Traffic Violation Detection System

Randhima Dinalankara
Department of Computer Engineering
University of Sri Jayewardenepura
Colombo, Sri Lanka
randima@sjp.ac.lk

Udaya Wijenayake
Department of Computer Engineering
University of Sri Jayewardenepura
Colombo, Sri Lanka
udayaw@sjp.ac.lk

Sahara Ameer
Department of Computer Engineering
University of Sri Jayewardenepura
Colombo, Sri Lanka
en91436@sjp.ac.lk

Abdul Rahuman
Department of Computer Engineering
University of Sri Jayewardenepura
Colombo, Sri Lanka
en91459@sjp.ac.lk

Vithushanth
Department of Computer Engineering
University of Sri Jayewardenepura
Colombo, Sri Lanka
en91424@sjp.ac.lk

Abstract*— Travel convenience is impacted by a number of factors, including the condition of the road, traffic, length of trip, accidents, speed, etc. The daily increase in accident rates poses the biggest concern in Sri Lanka. These events not only result in fatalities but also increase the nation's financial losses. Traffic congestion is brought on by road users' lack of discipline and sentiments, which may result in traffic offenses. The goals of this research are to curb any traffic infractions as well as reduce corruption and nepotism on the road. This system was developed as separate three models namely edge server, fine management system and centralized server. Edge server developed in python is to detect traffic violations such as parking, single line cutting, double line cutting, one way and u turn in real time according to the changing rules and constraints configured by the traffic police admin via the fine management system which supports a configurable platform designed using canvas designing tools. Upon edger server identifying violations, the functions in the centralized server are responsible to recognize the number plate of the violated vehicle will be triggered and send a sms to driver for further fine management and payment supported by the fine management system. The mobile application controlling server is responsible to track and inform the violations to nearest police officer in any case the system could not rectify the vehicle number plate. The system surpasses the existing solutions by offering a highly configurable platform. This adaptability allows for the creation of specific rules for diverse traffic scenarios ranging from setting parameters for traffic violation detection based on vehicle types, time periods and

Keywords—Traffic violation detection system, vehicle tracking, vehicle detection, number plate recognition, traffic fine management system,

I. INTRODUCTION

Sri Lanka has the poorest record for vulnerable road users in the South Asia region, according to the WHO's Global Status Report on Road Safety in 2018. The World Bank Group report shows that Sri Lanka has twice the average annual road crash deaths per capita as high-income countries. Despite lockdowns and health guidelines, in 2021, there were 2419 fatalities and 13,469 injuries. The average annual accident in Sri Lanka is 38,000, with 3000 fatalities and 8,000 serious injuries [1]. The true number of traffic accidents is substantially higher than the statistic. The severity of the prevailing situation regarding road safety is well described by the above figures. Non-compliance with national traffic laws is the primary cause of road accidents, with drivers disregarding most laws. Authorities introduce timely traffic rules through programs like lane discipline, but these rules are often forgotten. Despite a heavy fine system in 2018, accidents and traffic-related deaths continue to rise. Drivers and riders are not concerned about fines, but only obey rules when accompanied by a police officer [2].

Traffic police in Sri Lanka face inadequate facilities compared to developed countries, requiring physical presence and manual monitoring to catch offenders. In Sri Lanka, traffic monitoring by police officers is an important aspect of maintaining road safety and enforcing traffic laws. Police officers are often stationed at busy intersections or along major roadways to monitor traffic flow and ensure that drivers are adhering to traffic laws. They may use radar or laser equipment to measure vehicle speeds, and they may also direct traffic or issue citations to drivers who violate traffic laws. However, it is worth noting that Sri Lanka is facing a severe shortage of traffic police officers and the traffic situation in the country is not well managed due to shortage of officers. The government is working on to increase the number of officers as well as providing them with necessary training and equipment.

In the recent past, traffic monitoring via CCTV is also becoming increasingly a common way to enhance traffic management and to improve road safety in Sri Lanka. CCTV cameras are used to monitor traffic flow and identify bottlenecks, as well as to detect traffic violations such as illegal parking, reckless driving, and running red lights. The footages captured by the cameras are also used as evidence in the event of a traffic violation or accident. Sri Lanka's Road Development Authority (RDA) has been implementing CCTV cameras in major cities such as Colombo, Kandy, and Galle to monitor the traffic. The footage is monitored by the RDA's Control Room and the police to ensure that traffic laws are being followed, if not tickets are issued by the nearby officers upon receiving information from the control room. Additionally, the government is planning to install more CCTV cameras in main roads and highways in the country to improve the monitoring and enforcement of traffic laws.

Though the aim is to reduce the number of accidents and improve the overall traffic situation in the country, the limited human resources, coverage of duty, hours of operation and their cost are major drawbacks of manual traffic monitoring.

But traffic monitoring systems in developed countries effectively influence driver behavior, reducing road accidents and promoting compliance with traffic laws. With all these facts, it is clear that a lasting and system-wide approach to road safety is a required need in Sri Lanka. Our approach is to create an automatic traffic violation detection system that detects violations, issues online tickets, and provides end-of-term reports. The system uses edge sensors to capture violations and sends data to a centralized server for processing and delivering tickets. If the system fails to identify the vehicle's number plate, it sends the information to the nearest police officer for immediate fines. The system also allows for timely rule enforcing by authorities.

A. *Features and objectives*

The proposed solution offers the features that will facilitate automatic traffic violation management system. The features offered by the system are detecting traffic violations like illegal parking, single line cutting, double line cutting, one-way, and illegal U turn. It can be configured based on time, day, region, or vehicle. The system extracts number plate information, retrieves owner details, and issues a violation ticket. Real-time alerts for police officers to identify vehicle numbers, replace manual payment systems, and generate detailed reports for both police and vehicle owners on violations. Online payment system replaces hassled manual payments.

Thereby the system achieves the objective of Automated traffic violation detection that can reduce accidents by alerting drivers on violations, monitoring violations and charging penalties. A centralized ticketing system eliminates corruption and bribery, promoting equal justice. A configurable platform allows for flexible detection under different constraints, allowing authorities to adapt rules as needed.

II. Related work

Recent research on traffic violation detection using Computer Vision and AI has garnered significant attention. This section compares recent studies in the proposed work domain.

Roopa Ravish and Shanta Rangaswamy [3], propose a system for detecting various traffic violations using computer vision techniques. The primary violations targeted in their study include jumping red signals, riding vehicles without helmets, driving without seat belts, and vehicles crossing the stop line during red signals. It employs YOLOv3, a popular object detection algorithm, using the Darknet 53 architecture. The authors highlight that system accurately detects and localizes traffic violations in input video footage. The authors also discuss issuing tickets to vehicle owners for detected violations, using E-Challan, an existing electronic ticketing system in India. The system integrates with E-Challan to automatically generate, and issue tickets based on detected violations.

Anima Pramanik, Sobhan Sarkar, and J. Maiti [4], demonstrate a system using computer vision techniques and video processing algorithms to detect traffic pre-events like speed violations, one-way traffic, overtaking, illegal parking, and incorrect passenger drop-off locations. The system's real-time capabilities enable timely corrective actions, reducing road events and improving overall safety. It uses object detection to identify vehicles in video frames, object tracking to track them over time, and spatio-temporal granulation for vehicle detection. Vehicle tracing is achieved through feature-based tracking and data association techniques. The system's performance and accuracy were tested using CamSeq1 and ISLab-Pvd datasets.

The work by Dr. S. Raj Anand, Dr. Naveen Kilari, Dr. D. Udaya Suriya Raj Kumar, and Sai Nitisha Tadiboina [5] discusses a detection system which is based on three principles: vehicle classification, environment awareness, and traffic violation detection. Vehicle classification involves tracking multiple vehicles from input video data, while environment awareness classifies elements like traffic lights, zebra crossings, lanes, and traffic signs. Traffic violation detection combines these principles to detect violations like signal jumping, speeding, and vehicle count. The study employs deep learning models like YOLOv5 and Darknet-53 for vehicle classification and detection. The MSCOCO dataset is used for object detection tasks in computer vision. The system achieves 97.67% accuracy for vehicle count detection and 89.24% for vehicle speed detection. The detection time is lower for high dense traffic flow, indicating that system operation speed is dependent on traffic density.

Samir Ibadov, Ragim Ibadov, Boris Kalmukov, Vladimir Krutov [3] present an algorithm for detecting traffic rule violations at unregulated crosswalks using computer vision approaches. The work has utilized Fast RCNN for vehicle detection, while Motion-Based Multiple Object Tracking detects pedestrians. The proposed algorithm compares the detected frames to determine if a violation has occurred. Also the study highlights that this method is effective in identifying and localizing vehicles and pedestrians in video frames.

In the work by Akila Peiris, E.A.T.A Edirisuriya, C.D Athuraliya and Isuru Jayasooriya, Haar Cascade detection is used for vehicle detection, with other techniques like background subtraction and RCNN being tested. In their work tracking phase involves using bounding box values from vehicle detection to create a KalmanBoxTracker model, assigning unique IDs to each object. The study has utilized KalmanBoxTracker to maintain vehicle tracking even in occlusions and abrupt movement. The study uses CCTV footage from the Sri Lankan police as their dataset.

III. Methodology

The Traffic Violation Ticketing System consists of three modules that work together to manage traffic violations efficiently as shown in fig 1.

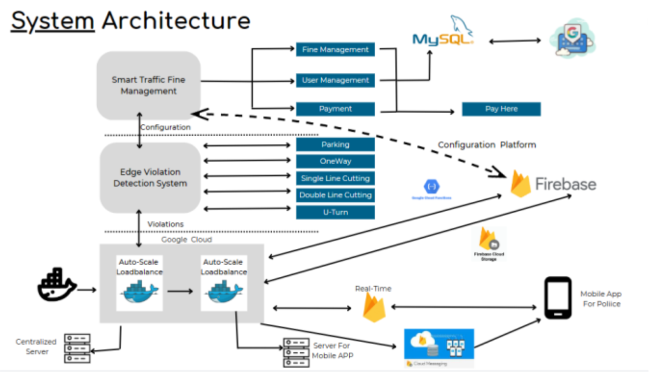


Fig. 1. Overall system architecture of Traffic Violation Detection System

The Smart Traffic Fine Management System streamlines ticketing by utilizing a MySQL database in Google Cloud and an integrated PayHere payment gateway. It automatically generates digital tickets with violation details, eliminating manual paperwork and ensuring accurate and timely fine issuance.

The Edge Violation Detection System utilizes advanced technologies like YOLOv5 and DeepSORT to detect real-time violations at the edge level. It monitors traffic scenarios, detects parking, one-way, single line cutting, double line cutting, and U-turn violations, and sends data to the Smart Traffic Fine Management System.

The Traffic Violation Ticketing System uses a centralized server system for enforcement and communication. It uses advanced computer vision algorithms to accurately detect vehicle number plates, send SMS notifications to drivers, and trigger notifications to nearby police officers if number plates cannot be recognized due to poor image quality or obstruction. The system also tracks police officers' real-time positions, optimizing response times and facilitating rapid intervention, enhancing overall effectiveness of traffic violation enforcement.

Hence the development steps are as follows;

1) Traffic violation detection implementation
2) Smart Traffic fine management system
3) Cloud function implementation
4) Centralized server implementation
5) Server for mobile application
6) Cloud hosting implementation

A. *Traffic Violation Detection Module*

The Traffic Violations Detection Module uses an edge server with multi-threading capabilities for efficient processing of incoming data streams, enabling real-time monitoring and analysis of traffic scenarios. It dedicates a separate thread for violation detection, while another thread saves detected violations to the Firebase database. The third thread module also monitors and updates camera views, ensuring accuracy and reliability in violation detection.

1) *Violation detection*

a) Vehicle detection: The Traffic Violations Detection Module relies on the YOLOv5 model for violation detection, which offers speed, scalability, accuracy, and transfer learning model size. YOLOv5 is known for its speed, efficiency in real-time object detection, and scalability, allowing it to handle various traffic scenarios without compromising detection accuracy. Its advanced deep learning techniques, such as anchor-based detection and feature pyramid networks, ensure reliable identification and classification of vehicles involved in violations. Transfer learning enables the model to leverage knowledge from pre-trained models on large-scale datasets, reducing training time and resources. The compact model size makes it suitable for edge computing scenarios with limited storage and computational resources, allowing faster deployment and efficient resource utilization [7].

b) Vehicle Tracking: The DeepSORT algorithm is used for tracking detected vehicles over time in traffic scenarios, enabling efficient multiple object tracking. It demonstrates robust tracking even in challenging situations, using sophisticated techniques like appearance-based matching and motion modeling. DeepSORT assigns unique identifiers to each tracked vehicle, maintaining consistent and accurate tracks. It also enables real-time tracking, ensuring timely response and intervention in traffic violations. This real-time tracking capability enables prompt detection and swift action [8].

c) Identifying violations: The Traffic Violations Detection Module uses algorithms for identifying specific traffic violations, analyzing vehicle behavior and making informed decisions based on predefined rules and patterns.

2) *Parking Violation Detection:*
The module uses vehicle tracking to monitor vehicle movement within a monitored area, assigning unique identifiers and maintaining trajectories. The non-parking areas are previously configured through the configuration platform by the respective authority. The algorithm analyzes the tracked vehicle's position and determines if it's within designated non-parking areas. If it falls within these boundaries, it's flagged for further analysis. The algorithm checks if the vehicle remains immobile for a threshold value, indicating potential parking violations. Workflow for parking violation detection is shown in fig 2.

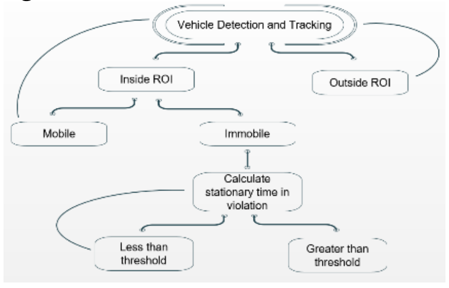


Fig. 2. Workflow of parking violation detection

3) *Single or Double Line Violation Detection:*
The module uses tracking algorithms to track vehicle movement and assign unique identifiers. Line polygon configuration is used to define single or double lines on the road, considering variations due to camera perspectives or road conditions. Coordinate analysis is used to determine if the vehicle falls within the configured polygons. Polygon intersection checks are performed to identify potential violations.

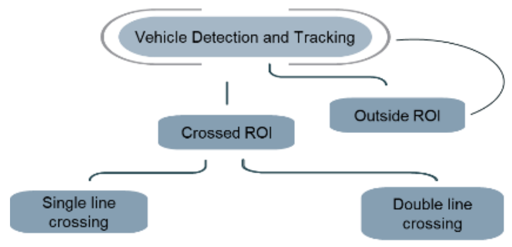
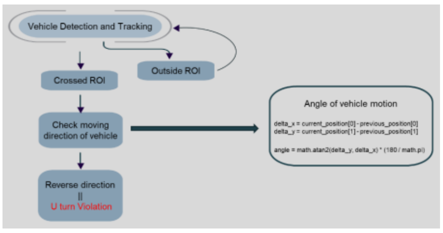


Fig 3. Workflow of single or double line cutting violation detection

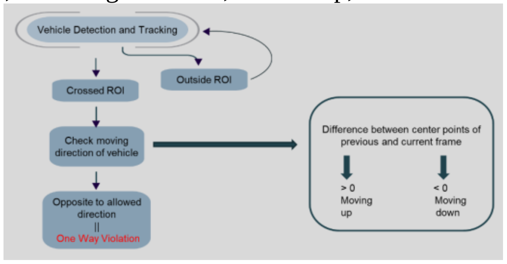
4) *U turn Violation Detection*:
The algorithm defines a U-turn line based on traffic regulations and road layout, tracks vehicle position, and checks its moving direction. It then calculates the angle of motion and maps it to a specific direction using if-elseif conditions. If the vehicle moves in reverse, it detects a U-turn violation and flags it in the Firebase real-time database. The algorithm also uses vehicle motion angle to determine the reverse direction after crossing the configured line area.



Vehicle Detection and Tracking
Crossed ROI
Outside ROI
Check moving direction of vehicle
Reverse direction || U turn Violation
Angle of vehicle motion
delta_x = current_position[0] - previous_position[0]
delta_y = current_position[1] - previous_position[1]
angle = math.atan2(delta_y, delta_x) * (180 / math.pi)

Fig 4. Workflow of U turn violation detection

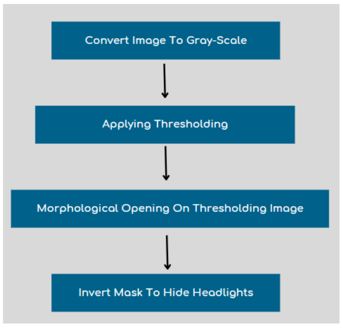
5) One way Violation Detection: The algorithm tracks vehicle movement, assigns unique identifiers, and monitors their positions over time. It defines road configuration and permits one-way travel. The algorithm analyzes vehicle positions, checks if detected direction matches authorized travel, and detects one-way violations by reporting the violation, including identifier, timestamp, and location.



Vehicle Detection and Tracking
Crossed ROI
Outside ROI
Check moving direction of vehicle
Opposite to allowed direction || One Way Violation
Difference between center points of previous and current frame
> 0 Moving up
< 0 Moving down

Fig 5. Workflow of one way violation detection

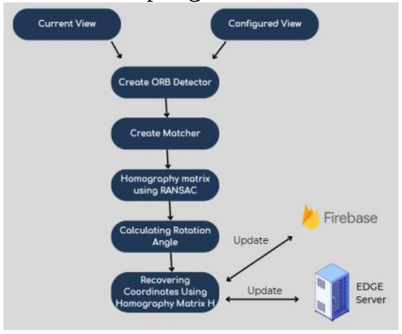
6) Violation Detection at Night: The system uses real-time identification to differentiate between daytime and nighttime conditions, allowing the server to apply specific processing steps for nighttime scenarios. Nighttime checks optimize computational resources, while headlight illumination reduction suppresses interference and enhances visibility. Grayscale conversion simplifies image processing, while thresholding segmented images and distinguishing headlights. Morphological operations remove noise and refine binary masks, while mask inversion transforms bright areas into dark areas, reducing headlight glare and enhancing violation detection. Mask application selectively retains regions not affected by headlight illumination, improving accuracy in identifying violations during nighttime conditions.



Convert Image To Gray-Scale
Applying Thresholding
Morphological Opening On Thresholding Image
Invert Mask To Hide Headlights

Fig 6. Preprocessing steps for night time vehicle detection

7) Camera Angle Change recovery: The function retrieves a reference image from Firebase Storage based on the camera ID and uses the ORB detector and matcher objects to detect key points and compute descriptors. It then calculates the homography matrix using RANSAC, enabling comparison and coordinate recovery. The function calculates the rotation angle of the current image compared to the reference image. The function uploads the image to Firebase Storage for future reference, calculates recovered coordinates, and updates the Firebase Realtime Database coordinates without interrupting violations detection.



Current View
Configured View
Create ORB Detector
Create Matcher
Homography matrix using RANSAC
Calculating Rotation Angle
Recovering Coordinates Using Homography Matrix H
Update
Firebase
Update
EDGE Server

Fig 7. Workflow of camera angle change recovery process

B. *Smart Traffic Fine Management System*

The smart traffic management system enhances traffic management and streamlines operations by providing user-friendly interfaces, secure login and registration, and password recovery. It enables traffic police officers to configure timely violation rules, issue fines for violations, maintain profiles, and manage data. Users can access individual fine records, payment history, and access advanced functionalities like data search, sort, and download options in formats like CSV, PDF, and Excel. Print options enable users to obtain physical copies of relevant data. Also drivers are facilitated with online payment option. Overall, this system streamlines operations, enhances compliance, and improves efficiency for all stakeholders involved in traffic management.

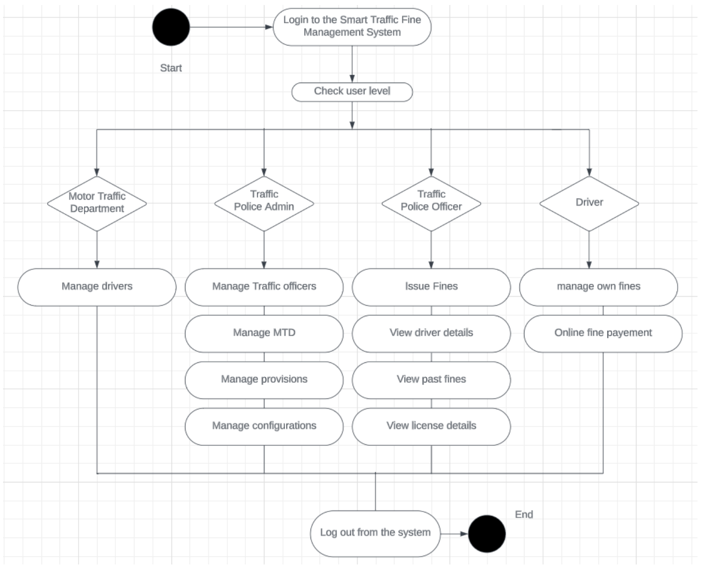


Fig 8. Activity diagram of Smart Traffic Fine Management System

1) Configuration platform: The configuration platform is a user-friendly tool for traffic police admins to upload images or videos and draw custom polygons as shown in fig. 9. It features a responsive design, allowing users to create and adjust polygon positions, and remove polygons if needed. The platform also provides coordinates for each polygon vertex, enabling accurate configuration and defining areas of interest. It also offers file selection options for local camera views and URL links for Google Storage-stored view footage.

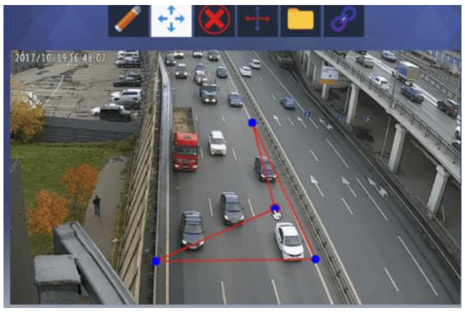


Fig. 9. Configured region of interest using configuration platform

C. Cloud function implementation

When a creation of violation event is occurred, in the specified path of the Realtime Database, the Cloud Function is automatically triggered and sends the event to the centralized server. The combination of Firebase Realtime Database and Cloud Functions follows an event-driven architecture which enables real time processing of the violation events. The event is sent with HTTP POST request to the centralized server with axios library.

D. Central server implementation

1) Vehicle detection: The centralized server detects vehicle types when a request is made, including violation details and the main attribute mentioned whether that particular vehicle is configured for the violation or mentioned as "all" where the violation is configured is for all type of vehicles. The detection model is YOLOv5, and vehicle classes include motorcycle, three-wheeler, bus, truck, van, and car. If the vehicle type is not mentioned for violation, the server ends the process. Further processing to identify the vehicle owner and issue a ticket will only be done if the type of the vehicle is configured for violation.

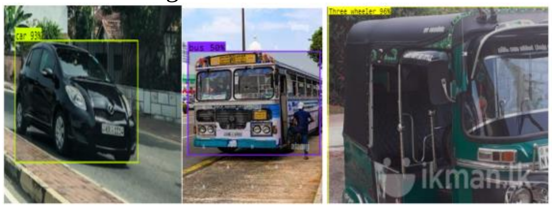


Fig. 10. Vehicle detection images with confidence rate

2) Number plate identification

a) Number plate detection: The YOLOv5 model is used to detect and localize the number plate region in vehicles. The model is trained on a custom-made dataset, using a combination of different number plates and lighting conditions. The pretrained model, trained on the MS COCO dataset, accelerates the training process and improves performance. The highest-confidence region is processed, and the bounding box region is cropped and stored for preprocessing and extraction as shown in fig. 11.

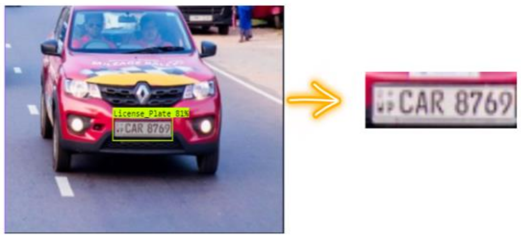


Fig. 11. License plate detection and cropped number plate region

b) Preprocessing number plate region: The optical character recognition algorithm uses image processing techniques to improve the quality of the number plate region. It converts the image to grayscale, removes noise and distortion using Iterative Bilateral Filtering, resizes the image with nearest neighbor, corrects skew angles using line detection, and applies affine transformation. Non-uniform illumination is rectified using top-hat transformation, and the image's intensity is enhanced by saturating 1% of the input image's intensity values and binarizing based on adaptive thresholding. This process ensures a binary license plate image with white text on a noiseless black background.

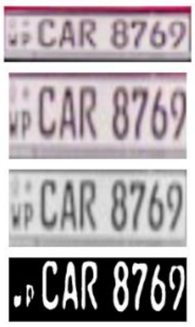


Fig. 12. Sequence of preprocessing steps done

OPTICAL CHARACTER EXTRACTION

PaddleOCR is an open-source toolkit optimized for accuracy and speed, providing pre-trained models for text recognition. The method uses preprocessed images, generating a list of lists with recognized text, coordinates, and confidence scores. The resulting text undergoes further processing using an ad-hoc algorithm.

AD-HOC ALGORITHM

Sri Lankan number plates are issued with lowercase characters and 4 digit numbers. An Ad-hoc algorithm is used to eliminate mis-recognized characters, and the ending contains only 4-digit numbers. When a hyphen or space is encountered, the text is split into 4-digit numbers and text or number parts. The provincial status and other constraints related to Sri Lanka are also eliminated.

3) Database operations and user notifications: Google Cloud SQL is a fully managed relational database that stores and manages user data. It is used to query for vehicle owners and issue fine tickets based on vehicle number plates. If the query returns null, the number plate is checked using a zoomed image. If the number plate is not identified, images from Firebase cloud storage are captured in close proximity to the camera, and the fine is recorded under the user's license ID. The details and proof of violation are sent to the user's mobile number using the Vonage SMS API.

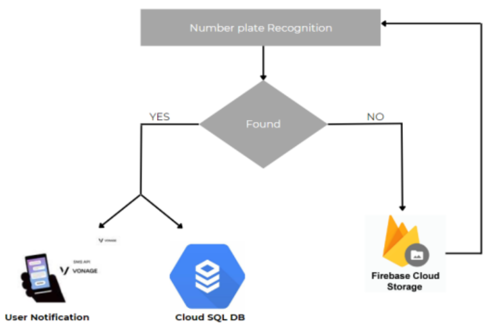


Fig. 13. Flow of number plate recognition in the central server

E. *Server for mobile application*

1) Mobile application development: The mobile application uses react-native to update location in real-time and receive notifications and data on violating vehicles. It is compatible with both Android and iOS devices and allows users to set location tracking options. Police officers can set the app to track the location when only open in the foreground or when running in the background. The app updates location every 10m changes, and the Global Positioning System (GPS) tracks the location. When the server sends violated vehicle data, the application receives notifications and details, allowing officers to issue tickets to the violators manually.

2) Server for mobile application: If the central server fails to match the correct vehicle number with the owner, then vehicle details, edge server location, fine details, and violation details are sent to the server, which processes notifications to the nearest police officer's mobile application. The server retrieves longitude and latitude coordinates from the Firebase real-time database, calculates the nearest point using Haversine Formula, and uses reverse geocoding to identify street names. The nearest three points are taken and then Google's reverse Geo coordinates API is used to get the police officer who is in the same street of violation occurred if the distance gap is a heuristic 20m else sent to the nearest police officer irrespective of street name. Firebase Cloud messaging sends notifications to user devices, ensuring timely updates.

F. *Cloud hosting impelemntation*

The mobile application's central server and server are deployed as Docker containers on Google Cloud Run, with URLs obtained and environment variables provided. Servers auto scale based on request load, and the integration and deployment pipeline is set through Github for seamless integration.

IV. EXPERIMENTS AND RESULTS

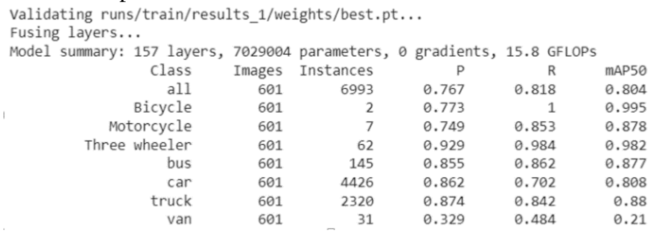
A. *Vehicle dataset gathering*

The project conducted a dataset gathering experiment to collect vehicle data for Sri Lankan vehicles under various climatic and lighting conditions. Methods included collecting open datasets, scraping Sri Lankan websites, and manually collecting road videos. The comprehensive dataset included totally 5925 images classified into 7 different classes namely 'Bicycle', 'Motorcycle', 'Three wheeler', 'bus', 'truck', 'van', 'car', 'Bicycle', 'Motorcycle', 'Three wheeler', 'bus', 'truck', 'van', 'car', which served as a valuable resource for training and evaluating Yolov5 for vehicle detection [9], [10], [11].

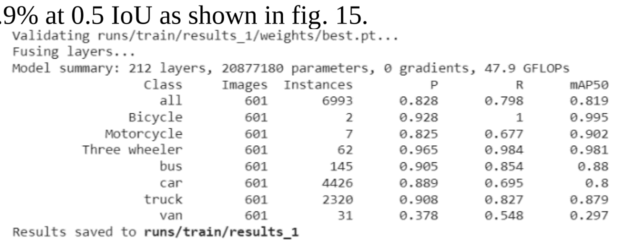
B. *Training Yolov5 for vehicle detection*

Training YOLOv5s model with following specifications, was able to achieve a mAP of 80.4% at 0.5 IoU as shown in fig. 14.

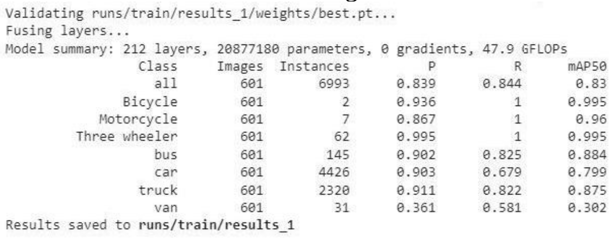
- Number of epochs: 100
- Batchsize: 16
- Optimizer: SGD


```
Validating runs/train/results_1/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7029004 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all        601       6993      0.767      0.818      0.804
               Bicycle        601          2      0.773          1      0.995
            Motorcycle        601          7      0.749      0.853      0.878
         Three wheeler        601         62      0.929      0.984      0.982
                   bus        601        145      0.855      0.862      0.877
                   car        601       4426      0.862      0.702      0.808
                 truck        601       2320      0.874      0.842       0.88
                   van        601         31      0.329      0.484       0.21
```
Fig. 14. mAP values for trained yolov5s model

Next, using the same specifications, the YOLOv5 medium model was used to improve precision with 25 blocks ranging from 0-24. To prevent overfitting, the first 15 blocks were frozen, allowing convolutional and batch normalization weights to be fixed. This allowed subsequent layers to fine-tune and learn specific features and to achieve a mAP of 81.9% at 0.5 IoU as shown in fig. 15.


```
Validating runs/train/results_1/weights/best.pt...
Fusing layers...
Model summary: 212 layers, 20877180 parameters, 0 gradients, 47.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all        601       6993      0.828      0.798      0.819
               Bicycle        601          2      0.928          1      0.995
            Motorcycle        601          7      0.825      0.677      0.902
         Three wheeler        601         62      0.965      0.984      0.981
                   bus        601        145      0.905      0.854       0.88
                   car        601       4426      0.889      0.695        0.8
                 truck        601       2320      0.908      0.827      0.879
                   van        601         31      0.378      0.548      0.297
Results saved to runs/train/results_1
```
Fig.15. mAP values for trained yolov5m model with frozen layers

Next approach was not to freeze layers, including initial blocks, enables end-to-end training, allowing for flexible optimization and learning more relevant features. This approach improves performance by leveraging the model's adaptability and learning throughout the training process, resulting in more meaningful representations from the data. The fully trained YOLOv5m model was able to achieve a mAP of 83% at 0.5 IoU as shown in fig. 16.


```
Validating runs/train/results_1/weights/best.pt...
Fusing layers...
Model summary: 212 layers, 20877180 parameters, 0 gradients, 47.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all        601       6993      0.839      0.844       0.83
               Bicycle        601          2      0.936          1      0.995
            Motorcycle        601          7      0.867          1       0.96
         Three wheeler        601         62      0.995          1      0.995
                   bus        601        145      0.902      0.825      0.884
                   car        601       4426      0.903      0.679      0.799
                 truck        601       2320      0.911      0.822      0.875
                   van        601         31      0.361      0.581      0.302
Results saved to runs/train/results_1
```
Fig.16. mAP values for fully trained yolov5m model

Fully trained Yolov5m model for vehicle detection gains highest precision, improving object detection and decreasing loss values as training progresses as shown in following figures.

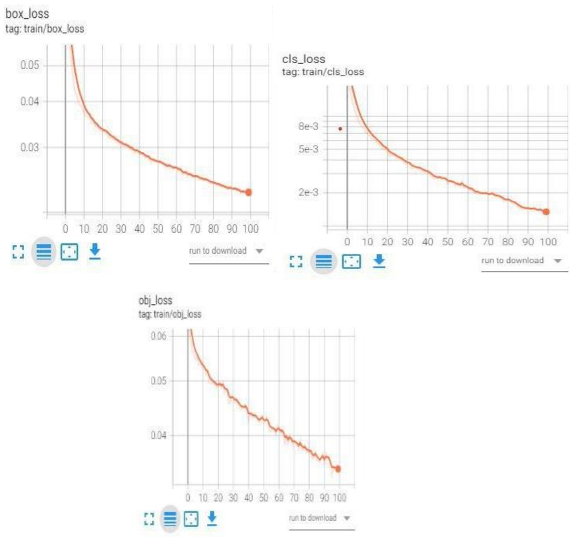


Fig. 17. Loss curves for training yolov5m model

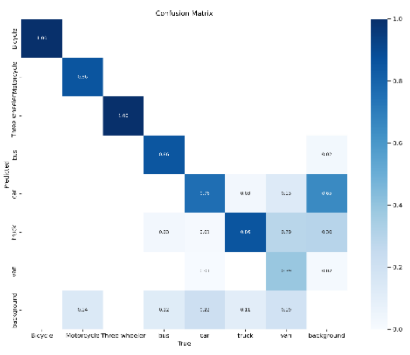


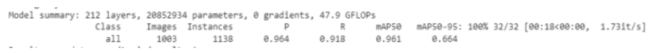
Fig. 18. Confusion matrix for trained yolov5m model

C. *Training for number plate detection*

The vehicle number plate datasets are derived from Roboflow, combining multiple number plate datasets from various environments [12], [13], [14], [15]. The dataset contains images from various environments, including bright light, cloudy sky, dark light, and smudged license plates. The class labels are changed to align with a single class label. The initial dataset is divided into training, validation, and test sets, with 7000, 2000, and 1000 images used for training and testing. The final dataset is prepared by selecting images from all datasets after preprocessing steps.

An augmentation technique is used to improve the model's performance. Cropping, grayscale, blurring, brightness, stretch, and shear are used to enhance the model's performance. The YoloV5 models are used to train and compare performance analysis on the dataset, with the mean average precision of the YoloV5m model being higher than small and nano models with slightly higher inference time.

The fully trained YOLOv5m model was able to achieve the highest mAP of 96.1% at 0.5 IoU and 66.4% mAP at range of 0.5-0.95 IoU as shown in the fig. 19.


```
Model summary: 212 layers, 20852934 parameters, 0 gradients, 47.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:18<00:00,  1.73it/s]
                   all       1003       1138      0.964      0.918      0.961      0.664
```
Fig. 19. Results after training yolov5m model for license plate detection

The training and validation metric and loss graphs for the YOLOv5m model are shown in figures. The number plate text extraction performed for 50 vehicles in Sri Lanka. The table 1 shows the testing results comparison with the OCR method, extracting number with OCR after performing preprocessing steps, extracting number with OCR after performing preprocessing steps and then applying ad-hoc algorithm which suits to Sri Lankan context.

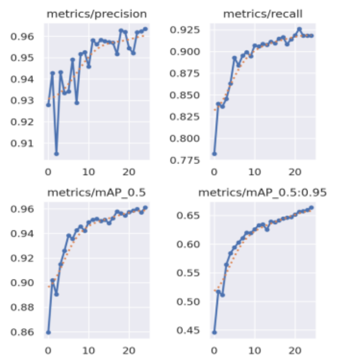


Fig. 20. Training metric for the YOLOv5m model

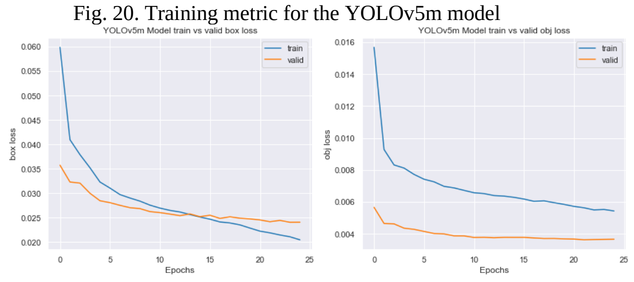


Fig. 21. Training vs validation loss in yolov5m model

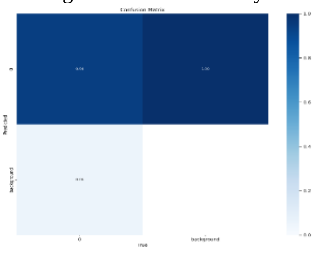


Fig. 22. Confusion metrics for trained model

TABLE I. NUMBER PLATE TEXT EXTRACTION TESTING RESULTS

Number of license plates	*Number of license plates*	*Correct Number plate detected after preprocessing and the OCR*	*Correct Number plate detected after preprocessing and Ad-hoc algorithm for Sri Lankan number plates and the OCR*
50	20 (40%)	27 (54%)	2 (84%)

D. *Discussion*

The traffic violation ticketing system streamlines the process of detecting and issuing tickets for traffic violations. It uses advanced technologies like YOLOv5 vehicle detection and DeepSORT vehicle tracking to detect violations in real-time. The system monitors traffic through cameras and sends relevant information to a centralized server for processing. The server then handles number plate detection using OCR algorithms, generating tickets via SMS. An online payment system is also included for convenient fine payments. If the number plate cannot be recognized, a mobile app alerts a nearby police officer, who can verify the violation, issue a ticket, and collect the fine.

The traffic violation ticketing system incorporates several resilience features to ensure smooth operation and accurate violation detection.

1) Number Plate Capturing Area for Vehicles: Traffic violation ticketing system uses camera-focused areas for effective number plate capture, enabling multiple attempts to recover number plate, even if not visible in the initial violation spot snapshot.

2) Multiple Attempts to Recover the Number Plate: The traffic violation ticketing system uses multiple attempts to recover number plate information, initially detecting the number plate from snapshots. However, initial snapshots may not provide enough information, so the system uses additional resources, including scanning within designated capturing areas, processing at least three carefully selected images to extract the number plate.

3) Informing the Nearest Police Officer: The system notifies the nearest police officer if a number plate cannot be recognized after multiple attempts, enabling manual verification and ticket issuance. It also identifies officers in the same street.

4) Identifying Camera View Changes and Recovering: The system monitors camera views and detects changes in field of view, ensuring accuracy and reliability. It recovers new views and updates configurations for violation detection.

E. Challenges

1) Number Plate Detection at Night Time: Nighttime visibility challenges number plate detection due to vehicle headlight intensity. Applying the illumination techniques to video frames while capturing the video can improve visibility at night.

2) Camera Angle Change: Camera angles and ROI configurations impact number plate detection accuracy; techniques like image rectification and perspective correction compensate for variations.

3) False violations till configured areas are recovered: Camera shakes or sudden movements can trigger false violations until the system recovers and recalibrates, causing misalignment between captured images and configured areas. Drivers can verify their actions via information sent in SMS.

4) 2D bounding boxes' constraint inaccurately capturing vehicle area: 2D bounding boxes for vehicle area capture limitations, sometimes recording larger ones. To address this, specific localization techniques were applied, such as reducing vehicle bounded area and considering bottom line coincidence in parking violations.

F. Future works

Enhance system flexibility by integrating additional violation detection modules, enabling detection and enforcement of a wider range of traffic violations beyond initial rules. Implement plugin architecture for easy addition and modification.

Implement hardware-based solution with cameras and edge devices for real-life usage, eliminating reliance on centralized servers and enabling faster processing and response times. This localized violation detection and ticketing process reduces dependency on external connectivity.

3D bounding boxes improve vehicle detection and estimation accuracy by capturing the object's volume and spatial extent. This allows for precise detection and estimation of vehicle boundaries, accommodating vehicles of different sizes and shapes. Utilizing LiDAR sensors or depth cameras enhances the violation detection process.

V. Conclusion

The traffic violation ticketing system automates the process of detecting and issuing tickets for traffic violations. This system is a promising solution to reduce the road traffic violations. The proposed system works resiliently by detecting violations in real-time using camera streams and issuing tickets to the vehicle owners. Furthermore, the system is configurable for varying sets of traffic rules, allowing it to adapt to changes in regulations over time with end user facilitations such as fine management and fine payment.
It includes traffic fine management, edge server, and centralized server components. The edge server uses the YOLOv5 algorithm for real-time vehicle detection, while the DeepSORT algorithm ensures continuous tracking. Separate algorithms are implemented for specific violations, such as parking, line cutting, U-turn, and one-way violations. The system uses a firebase real-time database for data storage and retrieval.

The centralized server uses advanced image processing and machine learning algorithms to extract number plate information from captured images, integrating with a vehicle registration database. The Google SQL database facilitates traffic fine management, storing violation-related data for efficient tracking, reporting, and processing of fine payments.

References

[1] S. Observer. (2022, January) Inferior road safety and discipline are an overall burden. [Online]. Available: https://www.sundayobserver.lk/2022/01/30/opinion/inferior-road-safety-and-discipline-are-overall-burden

[2] G. Docs. Police oic. [Online]. Available: https://drive.google.com/file/d/1fO0do-DmMK2MdHvm01trdSg0l3hQv/view?usp=sharing

[3] Intelligent traffic violation detection. [Online]. Available: https://www.researchgate.net/publication/355109949_Intelligent_Traffic_Violation_Detection

[4] A. Pramanik, S. Sarkar, and J. Maiti, "A real-time video surveillance system for traffic pre-events detection," Accid. Anal. Prev., vol. 154, no. 106019, p. 106019, 2021.

[5] A. Peiris, E. Eata, C. D. Athuraliya, and I. Jayasooriya, "Computer vision based approach for traffic violation detection," Kdu.ac.lk, Year. [Online]. Available: http://ir.kdu.ac.lk/bitstream/handle/345/2953/FOC%20136139.pdf?sequence=1&isAllowed=y

[6] R. J. Franklin and Mohana, "Traffic signal violation detection using artificial intelligence and deep learning," in 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 839–844.

[7] Ultralytics, "Yolov5: Object detection with pytorch," https://github.com/ultralytics/yolov5 , Year.

[8] L. OpenCV. (Year) Custom object detection training using yolov5. [Online]. Available: https://learnopencv.com/custom-object-detection-training-using-yolov5/

[9] "Indian vehicle dataset," Data set, 2023.

[10] "dataset-vehicles," Google Drive, retrieved February 8, 2023. [Online]. Available:

https://drive.google.com/drive/folders/1a-v4os2Ekr-IezLE-pGNJ7R0plZyf6bE

[11] "Vehicles-openimages object detection dataset," Data set, 2022.

[12] Yolov5, "Automatic-number-plate-recognition object detection dataset by yolov5," Roboflow, 2023, retrieved January 4, 2023. [Online]. Available: https://universe.roboflow.com/yolov5-pyvog/automatic-number-plate-recognition

[13] C. Platev1, "Car plate object detection dataset (v1, platev1) by car platev1," Roboflow, 2023, retrieved January 5, 2023. [Online]. Available: https://universe.roboflow.com/car-platev1/car-plate-dalcl/dataset/1

[14] cocotoyolo, "cocotoyolo object detection dataset (v6, 2023-01-07 4:58pm) by cocotoyolo," Roboflow, 2023, retrieved January 5, 2023. [Online]. Available: https://universe.roboflow.com/cocotoyolo-pwnbd/cocotoyolo-naltw/dataset/6

[15] B. Kilic, "plakalar object detection dataset by berkay kilic," Roboflow, 2023, retrieved January 3, 2023. [Online]. Available: https://universe.roboflow.com/berkay-kilic-dpfzs/plakalar