**International Journal of Innovation Scientific Research and Review**
*Vol. 06, Issue, 11, pp.7338-7352, November 2024*
*Available online at http://www.journalijisr.com*
*SJIF Impact Factor 2023: 6.599*

**Research Article**

**ISSN: 2582-6131**

# DEVELOPMENT AND APPLICATION OF AN INTELLIGENT TRAFFIC MANAGEMENT SYSTEM BASED ON YOLOV5 AND ALEXNET V3 FOR ROAD SAFETY ENHANCEMENT

**[1,] * Isatou K Njie, [2]Yongqian Sun, [1]Olumayowa O. Adedara, [1]Oyeleke Samuel Oluwafemi**

[1]MSc Software Engineering (College of Software), Nankai University, Tianjin, China.
[2]Professor, College of Software, Nankai University (Supervisor), Tianjin, China.

**Received 11th September 2024; Accepted 12th October 2024; Published online 30th November 2024**

**ABSTRACT**

Traffic congestion is a growing issue, especially in urban areas, requiring improved real-time traffic monitoring to enhance signal control and overall management. This research presents an intelligent traffic management system that optimizes traffic light operations based on real-time traffic density. Using image processing techniques with YOLOv5 and AlexNet V3, live camera feeds are analyzed to detect vehicles, monitor lanes, and adjust signal timings. This approach reduces congestion, improves road safety, and streamlines traffic flow. Python Open CV libraries and the Anaconda IDE were used to design and implement the system, ensuring efficient real-time data analysis for traffic management.

***Keywords:*** Road safety; traffic management; image processing; YOLOv5; Alexnet V3 Convolutional, Neural Network.

## INTRODUCTION

Urban transportation challenges are a global issue that impacts us daily, and many developing nations grapple with various difficulties related to traffic control, traffic management, and road safety. These obstacles encompass insufficient infrastructure, the absence of traffic monitoring systems, and limited resources for efficient traffic management.

The case study of The Gambia illustrates these challenges. Like many developing nations, the Gambia is dealing with traffic management and road safety challenges due to population growth, increased vehicles, and diverse road infrastructure. The inadequacy of proper roads contributes to frequent traffic accidents and congestion. Additionally, many urban areas face issues with fixed time cycles for intersectional traffic signals. Small activities, such as stopping a vehicle at an intersection or a vehicle breaking the traffic signal, cause a chain reaction that ultimately prompts colossal car influxes. This is confirmed in a study done by Dr. Gabor Oros z of the University of Exeter [1].

Effective management of vehicular movement is crucial for reducing congestion, safeguarding road users' well-being, and maintaining overall road safety. Conventional traffic management systems often strug-gle to handle growing congestion levels, leading to longer travel times, increased fuel consumption, and higher pollution levels. Fresh approaches are required to successfully address the limits of manual monitor-ing. According to research, artificial intelligence can efficiently address transportation concerns such as traffic management, safety, public transit, and urban mobility.

The existing traffic light system has a defined length and requires manual involvement from police officers to make changes. This approach is not very helpful since it relies on physically capturing offenders, which causes more congestion. Utilizing available information and data can lead to smarter services that enhance living conditions. An improved approach would involve determining signal times based on the number of cars crossing the intersection. If there is a higher volume of cars in a particular lane, the signal duration will be extended to alleviate congestion.

This paper aims to comprehensively study and explore how cutting-edge smart technologies integrated with data-driven insights can transform traffic management and enhance road safety. It suggests leveraging image processing from surveillance cameras and implementing a feedback mechanism in traffic light operations that considers traffic density during peak times.

Overall, this paper provides the following contributions:

- Development of an intelligent traffic management system utilizing AI technologies for real-time traffic monitoring and control.
- Implementing AI-powered traffic light control systems that dynamically adjust signal timings based on real-time traffic density.
- Introduction of an emergency vehicle prioritization mechanism that enables quicker response times through intelligent traffic signal adjustments.
- Application of image processing techniques to enhance road safety and reduce congestion in urban areas.

The rest of this article is organized as follows. Section 2 discusses the related work and technology background. Section 3 presents the experimental results and discussion. Section 4 describes the system architecture and explains the principles of operation. Section 5 shows the implementation and testing of the system. Finally, Section 6 concludes the paper.

## LITERATURE REVIEW

Over the years, there has been an increasing amount of literature on several works and research done to solve the problems of traffic congestion, and road safety using artificial intelligence techniques.

***Corresponding Author: Isatou K Njie,**
1MSc Software Engineering (College of Software), Nankai University, Tianjin, China.

In 2014, Kanungo *et al.,* in their work, proposed a system that utilizes video processing techniques for intelligent traffic light switching and real-time traffic density calculations at a four-way junction. The proposed system makes use of video cameras that are installed over red lights on each side of the junction. The cameras capture live feeds from the traffic and then process these feeds on a server using video image processing techniques [2]. The setback of the proposed system is it heavily depends on costly video cameras for its operation and has subpar performance in adverse weather conditions as well as concerns regarding scalability for larger traffic junctions.

Based on the research by Kanungo *et al.,* (2014) [2] Khekare, G.S. *et al.,* (2013) proposed the concept of VANETs (Vehicular Ad Hoc Networks). These networks serve as examples of technologies facilitating communication between vehicles and roadside units. VANETs play a role that significantly impacts the concepts behind smart city designs. The study focuses on a city framework designed to help drivers make intelligent decisions to avoid traffic congestion ultimately leading to reduced traffic jams. Additionally, it aims to provide real-time information on traffic conditions [3]. Their proposed system encounters a limitation in implementing VANET because it necessitates the installation of suitable hardware on every vehicle which can be difficult for two-wheelers. The entire framework is dependent on the user's decisions since traffic congestion will rely on them.

In 2010, Salama A.S., Saleh B.K., and Eassa M.M. introduced a system that uses sensors to manage traffic signals based on realtime vehicle movements. Their work was acknowledged by Kanungo *et al.,* in 2014 [2]. The system prioritizes roads with congestion and allows for emergency vehicles to receive priority using active RFID technology, adapting to traffic patterns and congestion levels [4].

In 2009, Haimeng Zhao *et al.,* introduced a traffic light system that uses a DSP, Nios II, and FPGA for dynamic control based on user demands. Both systems require ongoing analysis and maintenance and are vulnerable to damage due to challenging exterior conditions [5]. To enhance traffic flow and prioritize emergency vehicles, Varun Chava *et al.,* presented a smart traffic control system in 2023 that integrates the YOLOv4 and Mobile NetV2 convolutional neural network models. The system's primary objectives are to decrease the need for human involvement, provide precise traffic management results, and improve the effectiveness of real-time control. The entire number of cars on the road was counted, the average size of vehicles was calculated, and traffic signals were dynamically adjusted

based on the density of vehicles and the presence of emergency vehicles through the use of high-resolution cameras [6]. The proposed system also has limitations as it highly depends on hardware sensors like RFID, and high-resolution cameras which could pose a challenge in terms of maintenance, scalability of the system, and practical application. Tushar Deb Nath proposed an advanced Internet of Things-based road traffic control system in 2021. Intersection intelligent street lights monitor four important variables: number of cars, time it takes to activate, amount of waiting time, and emergency signals for each lane. To precisely count the number of cars on the road, this study uses an automated video processing technique that combines a Fully Convolutional Network (FCN) for precise pixel boundaries and a faster R-CNN for object detection (class + bounding box) [7].

In 2023, Sanjai *et al.,* suggested an image-processing-based method for detecting ambulances in traffic signals. Ambulance classification is the main topic. Their method classifies and identifies emergency vehicles according to their sort, make, or model using a convolutional neural network and VGG-16. VGG-16 CNNs and emergency photos are used in this strategy. Ambulance classification is made accurate and efficient with the use of these two [8].

A model for ambulance detection was proposed by Bhoomika (2022) [9] and Agrawal (2021) [10], as cited by Chava *et al.,(*2023), and it was created utilizing the YOLOv5 and YOLOv3 algorithms in different works. While Bhoomika (2022) employs the YOLOv5 method, [10] uses the YOLOv3 algorithm to classify vehicles in images collected from the footage as automobiles, buses, or trucks. Both models used predetermined algorithms. A pre-trained algorithm receives the cropped image of a vehicle that has been classified as a truck and uses it to determine whether or not it is an ambulance. However, the methods used by [9] and [10] both necessitate saving photographs each time which uses up storage space when processing image folders.

Using a convolutional neural network (CNN), Deepajothi *et al.,* (2021) [11], as referenced by Chava *et al.,* (2023) [6], developed a traffic management model for the detection of emergency vehicles. The Raspberry Pi is equipped with the CNN model and it will quickly decide whether to permit emergency vehicles to pass based on a traffic video input. However, the main purpose of this traffic system is for emergency vehicles. The output is red if there isn't an emergency vehicle visible in the input video. Other cars are not taken into account by this model, therefore traffic management for such a scenario is not offered.

Gandhi(2020) [12] and Rangari *et al.,* (2022) [13] suggested that the YOLO algorithm may be used to control traffic light signals consecutively. According to [12], the amount of traffic on the road determines when the green light should turn on. Thus, by employing YOLO (You Look Only Once) in image processing, the traffic density is determined. The JSON format is transformed from the YOLO to be used as an input (count of vehicles) for determining how long it will take for the green light to appear. Thus, the present traffic density determines when the green signal will turn on. [13] designed an intelligent traffic management system for India using YOLOv7. In terms of speed and accuracy, that iteration of the YOLO algorithm performs

better than any prior model for object detection.

Our model employs the hybrid approach of combining the strengths of YOLOv5 and AlexNet V3 to detect, classify, and count vehicles. The system in real-time can detect the video stream from a live camera. The implementation of this approach runs videos at 30-40 frames per second. This makes it detect objects and vehicles very quickly. It uses a low-power processor of 2.4 GHz. By using that, we can be able to achieve low-power operation - making this method the most suitable for traffic control. The system successfully deals with traffic data processing, vehicle recognition and classification, traffic forecasting, and real-time traffic control.

**Conventional Traffic Control System**

Many regions still use the conventional traffic control system to manage vehicular and pedestrian traffic flow on roads and intersections. The conventional traffic control system consists of:

- Traffic Signals and Signs: Traffic signals regulate the flow of traffic in urban areas. These areas utilize internationally standardized signals, with red, yellow, and green controlling the traffic movements. This automatic system has a disadvantage as it can cause excessive delays in traffic as it glitches and stops working at some point.

- Manual Traffic Enforcement: Police Officers are responsible for manually controlling traffic and making sure people comply with speed limits, seatbelt rules, and other traffic rules. Officers use a board, a sign light, and a whistle. This method is tedious and human errors are unavoidable, which results in compromises and uncontrollability of traffic.

### Limitations of Current AI Technologies in Developing Countries

- Infrastructural Limitations: Numerous developing countries may face obstacles regarding infrastructure which hinders the adoption of advanced AI technologies. These challenges include inadequate road networks, limited availability of high-speed internet, and a lack of extensive sensor systems necessary for gathering traffic information.
- A lack of technical expertise can be a hindrance. Insufficient local technical knowledge could challenge AI systems' advancement, implementation, and administration. The scarcity of AI and machine learning professionals may delay integration and increase reliance on foreign expertise, resulting in higher costs.
- AI systems, particularly our system that uses YOLO and AlexNet, require a steady source of energy to function effectively. Outages of electricity and inconsistent electrical distribution can interrupt traffic management systems, resulting in inefficiencies and a reduction in effectiveness.

## Theoretical Framework

1) Data Collection and Analysis: Plan methods for collecting data, observing with creativity and standardized techniques. Big data analysis and AI predict traffic trends and behavioral patterns.
2) Regulatory Framework and Enforcement: Cover elements for implementing AI-driven systems, aligning regulations with AI functionalities, ensuring enforcement, and establishing guidelines.
3) Stakeholder Engagement and Capacity Building: Collaborate with government agencies and community groups, investing in training programs to utilize intelligent technologies effectively.
4) Infrastructure Integration: Explore integrating intelligent technologies into existing infrastructure, evaluating suitability, and enhancing road infrastructure.

## Conceptual Framework for Implementation

Notwithstanding the recently portrayed hypothetical establishment, a solid reasonable structure that tends to both the key and functional parts of the combination cycle is expected for the effective execution of artificial intelligence-based traffic management frameworks.

1) Policy and Governance Framework: Establish protocols for data collection and utilization, ensuring AI solutions prioritize stakeholder needs.
2) Risk Assessment and Mitigation Strategies: Integrate risk assessments, examining challenges and cy-bersecurity implications, with proactive mitigation strategies.
3) Financial and Resource Planning: Focus on acquiring funds and resources, forming partnerships, and developing long-term financial strategies.
4) Performance Monitoring and Evaluation: Include methods for monitoring AI solutions and defining metrics for traffic signal efficiency and road safety outcomes.

This framework establishes the groundwork for the efficient implementation of AI-driven traffic management solutions.

# METHODOLOGY

The idea of developing a system using cutting-edge technology aims to enhance road safety, traffic flow, and signal control, with capabilities for detecting cars, trucks, buses, and pedestrians. This setup employs AI components to analyze visuals and predict traffic volume. A camera is installed alongside traffic lights to capture image sequences, which are processed to identify and count vehicles. The system uses Python's Open CV library for image processing to classify vehicle types accurately.

Utilizing AI technologies like YOLOv5 and AlexNet V3, the system performs real-time traffic analysis and management. Based on vehicle count, traffic signals are dynamically adjusted to alleviate congestion and prioritize emergency vehicles. This integration enhances precision and efficiency, enabling signal control, incident detection, and valuable insights for traffic control purposes. Figure 3.1 shows traffic lights with a camera installed alongside it to capture video feeds.

**Figure 3.1:** Traffic lights with Camera

## Models Used in the System

### AlexNet Convolutional Neural Network

AlexNet consists of eight layers: the first five are convolutional, and the last three are fully connected. The output from the last fully connected layer feeds into a 1000-way softmax, generating a distribution over 1000 class labels. AlexNet maximizes the multinomial logistic regression objective by enhancing the average log probability of correct label predictions. The second, fourth, and fifth convolutional layers connect only to kernel maps in the preceding layer on the same GPU, while the third convolutional layer connects to all kernel maps in the second layer. Neurons in fully connected layers interconnect with all neurons in the previous layer. Max-pooling layers follow both response normalization layers and the fifth convolutional layer. The ReLU activation function is applied to all layers. The first convolutional layer uses 96 kernels of size 11x11x3 with a stride of 4 on a 224x224x3 input image. The next layer operates on the pooled result from the first layer using 256 kernels of size 5x5x48. The third, fourth, and fifth convolutional layers are linked without pooling or normalization in between, with the third layer containing 384 kernels of size 3x3x256, the fourth layer having 384 sets of 3x3x192 bits, and the fifth layer containing 256 sets of 3x3x192 parts. Combined, each layer totals 4096 neurons. [14].

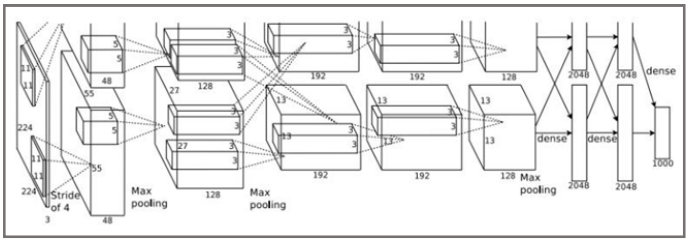


**Figure 3.2:** AlexNet Architecture

### YOLOv5 for Object Detection

The latest YOLOv5 version, "You Only Look Once," excels in quick and accurate object recognition in images and videos, making it highly effective for driving, traffic signal recognition, and surveillance. YOLOv5 has four versions: 5x, 5s, 5m, and 5l, each featuring a head section, configuration setup, and neck design. A model developed by experts [15].

Konala *et al.,* (2023) provide insights into enhancing YOLOv5 based on improved images, as discussed by Sheng *et al.,* (2022) [16]. A key aspect of the model is segmenting input images into a grid for analysis, predicting the number of bounding boxes in each cell and the likelihood of an object being present. Each bounding box includes five elements: a probability of containing an object, its width and height, and its center coordinates (x and y). The algorithm also predicts class probabilities for each object, indicating its category, such as a person, car, or dog.

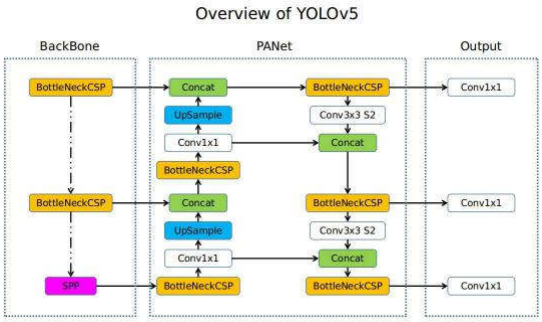


**Figure 3.3:** YOLOv5 Proposed Model

## Vehicle Detection

Figure 3.4 shows models in vehicle detection.

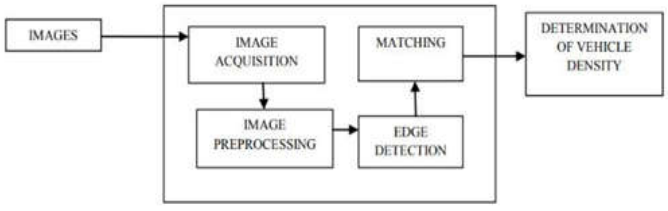


**Figure 3.4:** Models in Vehicle detection

1) Image Acquisition

An image is typically represented as a two-dimensional function f(x, y), where x and y are plane coordinates. The intensity of the image at any given point (f) is commonly referred to as the grey level. To transform an analog image into a digital format for storage in shared and drive databases, continuous x and y values must be converted into discrete ones. Each digital image comprises finite elements known as pixels [17]. Image acquisition is done by using an external Video. Capturing images involves utilizing a video source. For this project, the operating system of choice is LINUX as it is known for its open-source nature that undergoes updates.

2) Image Preprocessing

- Image Resizing/Rescaling

Image scaling is a common process in digital photography, where the size of an image is changed by adjusting the pixel grid. This resizing becomes essential when there's a need to alter the total number of pixels. The outcome can differ considerably based on the algorithm used, even if the same resizing operation is employed [18].

- Image Enhancement

Enhancing images entails modifying digital images to better adapt them for display or subsequent examination. For example, noise can be removed to facilitate the identification of important features. In low-contrast images, neighboring elements may blend during binarization. Therefore, it is essential to minimize the blending of these elements before applying a threshold to the image. This is where "Power-Law Transformation" becomes valuable as it enhances contrast and refines segmentation. The foundational version of power law transformation is:

$$s = cr^{\gamma}, \qquad (3.1)$$

In Formula 3.1, r and s symbolize the input and output intensities, with c denoting positive con-stants. Power law is utilized by various imaging devices for capture, printing, and display. The exponent in the power-law equation is commonly known as gamma. As a result, gamma correction is employed to address these power law response phenomena ensuring accurate image representation on computer screens.

In our project, Image preprocessing functions are imported from Python Open CV libraries and are included in the final Python program. This will automatically process the image when the program is invoked.

- Image Processing

Image processing techniques are used to improve the quality and usefulness of images captured by various devices, such as cameras, sensors on space probes and aircraft, or everyday photographs. It covers aspects related to how images are represented, methods for reducing file sizes with-out significant loss in quality, and advanced manipulations that can be performed on image data. These manipulations include processes like improving image clarity through sharpening or blur-ring, adjusting brightness levels, and enhancing edges among others. Image processing is a branch of signal processing where the input is an image (e.g., photos or video frames) and the output may be another image or a set of characteristics extracted from the original one [19].

- Edge Detection

The process of edge detection involves a set of mathematical techniques that are utilized to recognize areas in a digital image where there is a sudden change or disruption in brightness, indicating the presence of edges. These abrupt changes form curved lines and are crucial for tasks such as feature identification and extraction in fields like image processing, machine vision, and computer vision [20].

3) Image Matching

(Yamini 2022) [21] described image matching as a recognition method utilizing matching and involves using a prototype pattern vector to represent each class. When presented with an unknown pattern, it is assigned to the class that most closely resembles it according to a predetermined measure. The most straightforward method is the minimum distance classifier, which calculates the distance between the unknown pattern and each of the prototype vectors and selects the shortest distance for deciding. Another method relies on correlation, expressed directly in terms of images, and is quite intuitive. Our image-matching technique entails comparing a reference image with a real-time image pixel by pixel. Pixel-based comparison presents certain drawbacks; however, it is recognized as one of the most effective methods for the algorithm implemented in this project for making decisions. The original image is stored in a memory matrix, and the real-time image is similarly transformed into the required matrix. For two images to be deemed identical, their pixel values within the matrix must correspond. Subsequently, the percentage of alignment can be expressed as in Formula 3.2:

$$\% \text{ match} = \frac{\text{Number of pixels matched successfully}}{\text{total number of pixels}} \quad (3.2)$$

## Experimental Evaluation and Analysis of the System

In our system, each frame is taken to the YOLOv5 algorithm and the frame is read as input through Open CV's im read() method in Python. The YOLOv5 is trained using the COCO dataset which has about 330,000 images over 2.5 million object instances and has a large object collection, segmentation, and captioning collection. Training the YOLOv5 algorithm in the COCO dataset allows the algorithm to possess a much broader and more varied range of objects and situations, thereby improving its capacity to precisely identify and categorize objects [6]. The model has the coco.names file which contains the names of all the classes and objects that the system can detect on the custom detector.

The image input size is set to 416 x 416 pixels. The neural network model and dataset are trained over 200 epochs and use a batch size of 150, that is, the model weights are updated after every 150 samples are processed. The confidence threshold is set to 0.5 which indicates that the model must be at least 50% sure that the detected object belongs to a particular class (car, truck, people, etc) before it considers it a valid detection.

### Data Preprocessing

The system predicts traffic conditions by using Tensor Flow to construct YOLO and AlexNet models with several dense layers, indicating a focus on capturing complex patterns within traffic data. Before feeding the data into the model, it is split into training and testing sets. Assessing the model's performance post-training by comparing predicted and real traffic situations showcases how well the model can forecast traffic trends.

- Dataset Processing

  It is crucial to prepare the dataset before starting to ensure its compatibility with models. Tasks such as improving, organizing, and isolating components are some examples of actions that can enhance the accuracy and quality of the data used in training models.

- Splitting the Dataset for Training and Testing

  It is common in AI practices to split the dataset into training and testing sets for model evaluation. Our approach involves converting a Data Frame into an array simplifying data processing. To distinguish features and labels for model training and evaluation, the dataset is divided 90-10 into training and testing subsets respectively. The setup designates the column as a label and the first four columns of the dataset as predictors. This step is crucial to demonstrate how an AI model learns from the training set without exposure to test data. The training set is utilized to educate the model by allowing it to analyze the information while the testing set is employed to assess how well the model performs with data.

- Training and Testing Sets
  - Training Set: This segment making up 90% of the dataset is usually bigger. It is used to teach the model and helps the algorithms understand the patterns and relationships in the data.
  - Testing Dataset: The testing subset, constituting the remaining 10% of the data collection, plays a critical role in assessing the model's capacity for generalization. This portion facilitates an impartial evaluation of the model's performance with new data and showcases its ability to forecast traffic conditions outside the training set accurately.


**Figure 3.5:** Code showing the Creating, Training, and Testing of Datasets

### Performance Metrics of YOLOv5

The performance of each trained YOLOv5 model was evaluated using metrics such as accuracy, precision, recall mean average precision (mAP), and F1 score which are determined through calculations. Formula 3.3 represents Accuracy, Formula 3.4 Precision, Formula 3.5 Recall, Formula 3.6, and Formula 3.7 represents mean average precision.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.3}$$

$$Precision = \frac{TP}{TP+FP} \tag{3.4}$$

$$Recall = \frac{TP}{TP+FN} \tag{3.5}$$

$$\text{F1-Score} = 2 \times \left(\frac{P \times R}{P+R}\right) \tag{3.6}$$

$$\text{mAP} = \frac{1}{n}\sum_{k=1}^{k=N} AP_k \tag{3.7}$$

In the Formulas above, the scenario TP represents the count of predictions TN stands for the count of accurate negative predictions, FP indicates the count of incorrect (false) positive predictions, FN denotes the count of incorrect negative predictions, APk signifies the average precision (AP), for class k and n represents the number of confidence thresholds which is set at 0.5 or 50% [22].

Figure 3.6 represents graphs depicting the Accuracy, Precision, Recall, mean average precision (mAP), and f1-score trained over 200 epochs.

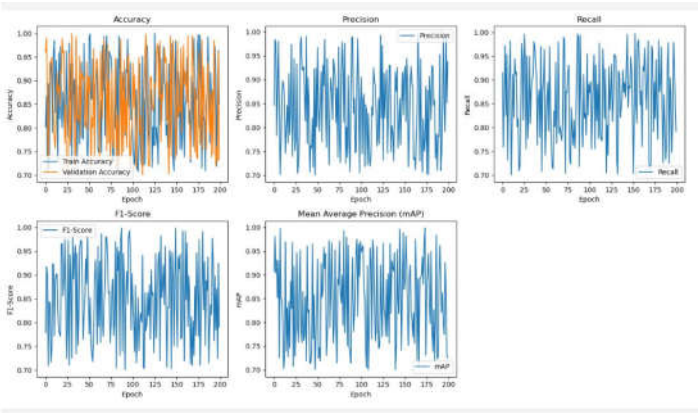


**Figure 3.6:** Graphs showing performance metrics of Accuracy, Precision, Recall, mean average precision (mAP), and f1-score obtained after training YOLO over 200 epochs.

### Performance Evaluation

The system's performance is evaluated by training the YOLOv5 and AlexNet neural networks to anticipate traffic outcomes. The mean squared error is used as the loss function throughout the training procedure. An independent test dataset is used to validate the model's potential for generalization. The trained model is tested using the test data (test_predictors and test_labels), yielding a final performance metrics. The model' s efficacy is further validated by comparing anticipated values to actual values using visual representations, demonstrating its accuracy in forecasting appropriate traffic cycle lengths. The test set is used to check the model's accuracy, and the precision is evaluated by comparing displayed and real optimal time cycles in bar graphs which will be shown in the implementation phase of the project.

### Experimental Results

After training and testing, the model and the size of the datasets are determined. The threshold helps to filter out weaker detection and reduce false positives, improving the overall accuracy of the object detection process. After 200 training epochs, the model's predictions are visually evaluated against actual values through line plots to demonstrate its efficacy in estimating optimal traffic light cycles. Non-maxima suppression is applied to the images to reduce redundancy among detected bounding boxes. Only the strongest bounding box is retained when multiple boxes overlap significantly and detect the same object. The weaker overlapping detection is eliminated, making sure each detected object is represented by a single bounding box, improving the result's accuracy and clarity.

The validation data set limits the batches of samples analyzed per epoch to 100. It provides test data for evaluating model performance at the end of each epoch. There is a limit on the number of validation batches that can be run. This setup helps monitor how well the model is learning and its ability to adapt to inputs.

This application includes components related to traffic control systems such as data collection, analysis, and display aimed at improving traffic flow. To effectively manage traffic signals involves creating datasets from traffic images, analyzing the vehicle density and types in the area, and utilizing this information. By using a state machine that adjusts signals based on real-time traffic data, the system mimics the functionality of traffic lights.

Additionally, we utilize tools like Matplotlib for visualization, Pandas for data manipulation, and techniques from the Image Processing module to study car types and traffic density in images. Through image processing algorithms applied to a series of photos, results are compiled into a CSV file. It also can generate graphs depicting lane-by-lane vehicle counts, traffic flow patterns, and optimal cycle duration. These functionalities offer insights into traffic behaviors and congestion issues that can aid in making decisions, for effective management of traffic signals.

In the model, regression is depicted as a function that connects the output (y) to input features (X). We use weights (w) and biases (b) at all levels of the model along with activation functions like ReLU in layers to predict the target value y. Ultimately, it produces a value as the prediction. During training, the model adjusts weights and biases to reduce the squared error between predicted values in the dataset. The formula for the regression can be represented as:

$$y = f(W_3(ReLU(W_2(ReLU(W_1X + b_1) + b_2)) + b_3) \tag{3.8}$$

The formula represents a feed forward neural network with two hidden layers and an output layer where:

X = [X1, X2, X3, X4] represents the input features. W1, W2, W3 are the weight matrices for each layer. b1, b2, b3 are the biases for each layer. ReLU(x) = max(0, x) is the Rectified Linear Unit activation function used in the first two dense layers. f (x) is the identity function in the output layer since it is a regression task. yˆ is the predicted output.

Formula 3.8 is the computation used in the hybrid system. YOLOv5 analyses full pictures in a single forward pass, predicting bounding boxes and class probabilities for objects using several convolutional layers and ReLU activation. This feature enables real-time recognition of cars and people, which aids in traffic flow monitoring and congestion detection. AlexNet, used for image recognition also employs convolutional layers and ReLU activations for feature extraction and classification. AlexNet can classify various vehicle kinds and analyze traffic patterns in a hybrid system.

The model has been trained to minimize the loss function, specifically by reducing the mean squared error between the predicted outputs and the actual labels. This error aids in the adjustment of the model's parameters, hence improving predictions.

The mean-squared loss function determines the difference between the model's predictions and the actual labels. This inaccuracy aids in adjusting the model's parameters, hence improving predictions. Labels are the known output or goal values that the model attempts to predict. Labels are required for supervised learning, and the purpose is to teach the model to generate predictions based on input data. The labels are the proper responses or outcomes for each example in our dataset. It is used in the classification and regression tasks to classify the datasets of images of vehicles.

Figure 3.7 shows the sample of images used to train the dataset. Figure 3.8 shows the images detected in real-time after the training and testing.

**Figure 3.7:** Sample of Images Used to train the dataset

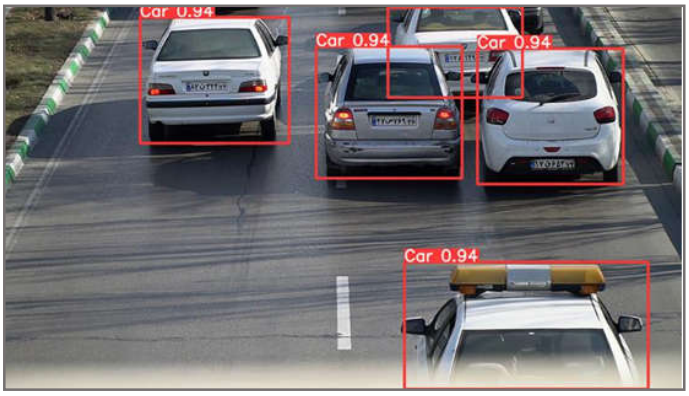


Figure 3.8: Vehicle Detection in real-time with Vehicle label

### Traffic Prediction

We test the system with four-lane samples in our model. After comparing four lanes, the lane with the most cars is given a green signal at that particular timeframe. For the subsequent timeframe, this process is repeated as shown in Table 3.1.

**Table 3.1:** Traffic Predictions and Green Signal Activation

| Lane Numbers and Car Count | Prediction | Green Signal |
|---|---|---|
| Lane 1: 10, Lane 2: 25, Lane 3: 30, Lane 4: 42 | Lane 4 has the most vehicles | The green signal turned on for lane number 4 |
| Lane 1: 60, Lane 2: 18, Lane 3: 42 | Lane 1 has the most vehicles | The green signal turned on for lane number 1 |
| Lane 2: 20, Lane 3: 39, Lane 4: 25 | Lane 3 has the most vehicles | The green signal turned on for lane number 3 |
| Lane 1: 76, Lane 2: 90, Lane 4: 46 | Lane 2 has the most vehicles | The green signal turned on for lane number 2 |
| Lane 1: 104, Lane 3: 52, Lane 4: 65 | Lane 1 has the most vehicles | The green signal turned on for lane number 1 |

## SYSTEM ANALYSIS AND DESIGN

### System Requirements

According to Tilley (2019), a system requirement is a characteristic or feature that must be included in an information system to satisfy business requirements and be acceptable to users. System requirements serve as benchmarks to measure the overall acceptability of the finished system [23].

Mahalank *et al.,* (2016) stated that the design of any AI or IoT-based system is rooted in two perspectives: Requirement Analysis. The initial perspective revolves around Functional Requirements which relate to the functions that the AI or IoT-based Traffic Density indicators can execute as a system. The second outlook includes Non-Functional Requirements, which denote the characteristics that enhance brand value for the design unit [24].

1) Functional Requirements

Functional requirements outline the activities, procedures, and capabilities that the system is expected to execute. The functional requirements of the system are:

- Design a user interface that is intuitive and easily accessible for individuals with varying degrees of technical knowledge.
- Real-Time Traffic Monitoring: The system needs to consistently monitor traffic conditions with cameras and sensors, and process data in real-time.
- Utilize YOLOv5 to accurately detect and categorize objects in the traffic, including automobiles, pedestrians, and cyclists, and AlexNet V3 for detailed classification such as identifying vehicle types and recognizing license plates.
- Traffic Flow Optimization: Analyzing traffic information to pinpoint congestion and adjust traffic light timing can improve traffic flow and minimize bottlenecks.
- Incident Detection and Response: Automatically identify road accidents or incidents and notify traffic management centers and emergency services.
- Implement AI-powered analysis to forecast traffic trends, detect bottlenecks, and recommend the best routes for drivers.
- Offer a complete user interface for traffic operators that combines real-time data representation, notifications, and the ability to manually intervene in AI decisions.

(a) Software Requirements Specifications

- PYTHON LANGUAGE
- ANACONDA IDE

(b) Hardware Requirements Specifications

Table 4.1 shows the hardware requirements specifications of the system.

**Table 4.1:** Hardware Specifications

| Hardware Components | Specification |
|---|---|
| System | Pentium Dual Core |
| Hard Disk | 120 GB |
| Monitor | 15''LED |
| Input Devices | Keyboard, Mouse |
| RAM | Minimum of 4 GB |

2) Non-Functional Requirements

The non-functional requirements outline the functioning of the system, emphasizing quality characteristics, efficiency, and operational criteria.

- High level of precision in object detection and categorization while reducing incorrect identifications and missed detections. The system must maintain a reliability of 99.9% uptime for consistent functionality.
- The system should grow and encompass additional areas without a decline in performance.
- Real-time processing allows for the rapid detection and categorization of objects, facilitating prompt decision-making in traffic management. Also, ensure that all methods of surveillance and data collection adhere to privacy laws and ethical standards.
- Ensure seamless integration with current traffic management infrastructure and emergency response systems to guarantee compatibility and interoperability.

## System Architecture

### Hardware Architecture

The suggested framework involves Raspberry Pi linked to four groups of LEDs, which simulate the traffic signals. The captured images and the reference images are fed manually to the Raspberry Pi currently. We have incorporated the Raspberry Pi model 3 into this setup [25]. The Raspberry Pi serves as the controller for the system. To manage the lighting, it employs a Python service that starts automatically. It captures an image and compares it with the one. If there is traffic on a road at the intersection compared to others, that road will be given priority and have a longer green light duration determined by the Pi based on matching rates with the reference image. For analysis purposes, the Pi also transmits data to the cloud. The location of the signal percentage of matches in each image and timestamps of when photos were taken are all included in the data provided in format.

### Software Architecture

The proposed model makes use of Open CV which allows companies to conveniently incorporate and adapt the code according to their requirements. The library contains a vast collection of over 2500 advanced algorithms, encompassing both traditional and cutting-edge techniques in computer vision and machine learning. It is compatible with Windows, Linux, Android, and Mac OS, and it provides interfaces for C++, C, Python, Java, and MATLAB. It is primarily designed for real-time vision applications and makes use of MMX and SSE instructions whenever possible. Developed in C++, Open CV also includes a templated interface that integrates smoothly with STL containers [26].

## Modules of the System

1) Image Acquisition Module

Image acquisition is accomplished through the use of an external video or videos saved in the path folder of the project. The operating system utilized in our project is LINUX, which is an open-source program that is often updated.

2) Image Preprocessing Module

Image preparation processes are imported and loaded from the Python Open CV libraries and used in the final Python program. When you run the program, the image will be processed automatically.

3) Object Detection Module

In our current study, fundamental components of image analysis include edges, lines, and points, with a focus on edges. Our research employs an object detection approach for image alignment, where the method identifies pixels within the image that align with the shapes of depicted objects. This process culminates in the generation of a binary image highlighting the detected object.

4) Vehicle Count Module

The process of vehicle counting commences with establishing a baseline image depicting an unoccupied road, which is stored in memory for reference. Subsequently, images captured from the four lanes undergo comparison with the reference image to ascertain the density of vehicles present and it is facilitated through the utilization of the Object Detection Module.

5) Light Control Module

After the vehicle detection process, reference and real-time images are compared to facilitate traffic light control by considering the number of vehicles in each lane. Varying durations for activating the green light are applied based on the number of vehicles. Initially, the minimum green light on time is set to 10 seconds for all lanes. At the point when lane 1 has more vehicles than other lanes, the green light is turned on for the lane longer. This process will continue checking for the lane with the heaviest traffic and allocating more time to it for smooth traffic flow.

6) Traffic Prediction Module

The system uses the AI framework Tensor Flow to analyze real-time traffic data. This part uses data analysis and scenario prediction to create traffic pattern models by utilizing Tensor Flows capabilities. During this procedure, databases containing data on traffic volume, speed, and congestion levels are used to train AI models. The system can therefore predict traffic conditions. Make suggestions for efficient traffic signal tactics. To lessen traffic and improve overall traffic flow efficiency, traffic light management is implemented with the help of the predictive feature.

## Feasibility Study

Before implementing a system, the organization, stakeholders, and developers need to evaluate the pro-posed system's feasibility cost, effectiveness, acceptance, and ability to enhance traffic management and road safety.

The feasibility study is particularly useful in thoroughly assessing the effectiveness of the proposed system in addressing the challenges

and requirements of urban traffic management and the evaluation goes beyond considerations such as market readiness, technological infrastructure prerequisites, financial sustain-ability, and potential socio-economic impacts.

The key considerations involved in the feasibility study include:

1) Technical Feasibility: To evaluate the technical feasibility of this project, a series of issues are considered when doing this study:
   - Assessing the present status of AI technologies applicable to traffic control and road safety, including examining the effectiveness of machine learning models, image processing methods, and real-time data analysis approaches in tackling traffic challenges.
   - Quality data is essential for training AI models, such as information on traffic patterns, incident records, and meteorological factors, which should be easily obtainable and reachable.
   - Supporting the implementation of AI solutions requires essential infrastructure such as traffic cameras, sensors, and computational capacity.
2) Economic Feasibility: In this study, the economic impact the system has on stakeholders is investigated. Investing in a cost-effective traffic management system will go a long way. The amount to be spent on the system is limited; procurement cost, which includes the cost of equipment like the traffic cameras, installation, and the cost of training users. The expenditure must be justified, and some of the technologies used in the developed system. Since the system will be deployed in a cloud, the update of the system will be done by service providers. The maintenance of hardware and training of individuals to get familiar with the system.
3) Social Feasibility: this part of the study explores the perception and acceptance of the system by road users, governments, and different stakeholders. It is important to keep in mind the process of training users on how to use the system. The users' willingness to adopt the system relies on their level of understanding and familiarity with its operations. Purchasers must consider the framework as essential as opposed to surveying it as a reason to worry.

## System Design

System design is a way of outlining the structure, elements, sections, interfaces, and information of a system to satisfy particular needs. It entails the transformation of user requirements into a detailed blueprint that guides the implementation phase. The objective is to improve traffic flow, reduce travel times, enhance safety, and support emergency response by coming up with an intuitive functional structure that meets the task at hand whilst putting performance, maintenance, and sustainability into account. This chapter's purpose is to describe the system's design phase.

### UML Diagrams

UML diagrams are designed to help us and users understand complicated software systems' concepts, code architecture, and potential implementations. Information about the system is converted into a graphical representation that is easier to read and comprehend. It adopts a standardized procedure for creating a system model and documenting conceptual theories as well. They assist developers in seeing the big picture of the system, as well as readers and people without programming experience in understanding software processes and operations.

### Use Case Diagram

A use case diagram is a graphical representation of a user's potential interactions with a technology. This definition of a use-case diagram is close to that of Yamini (2022) [21] who defined it as a form of behavioral diagram specified by and built from a use-case analysis whose objective is to offer a graphical picture of a system's performance in the form of actors. Its primary goal is to demonstrate which system functions are executed by which actor, as shown in Figure 4.1.

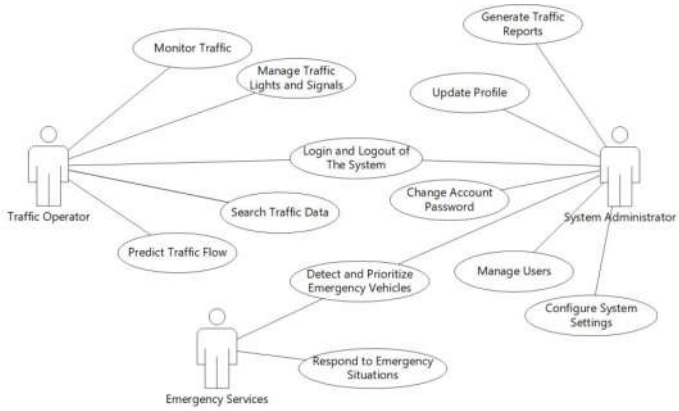


**Figure 4.1:** Use Case Diagram

### Class Diagram

This diagram showcases how the Traffic Light Controller, Vehicle Detection System, and Traffic Data Analysis Module work together to improve traffic flow. Demonstrate that the Traffic Operator, System Administrator, and Public User classes are derived from the User Interface class. This demonstrates common characteristics while highlighting their distinct functionalities. Also, specify that the Traffic Data Analysis Module could consolidate data from the Vehicle Detection System as well as other sources for analysis. Depict interconnections between classes such as the User Interface class depending on the Database class to store and retrieve user preferences and traffic data as shown in Figure 4.2.

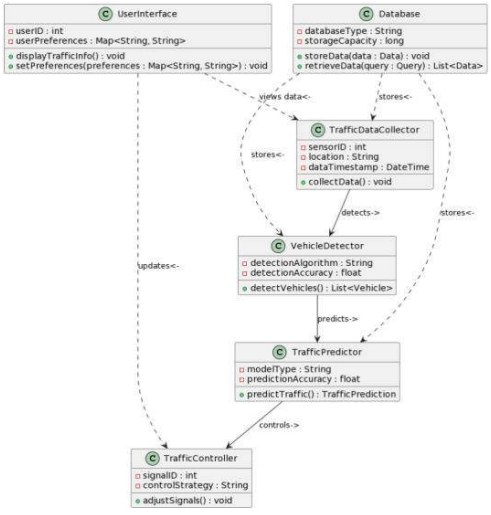


**Figure 4.2:** Class Diagram

### Component Diagram

This diagram offers an overview of the software components and their relationships within the system. Component diagrams illustrate implementation details and are crucial during the application's

implementation phase, impacting effectiveness and maintenance. Identifying system files, libraries, and relevant components, along with their interactions, is essential (Yamini 2022) [21].

The system performs critical functions like collecting traffic data, Data Processing, and recognition and categorization of vehicles. The Traffic Prediction tool anticipates patterns using historical and real-time data, and Traffic Control adjusts signals based on these predictions. A user interface enables traffic operators and administrators to monitor conditions and make modifications. The Database stores processed and raw data, configurations, and past traffic records, as shown in Figure 4.3.

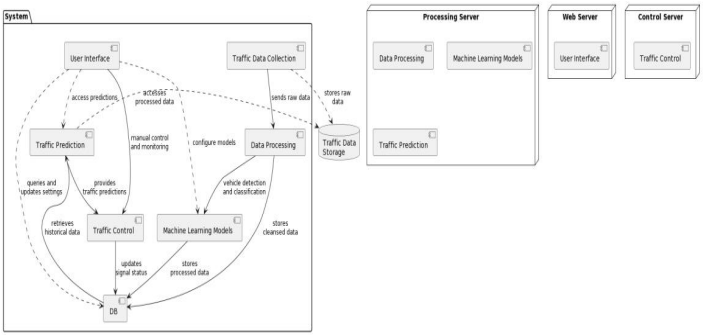


**Figure 4.3:** Component Diagram

### Deployment Diagram

Figure 4.4 shows the system's execution across hardware and network resources, highlighting key component interactions. [21] defines a deployment diagram as nodes representing hardware for application deployment, crucial for scalability, maintenance, and portability.

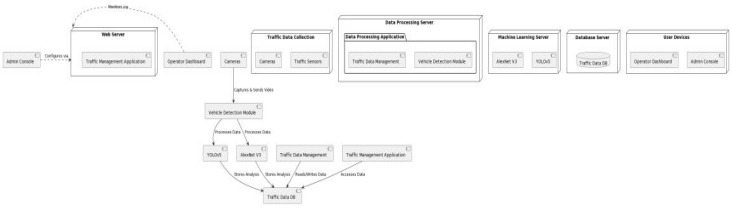

**Figure 4.4:** Deployment Diagram

### Database Design

The database structure and design facilitate the core operations of our traffic management system, such as collecting and analyzing data, identifying and categorizing vehicles, predicting traffic patterns, prioritizing emergency vehicle access, and managing user roles as shown in Figure 4.5.

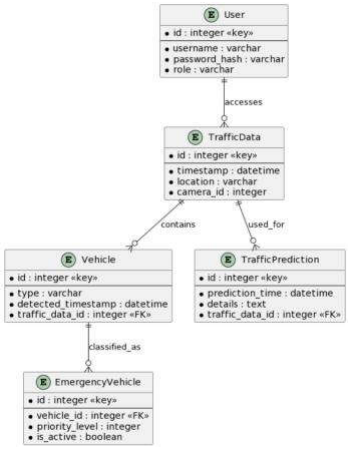


**Figure 4.5:** Database Design

### Entity Relationship Diagram

The ER diagram of our system shows different entities and specifies the entities, attributes, and relation-ships that exist between them. Our ER diagram has entities that are represented by key components such as the system, traffic data, vehicle, traffic prediction, and emergency vehicles as shown in Figure 4.6.

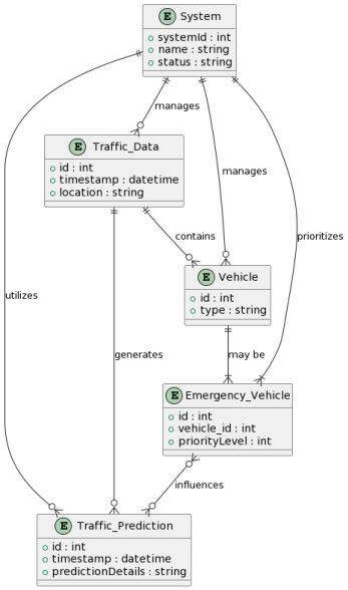


**Figure 4.6:** ER Diagram

## Proposed System Architecture

The use of the system starts with data collection from traffic cameras installed alongside traffic lights. The different modules of the system in Figure 4.7 are discussed in previous chapters. The videos are converted to frames, and images are put into the machine learning models (YOLOv5 and AlexNet V3) to detect vehicles in the images and identify, classify, and count them. Tensor Flow is employed to analyze data and predict traffic conditions. The predictions are used to adjust traffic signals accordingly. The user interface allows different users and provides them with access to traffic data and system controls. The number of vehicles on each lane and their labels are displayed, and datasets of the different lanes, graphs, and the view of the road can be viewed in the user interface. With the conda package, switching from different pages in the system is easy.

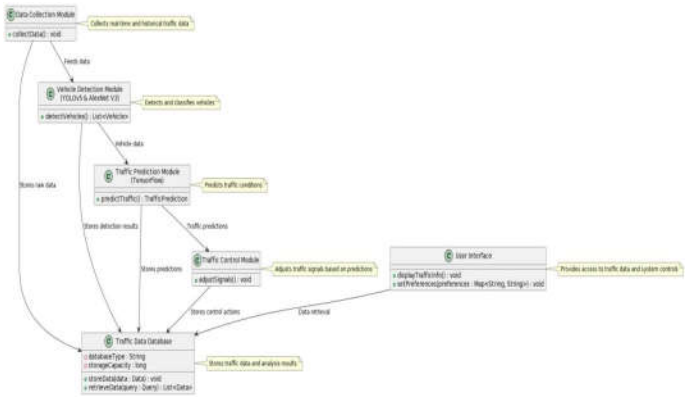

**Figure 4.7:** Overview of the Proposed System Architecture

### Flow Chart of System Architecture

This flowchart 4.8 shows the different stages involved in the traffic management system from how videos are captured and processed

using YOLOv5 for vehicle detection and labeling, further classified by AlexNet V3 where detected emergency vehicles are given priority to pass. Other traffic conditions are analyzed to predict traffic flow using Tensor Flow models. Forecasts such as this, real-time analysis, and comparison of the number of vehicles in each lane are used to manage and change traffic lights.

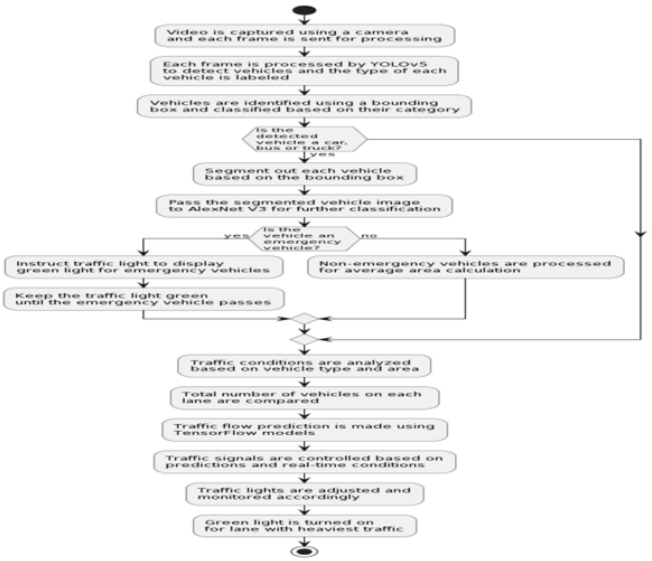


**Figure 4.8:** Flow Chart Showing Processes involved in the Traffic Management

## GUI (Graphical User Interface)

Our system makes use of Tkinter for the design of the GUI as it is a standard Python GUI library that provides an easy approach to developing GUI programs. Its inclusion in Python makes it accessible and simple to use. Its main advantage is it offers a variety of widgets for creating interfaces. The GUIs that are created show AI-detected objects, offer graphs and insights on traffic patterns, and enable manual or AI-assisted adjustments of the traffic lights just to name a few. The figure below shows a simple GUI of the system:

The GUI consists of:

- Traffic specification functions inter-green period, amber period, number of phases, initial delay, mini-mum time, the width of the four lanes, maximum cycle time, size of the dataset, and number of iterations.
- buttons that provide different views; Test image processing, run the traffic system, show the view of the road, create and display datasets, and data visualization.
- buttons to import the input folder and the output folder that contains the frames and images used in the system.

The traffic specifications can be changed and given different values.

# IMPLEMENTATION AND TESTING OF THE SYSTEM

The implementation stage of the system development life cycle entails transforming the system design into a fully operational and functional system. It involves all tasks associated with converting conceptual, architectural, and detailed designs into an active solution. Implementation acts as a link between the planned structure and functionalities and the practical realization and utilization of the system in an operational environment.

This phase is very important in the creation of the system. It includes testing and working with different videos, images, and datasets as well as releasing the system to its intended users and putting it to work in a real-world scenario. The result of this stage is dependent on testing, engaging with stakeholders, and verifying that the website adheres to all specified demands in advance of and following deployment.

## Report of the System

- It generates Datasets of different specifications used in the system
- You can easily save the Dataset and export it into CSV format.
- The traffic data can be represented in graphical format and saved to analyze traffic.

## Testing

### Unit Testing

Unit testing is a software testing approach to evaluate individual components or modules, ensuring they meet requirements and function correctly. Each field entry, page link, initial screen, and message response must work without delay (Yamini 2022) [21].

Our model confirms that the correct input and output folders are selected in the proper format—input for video frame images and output for trained models. Users should access the correct page, edit specifications, and upload folders without screens, buttons, or image loading delays.

Result: All buttons and links work correctly. There are no delays in loading the buttons and images.

1) Uploading Input and Output Folders

   A unit test was conducted to test the outcome of not uploading the correct image input and output folders. This resulted in the system not working.

2) View output of Image Processing

   The image processing algorithm's output was empty without uploading the right folders. When the right input and output folders are selected, the image processing algorithm output shows the images.

3) Dataset Size

   The dataset size was tested with different numbers of frames set in the traffic parameter specifications (50, 75, 100). The results obtained from the test are the exact size of the dataset specified in the traffic parameter specifications.

### Integration Testing

Integration testing ensures the smooth functioning of modules or programs requiring interaction without errors. It validates that the interfaces and connections between different system components are working as intended. After passing individual unit tests, integration testing focuses on assessing the control flow among modules and the exchanged data. This type of testing follows procedures similar to unit testing, such as creating a test plan with a series of tests, and is usually conducted by programmers and/or systems analysts. Integration testing often involves four approaches: user interface testing, use scenario testing, data flow testing, and system interface validation (Dennis 2009) [27].

### System Testing

System testing is crucial in the software development lifecycle, ensuring the fully integrated system meets specific requirements. It aims to identify faults and ensure error-free operation before launch, instilling confidence in quality and reliability. Comprehensive testing reduces the likelihood of malfunctions and facilitates a smooth transition to operational use. This phase occurs after unit and integration testing but before production deployment, with the primary goal of verifying that the system adheres to its intended design and performs as expected in real-world scenarios.

### White Box Testing

White box testing involves analyzing the framework's engineering and programming to evaluate how well vehicle recognition algorithms like YOLOv5 ensure identification under various conditions. It examines all possible combinations of conditions, loops, and code paths to ensure coverage and functionality. This testing also identifies overlooked issues, such as traffic patterns or unexpected problems, while focusing on security by detecting weaknesses that could compromise the system's integrity. It thoroughly assesses the system's performance, emphasizing speed and accuracy in vehicle identification and traffic forecasting.

### Black Box Testing

Black box testing is one procedure for evaluating the framework's usefulness. It's essential spotlight is on client cooperation. anticipated that results interestingly, should be the inside pecking order of codes. The most common way of checking that the framework can precisely distinguish, characterize, and change traffic lights for different vehicle types and traffic situations starts with testing. User interface testing ensures that traffic administrators can undoubtedly view and utilize the framework to simply decide. Execution testing surveys the framework's ability to handle enormous measures of traffic information and convey the necessary answers for testing circumstances.

### Functional Testing

Each function in the program is tested to ensure compliance with all standards and concepts. Every capability is examined by inputting data and verifying that the output meets predetermined criteria. The primary goal of testing is to assess whether the program operates according to its specifications, including input/output verification, user interface, database interactions, and API functionality. Various methodologies are employed, such as unit, smoke, integration, system, and sanity tests. The process involves creating test cases based on requirements, establishing a test environment, executing tests, recording defects, and retesting to identify functionality gaps, errors, or unexpected behavior while ensuring adherence to specified requirements.

The functional test of our system is focused on:

- making sure there are valid inputs. That is to say, only valid inputs are accepted in the system.
- invalid inputs will result in errors, and the system will not function normally.
- the indicated classes of application outputs must be exercised.

Table 5.1 shows the functional testing of the system with each iteration on different lanes.

**Table 5.1:** Functional Testing of the System

| Traffic Density | Number of Vehicles and Traffic flow | Lane 1 | Lane 2 | Lane 3 | Lane 4 | Total Time for one Iteration |
|---|---|---|---|---|---|---|
| No traffic | The minimum green light on time is set to 10 seconds for all lanes | 10 sec | 10 sec | 10 sec | 10 sec | 40 sec |
| Lane 1 has more vehicles than the other lanes | 19 vehicles with a traffic flow of 69.25 | 34 sec | 10 sec | 10 sec | 10 sec | 64 sec |
| Lane 2 has more vehicles than the other lanes | 21 vehicles with a traffic flow of 82.25 | 10sec | 39 sec | 10 sec | 10 sec | 69 sec |
| Lane 3 has more vehicles than the other lanes | 21 vehicles with a traffic flow of 64.0 | 10 sec | 34 sec | 10 sec | 10 sec | 64 sec |
| Lane 4 has more vehicles than the other lanes | 19 vehicles with a traffic flow of 70.25 | 10 sec | 10 sec | 10 sec | 34 sec | 64 sec |

### User Acceptance Testing

According to Yamini (2022), User Acceptance Testing (UAT) is crucial for any project and requires significant end-user involvement. End users, such as traffic controllers, test the intelligent traffic control system to ensure it meets operational requirements and user expectations in real or simulated scenarios. This testing confirms the system's performance, including vehicle identification accuracy, improved traffic flow, and interoperability with existing infrastructure. Feedback from UAT leads to final adjustments to align the system with its goals. The UAT results indicate that all test cases passed, confirming the system's readiness for deployment and its capability to enhance traffic control and safety measures. [21].

## Description of Modules of the System

1) First Page: Figure 5.1 is the first page accessed after the user runs the 'main.py' file in the terminal of the project in the Anaconda navigator.

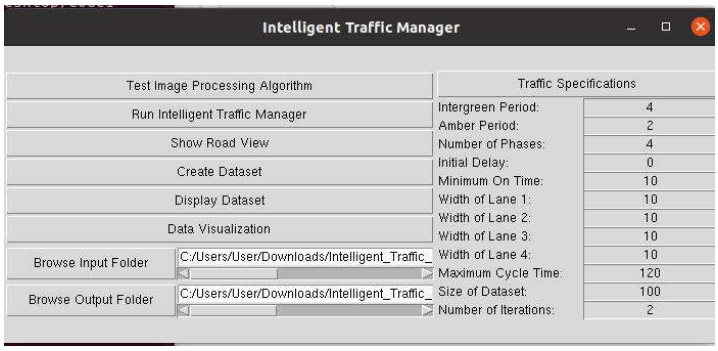


**Figure 5.1:** First Page

2) Traffic Specifications: Figure 5.2 shows buttons that, when clicked, will let us set the traffic specifications for editing and

changing the values to our preferred specifications. For example, the initial dataset size is set to 100 frames, which means 100 minutes of traffic is considered. The number of iterations as well as everything else can be edited to fit the user's demands.

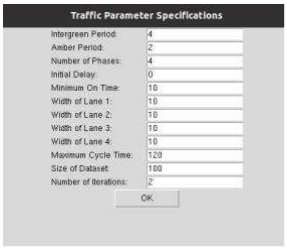


**Figure 5.2:** Traffic Parameter Specifications

3) Image Processing Algorithm: Before testing the image processing algorithm, the input folder that contains the images that were gotten from the video frames must be selected. The output folder must be selected too. After setting the input, output folders, and traffic specifications, the image processing algorithm can be tested. The system is going to test with one image which it will choose randomly from the images that were gotten from the frames converted from the videos captured. The actual image is compared to the reference image and the total number of vehicles detected is shown along with the labeling of the vehicles.

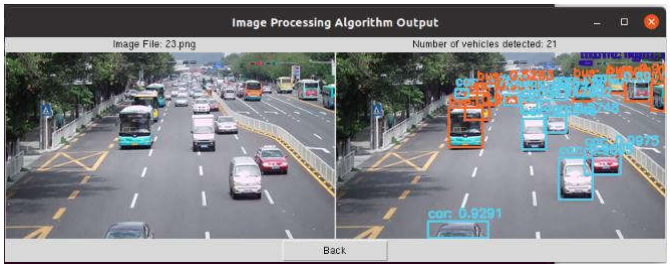


**Figure 5.3:** Process Image

4) Run Intelligent Traffic Manager: the detection of the total number of vehicles in each lane will lead us to run the traffic manager. This will compare the total number of vehicles in each lane turning on the green light to the lane with the heaviest traffic. Figure 5.4 shows lane 1 has 20 vehicles detected, thus a time of 34 seconds was allocated to that lane.

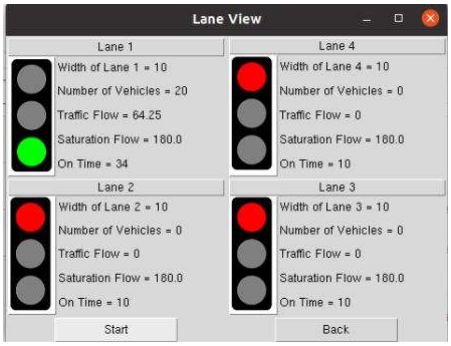


**Figure 5.4:** Traffic Light Control

5) Road View: Figure 5.5 shows different lane structures and shows the number of vehicles on each lane. This will be updated based on the files uploaded. Traffic lights will work on each lane depending on the vehicle density of the lanes and give priority to the lane with the most vehicles.


**Figure 5.5:** Road View

6) Create Dataset: Figure 5.6 shows the dataset of all the vehicles given along with the number of vehicles in each lane, traffic flow, optimum cycle time, delay time, etc. The size of the dataset is affected by the number of frames set in the specifications, which was set to a hundred minutes. Due to this, the dataset will be created and generated for all the data of 100 minutes. The dataset can be saved and this will create a CSV file where all the data will be stored.

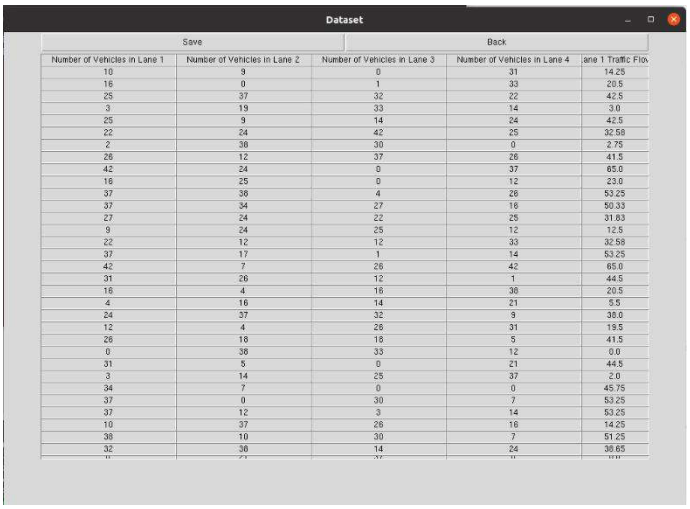


**Figure 5.6:** Creating Dataset

7) Display Dataset: Figure 5.7 shows the created and saved dataset displayed on the page.

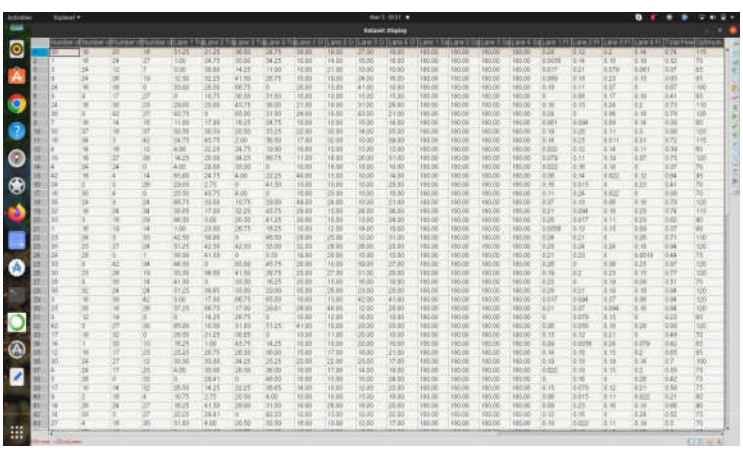

**Figure 5.7:** Displaying Dataset

8) Data Visualization: Under this, different parts of the system are represented in a graphical view for traffic analysis as shown in Figure 5.8. It consists of the number of vehicles in each lane, traffic flow in each lane, green light time, traffic flow ratio, total traffic flow, and optimum cycle time.

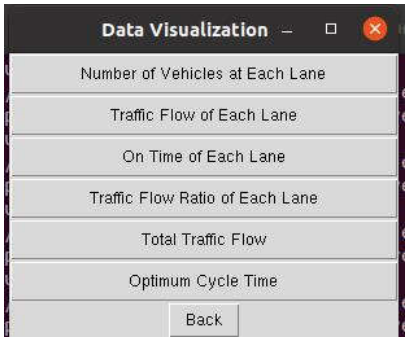


**Figure 5.8:** Data Visualization

9) Number of vehicles in Each lane: Figure 5.9 entails graphs showing the number of vehicles in each lane in the 100-minute time frame that was given.

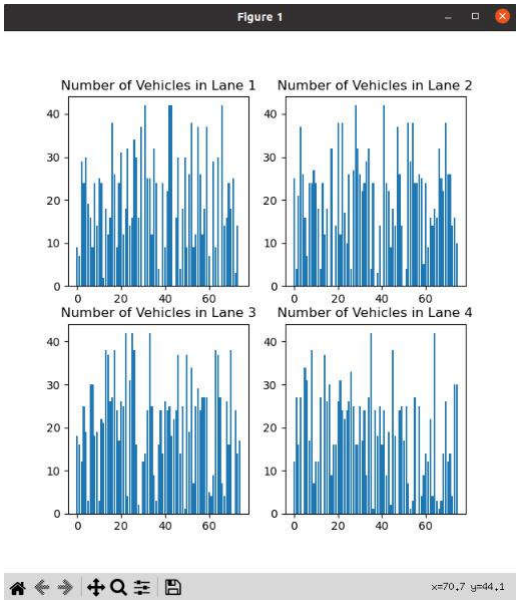


**Figure 5.9:** Graphs of Lanes

10) Total Traffic Flow: The Figure 5.10 shows the flow of traffic on each lane.

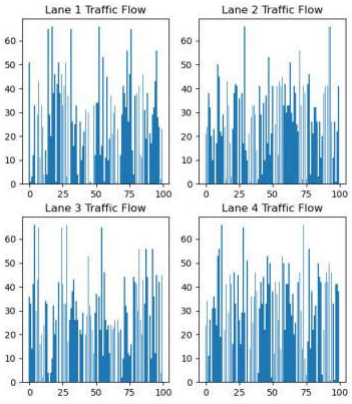


**Figure 5.10:** Traffic flow

11) Optimum Cycle Time: The optimum cycle time shows how much priority is given to each lane in each second. This can be compared to how much traffic the system has reduced compared to the existing system.

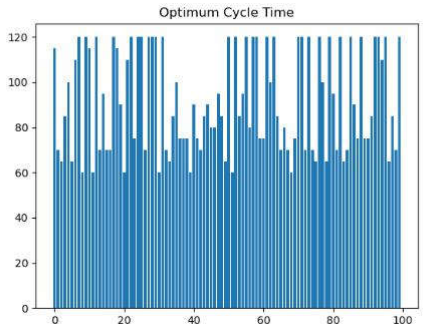


**Figure 5.11:** Cycle Time

12) Green Light On Time in Each Lane: Figure 5.12 depicts graphs showing how much time the vehicles on each lane waited.

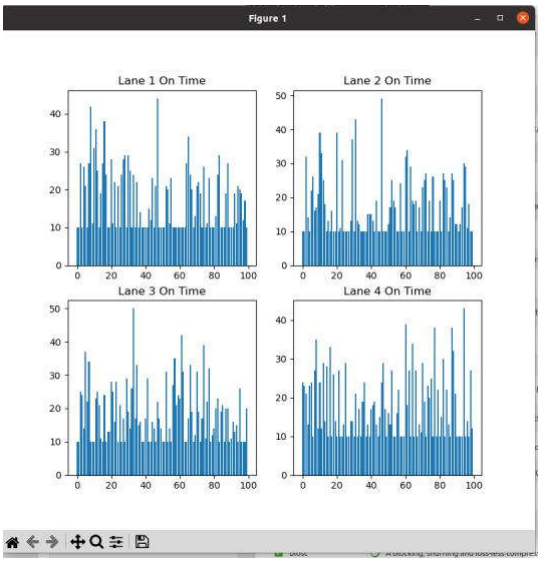


**Figure 5.12:** On Time

## CONCLUSION AND FUTURE WORK

This thesis provides insight into how artificial intelligence can improve traffic control and safety. It highlights the potential of using YOLOv5 and AlexNet V3 models to enhance traffic flow and safety measures, demonstrating the feasibility of integrating AI into urban traffic management. The main goal was to address road traffic accidents caused by negligent driving and congestion from increasing population and vehicles, developing a model to enhance road safety and traffic management. The research shows how AI can revolutionize traffic management by providing forecasts, accurately categorizing vehicles, and dynamically adjusting signals to reduce congestion and prevent accidents.

Future studies should assess the long-term effects of enhancing the traffic control system by broadening the dataset and exploring neural network models for improved accuracy, focusing on real-time traffic management to ease congestion. Integrating the system with online platforms can provide up-to-date traffic information and prioritize emergency vehicles, influencing planning and infrastructure upgrades for safety and efficiency. Public engagement is essential for system improvement, considering reduced emissions from optimized mobility.

The system does not work up to expectations in low-light conditions. Overcoming this would require switching over to a hard-coded system or night vision cameras could be installed to keep the dynamic system working at night.

The system's scalability can extend beyond urban areas, addressing data collection challenges. Although the algorithm performs well in daylight, it struggles in low-light conditions. Data augmentation techniques and methods like illumination maps can enhance performance. Finally, low vehicle volume on one side may lead to longer wait times, which should be addressed in future plans.

## BIBLIOGRAPHY


[1] G. Orosz, B. Krauskopt, and R. Wilson, "Traffic jam dynamics in a car following model with reaction-time delay and stochasticity of drivers," IFAC Proceedings Volumes (IFAC-PapersOnline), vol. 6, 2006.

[2] A. Kanungo, A. Sharma, and C. Singla, "Smart traffic lights switching and traffic density calculation us-ing video processing," in 2014 Recent Advances in Engineering and Computational Sciences (RAECS), pp. 1–6, 2014.

[3] G. S. Khekare and A. V. Sakhare, "A smart city framework for intelligent traffic system using vanet," in 2013 International Mutli-Conference on Automation, Computing, Communication, Control and Com-pressed Sensing (iMac4s), pp. 302–305, 2013.

[4] A. S. Salama, B. K. Saleh, and M. M. Eassa, "Intelligent cross road traffic management system (icrtms)," in 2010 2nd International Conference on Computer Technology and Development, pp. 27–31, 2010.

[5] H. Zhao, X. Zheng, and W. Liu, "Intelligent traffic control system based on dsp and nios ii," in 2009 International Asia Conference on Informatics in Control, Automation and Robotics, pp. 90–94, 2009.

[6] V. Chava, S. S. Nalluri, S. H. Vinay Kommuri, and A. Vishnubhatla, "Smart traffic management system using yolov4 and mobilenetv2 convolutional neural network architecture," in 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), pp. 41–47, 2023.

[7] T. D. Nath, "Iot based road traffic control system for bangladesh," International Journal of Recent Technology and Engineering, vol. 10, pp. 60–66, May 2021.

[8] P. Sanjai, J. J. Sam, S. Iyengar, and K. Kalimuthu, "Ambulance detection in traffic signals using image processing," in 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 927–931, 2023.

[9] G. M. Bhoomika, "Ambulance detection using image processing," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 2022.

[10] K. Agrawal, M. Nigam, S. Bhattacharya, and G. Sumathi, "Ambulance detection using image processing and neural networks," vol. 2115, no. 1, p. 012036, 2021.

[11] S. Deepajothi, D. P. Rajan, P. Karthikeyan, and S. Velliangiri, "Intelligent traffic management for emer-gency vehicles using convolutional neural network," vol. 1, pp. 853–857, 2021.

[12] M. M. Gandhi, "Smart control of traffic light using artificial intelligence," in Proc. 5th IEEE Inter-national Conference on Recent Advances and Innovations in Engineering-ICRAIE 2020, 2020. IEEE Record#51050.

[13] A. P. Rangari, A. R. Chouthmol, C. Kadadas, P. Pal, and S. K. Singh, "Deep learning based smart traffic light system using image processing with yolo v7," in 2022 4th International Conference on Circuits, Control, Communication and Computing (I4C), (Bangalore, India), pp. 129–132, 2022.

[14] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, pp. 1303–1314, June 2018.

[15] L. Hu, "An improved yolov5 algorithm of target recognition," in 2023 IEEE 2nd International Confer-ence on Electrical Engineering, Big Data and Algorithms (EEBDA), (Changchun, China), pp. 1373– 1377, 2023.

[16] S. Xu, Z. Guo, Y. Liu, J. Fan, and X. Liu, "An improved lightweight yolov5 model based on attention mechanism for face mask detection," Journal Name, September 2022.

[17] R. Harshitha, R. Chandan, K. Poornima, U. N. Navyashree, and P. S. Gowda, "Traffic light switching by traffic density measurement using image processing technique," International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 5, May 2017.

[18] S. S. Patra, S. S. Alam, S. Chandra, and H. N. Pratihari, "A survey on digital images zooming tech-niques," International Journal of Innovative Research and Studies, vol. 3, no. 3, pp. 865–872, 2014.

[19] R. R. Hoare and D. Smetana, "Accelerating sar processing on cots fpga hardware using c-to-gates de-sign tools," in 2014 IEEE High Performance Extreme Computing Conference (HPEC), (Waltham, MA, USA), pp. 1–6, 2014.

[20] "Digital image processing based intelligent traffic light," Dec. 16 2019. [Accessed: Feb. 16, 2024].

[21] G. Yamini, "Smart traffic control system using canny edge detection algorithm." MCA Project Report, University College For Women, Osmania University, 2022.

[22] P. K. Yadav et al., "Assessing the performance of yolov5 algorithm for detecting volunteer cotton plants in corn fields at three different growth stages," 2022. [Online]. Available: https://arxiv.org/abs/2208.00519. [Accessed: Feb. 16, 2024].

[23] S. Tilley, Systems Analysis and Design. United States of America: [Publisher], 12th ed., 2019.

[24] S. N. Mahalank, K. B. Malagund, and R. M. Banakar, "Non functional requirement analysis in iot based smart traffic management system," in 2016 International Conference on Computing Communication Control and automation (ICCUBEA), (Pune, India), pp. 1–6, 2016.

[25] Wikiversity, "Raspberry Pi — Wikiversity," 2021. [Online; accessed 21-February-2021].

[26] University of Oslo, "Introduction to c++ and opencv," 2018. Accessed on Mar. 6, 2024.

[27] R. R. Dennis, A. Wixom, and B. Roth, System Analysis and Design. John Wiley & Sons, Inc., 4th ed., 2009.


*********