Major Project Report

on

Intelligent Intersection Management: Minimizing Traffic Delays through Adaptive Timing

Submitted by

20BCS011 - Amit Appanna Talawar

20BCS014 - Anant Terkar

20BCS061 - Udit Jain

20BCS106 - Rahul Pentamsetty

Under the guidance of

Dr. Prabhu Prasad B M

Asst. Prof., Dept. of Computer Science and Engineering

and

Dr. Manjunath K V

Asst. Prof., Dept. of Data Science and Intelligent Systems

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

19/03/2024

Certificate

This is to certify that the project, entitled **Intelligent Intersection Management: Minimizing Traffic Delays through Adaptive Timing**, is a bonafide record of the Major Project coursework presented by the students whose names are given below during 2023-24 in partial fulfilment of the requirements of the degree of Bachelor of Technology in Computer Science and Engineering.

Roll No	Names of Students
20BCS011	AMIT APPANNA TALAWAR
20BCS014	ANANT TERKAR
20BCS061	UDIT JAIN
20BCS106	RAHUL PENTAMSETTY

<Supervisor name here>
(Project Supervisor)

Contents

List of Figures

List of Tables

1 Introduction

In the bustling streets of India, navigating through traffic has become a daily battle for millions of commuters. The country's rapid urbanization and burgeoning population have exacerbated long-standing challenges in managing traffic flow, resulting in a myriad of issues that plague its roadways. Among these issues, one of the most pressing concerns is the significant wait times experienced by motorists at intersections across major cities.

Each passing minute spent idling in traffic not only contributes to frustration but also leads to a wastage of precious hours for commuters, impacting productivity and quality of life. The lack of efficacy in managing traffic flow has emerged as a critical issue, leading to congestion, delays, and a host of associated socio-economic challenges.

In this project, we will be addressing these issues with an innovative solution that leverages technology and data driven approaches to optimize traffic flow, minimize wait times and enhance overall efficiency in transportation networks, primarily traffic junctions. We will be working upon a dynamic environment limited to junctions in which we will simulate traffic scenarios and rigorously test our adaptive model results and compare them against static traffic light timings. We are also working upon *Vehicle Detection and Counting* to further employ our model in *real-time environments.*

2 Related Work

In the beginning phases of the requirement and feasibility analysis of the project, it was revealed that there were little to no *on-site* cameras installed at the traffic lights in India. As a result there were no publicly available video dataset to perform the task. So we decided to employ our model to work in simulated scenarios. The following section will be a brief discussion about various related research we referred to; working in both simulated and real environments.

Dnyaneshwar Birajdar et al., 2021 [1] employed a specific methodology to gather real-time traffic data from intersections to facilitate the generation of dynamic traffic signal timings. To achieve accurate vehicle detection and classification, they utilized the YOLOv3-tiny algorithm implemented through Convolutional Neural Networks (CNNs). This enabled the system to precisely detect vehicles in images and generate counts. Subsequently, video footage from installed cameras was used to detect vehicles and provide real-time vehicle counts. These counts

were then fed into a scheduling algorithm to optimize traffic signal timings, considering vehicle counts for different lanes of an intersection.

The algorithm proposed by P. Gupta et al., 2015 [2] focuses on optimizing traffic light behavior by monitoring incoming traffic from all four directions at a junction and calculating traffic density on each edge. Unlike traditional methods, it considers the area between two junctions to communicate with neighboring traffic lights. By assessing traffic flow on opposing edges simultaneously, the algorithm dynamically adjusts traffic light timings to prioritize *heavily congested sides* during peak traffic hours. Additionally, it provides priority to emergency vehicles, such as ambulances and fire brigades, even on edges with lower traffic density.

Kavita Pandey et al., 2018 [3] presents a Traffic Signal Control algorithm with specific assumptions and notations. It assumes vehicles move in groups of two, each taking the same time to cross the intersection, and that all vehicles are of ideal size and shape. The algorithm analyzes n videos of respective lanes at a road intersection for traffic intensity over a time period t seconds. If n is greater than zero, it turns the lane with the longest clearance time green, calculated based on the number of cars in the lane and the time taken by a group to cross the intersection. This process repeats for *n-1* videos, adjusting the time period for each lane based on the duration of the previous green signal.

A lot of papers working their solution in real-time environments suffered with a very fundamental issue i.e. the amount of time their *model/algorithm* took to come up with the solution. Some of the solutions proposed in [5] [6] [7] [8] were lacking in a algorithmic approach to the problem that led to difficulties to adjust the timings at the traffic lights. L. Zhuo et al., 2017 [6] was dependent on a large dataset demanding heavy system requirements and failing to comply with them would just make the machine learning algorithm too slow to detect vehicles in real time.

Identifying and understanding the scenarios where the industry research was facing trouble, we decided to define multiple *heuristic values* in our approach that gives us the freedom to efficiently simulate various traffic scenarios and work upon them. We also employ an algorithmic approach to the problem with dynamic adaptations to the traffic *inflow and outflow rates*. We are simultaneously working on ways to employ the model in real-time environments with Vehicle Detection and Counting using frameworks such as *YOLO*. The next section will discuss the model in detail.

3 Data and Methods

3.1 Simulation of Junction

To simulate a simple four-cross junction we used the pygame library module of Python along with some supporting modules. Each road has one outgoing lane and one incoming lane. There are four traffic signals to manage the traffic at the junction. Each traffic signal has red, yellow, and green colored signals indicating stop, prepare to stop and go respectively. In a static traffic controller, the signal times are pre-decided and never change. For our simulation purpose, we assumed red, yellow, and green light times to be 10, 4, and 10 seconds respectively. The types of vehicles present in the simulation are cars, buses, trucks, and bikes with average speeds being 2.25, 1.8, 1.8, and 2.5 respectively. The "TrafficSignal" class has red, yellow, and green attributes. Attributes of the "Vehicle" class are lane number, vehicle type, direction number, and direction (for appropriate vehicle image file path). Random traffic for each lane is generated using the random module.

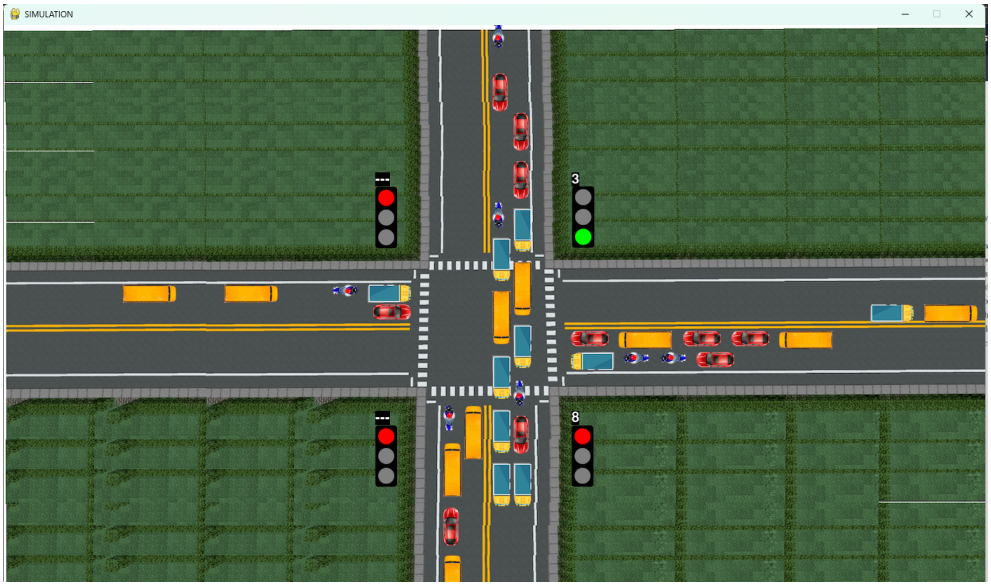


Figure 1. Simulation Snap

3.2 Adaptive Algorithm

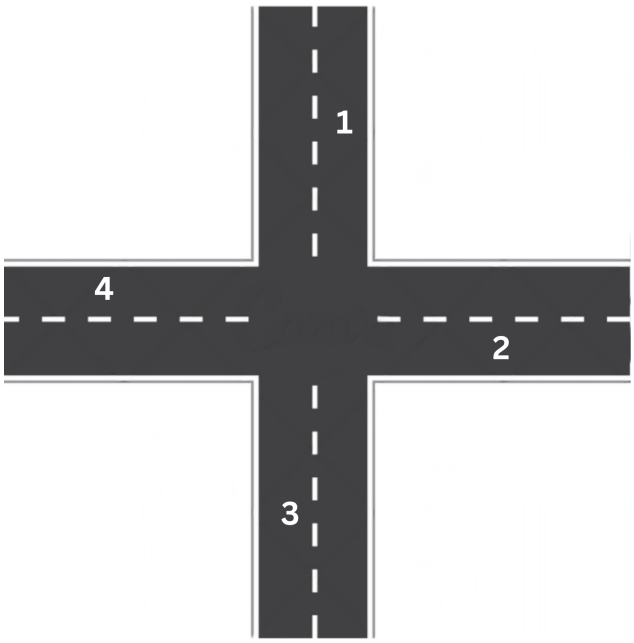


Figure 2. Junction

Each iteration within the algorithm is defined as a single instance of *One Green Light* allocated to a specific lane. To enhance comprehension of the algorithm, let's establish certain mathematical terms.

1. V_i^j : Number of vehicles in i^{th} lane in j^{th} session.
2. D_i^j : Traffic density in i^{th} lane in j^{th} session.
3. α : Queue Time Constant.
4. β : Outflow Constant.

3.2.1 Selecting a lane

To select a lane for the green light allocation at the start of an iteration; we use the V_i^j to find the lane with *maximum lane congestion*. $Next = argmax_i(< V_i^j >)$, Implies in our case as $Next = argmax_i(V_1^j, V_2^j, V_3^j, V_4^j)$.

3.2.2 Green Time

Using the $Next$ variable, *Green time* for that lane is calculated as $GreenTime = \alpha * V^j_{next}$. $GreenTime$ is further used to update V^j_{next}.

3.2.3 Updating Lane Traffic

In each iteration, traffic is simultaneously added to all lanes and removed from the $Next$ lane, pertaining to the variable *Traffic Density* D^j_i and the *Outflow Constant* β respectively.

$$V_i^{j+1} = V_i^j * (1 + \alpha * D_i^j - \alpha * \beta) \text{ if i = next.} \tag{1}$$

$$V_i^{j+1} = V_i^j * (1 + \alpha * D_i^j) \text{ if i} \neq \text{next.} \tag{2}$$

The congestion value at the $Next$ lane for the subsequent iteration is adjusted by incorporating the scale of Traffic Density, denoting incoming traffic, and deducting the scale of Outflow Constant, representing outgoing traffic.

The congestion value at every other lane is adjusted by just incorporating the scale of Traffic Density, denoting incoming traffic.

3.2.4 Picking Control Variables

To maintain a consistent flow of traffic and prevent either excessive buildup or depletion, it is imperative to strike a balance between traffic density D^j_i and the outflow constant β.

In every lane apart from $Next$ lane, $V^{new} > V^{old}$; to balance out the traffic we want $V^{new} < V^{old}$ for the $Next$ lane.

Considering 1 where the traffic is updated for $Next$ lane.

$$V^{new}/V^{old} = 1 + \alpha * (D_i^j - \beta) \tag{3}$$

. Implies

$$1 + \alpha * (D_i^j - \beta) < 1 \tag{4}$$

$$D_i^j < \beta \tag{5}$$

Using these set of equations we finally come to a constraint for our free variables D_i^j and β i.e. 5.

3.2.5 Calculation of Waiting Times

The calculation simplifies the process by considering waiting time on a per-lane basis rather than per-vehicle. We consider adaptive waiting time to be zero when i=next, i.e ith lane is green, henceforth ignoring trivial in-between waiting times.

1. Adaptive :

$$Dy_i^j = \alpha * V_{Next}^j, fori \neq next \tag{6}$$

$$Dy_i^j = 0, otherwise \tag{7}$$

2. Static :

$$S_i^j = \delta, fori \neq next \tag{8}$$

$$S_i^j = 0, otherwise \tag{9}$$

where δ is *Fixed Green Time* in static traffic management model.

1. Average Adaptive Wait Time :

$$\overline{Dy_i} = \frac{\sum_{j=1}^{N} Dy_i^j}{\sum_{j=1}^{N} V_i^j} \tag{10}$$

2. Average Static Wait Time :

$$\overline{S_i} = \frac{\sum_{j=1}^{N} S_i^j}{\sum_{j=1}^{N} V_i^j} \tag{11}$$

3.2.6 Control Variables and Initialization

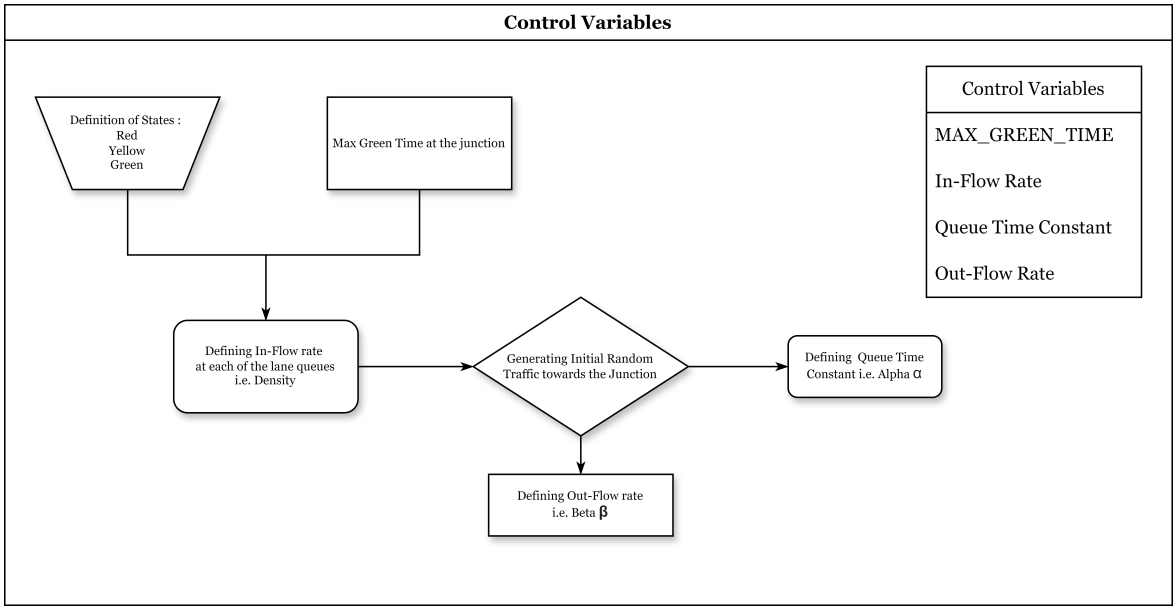


Figure 3. Control Variables

When configuring the model for its initial state, we commence by defining the three states for the traffic lights. Additionally, we set an upper limit for the $GreenTime$ parameter to ensure that our system can effectively manage situations of severe congestion on particular roads without excessively altering the $GreenTime$ value. Furthermore, we establish the Traffic Density, also known as Inflow Rates, for each lane converging towards the junction. These Density values undergo adjustments at every iteration. The first iteration of Traffic converging towards the junction is random. We further define the suitable Outflow Rate β pertaining the constraint 5. Lastly we define α, which is used in combination with Traffic Density rates to exactly determine the number of vehicles converging at the junction at each lane.

3.2.7 Work Flow Chart

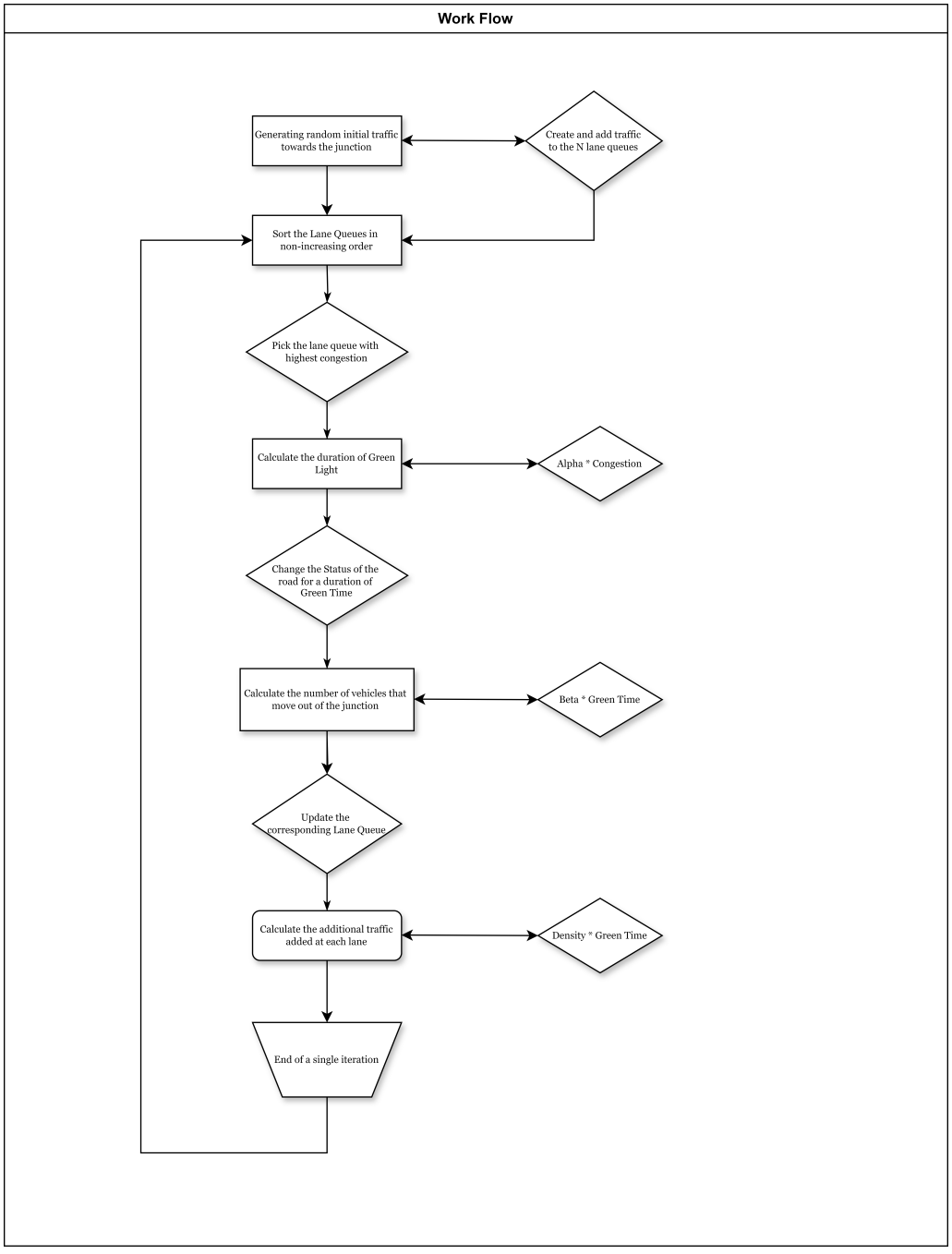


Figure 4. Adaptive Workflow

Let us examine the Workflow chart to gain an abstract understanding of the model. At the start of every iteration, the model begins with sorting the *Lane Queues* in a *Non increasing order* fashion. It then picks the lane with the highest congestion for changing the state of that lane to *Green*. Calculation of $GreenTime$ is performed using the equation $GreenTime = \alpha * V^j_{next}$. The model changes the state of lane to *Green* for $GreenTime$ duration. Now the model updates the traffic at each lane; meaning reducing the traffic at *Green* state lane and adding traffic to other lanes. The Outflow Constant β is used to calculate the number of vehicles leaving the junction. The Traffic Density Control variable is used to calculate additional traffic at each of the lanes. This marks the end of one iteration.

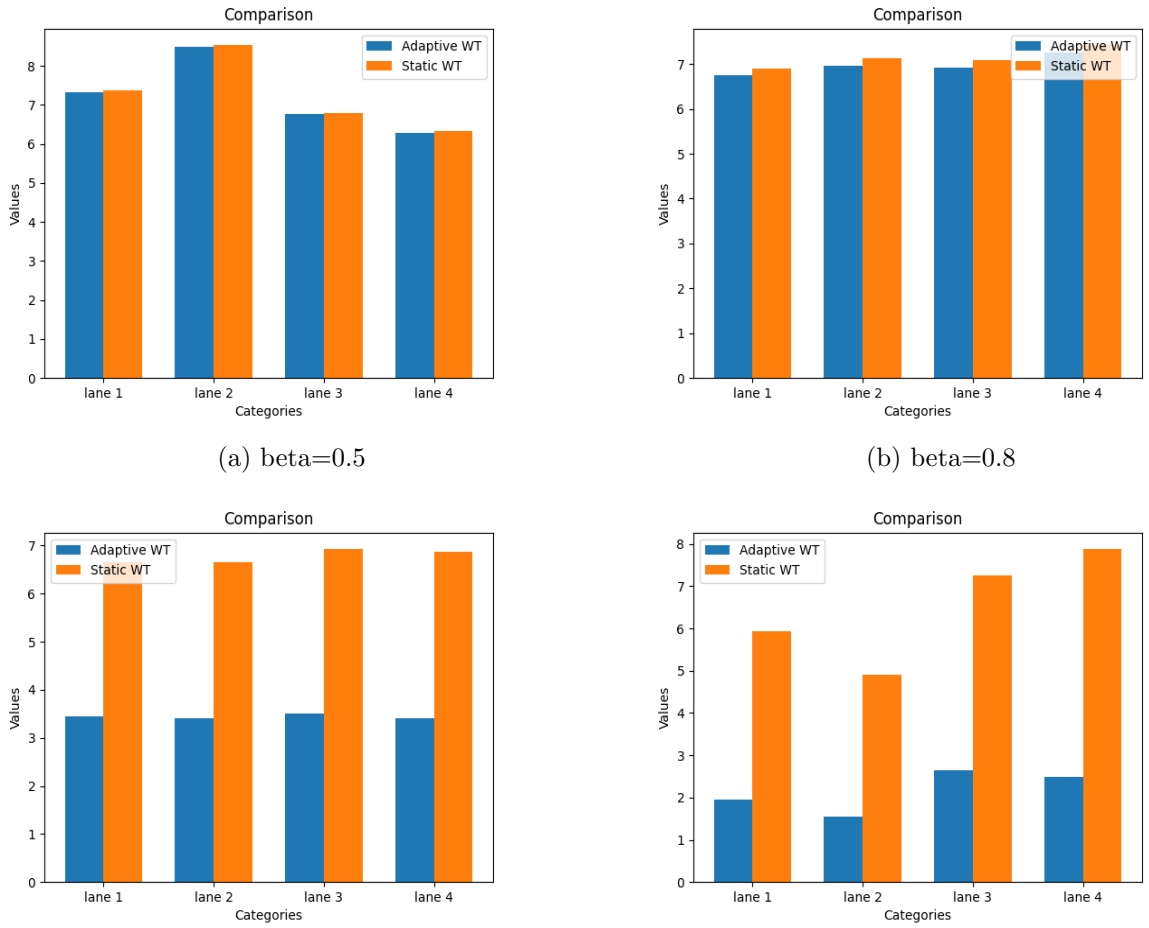
3.2.8 Vehicle Detection, Classification and Counting

Once we know the number of cars and apply the dynamic algorithm, we can reduce the average waiting time. To count the number of cars in situations with a large number of cars and low image quality, we needed to choose a model which works best in these scenarios. After experimenting with different models, we decided to proceed with the YOLO model.
On training Yolo V8 [9] with various labeled datasets, we found that most datasets fail with multi-car detection and lose confidence when there are a large amount of cars in the image. Since we needed an approximate number of cars in the image and a few mis-classified vehicles do not change the values for a major number of our variables which are a part of our dynamic algorithm, we decided to move on with the currently available datasets.
Keeping the lack of GPU resources in mind, we decided to train the model for 20 epochs for about 5000 images. On trial of multiple datasets from different sources, with varying epochs and different number of images with varying number of classes we found 2 datasets that the model Dataset 1 - [10], Dataset 2 - [11]

4 Results and Discussions

4.1 Performance Analysis of Adaptive Algorithm

We look at how our novel adaptive traffic light algorithm explained in section 3.2 performs in comparison to a static traffic light. The right metric for such comparision is average waiting time of vehicles. As it is quite tedious to keep track of waiting times of each and every vehicle in all the lanes, we compare a slightly different waiting time, lane-wise average waiting time.



(a) beta=0.5

(b) beta=0.8

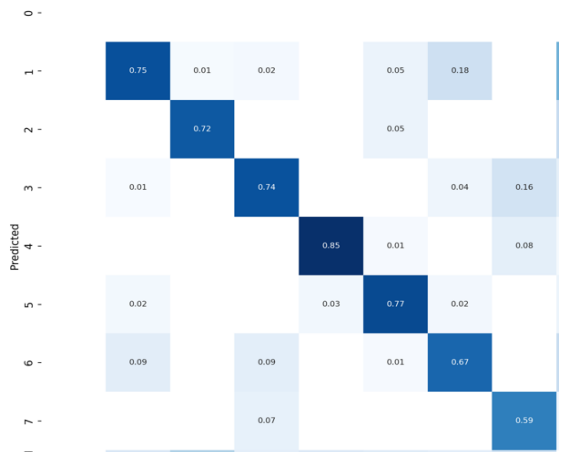
(c) beta=1.0

(d) beta=1.15

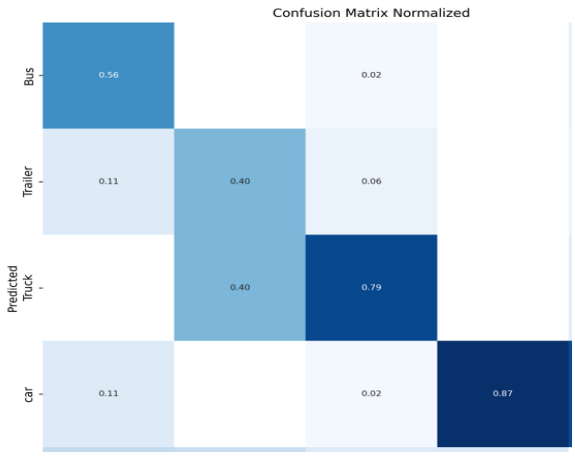
Figure 5. Adaptive vs Static Average Lane-wise Waiting Times

4.2 Yolo Model Training Results

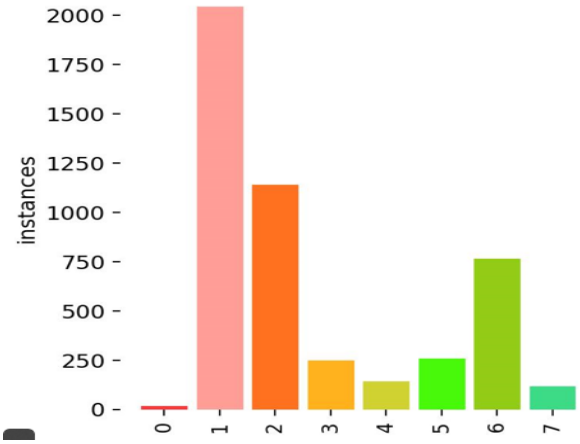
As mentioned above we finetuned Yolo [9] on a labeled dataset. The results for which are pasted below. On using 2 different datasets with different labels, we looked at the confusion matrix.We use Normalized Confusion matrix because there is a very high class imbalance of cars in the datasets used, as shown in figure 6c



(a) Dataset 1 - Classes 1-7



(b) Dataset 2 - Named Classes



(c) Dataset 1 - Class Instances

On looking at figure 6a we see that the diagonal contains high values, which shows that the model correctly identifies most classes correctly. In between certain classes there is a higher disparity, as there are classes named as Truck, Trailer, Big Truck which naturally had some

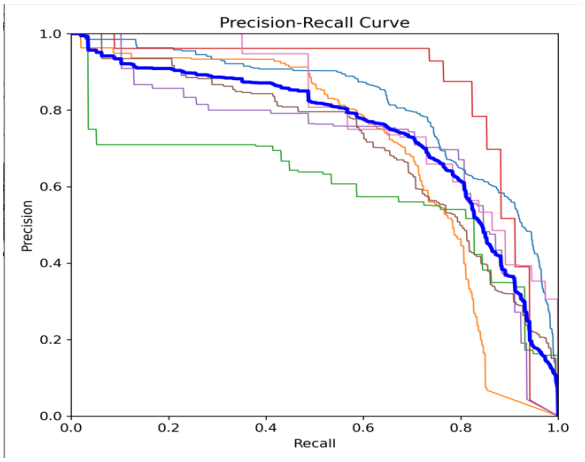
human labeling errors and those classes have few differences. On training with a different dataset, with fewer number of classes, we see that there is still a high disparity with Truck and Trailer but the model correctly identifies Cars which has a major number of instances as shown in Figure 6b.

(a) Labeled

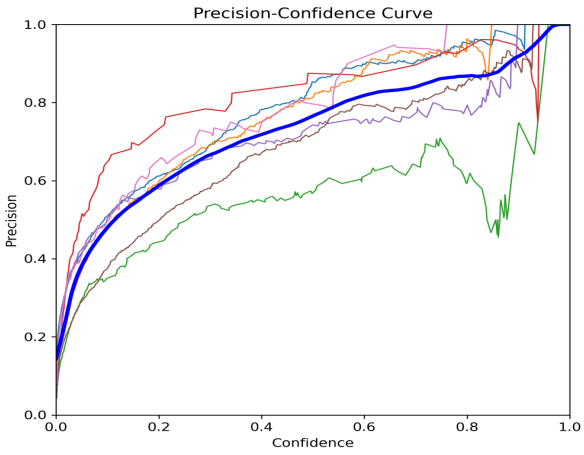
(b) Predictions

Looking at the above images, we can clearly see all vehicles have been clearly identified with the bounding boxes, which shows that the model can count the vehicles and predict which vehicles are there. However the confidence with the classification is low which is due to the image quality, and the distance from the camera for each of these cars.

Looking at Figure 8a we can see that the trained model works. The multiple lines in the figure represent the different classes. The light blue line in both figures which represents cars, which has the majority of instances in our dataset, shows that the model fares well. The lines that are below the bold blue line are classes that have a few instances and the model fared badly on those few instances.



(a) Precision Recall Curve



(b) Precision Confidence Curve

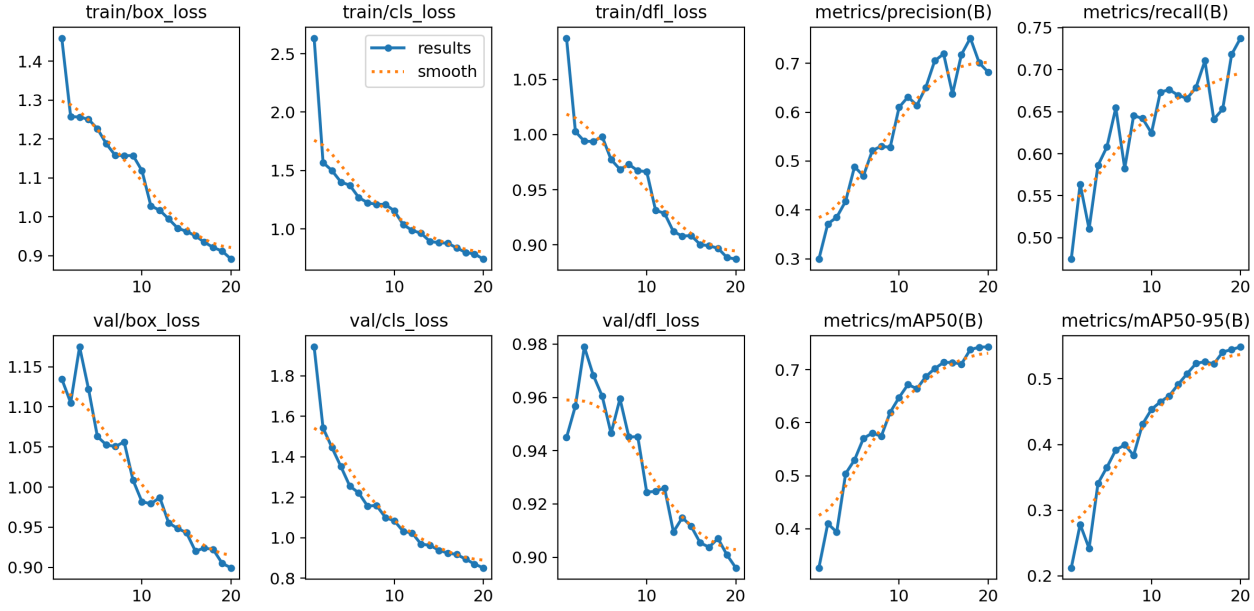


Figure 9. Training Graphs

Graphs in Figure 9 show the training losses while training the model for 20 epochs. We can see **cls_loss** which is the cross entropy loss for the classification model, that decreases exponentially as we train the model further. This loss is used to evaluate the model's ability to correctly classify vehicles into their respective classes. **dfl_loss** is used to make the model

learn more discriminitive features, and we see that the model is improving because the loss is steadily decreasing. The **mAP50** metric evaluates object detection performance by calculating the Average Precision (AP) for each class at an Intersection over Union (IoU) threshold of 0.5. This threshold requires a 50% overlap between predicted and ground truth bounding boxes for a positive detection.It is obtained by averaging the AP values across all classes, focusing on reasonably accurate bounding box predictions. By the end of training we reach a value of 0.6 to 0.7 which are reasonably good for 20 epochs. The only two values that shows room for improvement are **precision** and **recall**. Since these values are extremely volatile, it shows the instability in model training. All other metrics show that if the model is trained further, there is only room for improvement.

5 Conclusion

This study has presented an adaptive approach to intelligent intersection management that leverages dynamic traffic signal timing to minimize delays and congestion. Through a mathematical model and simulation framework, the proposed algorithm has demonstrated its efficacy in optimizing traffic flow and reducing average wait times compared to traditional fixed-time traffic light systems.
The key findings include:

1. The adaptive algorithm dynamically allocates green light durations based on real-time traffic conditions, enabling the system to prioritize congested areas and respond to changing patterns.

2. Simulation results indicate significant reductions in average lane-wise waiting times across various traffic density scenarios under the adaptive approach.

3. The incorporation of vehicle detection and counting capabilities, through fine-tuned YOLO models, enhances the real-world applicability of the system by providing accurate data inputs.

These findings highlight the immense potential of data-driven, mathematically-grounded strategies to address the pressing challenges of urban congestion. By dynamically optimizing traffic signal timings, the proposed system can improve transportation efficiency, reduce emissions, and enhance the quality of life for citizens.
Future research directions may involve further refinement of the adaptive algorithm, integration with other intelligent transportation technologies, and field testing to assess the system's performance and scalability in practical scenarios.
In conclusion, this study has demonstrated the significant potential of adaptive traffic management approaches to develop smarter, more efficient, and sustainable transportation systems.

References

[1] D. Birajdar, A. Bane, S. Darade and S. Chaudhari, "Real Time Traffic Control System Using Dynamic Scheduling," 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 2021, pp. 1-4, doi: 10.1109/CONIT51480.2021.9498316.

[2] P. Gupta, L. P. Singh, A. Khandelwal and K. Pandey, "Reduction of congestion and signal waiting time," 2015 Eighth International Conference on Contemporary Computing (IC3), Noida, India, 2015, pp. 308-313, doi: 10.1109/IC3.2015.7346698.

[3] K. Pandey and P. Jalan, "An Approach for Optimizing the Average Waiting Time for Vehicles at the Traffic Intersection," 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan, India, 2018, pp. 30-35, doi: 10.1109/PDGC.2018.8745757.

[5] C.M. Bautista, C.A. Dy, M.I. Mañalac, R.A. Orbe, and M. Cordel, "Convolutional neural network for vehicle detection in low resolution traffic videos", In Region 10 Symposium (TENSYMP), pp. 277-281, 2016.

[6] L. Zhuo, L. Jiang, Z. Zhu, J. Li, J. Zhang, and H. Long, "Vehicle classification for large-scale traffic surveillance videos using Convolutional Neural Networks". Machine Vision and Applications, Vol. 28, Issue 7, pp.793-802, 2017.

[7] Y. Tang, C. Zhang,R. Gu, P. Li, and B. Yang, "Vehicle detection and recognition for intelligent traffic surveillance system". Multimedia tools and applications, Vol. 76, Issue 4, pp.5817-5832, 2017

[8] M. Bommes, A. Fazekas, T. Volkenhoff, and M. Oeser, "Video Based Intelligent Transportation Systems–State of the Art and Future Development". Transportation Research Procedia, Vol. 14, pp.4495- 4504, 2016

[9] Jocher, G., Chaurasia, A., Qiu, J. (2023). Ultralytics YOLO (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

[10] https://universe.roboflow.com/baworntot/car_yolov7

[11] https://universe.roboflow.com/ilham-winar/venom