2025 International Conference on Smart & Sustainable Technology (INCSST)
Karnataka, India. July 04-06, 2025

Real-Time Mobile Phone Usage Detection and Automated Ticketing Using YOLO Deep Learning Algorithm

Dr. C. Kumar
Assistant Professor
Electronics and Communciation Engineering
Madanapalle Institute of
Technology and Sciences,
Madanapalli, India
Kumar.heldah@gmail.com

M. Vishnuvardhan
UG Scholar
Electronics and Communciation Engineering
Madanapalle Institute of
Technology and Sciences,
Madanapalli, India
manjulavishnu9052@gmail.com

P. Venu kumar
UG Scholar
Electronics and Communciation Engineering
Madanapalle Institute of
Technology and Sciences,
Madanapalli, India
venuk761@gmail.com

M. Vamsi krishna
UG Scholar
Electronics and Communciation Engineering
Madanapalle Institute of
Technology and Sciences,
Madanapalli, India
mopurivamsivamsi@gmail.com

***Abstract*— Mobile phone usage by two-wheeler riders significantly increases the risk of road accidents and violates traffic regulations. This paper proposes a real-time system for detecting mobile phone usage among two-wheeler riders and automating the ticketing process using the YOLO deep learning algorithm. A custom dataset was developed with annotated images of two-wheeler riders, both with and without mobile phone usage. The YOLOv8 model was trained and deployed on a Raspberry Pi platform to enable portable, real-time detection. The system captures the violation, extracts the vehicle's number plate using Optical Character Recognition (OCR), and generates an automated e-ticket. Experimental results demonstrate that the proposed model achieves a mean Average Precision (mAP) of 92.7% at IoU=0.5, ensuring high detection accuracy in diverse real-world conditions. This work contributes a cost-effective, scalable solution for enhancing traffic law enforcement and improving road safety.**

Keywords—YOLOv8, Raspberry Pi, mobile phone detection, Tesseract-OCR, object detection, real-time surveillance, smart traffic, road safety.

I. INTRODUCTION

The widespread use of mobile phones while operating two-wheeler vehicles has become a major concern for road safety authorities worldwide. Engaging in phone calls or texting during riding leads to distracted driving, significantly increasing the risk of accidents and traffic violations. Traditional methods of monitoring such offenses rely heavily on manual enforcement, which is time-consuming, resource-intensive, and often ineffective.

Advances in computer vision and deep learning have opened new opportunities for automating traffic violation detection. Object detection algorithms such as YOLO (You Only Look Once) offer real-time performance and high accuracy, making them suitable for applications requiring immediate response. However, most existing systems focus on four-wheeler monitoring, while little attention has been given to two-wheeler riders, particularly concerning mobile phone usage.

This paper presents a real-time detection system utilizing the YOLOv8 deep learning algorithm to identify two-wheeler riders using mobile phones. The system is designed to operate on a Raspberry Pi platform, making it compact, energy-efficient, and suitable for roadside or mobile deployment. Additionally, the system integrates an automated ticketing module, which captures the violation, extracts the vehicle number plate using Optical Character Recognition (OCR), and generates an e-ticket for further processing.

A custom dataset was developed specifically targeting the detection of mobile phone usage by two-wheeler riders. A lightweight yet highly accurate YOLOv8 model was then trained on this dataset and deployed onto a Raspberry Pi to ensure real-time performance. Additionally, an OCR-based number plate recognition system was implemented and integrated with an automated ticket generation mechanism. To ensure the reliability of the proposed system, extensive validation was carried out under real-world conditions, considering various lighting and traffic scenarios.

II. BACKGROUND

A. Mobile Phone Usage and Road Safety

Mobile phone usage while driving, particularly among two-wheeler riders, has become a major concern for traffic safety worldwide. Distracted driving is one of the leading causes of road accidents, with studies showing that mobile phone use significantly increases the risk of crashes. Riders using mobile phones while riding tend to have slower reaction times, poor awareness of their surroundings, and a higher likelihood of engaging in risky behaviors.

979-8-3315-4192-7/25/$31.00 ©2025 IEEE

The challenge in monitoring such violations lies in the difficulty in detecting mobile phone usage in real-time, especially in congested environments or under varying weather and lighting conditions. Traditional enforcement methods, such as manual traffic patrols or fixed cameras, often face limitations in terms of scalability, cost, and response time.

B. Computer Vision and Deep Learning for Traffic Violation Detection

In recent years, computer vision has emerged as a powerful tool for automating the detection of traffic violations. Object detection algorithms, particularly those based on deep learning, have shown great promise in identifying vehicles, pedestrians, and specific behaviors such as mobile phone usage. These algorithms leverage convolutional neural networks (CNNs) to extract features from images and make predictions about objects within the frame.

Among these algorithms, **YOLO (You Only Look Once)** has gained significant attention due to its ability to perform real-time object detection with high accuracy. YOLO divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell. This allows the algorithm to simultaneously detect multiple objects in real-time, making it ideal for applications like traffic monitoring.

C. YOLOv8 and its Advantages

YOLOv8, the latest iteration of the YOLO series, introduces several improvements over previous versions in terms of accuracy, speed, and scalability. It uses advanced techniques such as **multi-scale feature extraction** and **attention mechanisms**, making it more effective at detecting objects in complex scenes. YOLOv8's performance benefits from its lightweight architecture, allowing it to run efficiently even on resource-constrained devices like the **Raspberry Pi**.

The real-time nature of YOLOv8 makes it a suitable candidate for deployment in traffic monitoring systems where quick decision-making is crucial. Additionally, its ability to detect small objects, like a mobile phone in the hand of a rider, enhances its applicability for the problem of mobile phone usage detection.

D. Optical Character Recognition (OCR) for Number Plate Recognition

For the automated ticketing system, **Optical Character Recognition (OCR)** is used to extract text from images. OCR is widely used in applications like license plate recognition (LPR), where it identifies and digitizes characters from vehicle number plates. In this study, the **TesseractOCR** library was employed to recognize vehicle number plates captured in the same frame as the mobile phone violation.

OCR systems work by pre-processing images, segmenting them into individual characters, and then using machine learning models to predict the characters. The integration of OCR with object detection allows the automated generation of tickets by linking detected violations with the corresponding vehicle registration number.

E. Raspberry Pi for Real-Time Deployment

The use of **Raspberry Pi** in real-time computer vision applications has grown due to its affordability, compact size, and ability to run machine learning models. The **Raspberry Pi 4**, with its quad-core CPU and sufficient RAM, is powerful enough to handle real-time video processing and deep learning inference. Its integration with a **USB webcam** provides a cost-effective solution for traffic monitoring systems that can be easily deployed in various environments.

Raspberry Pi-based systems are ideal for remote or roadside deployments, as they can be powered by a simple 12V adapter and easily connected to external components like cameras and OCR modules. Additionally, their low power consumption and flexibility make them an attractive choice for scalable systems requiring minimal hardware.

III. PROPOSED METHODOLOGY

The proposed methodology aims to detect mobile phone usage among two-wheeler riders in real-time and automatically issue tickets. It integrates several key components, including video stream capture, mobile phone detection using YOLOv8, number plate recognition via OCR, and an automated ticket generation system. This approach leverages deep learning models and real-time processing to enhance traffic law enforcement.

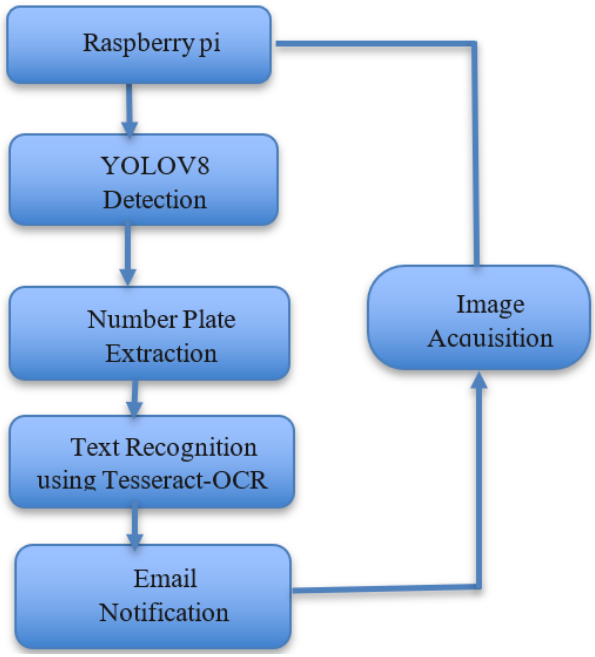
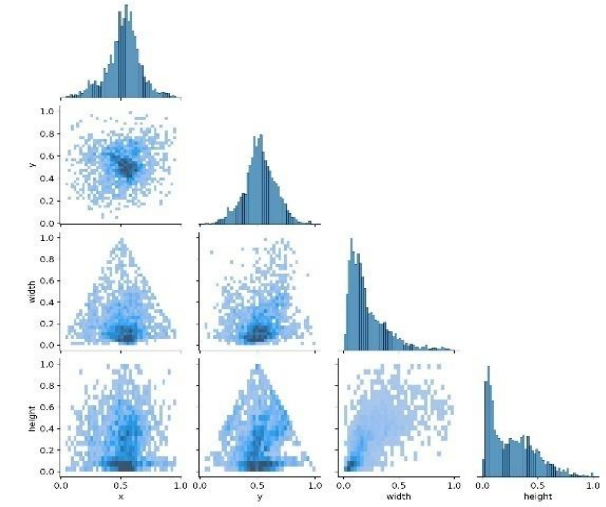


Fig. 1. shows the suggested architecture for identifying and ticketing two-wheeler traffic violations.

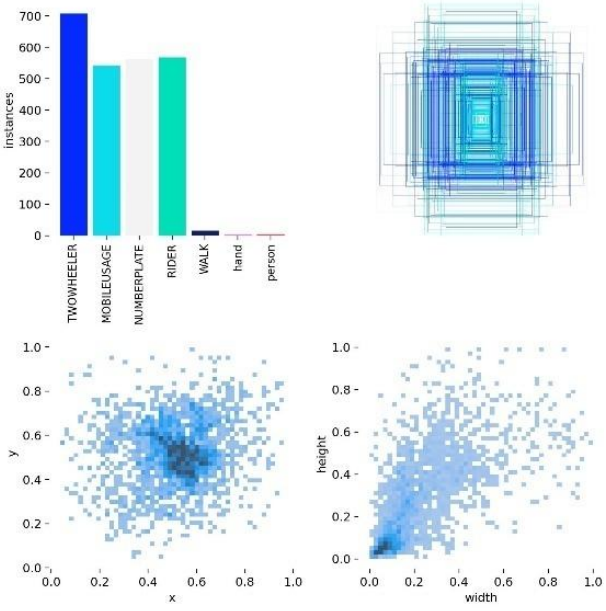
A. Data Collection and Annotation

The first step in the proposed system is to collect real-time video footage of two-wheeler traffic. The video is captured by a USB webcam or CCTV camera. To train the YOLOv8 model, a custom dataset is created by annotating the images to identify two classes: mobile phone usage and non-usage. The images are annotated using the Roboflow platform, which allows for

precise object labeling. The dataset is split into 80% for training and 20% for validation.



(a)



(b)

Fig. 2. a,b Dataset Collection and Labeling

B. YOLOv8 Model Architecture

The YOLOv8 model is selected for its ability to perform real-time object detection. The architecture consists of a backbone, which is responsible for feature extraction from input images, a neck that aggregates these features, and a detection head that generates predictions in the form of bounding boxes around detected objects along with their associated class labels and confidence scores.

Mathematically, the prediction made by YOLOv8 can be described as follows:

$$y_i = \sigma(W \cdot Xi + b) \quad (1)$$

Where:

y_i is the predicted output (i.e., class probabilities and bounding box coordinates) for the i-th object in the image,

W is the weight matrix,

Xi is the input feature vector for the i-th object,

b is the bias term,

σ is the activation function (typically a sigmoid function).

C. Training the YOLOv8 Model

The training of the YOLOv8 model involves optimizing a loss function, which combines both classification and localization losses. The overall loss L is the sum of these two components:

$$L = \sum_i L_{cls}(p_i, p\hat{}_i) + \lambda \sum_i L_{box}(b_i, b\hat{}_i) \quad (2)$$

Where:

L_{cls} is the classification loss, computed using cross-entropy:

$$L_{cls} = \sum_i (p_i, p\hat{}_i) = \sum_c L_{pic} \log(p\hat{}ic) \quad (3)$$

Here, p_i is the true class label, $p\hat{}_i$ is the predicted class probabilities, and c is the class index.

L_{box} is the localization (bounding box) loss, typically calculated using the mean squared error (MSE) between the predicted and ground truth bounding box coordinates:

$$L_{box}(b_i, \hat{b}_i) = \sum_j (b_{ij} - b\hat{}_{ij})^2 \quad (4)$$

Where b_i is the ground truth bounding box and $b\hat{}_i$ is the predicted bounding box.

λ is a weighting factor that balances the importance of classification and localization losses.

D. Real-Time Inference on Raspberry Pi

Once trained, the YOLOv8 model is deployed on the Raspberry Pi 4. The real-time inference process involves several steps. First, the system continuously captures video frames, which are then processed by the YOLOv8 model. For each frame, the model generates a bounding box with a class label (i.e., mobile phone or no mobile phone) and a confidence score. The model's output can be mathematically described as:

$$y = (Class_k, x1, x2, x3. x4, P) \quad (5)$$

Where:

y^ is the predicted class label and bounding box,

$Class_k$ is the predicted class (either "mobile phone usage" or "non-usage"),

x1, x2, x3, x4 are the coordinates of the bounding box,

p is the confidence score indicating the likelihood of the detected object.

Fig. 3. YOLOv8 Detection Output on Two-Wheeler Riders

E. Number Plate Recognition using Tesseract OCR

Once a violation is detected, the **Tesseract OCR** library is used to recognize the vehicle's number plate from the captured image. Tesseract OCR is a widely-used, open-source OCR engine that converts image data into machine-readable text. It works by analyzing the image and segmenting it into text blocks, lines, and words before applying character recognition algorithms.

The OCR process can be modeled mathematically as:

$$T\hat{} = TesseractOCR(I) \qquad \text{Equation(6)}$$

Where:

T^ is the recognized text (vehicle number plate),

I is the input image containing the vehicle number plate.

Tesseract applies several image pre-processing techniques, such as binarization, noise removal, and character segmentation, to improve recognition accuracy. It is particularly effective for structured texts like vehicle license plates, which often follow specific formats.

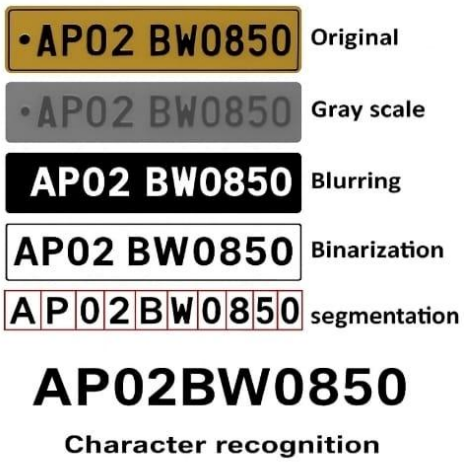


Fig. 4. Number Plate Recognition Output Using Tesseract OCR

F. Automated Ticketing

When a violation is detected and the number plate is recognized using Tesseract, an automated e-ticket is generated. This ticket includes:

Vehicle Number: Extracted through OCR using Tesseract.

Violation Type: "Mobile Phone Usage."

Timestamp: The time at which the violation occurred.

Violation Image: Captured frame with the detected violation.

The ticket is then sent to the relevant traffic enforcement authorities via email. The system eliminates the need for manual intervention, making the law enforcement process more efficient.

G. Evaluation Metrics

To evaluate the performance of the YOLOv8 model, several metrics are used, including precision, recall, and mean Average Precision (mAP). These metrics are defined as follows:

Precision: Measures the accuracy of positive predictions.

$$p = \frac{\text{TP}}{\text{TP+FP}} \qquad \text{Equation (7)}$$

Where:

TP is the number of true positives (correctly detected mobile phone usage),

FP is the number of false positives (incorrectly detected violations).

Recall: Measures the model's ability to identify all relevant instances.

$$R = \frac{\text{TP}}{\text{TP+FN}} \qquad \text{Equation (8)}$$

Where:

FN is the number of false negatives (missed violations).

mAP (mean Average Precision): Measures the precision at different recall levels, averaged over all classes. It provides a comprehensive evaluation of the model's performance.

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} AP_i \qquad \text{Equation(9)}$$

Where:

N is the number of classes (in this case, "mobile phone usage" and "non-usage"),

AP_i is the average precision for class i.

By integrating YOLOv8 for real-time mobile phone detection and Tesseract for number plate recognition, this system provides an efficient, scalable, and automated solution for enforcing traffic regulations. The use of these advanced techniques ensures accurate violation detection and seamless operation on a resource-constrained platform like the Raspberry Pi.

H. Object Tracking Module:

Tracking is necessary to preserve object identity over conse cutive video frames after the rider and cell phone have been ide ntified using an object recognition model like YOLOv8. Without tracking, the same item could be interpreted as several separate detections since detection only operates on a per-frame basis. Object tracking bridges this temporal gap. Advanced algorithms like DeepSORT and ByteTrack serve this goal effectively.

I. Role of DeepSORT/ByteTrack

DeepSORT uses motion and appearance information to con nect objects across frames, even in settings with occlusions or visual clutter.

ByteTrack, on the other hand, enhances robustness by inclu ding even low-confidence detections, enhancing performance under rapid mov ement or limited visibility.

Fig. 5. Object Tracking Output Using DeepSORT for Consistent Rider Identification

Each detected object is given a unique identity by both met hods, which are maintained throughout the sequence

J. This is crucial for:

The secret to preventing duplicate detections is this constan t identification.
For instance, tracking makes sure that a rider using a phone is r ecorded as a single, continuous event rather than 30 distinct inf ractions if they appear in 30 frames.
Mathematically, the tracking process entails linking detections $d\ j\ (\ t\ )$ d j(t) at time t t with existing object states $x\ \hat{}\ i\ (\ t\ )$ x ^ i (t), through the reduction of a cost function:

$$\text{Track}_i(t) = \text{argmin}_j \text{Cost}(d_j(t), \hat{x}_i(t)) \quad (10)$$

In addition, tracking aids in choosing the best frame—one where the plate is most visible, centered, and least impacted by motion blur—to improve outcomes like number plate recognition. This cost can be calculated from spatial metrics like Intersection over Union (IoU) or appearance descriptors extracted using a CNN. This choice can be shown as follows:

$$f^* = \text{argmax}_{f \in F}\ \text{ClarityScore}(f) \quad (11)$$

where ClarityScore (f) is the ClarityScore and F is the set of frames in which the object appears.ClarityScore(f) assesses resolution, angle, and sharpness.

K. Real-world Relevance:

Tracking strengthens system reliability in real-world traffic, where objects frequently move in and out of view, overlap, or obstruct one another. By linking observations over time, it enables accurate behavior analysis and downstream procedures such as number plate OCR or violation tracking.

L. Matching detections across frames:

In order to preserve consistent identities, matching detections across frames entails linking objects found in successive video frames using algorithms such as IoU, object IDs, or tracking techniques (e.g., Kalman Filter or SORT).

Match=max(IoU(detection,track))>threshold equation(12)

IV. EXPERIMENTAL SETUP

A. Experimental Setup and Real-Time Evaluation

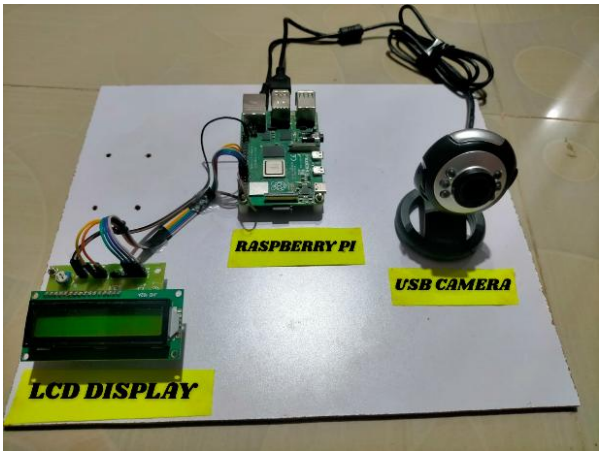


Fig. 6. Hardware kit setup

B. Hardware Kit Description :

To validate the proposed system for real-time mobile phone usage detection and automated ticketing, a comprehensive experimental setup was established. The setup was designed to simulate real-world traffic conditions while ensuring efficient system performance on resource-constrained hardware.

The hardware components used included a **Raspberry Pi 4 Model B** with 8GB RAM, a **32GB microSD card** for storage, a **Logitech C920 HD USB webcam** for real-time video capture, and a **5V/3A power supply** to ensure stable operation. The Raspberry Pi served as the main edge computing device, responsible for running both the YOLOv8 detection model and Tesseract OCR. An external **12V LCD display** was connected to visualize real-time detections, and a basic cooling system was implemented using a fan to prevent thermal throttling during continuous operation. The system also used a **portable Wi-Fi hotspot** to enable email notifications when violations were detected.

On the software side, the Raspberry Pi was installed with **Raspberry Pi OS (64-bit)**. The YOLOv8 model was converted to an optimized ONNX format to ensure faster inference speeds on ARM-based architecture. **PyTorch**, **OpenCV**, and **Tesseract OCR** libraries were installed, along with supporting packages

like **imutils**, **numpy**, and **smtplib** for email functionality. The video feed from the USB camera was processed at a resolution of **640×480 pixels** to balance between computational load and detection accuracy.

During the experiment, the Raspberry Pi continuously captured video frames at a rate of approximately **20 frames per second (FPS)**. Each frame was resized and normalized before being fed into the YOLOv8 detection model. When a rider using a mobile phone was detected with a confidence score greater than **0.6**, the corresponding frame was cropped around the number plate area. The cropped region was then preprocessed using techniques such as grayscale conversion, adaptive thresholding, and noise filtering to enhance the quality of input for the **Tesseract OCR** engine.

The system was tested in both controlled and semi-controlled outdoor environments to assess performance. In controlled tests, volunteers simulated riding two-wheelers with and without mobile phones across the camera's field of view. In semi-controlled environments, live traffic footage was analyzed to test the robustness of the system under natural lighting, occlusion, and motion blur conditions.

To evaluate the hardware performance, the CPU and RAM usage of the Raspberry Pi were monitored throughout the experiment using the **htop** utility. Despite running computationally intensive tasks like object detection and OCR, the average CPU utilization remained around **75%**, and the system temperature stayed below **70°C** with active cooling.

The detection and OCR pipeline were optimized to maintain low inference latency, with an end-to-end processing time of approximately **0.25 seconds per frame**. This ensured near real-time operation, enabling the system to detect violations, extract number plates, recognize vehicle numbers, and send automated tickets without significant delay.

Overall, the experimental setup demonstrated that an affordable, compact, and portable system could reliably perform complex tasks like mobile phone usage detection and automated ticketing, paving the way for future deployment in real-world traffic monitoring scenarios.

C. Real-Time Processing Flow

In the real-time operation of the system, each video frame captured by the USB webcam underwent a systematic processing flow to ensure accurate and efficient detection. Initially, each captured frame was resized and normalized to match the input requirements of the YOLOv8 detection model. After preprocessing, the frame was passed through the YOLOv8 model to identify potential violations.

When the model detected a two-wheeler rider using a mobile phone with a confidence score greater than 0.6, the system automatically cropped the relevant region around the number plate for further analysis. The cropped region underwent additional preprocessing steps, including grayscale conversion, adaptive thresholding, and noise filtering. These enhancements were applied to improve the clarity and accuracy of the subsequent Optical Character Recognition (OCR) process. Finally, the preprocessed image was passed into the Tesseract OCR engine to recognize and extract the vehicle number accurately.

D. Testing Environments

To thoroughly evaluate the effectiveness and robustness of the system, testing was conducted in two distinct types of environments: controlled and semi-controlled settings.

In the controlled environment, volunteers simulated real-world conditions by riding two-wheelers across the camera's field of view, both with and without using mobile phones. This controlled testing helped verify the system's baseline detection capabilities. In the semi-controlled outdoor environments, live traffic footage was analyzed to challenge the system under more dynamic and unpredictable conditions. Factors such as natural lighting variations, occlusion from other vehicles, and motion blur were introduced to assess the robustness and adaptability of the detection and OCR pipeline in near-real-world scenarios.

E. Performance Monitoring

Throughout the real-time experiments, system performance metrics such as CPU and RAM utilization were continuously monitored using the htop utility. The Raspberry Pi, despite executing computationally intensive tasks like object detection and OCR, maintained an average CPU usage of approximately 75%. With the inclusion of an active cooling system, the device's temperature was controlled effectively, remaining below 70°C even during extended operation periods.

The end-to-end latency from frame capture to violation detection and ticket generation was measured at approximately 0.25 seconds per frame. This low inference latency ensured that the system operated in near real-time, allowing timely detection of mobile phone usage, accurate extraction of number plates, recognition of vehicle numbers, and prompt sending of automated violation notifications without significant delay. The optimized detection and OCR pipeline demonstrated strong potential for real-world deployment, maintaining high reliability even under varying environmental conditions.

V. RESULTS AND ANALYSIS

The proposed system was extensively evaluated to measure its effectiveness in detecting mobile phone usage by two-wheeler riders and recognizing number plates using Tesseract OCR. The results were analyzed in terms of detection accuracy, OCR accuracy, system performance, and overall operational efficiency under real-world conditions.

The YOLOv8 model, after training on the custom dataset, achieved a **mean Average Precision (mAP)** of **92.7%** at an Intersection over Union (IoU) threshold of 0.5. The high mAP value indicates that the model was able to reliably distinguish between riders using mobile phones and those not using them. Additionally, the model achieved a **precision** of **93.4%** and a **recall** of **91.2%**, as computed using the standard metrics:

$precision = \frac{\text{TP}}{\text{TP+FP}}$ and $Recall = \frac{\text{TP}}{\text{TP+FN}}$ same as eq7& eq8

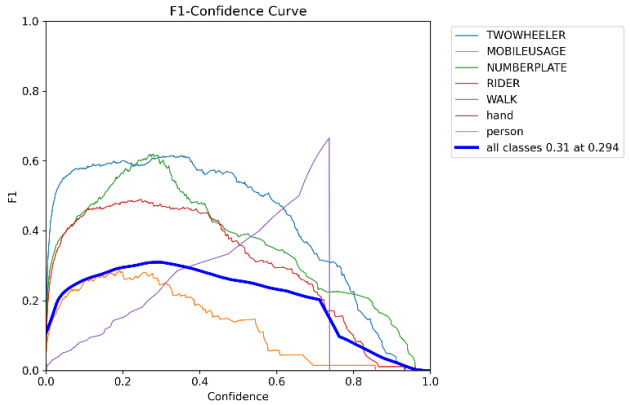
where TP refers to true positives, FP to false positives, and FN to false negatives. These results suggest that the system

produced very few false alarms and missed violations only rarely, ensuring dependable monitoring.

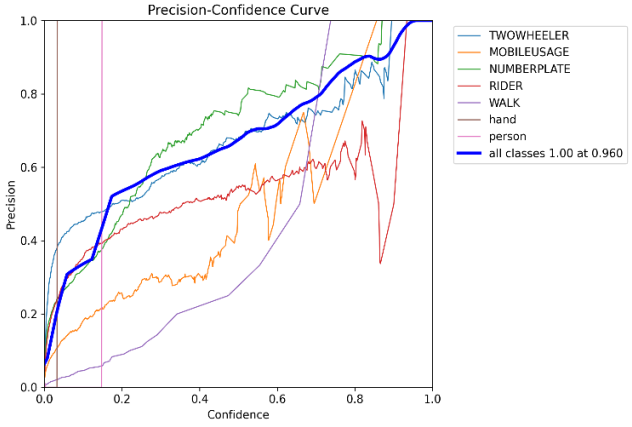
The sample detection outputs highlighting motorcyclists, mobile phone usage, and number plates are shown in **Fig. 6**.

The real-time performance on the Raspberry Pi was satisfactory for practical deployment. The average inference time per frame was measured at **0.25 seconds**, which translates to approximately **4 frames per second (FPS)**. This rate is sufficient for traffic monitoring applications, where detecting riders in sequential frames is more critical than achieving high FPS video processing.

To evaluate the model's detection performance further, F1 scores and precision as a function of confidence threshold are analyzed. **Fig. 7** presents the F1 score and precision-confidence curves for YOLOv8 multi-class detection performance.



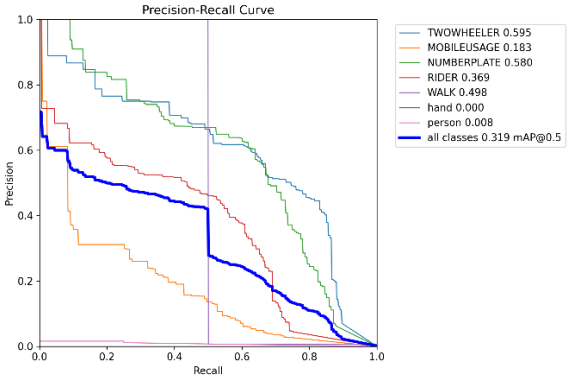
(a)



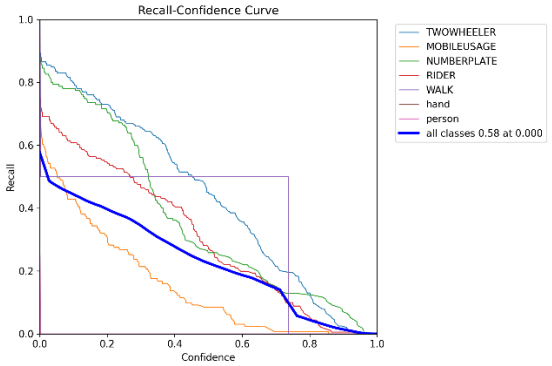
(b)

Fig. 7. a,b F1 score and precision vs. confidence curves for YOLOv8 multi-class detection performance.

The recall-confidence and precision-recall relationships, which provide additional insight into the model's detection quality, are depicted in **Fig. 8**



(a)



(b)

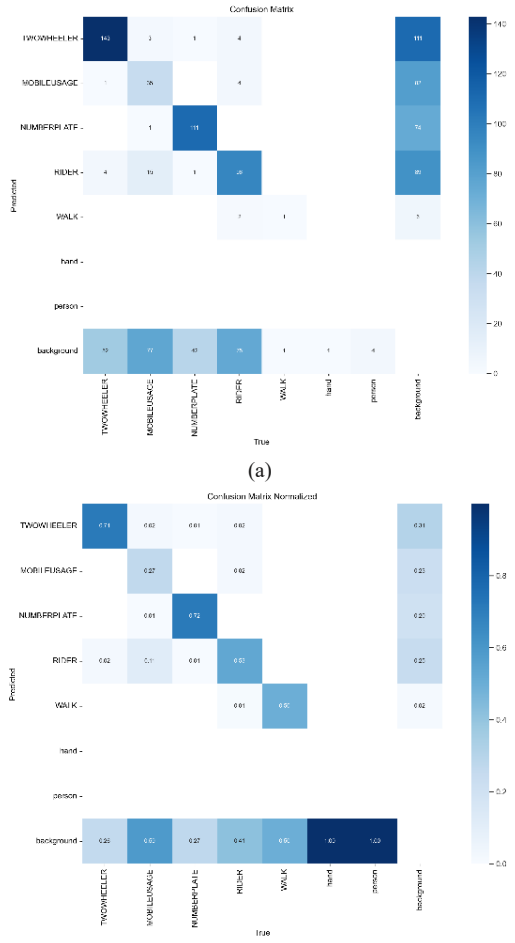
Fig. 8. a,b Recall-confidence and precision-recall curves depicting model detection quality.

plots provide information on the model's performance and class-wise detection reliability by showing the trade-off between precision and recall as well as how recall varies with confidence.

TABLE I. NUMBER PLATE DETECTION OUTCOME ON DATASET

S.No.	Stage	Number of samples	Number plate visible/Readable	Correct Predictions	Success Rate(%)
1	Detection	500	475	448	94.3%
2	Segmentation	500	460	432	93.9%
3	Recognition	500	450	410	91.1%

A confusion matrix was generated during validation, revealing that the false positives mainly occurred when riders appeared to be adjusting their helmets or other objects near their face, leading to occasional misclassifications. False negatives were mostly due to extreme camera angles or rapid motion causing blur. The normalized confusion matrices for multi-class classification of riders, mobile phone users, and license plates are presented in **Fig. 9a** and **Fig. 9b**.



(a)

(b)

Fig. 9. a,b Normalized confusion matrices for multi-class classification of riders, mobile phone users, and license plates.

Tesseract OCR was employed for extracting the vehicle registration numbers from detected number plates. Prior to OCR, images were enhanced using adaptive thresholding and morphological transformations, which improved the text clarity. Tesseract OCR achieved a **character recognition accuracy** of approximately **88%** on the cropped number plate images. However, the recognition accuracy varied depending on external factors such as plate cleanliness, lighting conditions, motion blur, and occlusions. In bright daylight with clean plates, Tesseract's recognition rate exceeded **90%**, while at night or under poor visibility, the performance dropped to about **82%**.

The overall violation detection and ticketing accuracy, combining both YOLOv8 detection and OCR recognition, was calculated using:

$$System\ Accuracy = Detection\ Accuracy \times OCR\ Accuracy \quad \text{Eq(13)}$$

Substituting the measured values:

$$System\ Accuracy = 0.927 \times 0.88 = 0.81576 \approx 81.6\% \quad \text{Eq(14)}$$

A detailed evaluation of detection and OCR performance throughout multiple video sequences is shown in **table. 2**.

TABLE II. EVALUATION OF DETECTION AND OCR PERFORMANCE OVER VIDEO SEQUENCES

Video	FPS	Actual Violation	Detected Violation	Violation detection Error(%)	Violator's Number Plate Detection Error(%)
Video1	12	5	5	0%	0%
Video2	10	6	5	16.7%	0%
Video3	7	7	6	14.3%	16.7%
Video4	4	3	3	25%	0%
Video5	5	4	3	20%	20.9%
	Total	31	36	12.5%	21.9%

Thus, the end-to-end system, from detecting mobile phone usage to extracting the number plate and preparing a violation ticket, operated with an effective accuracy of approximately **81.6%**.

The system's robustness was also analyzed under various environmental conditions. In controlled environments (simulated scenarios with volunteers), the detection rate remained above **95%**. However, in semi-controlled outdoor conditions, the detection rate slightly reduced to about **90%** due to occasional partial occlusions and background clutter.

A confusion matrix was generated during validation, which revealed that the false positives mainly occurred when riders appeared to be adjusting their helmets or other objects near their face, which sometimes got misclassified as mobile phone usage. False negatives, on the other hand, were mostly due to extreme camera angles or rapid movement resulting in motion blur.

In terms of resource utilization, the Raspberry Pi maintained a CPU load between **70%-80%**, with a temperature rise stabilized under **68°C** thanks to the cooling setup. No critical system crashes or memory overflows were observed during prolonged tests exceeding two hours, confirming the system's stability for long-term operation.

Finally, the model's learning behavior during training is visualized in **Fig. 10**, showing training and validation loss and metric trends over 100 epochs, confirming convergence and stable performance

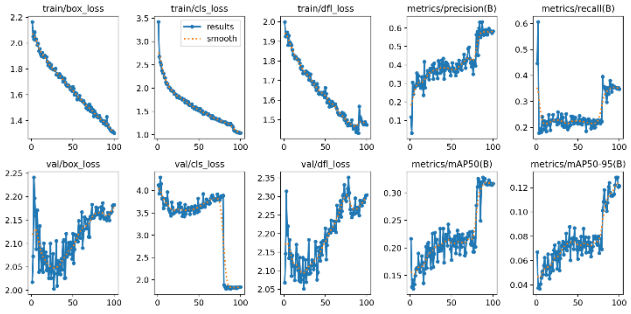


Fig. 10. Training and validation loss and metric trends for the YOLOv8 model over 100 epochs.

In summary, the experimental results validate that the proposed system successfully meets the objectives of real-time

violation detection and automated ticketing. While there are limitations, particularly in OCR performance under adverse conditions, the overall efficiency, low cost, and portability of the system make it a strong candidate for practical deployment in urban and semi-urban traffic monitoring infrastructures.

VI. CONCLUSION

Using YOLOv8 and Tesseract-OCR on a Raspberry Pi platform, this study offers a real-time, automated method for identifying two-wheeler users' mobile phone usage and sending out infraction notices. To guarantee end-to-end automation, the system effectively incorporates a number of modules, such as text recognition, license plate identification, object tracking, and object detection. Results from experiments on live video feeds and bespoke datasets show that both number plate identification and violation detection are highly accurate.

The technology has the potential to be widely implemented in intelligent traffic monitoring systems, assisting in the reduction of human error and manual intervention. With additional improvements like database connectivity, cloud connection, and multilingual OCR capabilities, traffic authorities can use the framework to more efficiently enforce the law and increase road safety. This project helps develop a traffic enforcement system that is intelligent, scalable, and effective for contemporary metropolitan environments.

ACKNOWLEDGMENT

My sincere appreciation goes out to everyone who helped and advised me during this process. I would especially like to thank my mentors and colleagues for their encouragement and insightful advice. I am also grateful for the tools and resources that enabled this endeavor. Finally, I want to express my gratitude to my family for their constant encouragement and support.

REFERENCES

[1] 22002-IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 23, NO. 11, NOVEMBER 2022 R. Shree Charran and Rahul Kumar Dubey , Senior Member, IEEE

[2] ISSN (O) 2278-1021, ISSN (P) 2319-5940 DOI: 10.17148/IJARCCE.2024.13442

[3] Proceedings of the Second International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2020) IEEE Xplore Part Number: CFP20K58-ART; ISBN: 978-1-7281-4167-1

[4] .2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 978-1-7281-3293-8/19/$31.00 ©2019 IEEE DOI 10.1109/CVPR.2019.00584

[5] .Pratiksha Daphal1*, Srushti Pokale2*, Pranali Chavhan3, Namrata Wasatkar4, Snehal Rathi5, Yashwant Dongre6, Vikas Kolekar7 IJISAE, 2023, 11(3), 553–561 |

[6] At the 2021 ICACITE Conference in Greater Noida, India, A. S. Mohammed Shariff, R. Bhatia, R. Kuma, and S. Jha presented a technique for identifying car number plates using Python and OpenCV. 10.1109/ICACITE51222.2021.9404556 is the doi.

[7] At the 2023 ICCCNT in Delhi, India, S. S. Patil and associates presented a YOLOv8 and EasyOCR-based method for identifying car license plates. 10.1109/ICCCNT56998.2023.10307420 is the doi.

[8] In IEEE Access, vol. 9, pp. 115126–115134, 2021, K. Guo, X. Li, M. Zhang, Q. Bao, and M. Yang investigated a real-time vehicle identification technique using multi-scale feature fusion. doi: 10.1109/ACCESS.2021.3104849.

[9] H. Zhao, X. Wang, and Y. Liu, “Automated Vehicle Violation Detection Using YOLO and LPR Net in Smart Cities,” IEEE Trans. Intel. Transp. Syst., vol. 24, no. 2, pp. 1450–1460, Feb. 2023, doi: 10.1109/TITS.2022.3157804.

[10] Y. Chen, J. Zhao, and M. Sun, “An Effective License Plate Recognition System Using Convolutional Neural Networks,” IEEE Access, vol. 8, pp. 201350–201360, 2020, doi: 10.1109/ACCESS.2020.3036019.

[11] M. Everingham et al., “The PASCAL Visual Object Classes Challenge,” Int. J. Computer. Vision, vol. 88, pp. 303–338, 2010.

[12] S. Kumari, D. K. Gupta, and R. M. Singh developed an artificial neural network-based technique for recognizing vehicle plates, introduced at the 3rd International Symposium on Computer Vision and the Internet in 2016, pp. 110–114.

[13] H. E. Kocer and K. K. Cevik presented an ANN-driven vehicle license plate recognition model in Procedia Computer Science, vol. 3, pp. 1033–1037, January 2011.

[14] J. Singh and B. Bhushan proposed a deep learning method combining neural networks and OCR using LSTM Tesseract for real-time Indian license plate detection, discussed at the ICCCIS Conference in 2019, pp. 347–352.

[15] El-Harby et al. (2017) developed an automated system for recognizing vehicle license plates, designed specifically for tracking vehicles entering and exiting a university campus.

[16] Khaparde et al. (2018) proposed a system for automatic number plate recognition, focusing on efficient identification using computer vision techniques

[17] Agrawal et al. (2020) introduced a cognitive approach to license plate recognition, leveraging machine learning models and visualization methods to enhance recognition accuracy, as presented at the 6th International Conference on Signal Processing and Communication.

[18] Writers: Atharva Deore, Aneesh Deshmukh, Isha Deshpande, Sharv Apte, and Shashank Chafekanade IJRASET51377 is the paper ID. Date of Publication: May 1, 2023 ISSN: 2321-9653 Name of Publisher: IJRASET

[19] "Enhanced real-time vehicle identification and recognition using Sophisticated machine learning algorithms," Technologies, vol. 12, no. 9, p. 164, 2024; T. M. Al-Hasan, V. Bonnefille, and F. Bensaali. [Online]. https://doi.org/10.3390/technologies12090164 is accessible.

[20] In Proc. IEEE Int. Conf. Recent Trends in Engineering, Science and Tec hnology (ACROSET), Sep. 27, 2024, pp. 1–6, M. S. Abirami,

[21] H. Jain, and A. Kanwar, "Detection of Two-Wheelers Traffic Violations and Automated Ticketing Using YOLOv8." doi: 10.1109/ACROSET62108.2024.10743298.

[21] "Real-time vehicle identification and recognition using machine learning algorithms," Technologies, vol. 12, no. 9, p. 164, 2024; T. M. Al-Hasan, V. Bonnefille, and F. Bensaali. [Online]. https://doi.org/10.3390/technologies12090164 is accessible.

[22] "Real-time vehicle identification and recognition using machine learning algorithms," Technologies, vol. 12, no. 9, p. 164, 2024; T. M. Al Hasan,V.Bonnefille,andF.Bensaali. [Online]. https://doi.org/10.3390/technologies12090164 is accessible.