signal time allocation by allocating time dynamically. The product will be used and deployed at traffic junctions specially at those junctions where traffic congestion is most. The system would enable the traffic to be managed by itself by allocating time dynamically and eliminating any traffic expert intervention which would prevent loses that occur due to wastage of time and resources and making day to day commute easier.

The first is to analyse images in each lane and calculate the number of vehicles in each lane. The model counts each vehicle type, making it easy to check for the presence of an emergency vehicle, and any future improvements can easily be added if some sort of vehicle category-based feature is added in the future.

The second function is to check the lane where an emergency vehicle may be. If emergency vehicles

are present in a lane, traffic in this lane is prioritized and receives the green signal first. If there is no emergency vehicle, the signals are given in cyclical order.
The third function is to assign time to each traffic light, i.e., how many seconds a traffic light has to be green. It is calculated based on the number of vehicles of each type in a lane. For a motorbike 1 second is added to the green signal time, for a car 2 seconds are added for each car and for buses and trucks 3 seconds are added, so this gives the total time calculated for each traffic light and each signal is calculated for the respective time green.

USER CHARACTERISTICS
There is almost nil user interaction with the system after the installation of the system at any junction. The target persons don't have to use the system, it will be a dynamic system which will work on its own. The system only needs one time installation at a traffic junction apart from maintenance and upgradation to be done.

ASSUMPTIONS
There is proper visibility at each traffic junction. The cameras are always working and capture reasonable area for analysis. There is proper light at each junction to capture quality images. The emergency vehicles are distinguishable from other vehicles.

SPECIFIC REQUIREMENTS
With this section and later, we will describe the requirements of the product in detail. Basically, we will categorize requirements in three which are namely external interface requirements, functional requirements and non-functional requirements. Except non- functional requirements, requirements of the product will be detailed under this section with brief information.

EXTERNAL INTERFACE REQUIREMENTS
User Interface Requirements: There is no special user interface required since no external commands are to be provided to the system to work, it works on its own.
Hardware Interface Requirements: The system requires 4 cameras to be installed at each traffic light system at a road intersection. The cameras are needed to provide a clear image of the roads to calculate the density of vehicles on each lane.
It also requires an obvious thing for the system to work and that is set of 4 traffic lights having the red, yellow and green colours with the colours meaning as usual.

PERFORMANCE REQUIREMENTS
System should take the image as an input where the image will be a real-time one taken by one of the cameras installed at the traffic signals. There should not be any kind of delay and the processing should be fast and synchronized. The system should be completely error free and smooth.

DESIGN CONSTRAINTS
Development Tools: The system shall be built using Python, TensorFlow, Keras.
Easy modification: The system should be designed in such a way that the time allocation algorithm can be easily changed if needed.
Image database: Database is required by the system's model for the purpose of training the proposed Neural Network for correct estimation of the number of vehicles.

NON-FUNCTIONAL REQUIREMENTS
Cost effectiveness – The system should be cost effective; the cost of installation should be reasonable so that it can be implemented on as many traffic signals as possible.

Usability – The system shall be easy to use and understand. User interaction is none to the system and so there must be proper automation in the allocation of time and synchronization among the lights at a particular junction.

Maintainability – Component driven development and modular structure shall ensure maintainable code.

3. MODEL PROPOSED

After going through all the papers, websites and tutorials, we have followed a process for the implementation of the project. First step will be to get the image dataset for the training of the model. So, we have collected the image dataset and we have also considered the point that image should be centred so that it will be easy for the better model training. Image is needed to go through the process of the pre-processing. Feature extraction and detection process is followed and then last stage will be image classification.

IMAGE ACQUISITION
The most important part for the given task is to acquire the real time traffic images because we need to train the model for real world vehicles for easy implementation. So, we need to have those background so that model is well trained. We went to the roads in Delhi to collect images at various roads of varying traffic. The example of the images is shown in the below figure:

Fig. 7: Dataset collected for training and testing

MODEL TRAINING USING YOLO MODELING

It predicts the bounding box coordinates and class probabilities for these boxes using the complete image in a single instance. The main benefit of adopting YOLO is its outstanding speed; it can process 45 frames per second. Additionally, YOLO is aware of generalised object representation. This is one of the best object detection algorithms and has demonstrated performance that is comparable to R-CNN algorithms.

YOLO first takes an input image:

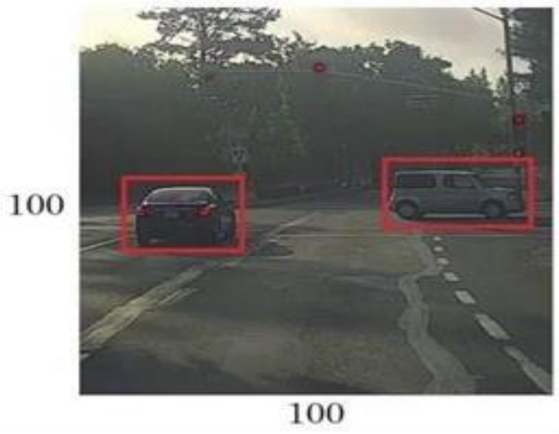


Fig: 8: Example of YOLO Model input image

The framework then divides the input image into grids (say a 3 X 3 grid):

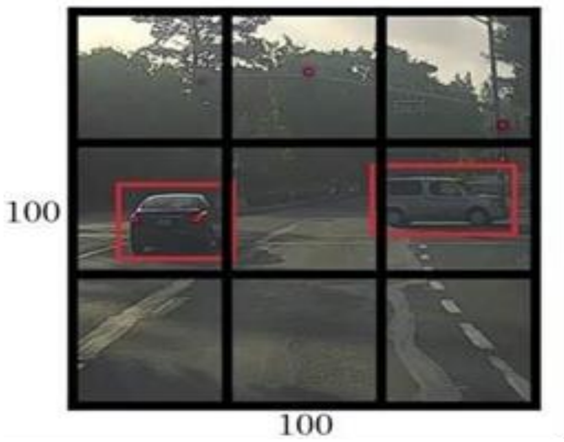
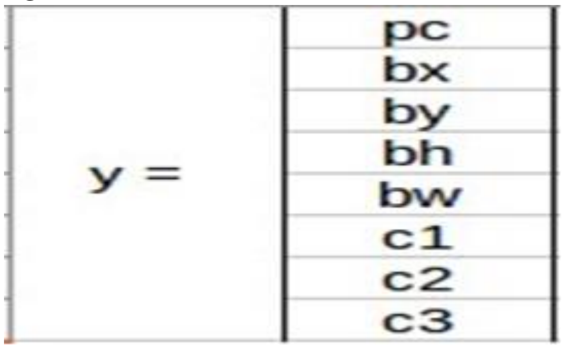


Fig: 9: YOLO model input image example

On each grid, image categorization and localization are applied. Then, YOLO forecasts the bounding boxes for objects and their accompanying class probabilities (if any are found, of course).

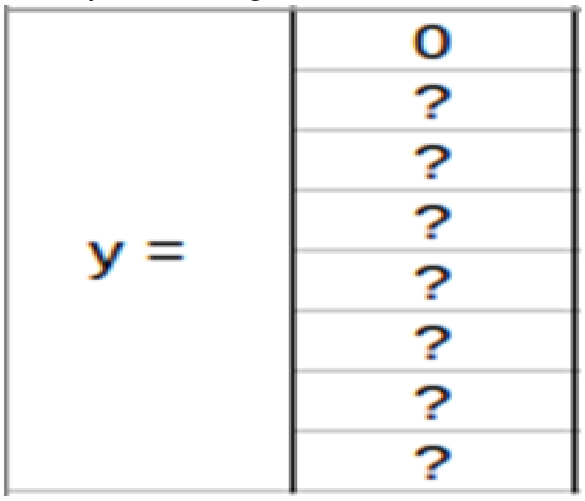
To train the model, we must transmit the labelled data to it. Consider dividing the image into a 3 x 3 grid with a total of 3 classes into which we want the objects to be classified. Let's imagine the classes are, respectively, Pedestrian, Car, and Motorcycle. As a result, the label y for each grid cell will be an eight-dimensional vector:



	pc
	bx
	by
y =	bh
	bw
	c1
	c2
	c3

Here, the computer determines whether or not an object is present in the grid (it is the probability) If there is an object, c1, c2, and c3 represent the classes while bx, by, bh, and bw specify the bounding box. Accordingly, c2 will be 1 if the object is a car, c1 & c3 will be 0, and so on. Let's choose the first grid from the example above:

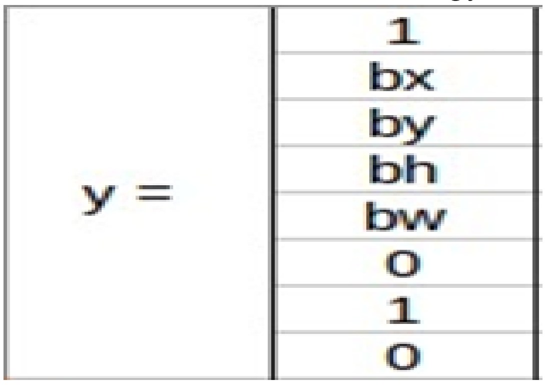
Given that this grid is empty, pc will be equal to zero, and the y label for this grid will read:



	0
	?
	?
y =	?
	?
	?
	?
	?

As there is no item in the grid, "?" in this case denotes that it is irrelevant what bx, by, bh, bw, c1, c2, and c3 contain. Take another grid where c2 = 1 represents a car:

It's crucial to comprehend how YOLO determines whether there is actually an object in the grid before we write the y label for this grid. As there are two objects (two automobiles) in the image above, YOLO will take the midpoint of the two and allocate the two things to the grid that contains the midpoint of the two objects. The center-left grid with the automobile will have the following y label:



$$y = \begin{bmatrix} 1 \\ bx \\ by \\ bh \\ bw \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Pc will be equal to 1 because there is an object in this grid, and bx, by, bh, and bw will be calculated in relation to the specific grid cell we are working with. Car being the second class, c1 and c3 are equal to 0, and c2 is equal to 1. We will therefore obtain an eight-dimensional output vector for each of the nine grids. The final product will be 3 X 3 X 8. As a result, we now have a goal vector and an input image. Our model will be trained as follows using the example above (input image - 100 X 100 X 3, output - 3 X 3 X 8):

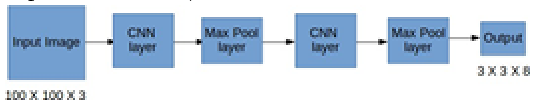


Fig. 10: YOLO model

To train our model, we will use both forward and backward propagation. We provide the model a picture during the testing phase, and we then execute forward propagation until we receive an output y. I've presented this using a 3 X 3 grid here to keep things easy, but in real-world circumstances, we typically use larger grids (perhaps 19 X 19).

Even if an object spans across multiple grids, it will only ever be given the grid where its mid-point is. By using more grids, we can lessen the chances that two objects will show in the same grid cell (19 X 19, for example).

ANCHOR BOXES

As we have seen, each grid is only capable of recognising one object. But what if a grid contains more than one object? In practise, that can happen quite frequently. This explores the notion of anchor boxes. Think of the following picture as a 3 X 3 grid:

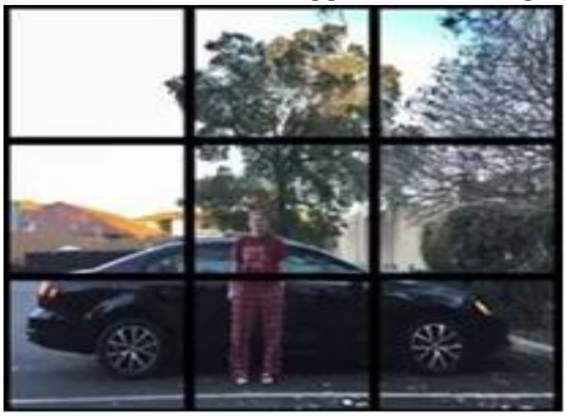


Fig. 10: Anchor boxes in YOLO model

Remember how we assigned an object to a grid? We took the midpoint of the object and based on its location, assigned the object to the corresponding grid. In the above example, the midpoint of both the objects lies in the same grid. This is how the actual bounding boxes for the objects will be:

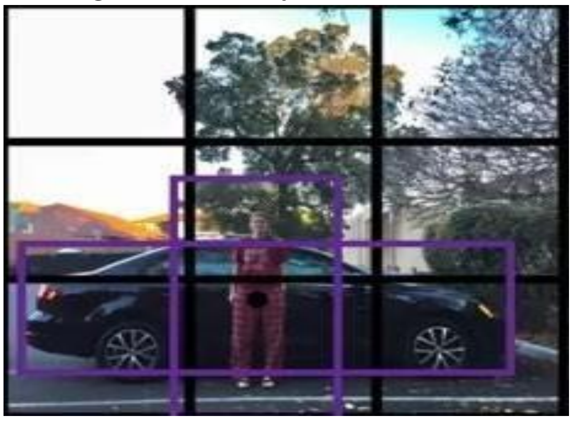
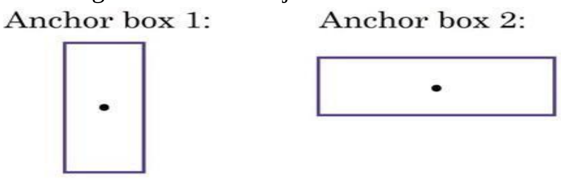
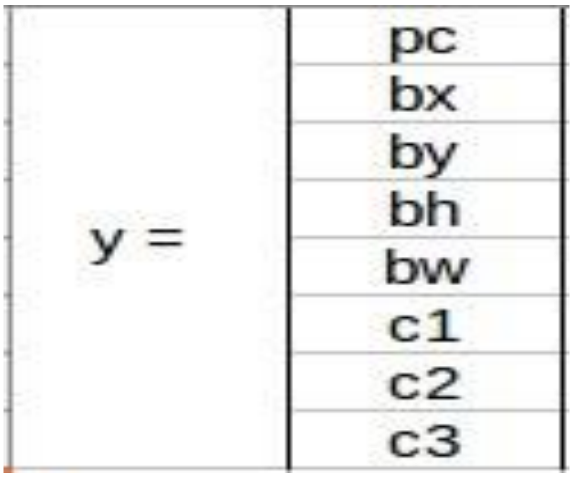


Fig. 11: Anchor boxes in YOLO model

Recollect how we allocated a grid to an object? We measured the object's midpoint and, using its position, placed it to the appropriate grid. The midpoint of both items in the aforementioned example is located in the same grid. The actual bounding boxes for the objects will look like this:

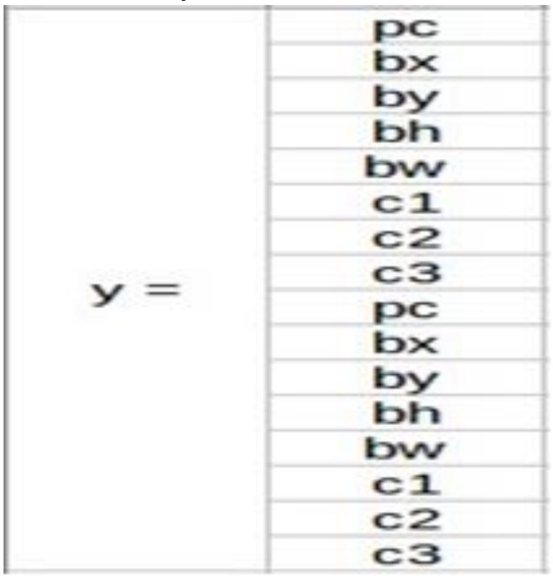


This is how the y label for YOLO without anchor boxes looks like:



y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

Given that there are two anchor boxes, what do you suppose the y label will be? Before you continue reading, I want you to pause and think about this. Got it? The label for y will read:



y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3
	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

The first 8 rows belong to anchor box 1 and the remaining 8 belongs to anchor box 2. Based on how closely the bounding boxes and the shape of the anchor box resemble one another, the items are assigned to the anchor boxes. The person will be assigned to anchor box 1, and the car will be assigned to anchor box 2, as the shape of the former is comparable to that of the bounding box for a person. In this instance, the output will be 3 X 3 X 16 rather than 3 X 3 X 8 (using a 3 X 3 grid and 3 classes) (since we are using 2 anchors).
In light of the quantity of anchors, we can thus detect two or more items for each grid. Let's put all the concepts we've discussed thus far together and incorporate them into the YOLO framework.

TRAINING
The input for training our model will obviously be images and their corresponding y labels. Given below is an image and its and we will create its y label:

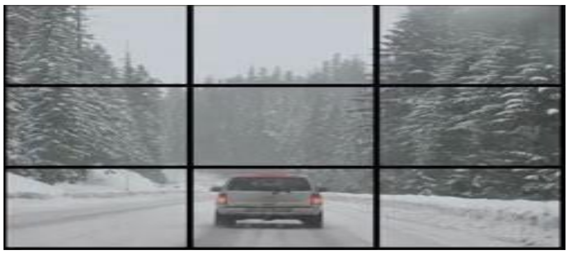


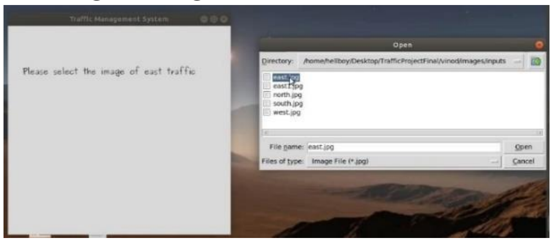
Fig. 11: Test image
Imagine that there are three different object classes and we are utilising a 3 X 3 grid with two anchors per grid. As a result, the matching y labels will be 3 X 3 X 16. Now imagine that there are now five classes and we have five anchor boxes per grid. The objective will therefore be 3 X 3 X 10 X 5 = 3 X 3X 50.
This is how the training procedure is carried out: a 3 X 3 X 16 target is used to map an image of a specific shape (this may change as per the grid size, number of anchor boxes and the number of classes).

TESTING
The same amount of grids that we selected throughout the training session will be divided into the new image. The model will forecast an output with the dimensions 3 X 3 X 16 for each grid (assuming this is the shape of the target during training time). This prediction's 16 values will have the same formatting as the training label's 16 values. The likelihood of an object appearing in that grid will be the first value in the first eight values, which correspond to anchor box 1. The bounding box coordinates for that object are represented by values 2 through 5, and the final three values indicate which class the object belongs to. The following 8 values—for anchor box 2—will follow the same style, with the probability coming first, followed by the bounding box coordinates, and then the classes.
To obtain a single prediction for each object, the Non-Max Suppression approach will next be applied to the predicted boxes.
With the model trained and prediction boxes for the objects generated, we have completed the theoretical part of understanding how the YOLO method operates. The YOLO algorithm's precise dimensions and steps are listed below:
accepts a shape image as input (608, 608, 3)
Sends this picture to a convolutional neural network (CNN), which produces a result with the dimensions (19, 19, 5, 85).
In order to obtain an output volume of (19, 19, 425), the final two dimensions of the output shown above are flattened:
• In this case, a grid of 19 x 19 cells yields 425 numbers.

• 425 = 5 * 85, where 5 is the number of anchor boxes in a grid and 85 is calculated by multiplying 5 by the number of classes we want to identify (pc, bx, by, bh, and bw).
Finally, in order to prevent selecting overlapping boxes, we perform the IoU and Non-Max Suppression.

4. EXPERIMENTS AND RESULTS

First of all, we have to select images for all the for directions from the images collected as dataset for testing of the model. We named the lanes of roads as east, west, north & south. We proceed on selecting the pictures as can be seen in the images shown below:
Selecting the images for the west direction



Please select the image of east traffic

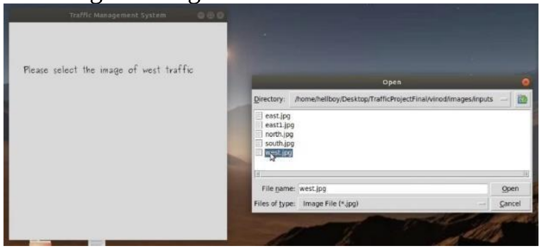
Directory: /home/hellboy/Desktop/TrafficProjectFinal/vinod/images/inputs

east.jpg
east1.jpg
north.jpg
south.jpg
west.jpg

File name: east.jpg
Files of type: Image File (*.jpg)

Fig. 12: Selecting image for east side of the traffic junction.

Selecting the images for the west direction



Please select the image of west traffic

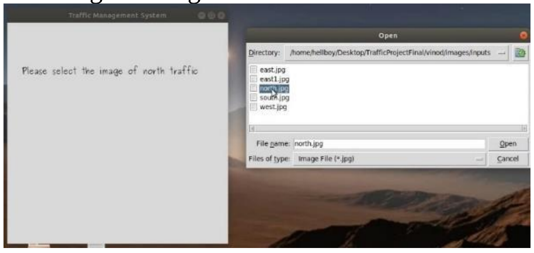
Directory: /home/hellboy/Desktop/TrafficProjectFinal/vinod/images/inputs

east.jpg
east1.jpg
north.jpg
south.jpg
west.jpg

File name: west.jpg
Files of type: Image File (*.jpg)

Fig. 13: Selecting image for west side of the traffic junction

Selecting the images for the north direction



Please select the image of north traffic

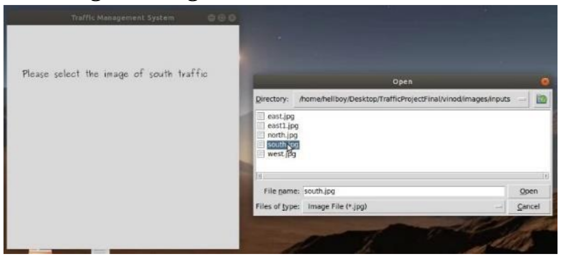
Directory: /home/hellboy/Desktop/TrafficProjectFinal/vinod/images/inputs

east.jpg
east1.jpg
north.jpg
south.jpg
west.jpg

File name: north.jpg
Files of type: Image File (*.jpg)

Fig. 14: Selecting image for north side of the traffic junction

Selecting the images for the south direction



Please select the image of south traffic

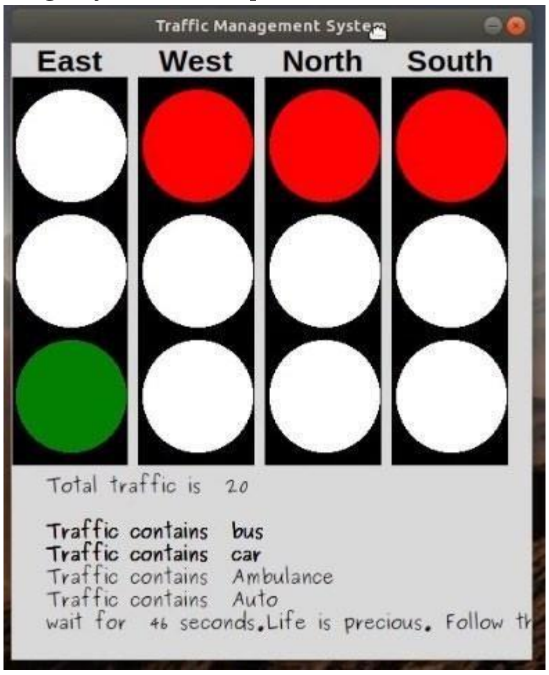
Directory: /home/hellboy/Desktop/TrafficProjectFinal/vinod/images/inputs

east.jpg
east1.jpg
north.jpg
south.jpg
west.jpg

File name: south.jpg
Files of type: Image File (*.jpg)

Fig. 15: Selecting image for south side of the traffic junction

After selecting images from all the directions, we will be promoted to close the image selection window and proceed for the results:
First of all, it checks for the presence of ambulance in any of the image and gives priority to that direction. In the above test case, there is an ambulance present in the east side image, so first of all the east direction will be given priority so that the emergency vehicles can pass.



Total traffic is 20

Traffic contains bus
Traffic contains car
Traffic contains Ambulance
Traffic contains Auto
wait for 46 seconds.Life is precious. Follow th

Fig. 16: East direction given the green signal first

As show in the above image, it can be noticed that the number of vehicles in the east direction in 20. So, the signal time is calculated for 20 vehicles. The calculation is done based on types of vehicles. For two wheelers, one second is added, for cars two seconds are added and for heavy vehicles 3 seconds are added to the time of signal for green light
As shown in the image, the total time calculated for the east direction is 46 seconds, i.e., east side is given a green signal for 46 seconds.
The image of the east direction can be seen in the below image for reference where the vehicles are bounded inside a bounded box by the model.

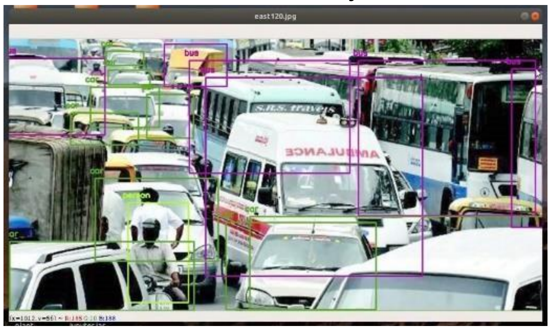
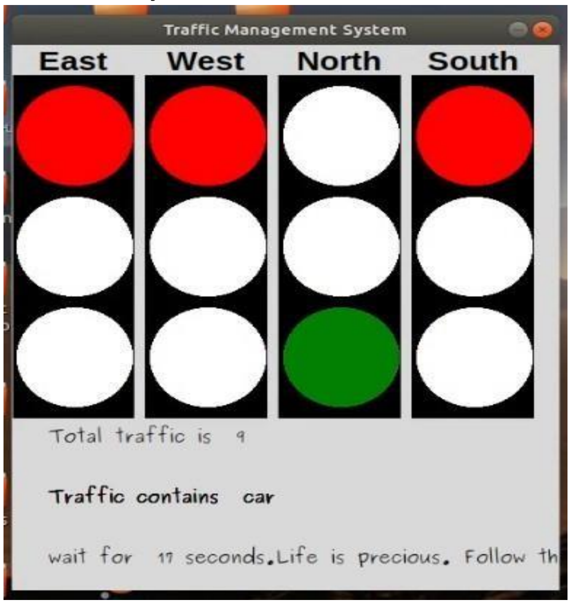


Fig. 17: The image feed into the system for east side of the traffic junction

After the east direction having an ambulance is dealt with, it's time to deal with other sides which don't have any emergency vehicle.

These directions are processed in cyclic order and the time allotted to each direction is dependent on the vehicle density as was int the above case.



Traffic Management System

East West North South

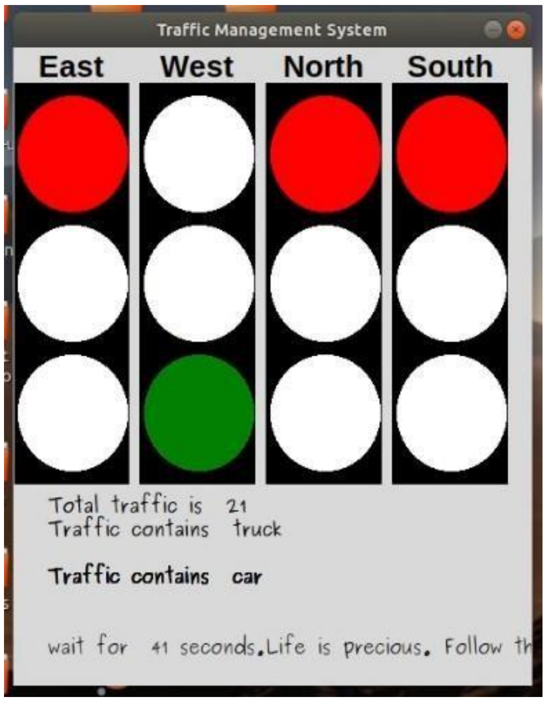
Total traffic is 9

Traffic contains car

wait for 17 seconds.Life is precious. Follow th

Fig. 18: North direction comes second in the cyclic order

As seen in the north directions result, there are a total of 9 vehicles and the north direction is allotted a time of 18 seconds. After the north direction, the west comes in cyclic order so the west signals is given a green signal. There are a total of 21 vehicles in at west side image, so the time allotted to west direction is 41 seconds.



Traffic Management System

East West North South

Total traffic is 21
Traffic contains truck

Traffic contains car

wait for 41 seconds.Life is precious. Follow th

Fig. 19: West direction's result.

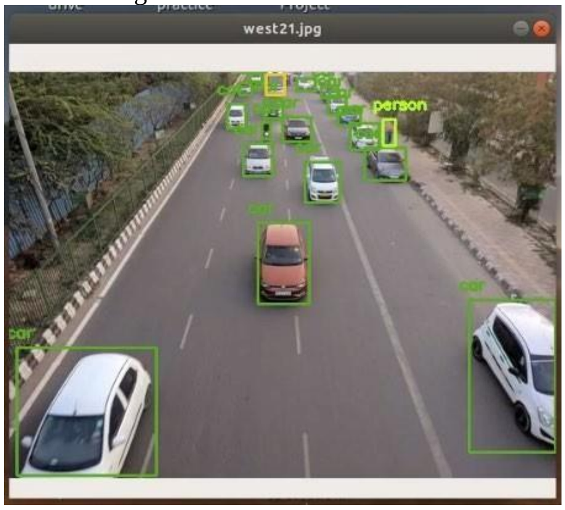
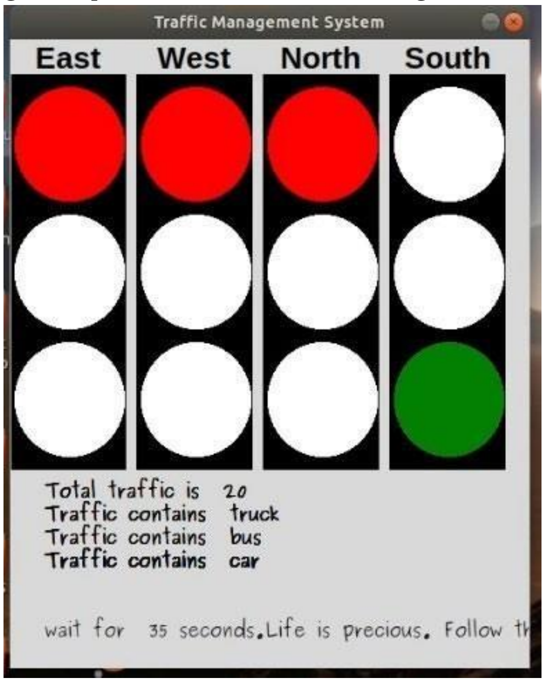


Fig. 20: The image feed into the system for west side of the traffic junction

Final turn comes for the vehicles at south direction, there are a total of 20 vehicles in this direction and the time allotted to this direction is 35 seconds. The signal output can be observed in the image below.



Traffic Management System

East West North South

Total traffic is 20
Traffic contains truck
Traffic contains bus
Traffic contains car

wait for 35 seconds.Life is precious. Follow th

Fig. 21: The signal for the south direction.

A traffic signal which has a static time of say 60 seconds for each direction will take a total of 240 seconds for the above vehicles to pass through the junction.

The above dynamic approach reduced the time to a total of 140 seconds, thus saving a total of 100 seconds of time and a lot of fuel that is burned in that duration.

% improvement = 100 / 240 * 100 = 41.66 %

Thus, the above approach if implemented successfully on real traffic lights would save a lot of time and effort of day-to-day commuters.

5. CONCLUSION

It is clear from the above approach that dynamic traffic management is a very good solution for the problem of traffic congestion that is very widely faced by the people commuting on roads on their vehicles, especially in urban cities.

The static traffic lights that are installed currently on the traffic junctions have a fixed time hard coded into their system which isn't able to manage the flow of traffic properly. Most of the time the traffic flow is high only on few of the lanes and other roads are empty, but the static traffic light still provides a fixed time to that road which lead to the congestion on those roads which already have a high flow of traffic, but are waiting for the signal to be green, and no traffic is flowing from the other end. This kind of situation also leads to the accidents that happen at such traffic lights due to some people trying to jump the signal while no traffic in flowing from other ends which sometime leads to major accidents which also leads to deaths. So, an improvement in the current system is highly needed.

If the static traffic signal system is replaced by some sort of dynamic traffic signal system which takes into account the density of the vehicles so that time can be saved of the people commuting and the fuel resources can also be saved. If it would improve the congestion even by only a few percent, even then resources worth lakhs will be saved. Thus, such kind of traffic signal system is highly desirable.

It will also save the efforts a traffic police personnel have to give to manage the traffic when a heavy traffic congestion takes place.

The system can be improved even further if all traffic signals in a city are connected and synchronized to each other so that a smooth flow of traffic signal can be managed. The above dynamic approach reduced the time to a total of 140 seconds, thus saving a total of 100 seconds of time and a lot of fuel that is burned in that duration.

PROS AND CONS

PROS:

1. Installation Cost of our system: Comparatively speaking, it is quite little because we only need the live stream, which is readily available from the security cameras placed at every traffic light. Most of major cities and the traffic junction where traffic remains high have cameras already installed.
2. Use of Data analytics: Use of technology in Traffic management is a known thing. However, it is the use of data from different sources in real-time and processing information to take immediate decisions that is the key to a successful traffic management in our cities. It is the need of the hour to leverage enormous amount of data around us and create a more meaningful and smooth living for us. Thus, a significant portion of the expense of the gear is eliminated; at most, some cameras would need to be moved or changed.
3. Saving of Fuel: Our model reduces the wastage of fuel while sticking in traffic jams by reducing the amount of traffic congestion and jams. Our model smartly tackles traffic and thus saves fuel which is need of the hour.
4. Unlike other traffic monitoring systems that use pressure mats, which frequently experience wear and tear because of their placement on roads where they are constantly subjected to intense pressure, our system's maintenance costs are practically non-existent. This is because our system does not use any additional hardware components.
5. Emergency Vehicle Passage: One great feature is that the emergency vehicles can be easily provided with a passage by providing the lane with such emergency vehicle priority over the other lanes and thus making it easy for the emergency vehicles to pass.

CONS:

1. After Sunset or Low Light Circumstances: In this situation, reduced light conditions prevent the system from performing as expected; in that case, we could switch to a hard-coded system during the night. Otherwise, we could set up night vision cameras to maintain the dynamic system's functionality as it does during the day.
2. No interconnectivity among different junction: The current system is for only junction; thus, it has to be installed on each junction separately and different junctions don't communicate with each other which if implemented would further improve the overall function and performance of the system.

ACKNOWLEDGEMENT

This is our chance to thank everyone who helped make this project possible. Without their assistance, it would not have been possible.

We truly appreciate Prof. Gauri Rao Ma'am, our mentor, for her advice, criticism, and support that helped us finish the assignment. We gained a solid understanding of the subject and developed our ideas during the frequent brainstorming sessions we had with her. We are grateful to her for providing us with this chance and for her unwavering support.

We would like to offer our sincere gratitude to our college, BHARTI VIDYAPEETH DEEMED TO BE UNIVERSITY, COLLEGE OF ENGINEERING, PUNE, for providing us with this chance for research and development as well as a top-notch library, all of which allowed us to work together and finish our project.

Finally, we would want to express our gratitude to our parents for always being there for us. Their confidence in our ability encouraged us to provide our best effort.

REFERENCE

[1] Chandigarh has highest number of per capita vehicles in Indiahttps://www.motoroids.com/news/chandigarh- highest-number-per-capita-vehicles- india/

[2] Congestion costs incurred on Indian Roads: A case study for New Delhi, Neema Davis, Harry Raymond Joseph, Gaurav Raina, Krishna Jagannathan Department of Electrical Engineering, Indian Institute of Technology Madras.http://www.ee.iitm.ac.in/~krishnaj/Publications_fil es/papers/Draft_journal_c ost.pdf

[3] Python – The Fastest Growing Programming Language K. R. Srinath Associate Professor, Department of Computer Science, Pragati Mahavidyalaya Degree and PG college, Hanuman Tekdi, Koti, Hyderabad, Telangana, Indiahttps://www.irjet.net/archives/V4/i12/IRJET- V4I1266.pdf

[4] OpenCV, an open-source computer vision and machine learning software library https://opencv.org/about/

[5] TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems (Preliminary White Paper, November 9, 2015) http://download.tensorflow.org/paper/whitepaper2015.pdf

[6] A Practical Guide to Object Detection using the Popular YOLO Framework https://www. analyticsvidhya.com/blog/2018/12/practical-guide-object-detection- yolo-framewor- python/

[7] You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon University of Washington, Santosh Divvala, Allen Institute for Artificial Intelligence, Ross Girshick, Facebook AI Research, Ali Farhadi, University of Washingtonhttps://pjreddie.com/media/files/papers/yolo.pdf

[8] Unlocking cities, research on traffic analysis in India by Boston Consulting Grouphttp://image-src.bcg.com/Images/BCG-Unlocking- Cities-Ridesharing-India_tcm9- 185213.pdf

[9] Khekare, G.S.; Sakhare, A.V., "A smart city framework for intelligent traffic system using VANET," Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on, vol., no., pp.302,305, 22-23 March 2013

[10] Badura, S.; Lieskovsky, A., "Intelligent Traffic System: Cooperation of MANET and Image Processing," Integrated Intelligent Computing (ICIIC), 2010 First International Conference on, vol., no., pp.119,123, 5-7 Aug. 2010.