International Journal of Integrated Science and Technology (IJIST)
Vol.2, No.8, 2024: 767-794

AI-Enabled Traffic Light Control System: An Efficient Model to Manage the Traffic at Intersections using Computer Vision

Majid Ayoubi[1], Hasibullah Aman[1], Rohullah Akbari[1], Hedayatullah Lodin[1]

Faculty of Computer Science, Kabul University, Afghanistan

Coressponding Author : Majid Ayoubi : ayoubi.m@ku.edu.af

A R T I C L E I N F O

Keywords: Traffic Light, Object Detection, Computer Vision.

Received : 6, June
Revised : 10, July
Accepted: 25, August

A B S T R A C T

Traffic congestion is a significant issue with studies indicating it costs cities billions annually and averages 54 hours of wasted time per traveler each year. This situation necessitates the implementation of efficient traffic management systems, especially at intersections. In response to this challenge, our work introduces an artificial intelligence-based system designed to analyze and predict traffic flow using machine learning algorithms and deep learning methods in conjunction with traffic cameras. The model comprises two main components: real-time data collection and predictive modeling. It employs object detection to identify and classify vehicles and adjusts traffic signal timings based on the necessary passage time and predetermined constraints. Additionally, data accumulated during operation facilitates the development of a predictive model for traffic flow over time, allowing for proactive traffic management. Evaluations are done to showcase the accuracy of the model and corresponding simulation and physical implementation further approved the applicability of our approach. Finally, this work aims to enhance urban transportation efficiently, reduce commuting stress, and improve the quality of life for city residents.

INTRODUCTION

Traffic congestion is a persistent issue faced by cities worldwide, regardless of their size or level of development. Effective traffic management is critical in addressing this challenge, especially at intersections where congestion and safety concerns frequently arise due to inadequate traffic signal timing [1]. This causes a long queue of vehicles at junctions, generating noise through continuous revving and frequent braking, delaying the driver, consuming more fuel, and increasing air pollution.

These problems have to be resolved through various means by analyzing the patterns in traffic through machine learning algorithms and deep learning methods. It aims to optimize the control of traffic signals at junctions by coming out of fixed or manual control methods that normally cause unwanted, longer waiting periods for drivers. This not only leads to frustration, but also creates increases fuel consumption and accident risks due to violations of traffic signals [2].

On the other hand, Computer Vision [3] plays a crucial role in finding an effective way to make the traffic signal system dynamic. Compared to the technique that uses expensive sensors and complex infrastructure, this method can be better and more economical from the pricing point of view.

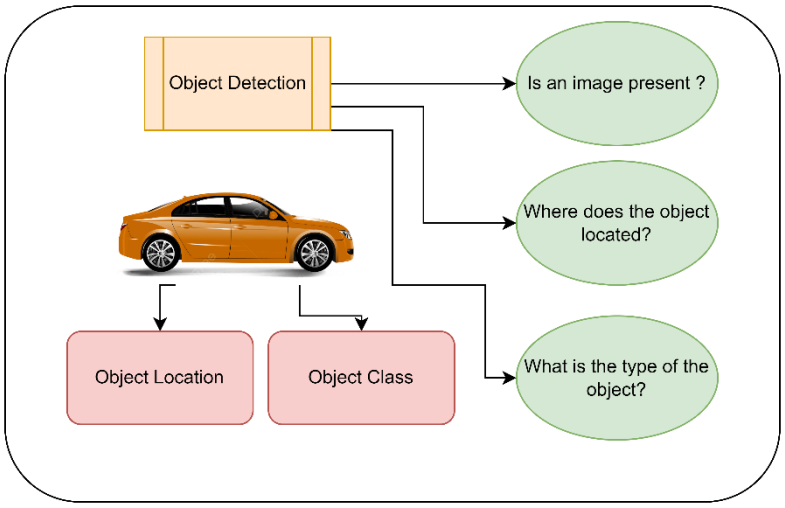


Figure 1: Object Detection Tasks

Our intelligent traffic light control system analyzes real-time data to make an accurate assessment of the traffic volumes and implements dynamic signal predictions and timing adjustments. Dynamic adjustment of signal timing will improve safety for all modes of users and thereby foster sustainable transportation by reducing idle time at intersections and lowering air pollution.

Our work consists of two primary components. The first incorporates real-time data collection, achieved by capturing video footage from traffic cameras at a four-way intersection. These videos are processed using object detection techniques that the specifications of its tasks are visualized in Figure 1. Specifically YOLOv8[1] object detection technique is proved to categorize vehicles in the best way and communicate the timing needs to the traffic signal control system.

The second component of our model addresses the challenge of predicting traffic congestion while mitigating camera limitations. We employ computer vision models to count vehicles on specific routes, storing the data for analysis. This information is used to forecast traffic flow over the next time interval, allowing us to adjust signal timing proactively, thereby reducing the likelihood of congestion and enhancing the efficiency of urban traffic management.

Our contribution is as follows:

- We have created a traffic light control dataset and preprocessed it for applying different algorithms on it.
- We have implemented different algorithms for object detection and provided a through comparison between them.
- We designed a traffic signal switching algorithm that combines the historical and real-time data to handle the traffic proactively and effectively.
- A virtual and physical simulations have been carried out to showcase the applicability of our signal switching approach.

The rest of the paper is written as follows. The section 2 reviews the current literature, section 3 discusses the methodology, followed by the simulation and discussion at section 4 and 5 respectively. We conclude the paper in section 6.

Literature Review:

The over-reliance on obsolete traffic management systems that is characterized by fixed-time controls and manual operation of the traffic signals has numerous disadvantages. These deficiencies have made countries and governments seek innovative solutions. Various companies and individuals have contributed to this advancement, identifying efficient methods and implementing them in respective communities. We are going to point out some of these works and the strategies they have used.

Overall, most organizations apply the use of traffic sensors in managing traffic at road intersections. For instance, a company called Sydney Coordinated Adaptive Traffic System, SCATS[2] has implemented a UTC system in Sydney containing 1,000 sets of traffic signals within an area measuring 1,500 square kilometers. This system encompasses 11 field minicomputers operating under the supervision of a central computer. Normally, a system that uses the dynamic management of time, division, and flow is more efficient in reducing delays as compared to time-based signals used traditionally Click or tap here to enter text..

Another method is called SCOOT (Split, Cycle and Offset Optimization Technique) that is used for the dynamic adjustment of traffic signals. This technique is based on the analysis of a detector installed in the vehicles by an online computer that contains programs that pre-schedules the vehicles at the intersections. The data is calculated and implemented to minimize congestions. Research and development of this system has been done by TRRL[3] and in cooperation with three other companies GEN, Ferranti, and Plessey. This system is implemented in Glasgow and Coventry. and traffic checks have been done by TRRL on a total of 62 signals. which had a better result than the traditional traffic control methods and was able to reduce the delay of cars by 12% .

However, the peripheral metal of the vehicle has an opposite effect on the inductance due to eddy currents that are produced. The decrease in inductance tends to decrease the electrical impedance of the wire to alternating current. The decrease in impedance actuates the electronics unit output relay or solid-state optically isolated output, which sends a pulse to the traffic signal controller signifying the passage or presence of a vehicle Click or tap here to enter text..

There are still companies and institutions that have used a better and more effective method than using physical sensors and have obtained better results from Google Map data. Google has created a system called Google Traffic that provides traffic information and routing suggestions to users, which uses the GPS of its users and also Google Map data. By analyzing this data, Google can help the traffic flow. And there are other processes and firms that use this method, such as Traffic Technology Services (TTS), Miovision [6], and etc. Using this method is better than using physical sensors that require huge infrastructure and investment and can be used all over the world. These methods, which use physical sensors or the use of false data and historical data, cannot be effective

[2] https://www.scats.nsw.gov.au/home
[3] https://assets.publishing.service.gov.uk/media/5ccc167c40f0b6332c726a44/LR1122.pdf

for some countries, among others, the following shortcomings can be pointed out as follows.

Existing traffic management methods have a few drawbacks, mainly in developing countries where the infrastructure is usually inadequate or totally absent. The use of traffic sensors that would be very helpful in providing data is thus rendered either impossible due to lack of proper infrastructure or very expensive for economically weak governments struggling to allocate funds to such technologies. Also, companies highly involved in traffic management solutions do not usually focus their efforts in such regions, and therefore the problem is aggravated.

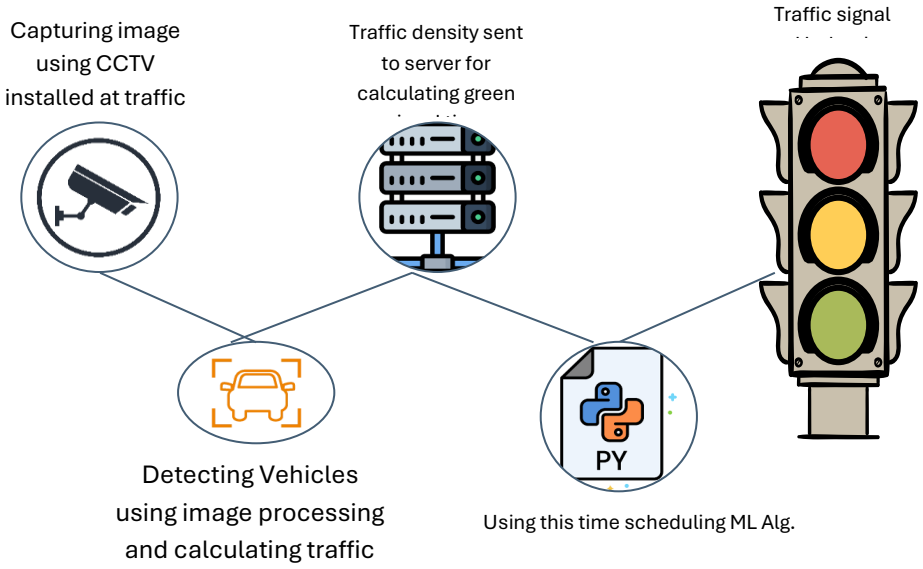
In most developing countries, expensive internet access inhibits the gathering of data from platforms such as Google, which relies on user-generated information. Even though Google Maps might be useful in standard intersections with low pedestrian volume, it becomes ineffective in an environment where the roads are usually crowded with pedestrians. Most of the traffic incidents in these areas are therefore not reported in Google's data [7], while relevant institutions have little to no access to reliable information on traffic. It highlights a disconnection and an important lapse in the unsuitable traffic management strategies of developing nations.[8].

We discuss a method that can be suitable for the places with less access to live or satellite-based traffic information. Our approach tries to manage a crossroads by AI, which can be a low-cost and effective method for the abovementioned places and it could also be useful in places that it needs minimum equipment and low price. Our approach also helps to collect data for another aspect of traffic management, which is the use of historical data, that can significantly improve the accuracy of the decision on traffic light control.

Using artificial intelligence and computer vision for traffic management offers a cost-effective and efficient solution compared to traditional methods. It minimizes infrastructure requirements, reduces congestion and delays, and enhances road safety. This approach is particularly beneficial for developing countries with limited resources and can provide effective traffic control.

METHODOLOGY

Our model consists of two components: First component operates in real-time, while the other component analyses the data which was collected throughout the system's operation. Our model analyses the incoming live traffic through Computer Vision technique and identifies different vehicle, while uses ML to



analyze the previously stored data for decision making (Figure 2). We start by explaining our dataset and the preprocessing, followed by the model selection and how we model the historical data in the process of traffic light control.

A. Preprocessing & Dataset

Dataset:

Basically, to manage the traffic light by the AI effectively, one would have to make a fully comprehensive and well-tuned dataset. It should include such information as patterns of traffic flow, conditions of an intersection, and other kinds of associated contextual information. Some of the key considerations in making or acquiring a dataset to work out all these activities include the following [9].

There are various sources that can provide traffic-related datasets: traffic volume data provided by sensors or GPS, traffic flow patterns, historical data, and data

Figure 2: The Process of the Traffic Light Control System

that are real-time. Considering the problems of previous systems and features of the system that we are to adapt, we prepared a real-time dataset taken by traffic cameras.

i. Collecting Data

We need to prepare a dataset that is a collection of photos of cars on a road or highway. We have used public videos as well as videos taken by ourselves, in total we have 185 minutes video for consideration of detection of vehicles.

ii. Making Frames

We made our data into different frames to detect specific time for traffic observations. We have converted the videos into frames by the convert_video_to_image.py module. In total we have 335 images after this operation.

iii. Labeling

For the labeling stage, we have used the LabelMe [10] program, which provides a tool for labeling, LabelMe is a graphical image annotation tool, it is written in Python and uses Qt for its graphical interface. The output of the LableMe program is a Json file, which in order to enter the preprocessing stage, we converted it to XML format, for which the Json_to_Xml.py module is used.

B. *Preprocessing:*

Before we can proceed to the modeling phase, it is essential to conduct a thorough analysis and preparation of our desired model for object detection within this system. To achieve this, we must accurately complete the preprocessing stage prior to entering the modeling stage. We have done the following stages for preprocessing:

i. Data Cleaning

After labeling stage, we eliminated the data which has fewer information or there was no object, in this process we have removed almost one third of the frames.

ii. Data Integrity:

To ensure the data integrity, we have followed the following steps: We standardized the all-image formats to make it possible to apply different approaches and we also labeled and annotated the data, in which we have converted the JSON to xml format for better processing. Meanwhile, we also made sure that all of our video is in high quality. And finally, we removed all the data which didn't use to provide insights or information for our process.

iii. Image Normalization

At this stage, to standardize the frames used (8-bit images), we have divided it by 255, which is a suitable range for analysis and training. Operations have been performed to normalize between 0 and 1.

iv. Image Augmentation

After the needed dataset is created, in order to make our model work better, we have performed Image Flipping, Random Cropping, Rotation, Scaling and Resizing, Translation/Shifting, Color Jittering/Hue/Saturation Adjustment. These operations have been done considering the appropriate frames.

v. Image Cropping

Image cropping is a common technique used in computer vision and image processing to extract a specific region of interest from an image, although it should be noted that this operation is performed before we use the model.

Model Selection

It is quite difficult to compare these different object detectors because there is no clear answer about which model is superior. In real life, we have to make a trade-off in accuracy versus speed when making our choices. Other factors that affect the performance, in addition to the detector type, are choices of feature extractors (VGG16, ResNet, Inception, MobileNet [11]), output strides for the extractor, input image resolutions, and the matching strategy along with the intersection over union (IoU) threshold that determines how predictions are excluded during loss calculations. Additional considerations include the non-max suppression IoU threshold, hard example mining ratio (the balance between positive and negative anchors), the number of proposals or predictions, boundary box encoding, data augmentation, and the training dataset itself.

That makes it even more difficult because technology in this area is changing so fast that the results get outdated and unreliable in no time. We include a summary of the findings of various techniques to provide clarity as they can be viewed collectively for better comparison [12].

i. **Faster R-CNN**

Faster R-CNN (Convolutional Neural Networks) [13] an object detection model that improves on Fast R-CNN by using a Region Proposal Network (RPN) with the CNN model. The RPN bonds full-image convolutional features with the detection network, allowing nearly cost-free region suggestions.

ii. **R-FCN**

Region-based Fully Convolutional Networks, or R-FCNs [14], are a type of region-based object detector. In contrast to previous region-based object detectors such as Fast/Faster R-CNN that apply a costly per-region subnetwork

hundreds of times, R-FCN is fully convolutional with almost all computation shared on the entire image.

To achieve this, R-FCN utilizes position-sensitive score maps to address a dilemma between translation-invariance in image classification and translation-variance in object detection.

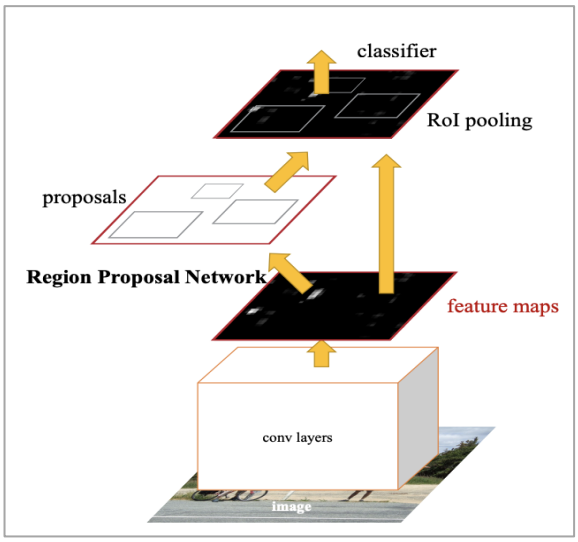


Figure 3 : Faster-RCNN-workflow

iii. YOLO

You Only Look Once (YOLO) using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.

Traditional object recognition algorithms have multiple steps to find objects, but YOLO only has one step to process the whole picture. This makes it faster and more efficient. The YOLO algorithm also uses anchor boxes to improve detection accuracy and handle objects of different sizes and aspect ratios YOLOv8 Click or tap here to enter text.is the latest version of the YOLO algorithm, which outperforms previous versions by introducing various modifications such as spatial attention, feature fusion, and context aggregation modules. These improvements result in faster and more accurate object detection, making YOLOv8 one of the key object detection algorithms in the field.

Modeling on Historical Data

As the second element, to select a specific route and use the previously conventional model to count the number of vehicles on that route, we also take their types into account. Subsequently, we utilize the Pandas library to store the

data in a CSV file. In the following stage, we apply machine learning models to forecast traffic flow for a particular segment. To do this, we initially compare various models and evaluate their performance, after which we choose the model that delivers the highest accuracy for our prediction task.

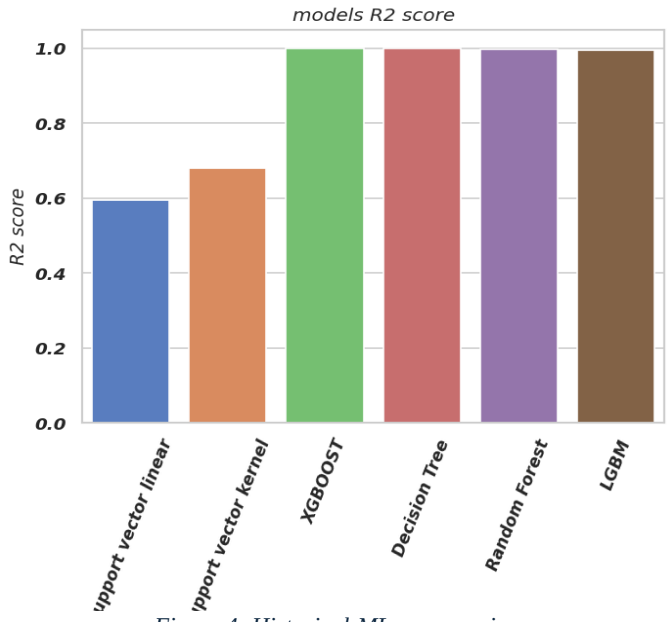


Figure 4: Historical-ML-compression

As illustrated in the Figure 4, the data shows predictable accuracy, which can be attributed to the preprocessing methods applied earlier, and the available models have yielded satisfactory results. Among these, the Decision Tree model has demonstrated the highest accuracy, making it our choice for predicting traffic conditions on a specific road. In the second step, where we combine four routes into a single dataset at one intersection. We then use the Decision Tree model on our dataset to develop a new model that forecasts whether each route is experiencing heavy traffic, normal conditions, or other states.

Some of the features of our dataset is as follows:

Table 1: The Dataset Features their and Descriptions

Feature	Description:
Time	This indicates the specific time of day when the data was recorded, typically in hours and minutes (e.g., HH:MM).

Date	This shows the date on which the data was collected, providing context for the recorded traffic conditions.
Day of the week	This specifies the day of the week (e.g., Monday, Tuesday) during which the traffic data was gathered, which may help in analyzing patterns based on weekdays versus weekends
CarRoad1	This represents the number of cars observed on the first road segment being monitored.
CarRoad2	This represents the number of cars observed on the second road segment being monitored.
BikeRoad1	The number of bicycles counted on the first road segment.
BikeRoad2	The number of bicycles observed on the second road segment.
BusRoad1	The number of buses that passed through the first road segment.
BusRoad2	This captures the count of buses on the second road segment.
TrukRoad1	This denotes the number of trucks observed on the first road segment.
TruckRoad2	Indicates the number of trucks recorded on the second road segment.
TotalRoad1	The total number of vehicles (cars, bikes, buses, and trucks) counted on the first road segment.
TotalRoad2	This captures the total count of vehicles observed on the second road segment, summarizing all vehicle types.

Once we have this dataset, we proceed to the third model in the historical analysis phase, where we predict traffic conditions for the next time period. The output of this model is then utilized in our real-time component, serving as a constant to adjust the duration of the green signal—either increasing or decreasing it based on the predictions.

Results:

A. Model Performance

In order to measure the performance of our model, we have used the following metric measures.

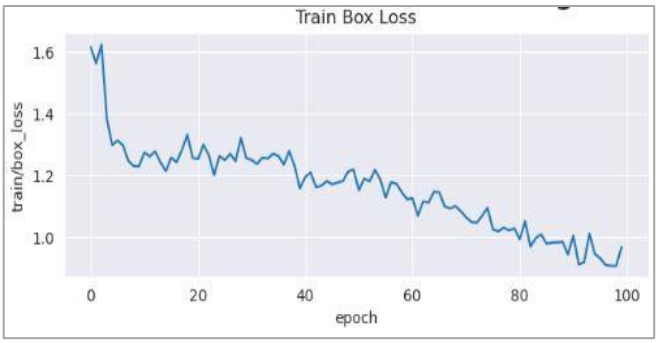


Figure 5 : Train box loss

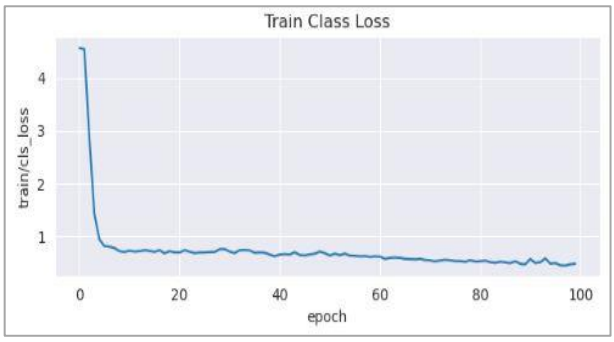


Figure 6 : Train Class Loss

Train Box Loss

The train box loss metric Figure 5 measures the difference between the predicted bounding boxes and the actual bounding boxes of the objects in the training data. A lower box loss indicates that the model's predicted bounding boxes more closely align with the actual bounding boxes.

Train Class Loss

The train class loss Figure 6 metric measures the difference between the predicted class probabilities and the actual class labels of the objects in the training data. A lower-class loss shows that the model's predicted class probabilities more closely align with the actual class labels.

Train DFL Loss

The train DFL (Dynamic Feature Learning), loss metric (Figure 7) measures the difference between the predicted feature maps and the actual feature maps of the objects in the training data. A lower DFL loss means that the model's predicted feature maps more closely align with the actual feature maps.

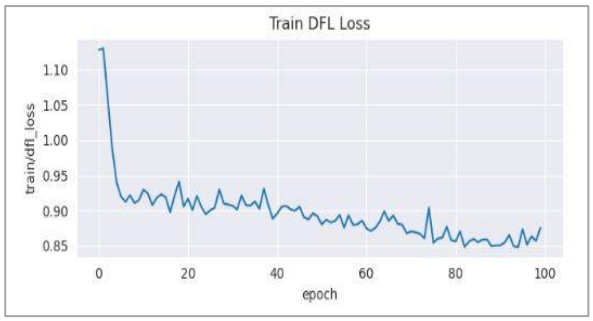


Figure 7 : Train DFL loss

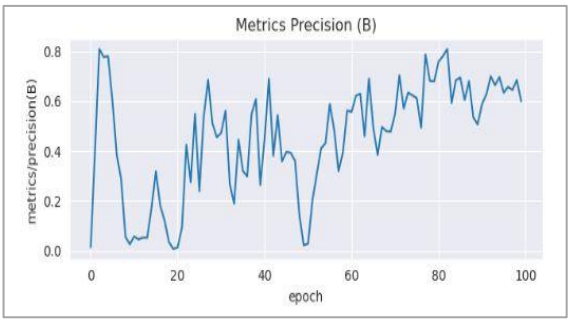


Figure 8 : Metrics Precision (B)

Metrics Precision (B)

The metrics precision (B) metric (Figure 8) measures the proportion of true positive detections among all the predicted bounding boxes. A higher precision means that the model is better at correctly identifying true positive detections and minimizing false positives.

Metrics Recall (B)

The metrics recall (B) metric (Figure 9) measures the proportion of true positive detections among all the actual bounding boxes. A higher recall means that the model is better at correctly identifying all true positive detections and minimizing false negatives.

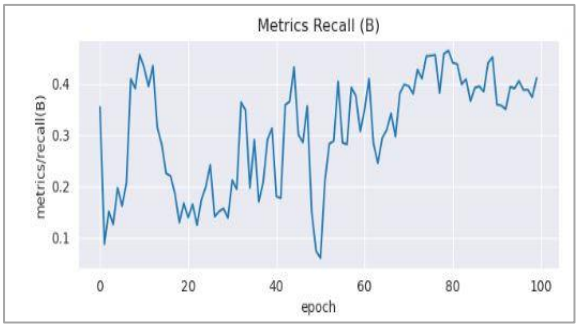


Figure 9 : metrics recall (B)

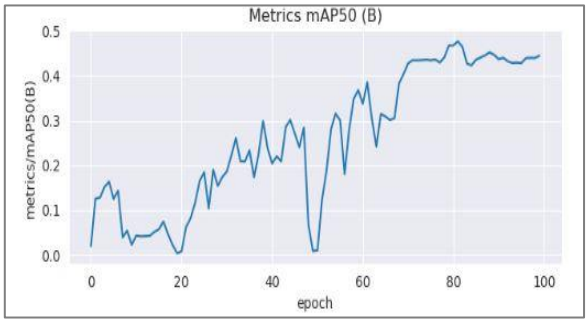


Figure 5 : metrics mAP50 (B)

Metrics mAP50 (B)

The metrics mAP50 (B) metric (Figure 10) measures the mean average precision of the model across different object categories, with a 50% intersection-over-union (IoU) threshold. A higher mAP50 means that the model is better at accurately detecting and localizing objects across different categories.

Model Evaluation

Once the training of our model is complete, we aim to evaluate its performance on samples that have not been observed or encountered previously, utilizing the Dataset defined earlier. To accomplish this, we employ two measurement metrics, which we will outline below.

Mean Average Precision (mAP) Metrics [17]

The mAP is a widely used evaluation metric in object detection tasks, including those involving the YOLO model. It assesses the accuracy of an object detection model by evaluating its ability to identify objects within an image and the precision of those detections. In the context of YOLO, mAP is particularly critical as it quantifies the model's effectiveness in detecting relevant objects. A higher mAP score (Figure 12) indicates the model's stronger capability to accurately identify objects in an image.

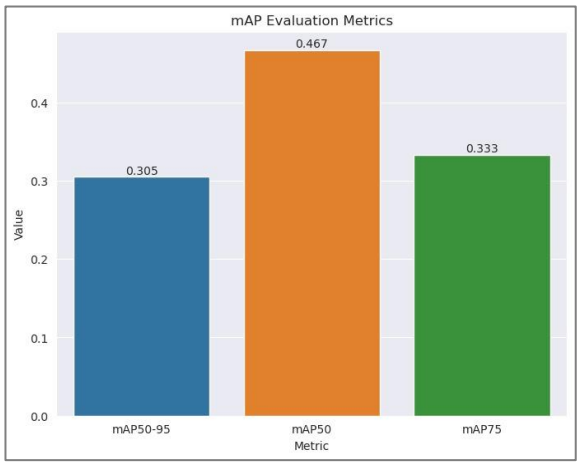


Figure 7 : mAP Evaluation metrics

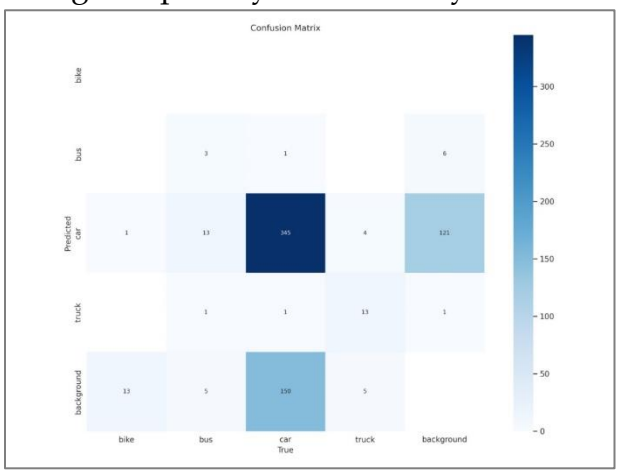


Figure 6 : Confusion Matrix

Confusion Matrix

The confusion matrix is an essential tool for evaluating the performance of object detection algorithms such as YOLO. In the context of object detection, it allows for the calculation of various performance metrics, including precision, recall, and F1 score. For instance, when detecting ships using YOLO in aerial images, the confusion matrix serves as a valuable means of assessing the model's effectiveness. Figure 13 shows the predicted and actual vehicles provided by our model.

The Proposed Signal Switching Algorithm

The Signal Switching Algorithm determines the duration of the green signal based on the traffic density provided by the vehicle detection module and updates the red signal timers for other lights accordingly. It cyclically switches

between signals based on the timers. The algorithm receives information about detected vehicles from the detection module, as detailed in the previous section, in JSON format, where the object label serves as the key, and the confidence score and coordinates are the corresponding values.

This input data is parsed to compute the total number of vehicles for each class. Based on this information, the algorithm calculates and assigns the appropriate green signal time for the active signal while adjusting the red signal times for the other lights as necessary. The algorithm can be adapted to accommodate any number of signals at an intersection.

While developing the algorithm, the following factors were taken into consideration:

1. The processing time required to calculate traffic density and determine the green light duration, which influences when to acquire an image.

2. The total count of vehicles in each class, such as cars, trucks, motorcycles, etc.

3. The traffic density derived from the above factors.

4. The additional time caused by lag experienced by vehicles at start-up and the increasing lag for vehicles positioned further back.

5. The average speed of each vehicle class when the green light engages, which reflects the average time needed for each class to cross the signal.

6. The minimum and maximum limits for the green light duration to avoid vehicle starvation.

Algorithm 1: Traffic Light Control System

Input:

1. Initial time for the first signal of the first cycle (default value).

2. Vehicle detection data

3. Average crossing times for each vehicle class.

4. Number of lanes at the intersection.

5. Time duration for yellow signal (set to 5 seconds).

Output:

1. Optimized green signal time (GST) for the current signal.

2. Timers for the next signals in the cycle.

Initialization:

1. Set the default time for the first signal (default_time).

2. Initialize a timer for the current green signal.

3. Initialize separate threads for vehicle detection in each direction.

4. Set the current signal cycle count (cycle_count = 1).

while true:

> **start** the countdown timer for the current **green** signal.
>
> Continuously **monitor** and detect **vehicles** in each direction.
>
> **if** the current **green** signal timer reaches **zero** or the red-light timer for the forthcoming green signal hits zero:
>
> > **capture** an image of the traffic at the intersection.
> >
> > **parse** the image to determine the number of vehicles in each class.
> >
> > **calculate** the green signal duration (GST) using the formula 1:
>
> **end if**
>
> *Signal Management*
>
> **set** the **green** signal timer for the current signal to the computed GST.
>
> **set** the **red** signal time for the **next** light based on the current settings and the traffic conditions.
>
> **allow** 5 seconds for processing after image capture before transitioning to the **next** signal.
>
> **Signal Switching**

transition to the **next** signal in a predetermined sequence after the green signal duration is complete.

include a **yellow** signal duration (5 seconds) as part of the signal switching process.

repeat the cycle from step 1, **incrementing** the cycle count.

end while

Upon the algorithm's initial execution, a default time is assigned to the first signal of the first cycle, while times for all other signals during this and subsequent cycles are determined by the algorithm. A separate thread manages vehicle detection for each direction, while the main thread oversees the timer for the current signal. When the green light timer of the current signal (or the red-light timer of the forthcoming green signal) reaches zero, the detection threads capture an image of the next direction. This result is parsed, and the timer for the next green signal is established. This entire process occurs in the background while the main thread continues counting down the timer for the current green signal, ensuring smooth transitions and preventing delays.

The image is captured when the timer for the upcoming green signal hits zero. This provides the system with five seconds (equivalent to the yellow signal timer) to process the image, evaluate the number of vehicles in each class, calculate the green signal duration, and set the timers for this signal along with the red signal time for the next light. To determine the optimal green signal time based on the number of vehicles present, the average speeds and acceleration times of vehicles at start-up, which allows for estimating how long each vehicle class requires to cross the intersection are considered. The green signal duration is then calculated using the following formula.

$$GST = \frac{\sum_{vehicleClass}(NoOfVehicles_{vehicleClass} * AverageTime_{vehicleClass})}{(NoOfLanes + 1)}$$

where:

- GST is green signal time
- $noOfVehicles_{vehicleClass}$ is the number of vehicles of each class of vehicle at the signal as detected by the vehicle detection module,
- $averageTime_{vehicleClass}$ is the average time the vehicles of that class take to cross an

intersection, and
- noOfLanes is the number of lanes at the intersection.

The average time required for each vehicle class to cross an intersection can be customized based on various factors, such as region, city, locality, or even specific intersection characteristics. This approach enhances traffic management effectiveness and can be informed by data analyzed from relevant transport authorities.

The signal switching occurs in a cyclic manner rather than prioritizing the densest traffic direction first. This aligns with the existing system, where signals turn green sequentially in a predetermined pattern, minimizing the need for public adjustment and preventing confusion. The order of the signals remains consistent with the current arrangement, and provisions for yellow signals have also been incorporated. Here also we briefly explain in more details how the change in the order of the signals is taking place.

Order of signals

Initially, all traffic signals are initialized with default values, with the exception of the red signal time for the second signal, which is determined based on the green and yellow times of the first signal.

The leftmost column displays the status of each signal (i.e., red, yellow, or green), followed by the traffic signal number and the current timers for red, yellow, and green. For instance, traffic signal 1 (TS 1) transitions from green to yellow. As the yellow timer counts down, the vehicle detection algorithm processes results, yielding a green time of 9 seconds for TS 2. Since this duration is shorter than the minimum green time of 10 seconds, the green time for TS 2 is adjusted to 10 seconds. When the yellow timer for TS 1 reaches zero, TS 1 turns red and TS 2 turns green, at which point the countdown continues. The red signal time for TS 3 is also updated to the total of the yellow and green times of TS 2, totaling 5 + 10 = 15 seconds.

After completing a full cycle, TS 1 once again shifts from green to yellow. As the yellow timer counts down, the vehicle detection algorithm runs again, calculating a green time of 25 seconds for TS 2. Since this value falls between the minimum and maximum green time thresholds, the green signal duration for TS 2 is set to 25 seconds. When the yellow timer for TS 1 reaches zero, TS 1 turns red and TS 2 turns green, continuing the countdown. The red signal time for TS 3 is updated to reflect the sum of the yellow and green times for TS 2, amounting to 5 + 25 = 30 seconds.

Model Evaluations:

To assess our model based on the dataset provided, we evaluated it in two ways. Initially, we have evaluated the model in terms of next 15 minutes, which the (Figure 14) shows the scatter plot and the corresponding line plot of the predicted and actual values. Secondly, we have predicted the corresponding traffic at the cross road and visualized the scatter plot and the line plot (Figure 15) that shows high accuracy of our model and indicates that it can provide more reliable decision making at the traffic light control system.

Once we have validated the performance of our model and completed the evaluation, the next step was to make predictions using samples from our Dataset. The result of the process can be seen in Figure 16.

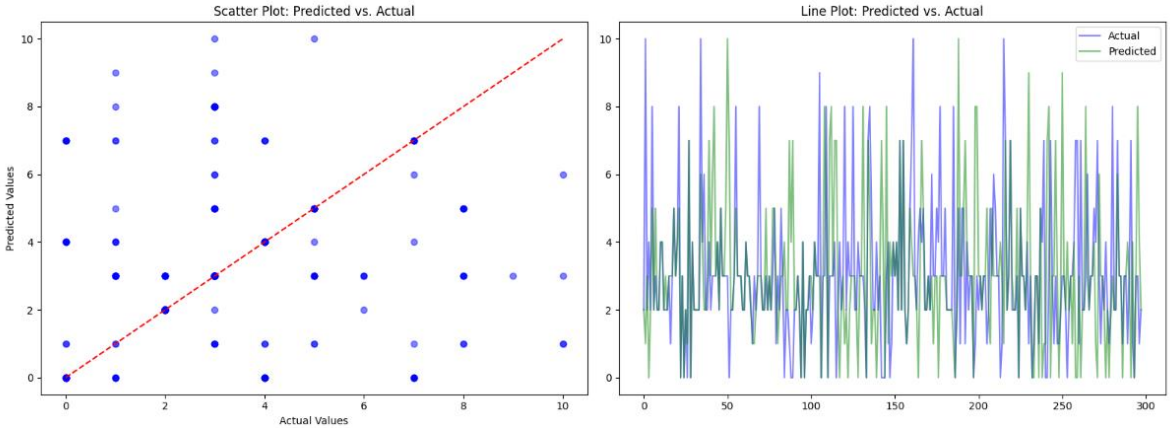


Figure 8 : Evaluation result to predict next 15 min

In this work, we had two main components. First, we concentrated on gathering video data and converting it into frames for use in Deep Learning and Machine Learning models. We labeled these frames using the Labelme tool. The next phase involved the Real-Time section, where we preprocessed the data and formatted it for input into our model. For modeling, we chose YOLO because of its effective features for our specific task and trained our Object Detection model, which we subsequently tested with satisfactory results. In the Real-Time component of our model, which is responsible for vehicle counting and traffic signal management, we completed these stages effectively.

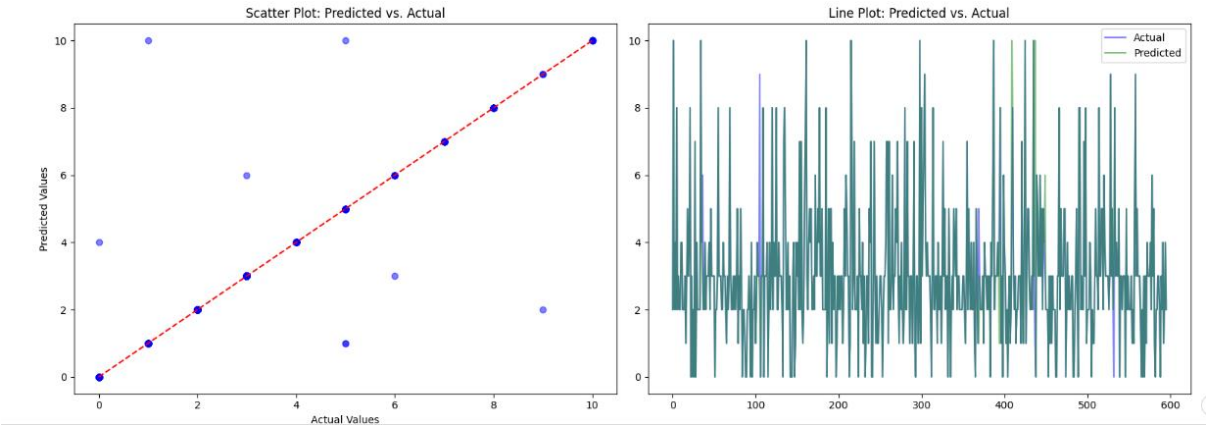


Figure 15: Traffic Predictions for Crossroad

Figure 9 : Model Prediction

In the second part, we utilized the model created in the to handle real-time situations. The steps performed before the prediction are similar to the first part of our system. In the second part, we considered a specific direction and stored the number of vehicles on that route in a CSV file every 15 minutes. Then, we built a model to predict traffic flow for one direction. In the next step, by combining the data from all four directions at an intersection, we constructed a model to predict congestion at the intersection. We used a Decision Tree algorithm to train the model and predict the traffic flow. This way, we could determine which direction at the intersection would have more congestion or in what state it would be. We then added this information as an additional input to the Real-Time section. Overall, these steps are carried out in this part of the system.

Simulations:

We have applied two types of simulations, a software based and simulation and another an Arduino based physical simulation to showcase the applicability of our model.

A simulation was created using Pygmy to replicate real-world traffic scenarios, aiding in the visualization of the traffic system and allowing for comparison with the existing static system. This simulation features a 4-way intersection equipped with two traffic signals. Each signal is accompanied by a timer that indicates the remaining time until the signal changes from green to yellow, yellow to red, or red to green.

Additionally, the number of vehicles that have crossed the intersection is displayed next to each signal. Vehicles, including cars, bikes, buses, trucks, and rickshaws, approach the intersection from all directions. To enhance practicality, some vehicles in the rightmost lane are programmed to turn while crossing the intersection.

The decision for a vehicle to turn is randomly determined at the moment the vehicle is generated. The simulation also includes a timer that tracks the elapsed time since the beginning of the simulation.

To showcase our model, we applied a simulation using pygmy model to run the model for 15 minutes. We have used the following devices for our experiment (Figure 17).

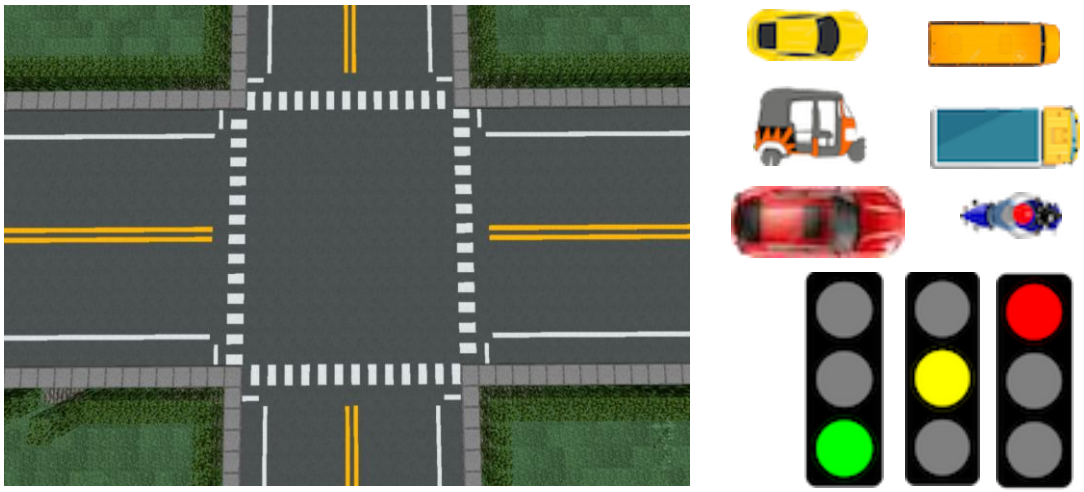


Figure 10: Stimulation Related Icons and Images

Key steps in development of simulation includes the following table.

Table 2: The Attributes of the Simulation

The Attribute	*Description*
speeds	We indicate the average speeds of vehicles
Coordinates of start	we set the coordinates of start for 4 side

Coordinates of stop lines	we set the coordinates for 3 types of stop and also for medium stop as necessary time
TrafficSignal	The green time of the signals is set according to the algorithm, by taking into consideration the number of vehicles at the signal. The red signal times of the other signals are updated accordingly.
Vehicle	Generation of vehicles according to direction, lane, vehicle class, and whether it will turn or not all set by random variables. Distribution of vehicles among the 4 directions can be controlled. A new vehicle is generated and added to the simulation after every 0.75 seconds
Move function	For how the vehicles move, each class of vehicle has different speed, there is a gap between 2 vehicles, if a car is following a bus, then its speed is reduced so that id does not crash into the bus
printStatus Function	For displaying the time elapsed since the start of the simulation. And finally,
updateValues function	For updating the time elapsed as simulation progresses and exiting when the time elapsed equals the desired simulation time, then printing the data that will be used for comparison and analysis.

The process of the simulation is as follows:

Right after start, the simulation showing red and green lights, green signal time counting down from a default of 20 seconds and red time of next signal blank. When the signal is red, we display a blank value till it reaches 10 seconds. The number of vehicles that have crossed can be seen beside the signal, which are all 0 initially. The time elapsed since the start of simulation can be seen on top right.

Simulation showing green time of signal for vehicles moving up set to 10 seconds according to the vehicles in that direction. As we can see, the number of vehicles is quite less here as compared to the other lanes. With the current static system, the green signal time would have been the same for all signals, like 30 seconds. But in this situation, most of this time would have been wasted. But our adaptive

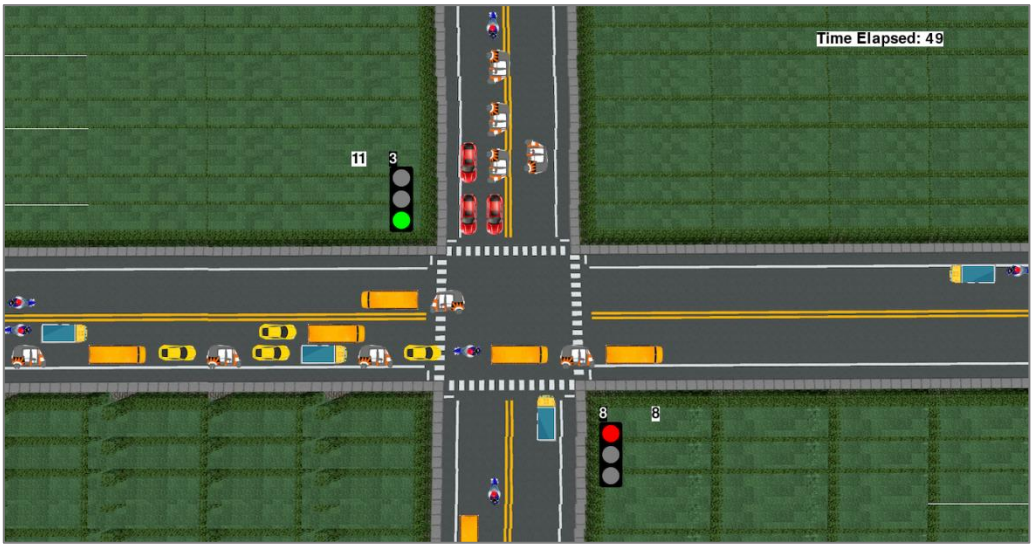


Figure 11 : Screenshot of the Stimulation's Final Output

system detects that there are only a few vehicles, and sets the green time accordingly, which is 10 seconds in this case.

Physical Simulation:

In this section, we address our next challenge: deploying our system in a real-world environment. We have conducted simulations and implemented our project using the necessary tools and equipment, with the overall process

detailed in the previous two sections. At this stage, we concentrated on executing our system within a physical simulated environment.

Our efforts have brought us to a refined point in the project, and we anticipate that our system will meet the requirements we established. We connect to the Arduino and then we apply in this simulation (Figure 19):

Figure 12 : Physical Simulation

In other words, we saw two ways to run our program effectively for this work. First, using the Pygame library, the system at a four-way intersection was displayed, then visualizing the process of management of the traffic in real-time.

Followingly, in order to implement it in the real world, Arduino was used, where it would physically represent the lights at the traffic lights and our system would be synchronized with it. Therefore, a fully workable system was made for a four-way intersection traffic management.

CONCLUSION:

The world's population continues to increase every day. With this, there is a rapid increase in traffic congestion within the city. The challenge of traffic keeps increasing with the increasing population. Several countries have tried to control this situation from the simplest device of traffic signs to fixed-time traffic signals which has brought some comfort. Knowing the deficiencies of fixed-time signals normally used in our area, the present work was an attempt to improve the flow of traffic at road intersections with the help of AI. Since some of these effective solutions require infrastructure and financial resources that are not easily available in all the environments, we considered only those that could be feasible.

We first created a dataset using video footage from cameras at the intersections, by splitting the videos into frames and labeling them to enable object detection. The YOLOv8 model was thereafter implemented due to the simple reason that its result performance came out to be greater than others. Although it was initially trained on a larger dataset of 1,000 object classes, we fine-tuned it to be more accurate in recognizing four classes of vehicles that were of interest for our study. We presented a variety of metrics that quantify the effectiveness of the trained model. A traffic signal control algorithm has been proposed that provides solutions with minimum infrastructure setup. We then presented the findings through two practical methods: a physical traffic simulation constructed with Arduino technology, which dynamically changed the timings of the signals at an intersection according to the output from our model, and another software-based to model the flow of this traffic using the Pygame library, based on the model's predictive abilities.

It is not always possible that the simulated implementation yields insights with applicability of the algorithm, which needed to be addressed in future research applying our findings in real-world scenarios.

FUTURE WORK

It is essential to program our system to prioritize the passage of emergency vehicles, such as ambulances, fire engines, and police cars, especially when their sirens are activated. This can be implemented by augmenting the existing model with additional data and developing a new model.

REFERENCES

N. Nigam, D. P. Singh, and J. Choudhary, "A Review of Different Components of the Intelligent Traffic Management System (ITMS)," Mar. 01, 2023, *MDPI*. doi: 10.3390/sym15030583.

S. Siri, C. Pasquale, S. Sacone, and A. Ferrara, "Freeway traffic control: A survey," *Automatica*, vol. 130, Aug. 2021, doi: 10.1016/j.automatica.2021.109655.

A. G. Sims and K. W. Dobinson, "The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits," *IEEE Trans Veh Technol*, vol. 29, no. 2, pp. 130–137, 1980, doi: 10.1109/T-VT.1980.23833.

P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton, "SCOOT-a Traffic Responsive Method of Coordinating Signals," 1981. [Online]. Available: https://api.semanticscholar.org/CorpusID:108797003

A. Wilbur, "NOTICE QUALITY ASSURANCE STATEMENT." [Online]. Available: http://www.tfhrc.gov

"Home: Traffic Technology Services." Accessed: Aug. 18, 2024. [Online]. Available: https://www.traffictechservices.com/

"Google Maps Platform Incident Management | Google for Developers." Accessed: Aug. 18, 2024. [Online]. Available: https://developers.google.com/maps/incident-management

A. M. de Souza, C. A. R. L. Brennand, R. S. Yokoyama, E. A. Donato, E. R. M. Madeira, and L. A. Villas, "Traffic management systems: A classification, review, challenges, and future perspectives," *Int J Distrib Sens Netw*, vol. 13, no. 4, Apr. 2017, doi: 10.1177/1550147716683612.

S. B. Rosende, S. Ghisler, J. Fernández-Andrés, and J. Sánchez-Soriano, "Dataset: Traffic Images Captured from UAVs for Use in Training Machine Vision Algorithms for Traffic Management," *Data (Basel)*, vol. 7, no. 5, May 2022, doi: 10.3390/data7050053.

"GitHub - labelmeai/labelme: Image Polygonal Annotation with Python (polygon, rectangle, circle, line, point and image-level flag annotation)." Accessed: Aug. 19, 2024. [Online]. Available: https://github.com/labelmeai/labelme

A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications."

X. Wang, T. E. Huang, T. Darrell, J. E. Gonzalez, and F. Yu, "Frustratingly Simple Few-Shot Object Detection." [Online]. Available: https://github.com/ucbdrive/

R. Girshick, "Fast R-CNN." [Online]. Available: https://github.com/rbgirshick/

J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks." [Online]. Available: https://github.com/daijifeng001/r-fcn.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection." [Online]. Available: http://pjreddie.com/yolo/

"YOLOv8: A New State-of-the-Art Computer Vision Model." Accessed: Aug. 19, 2024. [Online]. Available: https://yolov8.com/

P. Henderson and V. Ferrari, "End-to-end training of object class detectors for mean average precision," Jul. 2016, [Online]. Available: http://arxiv.org/abs/1607.03476