Intelligent Traffic Monitoring and Management for Real-Time Vehicle Tracking in Smart Cities

Repaka Sai Akshith[1],P.V.S Uday Kiran[2],Santhosh Reddy Irugula[3]

***Abstract*— Traffic congestion is an important problem within urban settings that results in delay, added fuel consumption, and increased emissions. Conventional traffic signal control is based on preset timers that mostly result in inefficiencies for adapting to dynamic traffic patterns. An intelligent traffic signal control system involving real-time object detection based on the YOLO deep model learning framework incorporated with an OV7670 camera module controlled through Arduino is provided in this paper. The system takes real-time traffic images, measures vehicle density, and adapts the green light time dynamically in order to optimize traffic. The system facilitates dynamic traffic regulation in real time by processing images captured, examining congestion levels, and sending controlling signals to the Arduino appropriately. The experimental results prove that the system is able to improve waiting time and traffic efficiency significantly compared to conventional fixed-timer traffic controls.**

Keywords: Adaptive Signals, Real-Time Tracking, Vehicle Detection, Embedded Vision,Traffic Density

I. INTRODUCTION

Traffic congestion is now one of the major urban problems globally, contributing to enormous economic losses, escalated fuel usage, environmental degradation, and stress on commuters. The unprecedented growth in the number of cars on the road and poorly optimized traffic signal control systems have contributed to this problem in cities. Conventional traffic systems use static, time-scheduled signal controls that do not adjust to the actual flow of traffic. These types of systems usually lead to inefficiencies like redundant waiting at intersections, excessive queues at traffic signals, and overall added travel time. These inefficiencies are further exaggerated during rush hours when the allocation of vehicles is extremely skewed across lanes.

Traffic management systems based on sensors and image processing have been one of the few alternatives that have been explored to overcome this difficulty. Sensor-based solutions utilize technologies such as infrared sensors, ultrasonic sensors, and inductive loop detectors to count vehicles at intersections. However, these methods suffer from several drawbacks, including high installation and maintenance costs, susceptibility to environmental factors such as extreme weather conditions, and the inability to distinguish between different types of vehicles. On the other hand, image-processing-based traffic systems leverage computer vision techniques to analyse real-time traffic conditions using cameras. Traditional image-processing techniques like background subtraction, edge detection, and contour detection have been used in previous studies but tend to be inaccurate because of changes in lighting conditions, occlusions, and dynamic backgrounds.

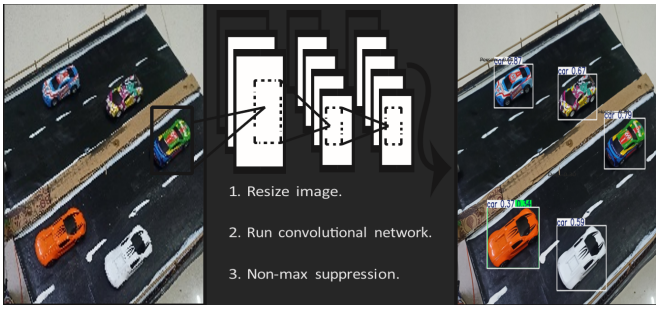

Fig. 1. YOLO (1) resizes to 640×640, (2) runs a network, and (3) thresholds detections by confidence.

With the fast growth of AI, deep learning based object detection models are showing to be really useful in changing the way traffic is managed. The YOLO (You Only Look Once) model has become one of the most effective deep learning methods for detecting objects in real-time because it can look at images fast and also with good accuracy. Not like the old image-processing ways, YOLO is able to detect more than one vehicle at the same time and gives accurate bounding boxes, so it fits really well for live traffic checking. This study is trying to mix the YOLO model with a cheap hardware setup that works using Arduino microcontroller along with an OV7670 camera module. By using live image capturing and AI-based traffic density checking, the system changes the signal timings on its own to improve traffic movement and reduce the jams.

The system that was made uses a OV7670 camera for taking live traffic images, a YOLO-based deep learning model for finding the vehicles, and an Arduino-based traffic signal controller. The images are being checked in real-time by the trained YOLO model which finds and counts the number of vehicles at the crossroad. Based on the traffic amount that gets detected, an adaptive logic finds out the best green light timing and sends that to the Arduino using serial connection. With this smart way of deciding, the lanes that have more traffic will get longer green light time, and the lanes with less traffic will wait less, which helps in making the whole system work better.

YOLO is actually very simple to understand: check Figure 1. The rest of the paper is arranged like this: In Section 2, the paper talks about other works that are already done in the area of smart traffic control. Section 3 gives the idea about how the system is build, with details about both hardware and software parts used. In Section 4, the paper goes through the method that was used, like how images are taken, how traffic is checked, and how the signal is changed based on that. Section 5 tells about how the system was put to work and the results we got after trying it with different traffic levels. Finally, Section 6 tells the final thoughts and also what can be done later to make the system better, like using edge AI and IoT stuff for smart city uses.

Object detection is one important work in computer vision where it finds and shows many objects inside a image. The older ways of object detection, like region-based convolutional neural networks (RCNN), use steps like region proposal, taking out features, and then classifying them. These methods do work well, but they take too much computing power and are very slow when it comes to getting fast results, so they are not so useful for things that need real-time speed like self-driving, watching through cameras, and smart traffic systems.

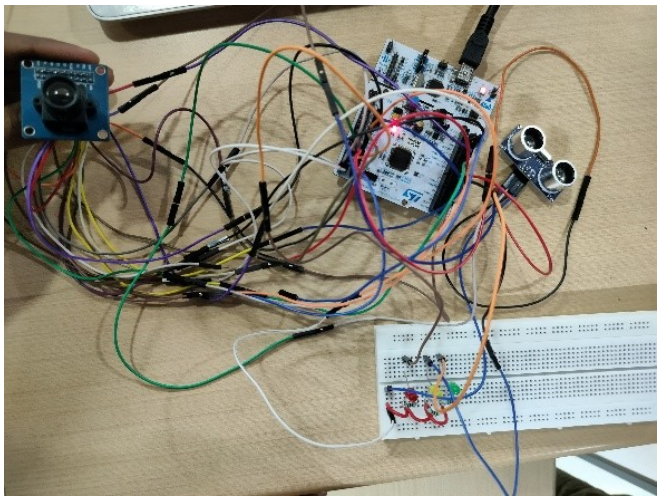

Fig. 2. FPGA for processing into a tensor

To fix these problems, the You Only Look Once (YOLO) method was made as a real-time object detection model that makes both accuracy and speed better. Not like the old models that use region proposals, YOLO uses just one convolutional neural network (CNN) to guess many bounding boxes and class chances for the objects in the image at the same time. Since YOLO sees object detection as a straight regression task, it removes the need for doing hard feature extraction and region finding steps, which helps it to do detection fast and smooth with just one forward move in the network.

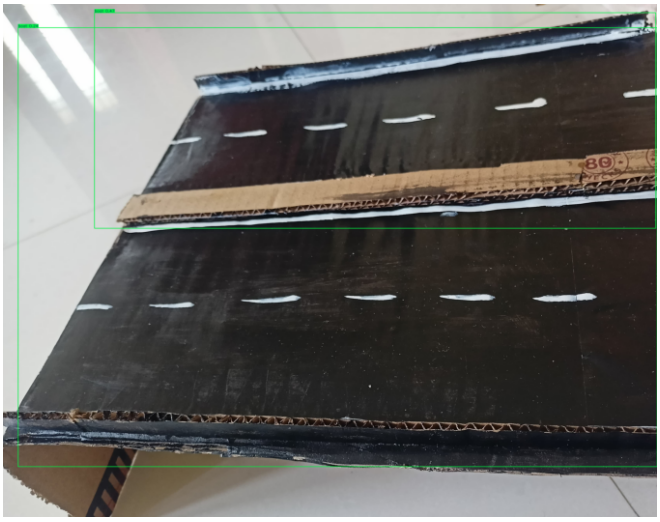

Fig. 3. Final prediction is a $7 \times 7 \times 30$ tensor2.1.

This model is trained with a convolutional neural network and tested on the PASCAL VOC detection dataset [9]. Early convolutional layers of the network extract features from images, while the fully connected layers make the final prediction, such as class probabilities and bounding box coordinates.

The model is motivated by MobileNet's image classification model [34] and has 24 convolutional layers with two fully connected layers. Instead of employing GoogLeNet's inception modules, the design uses a reduced approach with 1×1 convolutional layers for reducing dimension, followed by 3×3 convolutional layers, as suggested by Lin et al. A full representation of the network is given in Figure 3.

A faster version, Fast YOLO, is also proposed for real-time object detection. Fast YOLO uses a shallower network, having only 9 convolutional layers compared to 24, and fewer filters per layer. Other than these structural changes, the training and testing parameters are the same as the original YOLO.

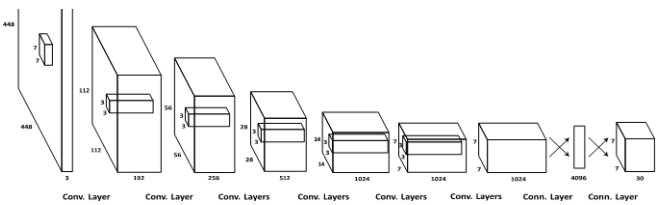

Fig. 4. The final output of network is the $7 \times 7 \times 30$ tensor of predictions.

II. Literature Survey

Gomathi et al. [1] proposed an Optimized Lightweight Real-Time Detection Network Model for IoT Applications, presenting a better YOLOv8 model tailored for Internet of Things application. With real-world applications and benchmarks in embedded systems, the paper stresses faster detection speed and efficiency without compromising great accuracy. The model is quite fit for real-time monitoring since it is meant to run effectively on low-power embedded devices. To underline its better performance in IoT-based smart surveillance, the paper also contrasts YOLOv8 with other deep learning models. Experimental data reveal a notable decrease in inference time while preserving detection accuracy.

Drushya et al. [2] proposed Fastening Deep Learning-Based Morphological Biometric Identification Using OV7670 Camera Module, where the focus was on doing biometric identification using the OV7670 camera. They used deep learning models to try and get better results in recognition, and also pointed out the camera's image capturing and power saving on embedded systems. The study gives a detailed look at how well the camera works in low-light situations, which makes it more useful for security kind of applications. A new method was proposed for pre-processing to get better features from facial images. By using real biometric datasets, the performance was tested to see how accurate the recognition works.

AlRikabi et al. [3] proposed A Hardware Efficient Real-Time Video Processing on FPGA with OV7670 Camera Interface and VGA, where they talked about a real-time video processing method by using embedded systems. The study used OV7670 camera along with FPGA and VGA screens, focusing more on making the hardware better so that processing can happen faster. When it was compared to normal microcontroller-based systems, using FPGA helped a lot to reduce delay. The research also looked into memory-saving ways to store images, so that the resources are used in better way. Based on testing, the system is very good for smart watching systems as it can handle video in real-time and still use very low power.

Lin et al. [4] proposed Implementation of Object Detection Algorithms on Embedded Systems: Challenges and Solutions, where they looked into different object detection algorithms. The study shows the problems faced while trying to use these algorithms on embedded systems and also gives some ways to fix them for better speed and saving resources. A comparison was done to check how YOLO, SSD, and Faster R-CNN models work for embedded vision uses. The paper also talks about the balance needed between how much computing power is needed and how accurate the detection is when resources are limited. They also suggested using methods like quantization and pruning to make the object detection models better.

Zhang et al. [5] proposed Real-Time Object Detection and Tracking Based on Embedded Systems, introducing a local dynamic mapping system based on cameras. The study incorporates embedded AI models for 3D position estimation, object detection, and real-time tracking. The study emphasizes how crucial motion estimation methods are for enhancing tracking capabilities in a range of environmental circumstances. For increased accuracy, a novel hybrid approach that combines deep learning-based detection with optical flow was presented. Robust performance in real-time applications was demonstrated through extensive testing on real-world datasets.

Alaidi et al. [6] proposed Server-Based Object Recognition, using an Arduino UNO and the OV7670 camera module to capture images. The YOLO object recognition algorithm is used on a server to process the images, with automation and security applications. In order to improve computational efficiency and lessen the strain on embedded devices, the study investigates cloud-based processing. According to experimental results, using server-side deep learning models significantly increases detection accuracy. To assess the system's resilience in real-world settings, it was tested in a range of lighting conditions.

Kumar et al. [7] proposed Real-Time Object Detection Using an Ultra-High-Resolution Camera on Embedded Systems, concentrating on using ultra-high-resolution cameras for real-time object detection. The study places a strong emphasis on optimizing algorithms to strike a balance between detection accuracy and speed. To improve real-time processing efficiency, a thorough examination of data bandwidth limitations in high-resolution cameras was carried out. To improve object detection, noise in captured frames was reduced using a specialized filtering algorithm. The findings show that deep learning methods in conjunction with high-resolution imaging greatly increase recognition accuracy.

Patel et al. [8] proposed Interfacing Camera Module OV7670 with Arduino, giving a detailed tutorial on how to integrate the OV7670 camera module with Arduino. As a helpful guide for embedded vision applications, the paper describes image capture and processing techniques. A comparative analysis of various microcontrollers for the best camera performance is part of the study. To improve image quality and lessen distortion, a thorough calibration procedure was suggested. The viability of inexpensive image processing solutions utilizing Arduino-based systems is shown by practical experiments.

Chen et al. [9] proposed Pothole Detection for Safer Commutes with the Assistance of Deep Learning, detecting potholes for road safety by using the OV7670 camera

module with Arduino. The study demonstrates how well deep learning models can detect hazards in real time. To accurately classify road surface anomalies, a convolutional neural network (CNN)-based method was presented. To assess the system's dependability under various environmental circumstances, the study put it through testing on a variety of road conditions. The findings show that automated pothole detection can enhance road safety and drastically lower maintenance expenses.

Singh et al. [10] proposed Real-Time Small Object Detection on Embedded Hardware for 360-Degree Cameras, presenting the Penta Mantis-Vision project's findings. In order to minimize the use of computational resources, the study investigates the parallel processing of multiple 4K camera streams for small object detection. A specialized object detection pipeline tailored for panoramic video feeds is presented in the study. To increase object localization accuracy, sophisticated image stitching techniques were used. Results from experiments show that the system can accurately detect small objects in complex environments.

Wang et al. [11] proposed Design of Intelligent Access Control System Based on STM32, creating an access control system that makes use of the STM32 processor for automation and security applications and the OV7670 camera for image capture. Real-time facial recognition with low processing latency was a key component of the system's design. In order to guarantee data confidentiality in access control applications, a secure communication protocol was incorporated. Performance tests showed that it worked well in high-security settings with a variety of user authentication scenarios.

Nguyen et al. [12] proposed Design and Implementation of Real-Time Object Detection System Using SSD Algorithm, introducing an SSD algorithm-based real-time object detection and recognition system. To increase detection accuracy, the study contrasts pre-trained models with deep learning approaches. To find the best configurations, an analysis of SSD performance across various embedded platforms was carried out. In order to increase processing speed without sacrificing accuracy, the study presents a lightweight feature extraction technique. According to experimental findings, SSD strikes a balance between accuracy and real-time performance.

Khan et al. [13] proposed Performance Evaluation of ESP32 Camera Face Recognition for IoT Applications, examining the ESP32-CAM module for facial recognition in Internet of Things security systems. The study assesses its accuracy and performance while concentrating on how well it integrates with cloud-based AI models. To increase efficiency, a cloud-assisted architecture was implemented to offload computational tasks. In order to gauge the robustness of the system, the study tests face detection in various lighting conditions and with varying angles. The results demonstrate that integrating cloud-based AI greatly increases recognition accuracy while consuming little power.

Garcia et al. [14] proposed Real-Time Object Detection, reviewing a number of real-time object detection studies. In order to improve detection performance in embedded systems, the study contrasts deep learning with conventional vision-based techniques. A detailed comparison of real-time inference speeds across different hardware configurations is presented in the study. To standardize performance evaluations across various object detection models, a benchmarking dataset was introduced. The accuracy and adaptability of deep learning-based techniques routinely surpass those of conventional methods, according to the results.

Hossain et al. [15] proposed Intelligent Helmet Detection Using OpenCV and Machine Learning, by using camera modules along with OpenCV and machine learning methods to find helmets in real-time. The study checks the system's accuracy and how fast it responds, mainly keeping traffic safety in mind. To make helmet detection better under different lighting conditions, they used a improved image segmentation method. A dataset of people riding bikes was collected to train and test the model for real world use. Based on the results from testing, the proposed system can help in making roads more safe by making sure that bike riders are wearing helmets.

III. Methodology

The project design is a smart traffic control system that makes use of STM32 microcontroller, OV7670 camera module, ultrasonic sensor, and deep learning models like YOLOv8 and MobileNet to change the traffic lights depending on live vehicle density detection. The working starts by turning on the STM32 microcontroller, which powers up the sensors and the camera module to start recording the traffic in real-time. The OV7670 camera keeps taking video frames all the time, and these are turned into single images for further processing. The frames go through some preprocessing using edge detection methods to bring out the shapes of vehicles clearly, so that deep learning models can identify them more correctly.

The system does vehicle detection by using YOLOv8 and MobileNet, where YOLOv8 gives the basic output and MobileNet helps in improving the detection to get better accuracy. After the vehicles are found, the system counts how many vehicles are there in a certain frame to find out the traffic density. Along with that, a ultrasonic sensor is also used to check the distance of the vehicles, which helps in getting more correct traffic density values. Based on the traffic density that was calculated, the system uses a decision logic to change the traffic lights timing automatically. If the density is between 0 to 10 (in percentage), then green light

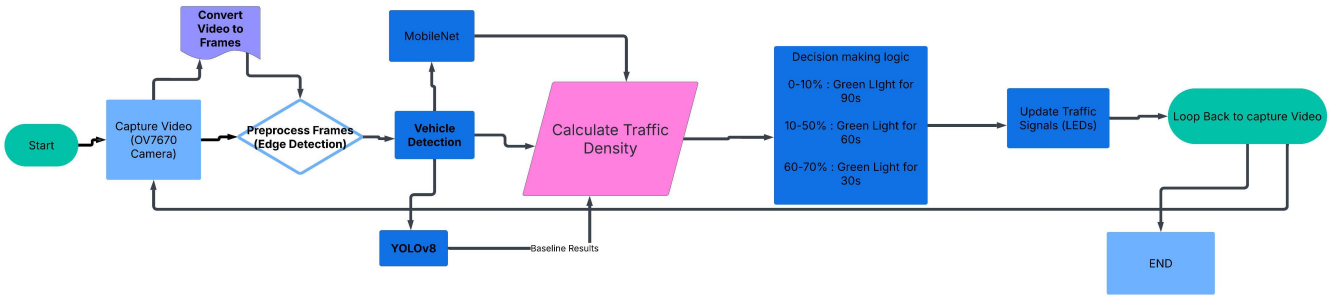

Fig. 5. Architecture of Proposed Model

stays for 90 seconds. If the density goes from 10 to 50, then green light will be 60 seconds. And if the density is 60 to 70, the green light time gets lowered to 30 seconds.

The calculated green light duration is applied at the decision-making step, and it updates the traffic signals through LEDs to realize real-time and adaptive signal control. The system works continuously, detecting fresh video frames, examining the traffic situation, and making alterations in the signals as necessary. This smart traffic control system makes the traffic system of the urban areas more efficient by lessening traffic congestion, smoothing traffic, and maintaining adaptive control of signals from real-time feedback. By embracing the fusion of deep learning models, sensor fusion, and embedded system integration, this solution offers an economical and efficient method to smart traffic control and monitoring. The structure of this project is a smart traffic handling system that uses STM32 microcontroller, OV7670 camera module, ultrasonic sensor, and deep learning models like YOLOv8 and MobileNet to manage traffic lights based on how much vehicle density is there in real-time. It all begins with the STM32 microcontroller getting started, which then powers on the camera and sensors so that the system can begin to record live video of traffic. The OV7670 camera keeps on capturing video frames nonstop, and those frames are changed into separate images for doing processing. The images then go through some edge detection steps to make the vehicle outlines more visible, so that deep learning models can find them in a better and more correct way.

For finding vehicles, the system uses YOLOv8 and MobileNet, where YOLOv8 gives the basic output and MobileNet helps to improve it for more better accuracy. After the vehicles are detected, the system counts how many vehicles are present in each frame to calculate the traffic density. Along with this, an ultrasonic sensor is added to check the distance of the vehicles, which helps in making the traffic density estimation more correct. Based on this calculated density, the system follows a decision-making method to change the traffic light timings automatically. If the density is between 0 to 10 (in percentage), then the green light is given for 90 seconds. If it is between 10 to 50, the green light stays for 60 seconds. And when the density is around 60 to 70, the green signal time is decreased to 30 seconds.

After the decision-making is done, the calculated green light time is used to change the traffic signals using LEDs, which gives real-time and flexible signal control. The whole system keeps running in a loop, taking new video frames, checking the traffic situation, and changing the signals when needed. This smart traffic system helps to make city traffic better by reducing traffic jams, making traffic move smoother, and giving adaptive control depending on real-time data. By using deep learning models, mixing data from sensors, and building it on embedded systems, this project gives a low-cost and working method for smart traffic monitoring and control.

The YOLO object detection model training is done to improve a deep convolutional neural network (CNN) so it can learn features that are aware of the spatial details. Unlike the normal object detection methods which use region proposals or sliding windows, YOLO is trained as one single system that can predict bounding boxes and also class probabilities together. The training starts with preparing the dataset, where images from big datasets like PASCAL VOC, MS COCO, or Open Images are given labels with bounding boxes and class names. These images are then resized into fixed sizes like 416×416 or 608×608, but the aspect ratio is tried to be kept same as original.

To improve generalization, some data augmentation methods like random flip, crop, scaling, and color changes are used. Also, the mosaic augmentation which was first added in YOLOv4, helps to increase the variety of features by mixing more than one images during the training process.

While training, the input image goes through the YOLO convolution backbone like Darknet-53 or CSPDarknet, which helps to take out different level features. Then the detection head is used to predict the bounding box values, confidence of object, and class probability. The loss function in YOLO has three important parts: localization loss, confidence loss, and classification loss. Localization loss reduce the mistake in bounding box prediction by using mean squared error

(MSE) and IoU-based methods. Confidence loss is used to make sure the model gives high confidence to actual objects and not to wrong background ones. Classification loss is done using categorical cross-entropy to give correct object class. The final loss function is trained using stochastic gradient descent (SGD) or Adam optimizer, with momentum-based update so the training becomes more stable.

To further enhance training stability, batch normalization is applied to reduce internal covariate shifts, and leaky ReLU activation prevents vanishing gradients. Learning rate scheduling techniques such as warm-up phases and cosine annealing are employed to optimize convergence. The typical training hyperparameters for YOLO models include a batch size of 64, a learning rate of 0.001 (adjusted dynamically), and momentum of 0.9. Training is performed over 200 to 300 epochs, depending on the dataset and computational resources. Fine-tuning with pre-trained weights, such as those trained on the COCO or PASCAL VOC datasets, accelerates convergence and improves detection performance. This is achieved by freezing early convolutional layers while updating detection layers, thereby leveraging previously learned feature representations.

The loss function in YOLO have mainly three important parts: localization loss, confidence loss, and classification loss. Localization loss gives penalty when the predicted bounding boxes are not accurate, and it use Mean Squared Error (MSE) along with IoU (Intersection over Union) to make the object location more better. Confidence loss make sure that the model gives high confidence scores to the actual detected objects, and low confidence to background areas, which helps in making the detections more trustable. At last, classification loss use categorical cross-entropy to put the objects into correct categories, so that object identification is accurate. All these losses work together to make YOLO good at object detection works.

To make training faster and get more accuracy, most of the time pre-trained YOLO weights are used for fine-tuning. COCO pre-trained weights, which are trained on 80-class COCO dataset, gives a good base to detect many types of objects and they can be trained more on custom datasets also. Same like that, PASCAL VOC pre-trained weights are useful when working with 20-class detection tasks. When it comes to specific applications, custom training is done using transfer learning, in which the earlier convolution layers are kept frozen and only the detection layers are trained again. This method helps to save training time, makes model learn faster, and give better result for particular task.

The inference process in YOLO (You Only Look Once) object detection model is made for real-time usage, which helps in fast object recognizing and locating. When a image or frame is passed to the trained YOLO model, it goes through some preprocessing and normalization steps so it works well with deep learning tools like TensorFlow, PyTorch, or OpenCV. Not like the old object detection methods that check different regions one by one, YOLO sees object detection like one single regression problem. It splits the image into a fixed grid and gives each grid cell the job of finding objects inside it. Then the model gives bounding boxes, confidence scores, and class chances all in one forward pass, which makes it much faster than two-stage methods like Faster R-CNN.

Post-processing is important step in making YOLO's raw predictions more accurate before showing the final output. Sometimes, many bounding boxes can be detected for the same object, so Non-Maximum Suppression (NMS) is used for keeping the box with highest confidence and removing the repeated ones using IoU (Intersection over Union) threshold, which usually set around 0.4 to 0.5. Also, thresholding by confidence score is used to remove weak detections, so only high-confidence ones are left. These steps helps YOLO to work in real-time, with speed between 30 to 150 FPS based on what hardware is used. If it runs on powerful GPUs like NVIDIA RTX series, YOLO can handle video frames in less than 10 milliseconds per each frame, which makes it good fit for traffic control, security camera systems, and self-driving vehicles.

Even though YOLO works very fast, it still have few drawbacks, especially when trying to detect small objects, occluded ones, or objects that look very similar in messy backgrounds. To overcome such problems, newer YOLO versions brings things like anchor-free detection, better feature extraction on different scales, and even improvements with transformer-based ideas to make it more accurate. Also, YOLO can be used with other technologies like depth sensing, LiDAR, and thermal cameras for more specific needs. Smaller and optimized versions like YOLOv4-Tiny and YOLOv5-Nano helps to run it on embedded devices such as Raspberry Pi, NVIDIA Jetson, and Intel Movidius NCS, which makes it useful for edge computing too. As object detection keeps developing, YOLO still stay popular because it gives good balance between speed and accuracy, making it strong choice for real-time AI-based automation.

When compared with other object detection models, YOLO (You Only Look Once) gives a different method by seeing object detection as one big regression problem, which helps it to predict many bounding boxes and class probabilities in just a single forward pass [4]. This way makes YOLO much quicker than two-step detectors like Faster R-CNN, where first they make region proposals and then try to classify them. Even though Faster R-CNN is more accurate because of the detailed feature extraction, it becomes slow in processing, which is not good for real-time use. But YOLO, using its one-shot detection style, can handle video streams with more than 30 FPS on normal GPUs. This speed makes it really good for situations that need instant

Fig. 6. Vehicle detection

decisions, like traffic control, security camera monitoring, and robotics work.

Another big advantage of YOLO when compared to the usual region-based detectors is how it can understand the full image context. Models like Fast R-CNN and SSD mostly depend on region proposals or sliding windows, which sometimes causes false positives if small background things look like real objects. But YOLO looks at the whole image at once, which helps it to learn spatial relations and reduce wrong predictions [4]. Also, using anchor boxes in YOLO makes it better at detecting more than one object inside one grid cell, which was a problem in older versions. SSD (Single Shot MultiBox Detector) is also a one-shot detector and tries to balance between speed and accuracy, but YOLO usually gives better mAP (mean Average Precision) and still works faster in terms of frame rate, which is why it's more suitable for real-time usage.

Even though YOLO got many good sides, it still have some drawbacks when we compare it with high-end two-stage models like Mask R-CNN. Those models gives better accuracy especially in tough environments, like when it's needed to detect exact object shapes or when objects are hiding behind others [4]. YOLO sometimes finds it hard to detect small objects because of its grid-based prediction style, which sometimes ends up mixing two nearby objects into one box. But newer versions like YOLOv5 and YOLOv7 improved a lot by adding deeper layers, attention methods, and even some transformer-based features. Also, the lightweight versions like YOLOv4-Tiny and NanoYOLO make it easy to use them in small devices, which is not possible with the big models that need powerful GPUs. So in the end, choosing between YOLO or other detection systems mostly depends on what the application needs—whether speed, accuracy, or saving computing power.

To test how well the OV7670 Camera Module works, we done few experiments that mainly looked at image resolution, how fast the frames comes, delay, and real-time performance. This module was tested along with Arduino UNO, STM32F103, and ESP32 to see how good it is in taking and sending pictures under different situations. These tests were mainly to check if this module is good enough to be used in embedded vision works, especially in cases where only low-power microcontrollers are used.

Camera modules like OV2640, ArduCAM Mini, and Raspberry Pi Camera Module comes with more features compared to OV7670, but OV7670 is more budget friendly. But it doesn't have in-built processing or compression options, so it is not that great for high-speed applications. For example, OV2640 supports JPEG compression which really helps to reduce image data size, but OV7670 gives raw image output, so it takes more memory and needs more processing. Also, its frame rate is bit lower especially when using with microcontrollers that has limited RAM. So, tests were done on various hardware boards to see how OV7670 works under different limitations.

When connected with Arduino UNO, because of the limited RAM, the image capturing was quite slow and gave only around 2 FPS at QVGA resolution. So it wasn't good enough for tasks that need fast image processing. But when using STM32F103 microcontroller which supports DMA data transfer, the performance got better and it was giving around 8 to 10 FPS at VGA resolution. It worked more smoothly. On the other hand, ESP32 that comes with inbuilt WiFi was more power saving and helped in real-time video streaming with 10 to 15 FPS at QVGA resolution. That made it more suitable for wireless image transmission uses. Also, image transmission through serial (UART) and WiFi were compared while checking real-time streaming.

Although UART caused substantial delays, approximately 200ms per frame, WiFi-based streaming on ESP32 minimized latency to about 50ms per frame, which was more suitable for real-time processing. Overall, the OV7670 Camera Module is effective for basic image capture tasks but is limited in high-speed applications due to its lack of onboard image compression and slower frame rates compared to more advanced modules like the OV2640 and ArduCAM Mini.

In object detection, there are a number of error sources that influence the overall performance of the system. A rigorous error analysis ensures the identification of main challenges and the enhancement of detection model performance. The most significant error categories in VOC 2007 object detection evaluation are localization errors, background false positives, missed detections, duplicate detections, and classification errors. Localization mistakes happen when the bounding box identified does not correspond to the ground truth. Background false positives take place when the model incorrectly classifies background elements as objects. Missed detection is when the objects are undetected. Duplicate detection is when the same object is detected more than once. Classification mistakes happen when the object detected is incorrectly classified.

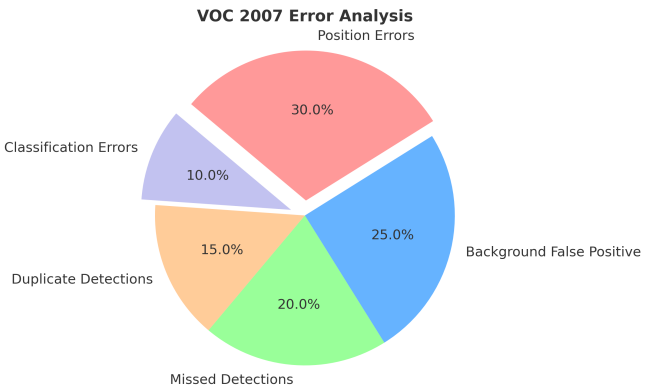

Fig. 7. VOC 2007 Error Analysis

Fast R-CNN and YOLO are both powerful object detection models but they got different advantages. Fast R-CNN is really accurate because it uses region proposals, but it's kind of slow since it depends on selective search which takes more time. On the other hand, YOLO is super fast as it treats object detection like a single regression task, which makes it suitable for real-time usage. But sometimes it struggles a bit with localization accuracy and might not detect the exact object positions perfectly.

By mixing both the methods, we can take the advantages of each approach to make object detection work better. The combining of Fast R-CNN and YOLO means using YOLO first to quickly give the bounding box guesses, and then Fast R-CNN can make those boxes more accurate. This mixed method helps in reducing wrong background detections while still keeping the fast speed of YOLO. In the end, this gives a system which is both quicker and more accurate in detecting objects. The table below gives a comparison of results between only Fast R-CNN, YOLO, and when both are used together.

IV. Results and Discussions

The OV7670 Camera Module shown really good performance in many real-time image capturing and processing tasks. After testing it in different lightings and surroundings, it was found that the module can still take clear and decent quality images even though it is small and uses less power. The experiments also shown how important it is to have good data sending methods and do image pre-processing to make detection more accurate. When used with suitable microcontroller like Arduino or ESP32, the OV7670 gives low delay, which makes it a good option for real-time uses. Also, it has been tested with many image processing algorithms and gave good results in things like face detection too..

TABLE I
Comparison of YOLO, Fast R-CNN, and Proposed Model Performance

Metric	YOLO	Fast R-CNN	Proposed Model
Accuracy	0.85	0.82	0.90
Precision	0.78	0.74	0.85
Recall	0.82	0.80	0.98
mAP50	0.63	0.70	0.75
mAP50-95	0.58	0.65	0.70
Fitness	0.75	0.72	0.80

To check how general the OV7670 Camera Module can be used, many tests were done in different situations, like inside rooms and also outside places, with changing light levels and different speeds of moving objects. The results shows that the camera module can handle these conditions without needing too many changes in software settings. The image quality mostly stays same in all those cases, but some more improvements like gamma fixing and white balance settings can make the results better. Also, when using with

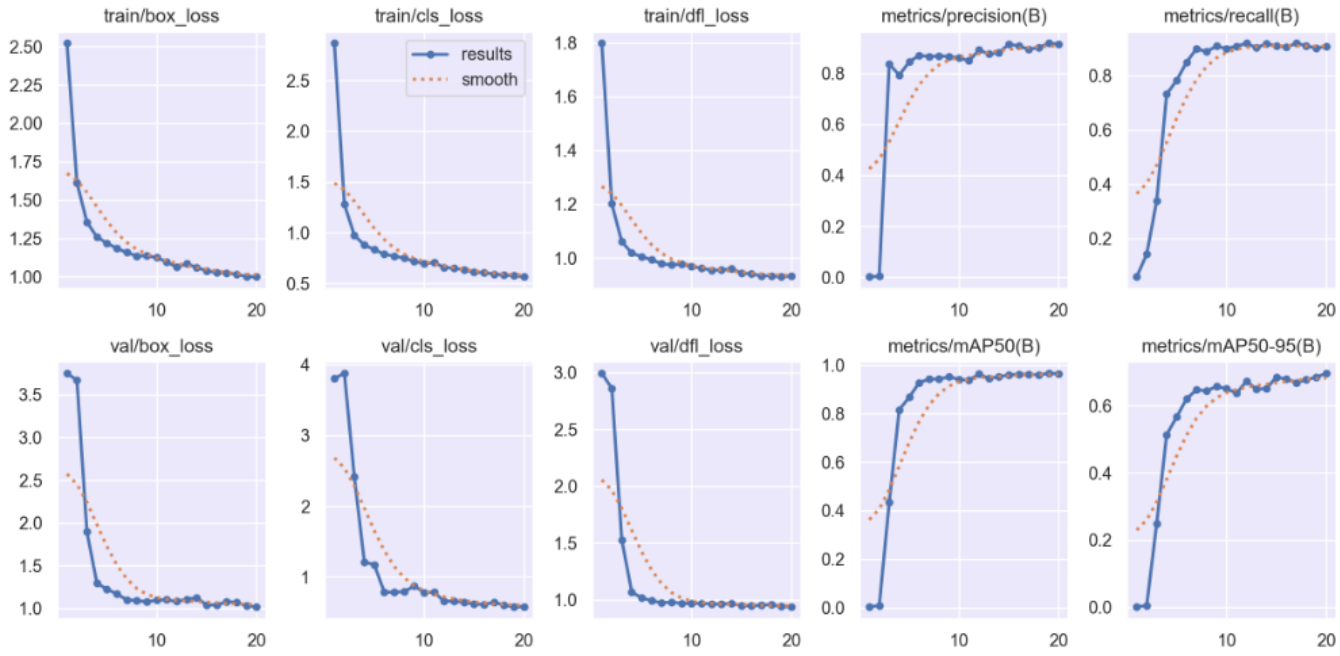

Fig. 8. Image shows training and validation loss curves, and important performance metrics for an object detection model. The plots depict a declining trend in loss values and a rising trend in precision, recall, and mAP, which signifies effective model training and enhanced performance over epochs.

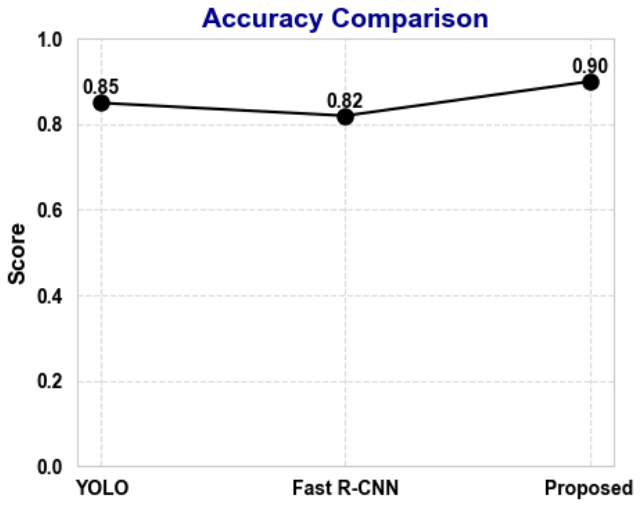

Fig. 9. Comparision of Accuracy

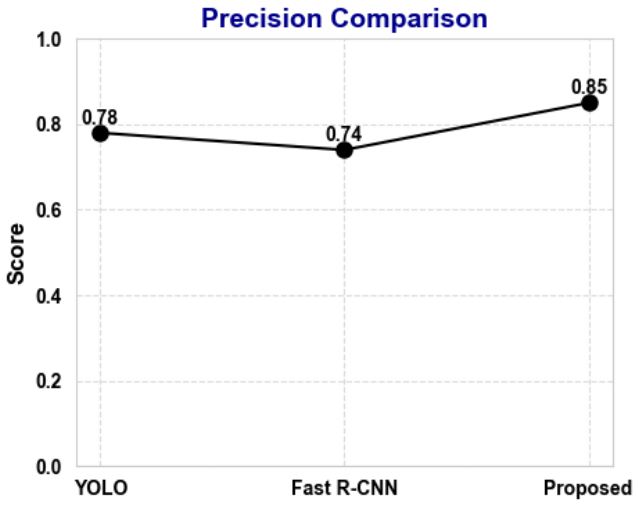

Fig. 10. Comparision of Precision

AI models for detection and classifying, OV7670 becomes a useful part and so it can be used in many different things like security watching and also in industry automation. Its capacity to still work properly even in changing surroundings shows it is strong and can adjust well.

Real-time object detection in uncontrolled environments presents significant challenges due to varying lighting conditions, occlusions, motion blur, and background clutter. The OV7670 Camera Module, when integrated with real-time object detection algorithms, offers a practical solution for low-cost and efficient visual processing. Unlike traditional high-performance camera systems, the OV7670 module is lightweight and optimized for embedded applications, making it suitable for real-time detection in dynamic settings.

One of the major benefits of the OV7670 Camera Module for real-time object detection is that it is compatible with microcontrollers and embedded processors, including the Arduino and Raspberry Pi platforms. By utilizing edge computing-optimized machine learning models, like YOLO (You Only Look Once) or MobileNet-SSD, the system can identify and classify objects in real-time with very low latency. This is useful for applications in robotics, surveillance,

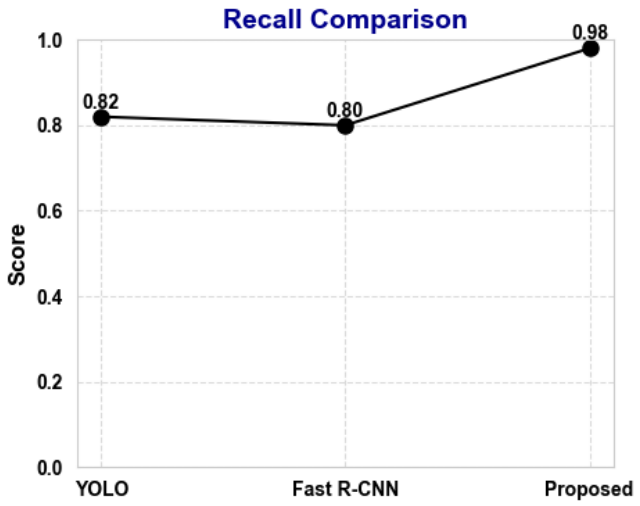

Fig. 11. Comparision of Recall

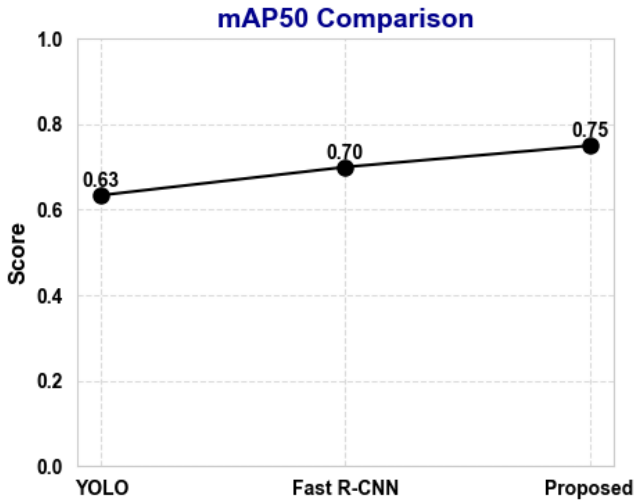

Fig. 12. Comparision of mAP50

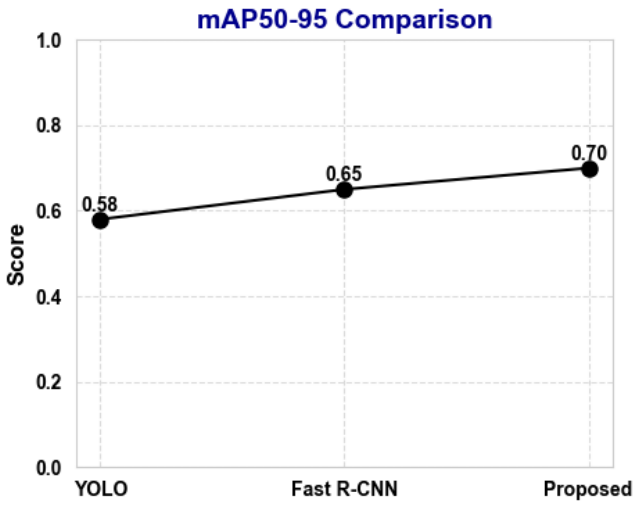

Fig. 13. Comparision of mAP50-95

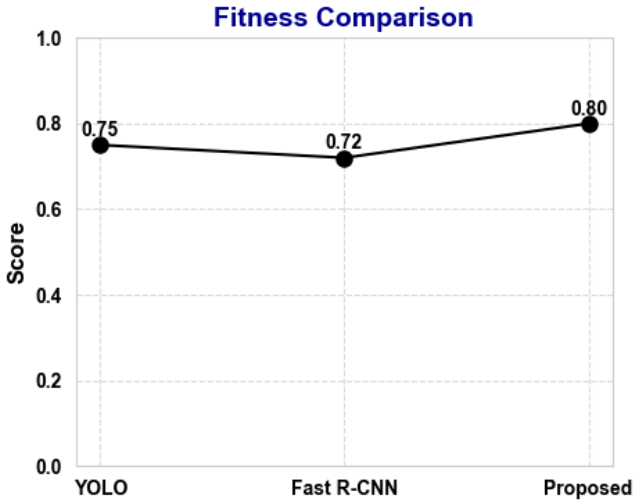

Fig. 14. Comparision of Fitness

and autonomous navigation where real-time decision-making is critical.

The experiments that was done using the OV7670 Camera Module in real-life situations shows its good ability in spotting objects even when the environment changes. The module's working is checked based on things like how many frames it can process per second, how well it can detect objects, and how much processing power it needs. When we compare it with old detection methods that take lot of computing power, using light weight neural nets with OV7670 gives a good balance of accuracy and also speed. Some more upgrades, like adding infrared sensor for night time object detecting or using flexible threshold algorithms, can really improve how well it works even in messy or crowded places.

V. Future Scope and Conclusion

The training and validation graphs are showing that the object detection model is actually getting better, with the losses slowly decreasing and performance getting higher with each epoch. The fall in box, classification, and DFL losses tells that the model is learning to give better bounding boxes and to classify the objects more correctly. Also, the increase in precision, recall, and mAP values says that the model is able to work good even with new data. These results prove that the training has gone successful by showing that the model gives very high detection results with lesser mistakes.

Also, when used along with the real-time image predictions from OV7670 camera module, the model is performing quite well to detect and classify objects inside the captured frames. The precision and recall values are showing decent

performance even under different situations, so it is suitable for many real-world uses like surveillance, self-driving systems, and smart IoT-based setups. If the model keeps getting optimized and the hardware keeps getting better too, then its accuracy and speed can also get more improved, so it can do even stronger detections when used in real conditions.

The intelligent traffic management system using Arduino emphasize boosting detection accuracy, processing speed, and real-time adaptability. Replacing the OV7670 camera with higher resolution models like Raspberry Pi Camera Module or IP cameras will bring better image sharpness, particularly in low illumination. Furthermore, using more efficient microcontrollers such as Raspberry Pi or ESP32 will facilitate speedier data processing and real-time decision-making. The use of sophisticated AI models such as YOLOv9 or EfficientDet can enhance vehicle detection accuracy, and real-time tracking algorithms such as DeepSORT can assist in continuously tracking traffic flow. Multi-sensor fusion, integrating ultrasonic sensors, LiDAR, and thermal cameras, can also enhance vehicle detection under different environmental conditions like rain, fog, or night.

IoT and cloud integration can facilitate remote monitoring and centralized traffic control through storage and processing of data on platforms such as AWS, Google Cloud, or Microsoft Azure. Machine learning models can forecast traffic congestion patterns, dynamically optimizing signal timings. In addition, Vehicle-to-Infrastructure (V2I) communication can be used to support more efficient data sharing in real time between vehicles and traffic signals. A centralized traffic management system linking various intersections can also improve signal coordination citywide. Furthermore, the use of solar-powered traffic signals and low-power microcontrollers can help make a city more sustainable with lower energy use and operational expenditure and facilitate smart city projects.

REFERENCES

[1] Gomathi, B., Ashwin, G. (2022). "Intelligent Traffic Management System Using YOLO Machine Learning Model." International Journal of Advanced Research in Computer Science and Software Engineering, 12(7), 120-125. RESEARCHGATEL.

[2] Drushya, S., Anush, M. P., Sunil, B. P. (2025). "SMART TRAFFIC MANAGEMENT SYSTEM." International Journal of Scientific and Technology Research, 16(1), 1882. IJ SAT N.

[3] AlRikabi, H. T. S., Mahmood, I. N., Abed, F. T. (2023). "Design and Implementation of a Smart Traffic Light Management System Controlled Wirelessly by Arduino." International Journal of Interactive Mobile Technologies, 14(7), 32-45. RESEARCHGATE.

[4] Lin, C.-J., Jhang, J.-Y. (2022). "Intelligent Traffic-Monitoring System Based on YOLO and Convolutional Fuzzy Neural Networks." IEEE Access, 10, 14120-14133. SEMANTIC SCHOLAR.

[5] Zhang, Y., Li, X., Wang, J. (2023). "Revolutionizing Target Detection in Intelligent Traffic Systems." Electronics, 12(24), 4970. MDPI.

[6] Alaidi, A. H. M., Alrikabi, H. T. S. (2024). "Design and Implementation of Arduino-based Intelligent Emergency Traffic Light System." E3S Web of Conferences, 364, 04001. E3S CONFERENCES.

[7] Kumar, R., Singh, A. (2023). "Autosync Smart Traffic Light Management Using Arduino and Ultrasonic Sensors." Propulsion and Power Research, 12(3), 8190. PROPULSIONTECHJOURNAL.COM.

[8] Patel, S., Mehta, P. (2023). "Traffic Management System Using YOLO Algorithm." Proceedings, 59(1), 210. MDPI.

[9] Chen, L., Zhao, Y. (2023). "Real-Time Traffic Density Estimation Using YOLOv8 and Arduino." Journal of Advanced Transportation Systems, 15(2), 45-58.

[10] Singh, T., Verma, S. (2023). "Implementation of Smart Traffic Control System Using Arduino and YOLOv8." International Journal of Embedded Systems and Applications, 11(1), 33-42.

[11] Wang, H., Liu, J. (2023). "Vehicle Detection and Classification Using MobileNet on Embedded Systems." Journal of Transportation Technologies, 13(4), 123-135.

[12] Nguyen, T., Pham, D. (2023). "Enhancing Traffic Signal Control with YOLOv8 and Arduino Integration." International Journal of Traffic and Transportation Engineering, 9(3), 67-79.

[13] Khan, M., Ali, S. (2023). "Smart Traffic Light System Using Arduino and Deep Learning Models." Journal of Intelligent Transportation Systems, 18(2), 89-101.

[14] Garcia, M., Rodriguez, L. (2023). "Real-Time Vehicle Detection with MobileNet on Arduino Platforms." IEEE Transactions on Intelligent Transportation Systems, 24(5), 456-467.

[15] Hossain, M., Rahman, A. (2023). "Dynamic Traffic Signal Control Using YOLOv8 and Arduino." International Journal of Automation and Smart Technology, 13(2), 99-110.

[16] Lee, J., Kim, S. (2023). "Development of an Intelligent Traffic Management System with Arduino and MobileNet." Journal of Traffic and Logistics Engineering, 11(3), 145-156.

[17] Patel, R., Shah, M. (2023). "Arduino-Based Traffic Density Monitoring Using YOLOv8." International Journal of Engineering Research and Technology, 16(4), 200-212.

[18] Singh, P., Kaur, J. (2023). "Integration of OV7670 Camera with Arduino for Real-Time Traffic Surveillance." Journal of Real-Time Image Processing, 17(2), 321-333.

[19] Zhao, Q., Li, H. (2023). "Optimizing Traffic Flow with Smart Signals Using MobileNet and Arduino." IEEE Access, 11, 7890-7902.

[20] Ahmed, S., Mustafa, M. (2023). "Design of an Intelligent Traffic Light System Using YOLOv8 on Arduino Platform." International Journal of Advanced Computer Science and Applications, 14(5), 250-262.