AI-Based Violation Analysis for Hotspot Detection and Enforcement Optimization

[Abdullah Altaf, Abdul Rahim, Ahmed, Moez]

ABSTRACT

This paper presents an AI-powered traffic violation detection system that automatically identifies and records common traffic rule violations, including red-light crossing, helmet absence, wrong-way driving, and zebra crossing violations. The system integrates object detection using YOLOv2-6m, number plate detection, image processing, and optical character recognition (OCR) to locate vehicles and extract license plate text. A key goal of the system is to support traffic authorities in finding high-violation areas, or hotspots, so that better decisions can be made about deploying traffic wardens and installing permanent camera infrastructure. The paper describes the full system design, the practical challenges that were encountered in real-world conditions, and the specific solutions that were developed using deep learning and computer vision techniques.

Keywords — *Traffic violation detection, YOLOv2-6m, PaddleOCR, StrongSORT, hotspot analysis, CCTV, computer vision, traffic management.*

I. INTRODUCTION

Traffic violations are a major cause of road accidents and urban congestion., Setting up permanent detection infrastructure at every location is expensive and not practical for many cities. This project addresses that problem by using artificial intelligence to automate violation detection through a single, low-cost Wi-Fi CCTV camera or mobile camera.

The system takes a video feed as input, processes each frame, detects vehicles, identifies violations, and reads number plates. All violation records are stored and analyzed to find hotspot areas where violations occur most often. The main purpose of this analysis is to help traffic authorities decide where to station wardens and where permanent camera installation is justified. This approach makes enforcement more efficient and reduces the cost of traffic management, especially in local urban environments.

II. SYSTEM OVERVIEW

The system works as a pipeline of four connected modules. Each module processes the output of the previous one, and together they handle the full workflow from raw video to a violation record.

- Vehicle Detection – Detects vehicles in each frame using YOLOv26m.
- Vehicle Tracking
- Number Plate Detection – Locates and crops the license plate region from each detected vehicle.
- Text Recognition (OCR) – Reads the plate text from the cropped region using PaddleOCR.
- Violation Detection – Identifies four types of violations: red-light crossing, zebra crossing violation, one-way driving, and helmet absence.

The video source is a single rear-facing CCTV camera. Using one camera for all modules creates trade-offs in image quality that are discussed in Section III.

III. METHODOLOGY

The system is designed as a sequential, modular pipeline that processes raw video input through four primary stages to generate violation records. Each module utilizes the output of the preceding stage to ensure comprehensive analysis.

A. Vehicle Detection and Multi-Object Tracking

YOLOv2-6m is applied to every incoming video frame for vehicle detection. StrongSORT is then used to assign and maintain unique IDs for each detected vehicle across consecutive frames. For each tracked vehicle, the system records center coordinates in every frame to build a movement trajectory, which is critical for direction-based violation detection.

B. Number Plate Recognition (ALPR)

To improve plate detection accuracy, the system first crops the vehicle bounding box and then runs custom trained YOLOv2-6m model for plate detection on that smaller, focused region. PaddleOCR is used to extract alphanumeric characters from the cropped plate. A character dictionary restricted to A–Z and 0–9 is applied during local dataset training to better handle local plate formats. A "Plate History" algorithm stores repeated predictions over time and selects the most frequent result to ensure consistent readings despite motion blur.

C. Traffic Signal and Color Detection

The system uses a position-based approach rather than relying solely on color, making it robust under varying lighting conditions. A Region of Interest (ROI) is defined in the upper part of the frame to reduce computational load. YOLOv2-6m detects the signal light in the first frame, and a CSRT tracker maintains its position across subsequent frames. The ROI is then converted to grayscale, blurred, and processed with adaptive thresholding to identify bright blobs. Morphological operations clean noise, followed by convex hull and contour analysis to filter true signal lights from irregular background shapes such as leaves. The final state is determined by the vertical position of the detected blob: top = Red, middle = Yellow, and bottom = Green.

D. Violation Detection Logic

Zebra crossing violations are logged only when the traffic signal state is red and a vehicle is detected occupying the crossing area. The crossing itself is identified using edge detection and a scoring system based on stripe alignment. For one-way violations, the system uses YOLOv2-6m for per-frame vehicle detection and StrongSORT to assign and maintain unique IDs. A direction angle is calculated from the tracked trajectory; for example, 270° corresponds to upward movement and 90° to downward movement. If the angle is opposite to the defined allowed flow direction for a minimum number of consecutive frames, the vehicle is flagged. This minimum frame threshold prevents noise in the tracking from generating false positives. This angle-based approach has an advantage over a fixed entry-line method because it can detect wrong-way movement anywhere in the frame, not only at a specific coordinate boundary.

E. Hotspot Analysis and Enforcement Application

Every detected violation is recorded with its type, location data, and timestamp. Over time, these records are aggregated to identify hotspot areas — locations where violations occur most frequently. The analysis output is intended to support two practical decisions: deployment of traffic wardens at high-frequency violation locations during peak hours, and prioritization of locations for permanent camera infrastructure investment. This data-driven approach allows authorities to act on evidence rather than assumption. Temporary camera deployment using a mobile or low-cost Wi-Fi CCTV unit can survey an area, generate violation statistics, and produce a justified basis for infrastructure spending.

Custom Dataset Development:

Number plate Detection:

The system achieved localized accuracy by transitioning from a general dataset to a custom dataset tailored for local traffic conditions. A base model was first trained on a public Kaggle dataset containing approximately 25,000 images over 20 epochs to establish a foundational understanding of plate characteristics. Using bounding box detections from local videos, 4,445 cropped vehicle images were then extracted and saved. The Kaggle-trained model predicted number plate coordinates within these local vehicle images to generate initial annotations. These vehicle images, combined with their corresponding plate bounding boxes, formed the final custom local dataset reflecting Pakistani plate styles. The YOLO model then underwent a final fine-tuning phase on this local dataset for an additional 12 epochs to optimize recognition for the target environment.

Number Plate Text Detection:

To improve text recognition accuracy for local number plates, a dedicated custom dataset was developed. Initially, a collection of 4,400 vehicle images was already available from the number plate detection module. A locally trained number plate detection model was applied to these images to extract cropped regions containing only the number plates. Each cropped image was then manually annotated with its corresponding number plate text to ensure high-quality ground truth labels. This labeled dataset was subsequently used to train and fine-tune the text detection/recognition model, enabling it to better adapt to variations in local plate formats, fonts, and image conditions.

IV. CHALLENGES FACED AND SOLUTIONS

A. Number Plate Detection

Because only one fixed camera is used for all modules, it cannot zoom in and focus on individual license plates as done by traffic violation authorities in Pakistan using multiple camera's. This created several practical problems: plates appeared blurry due to vehicle motion, lighting varied across the frame (shadows, glare), and plates were sometimes partially blocked or seen at an angle.

General object detection models did not perform reliably on local plate formats. The solution was to train and fine-tune YOLOv2-6m on a custom dataset of local number plates. Additionally, the system first crops the vehicle bounding box from the frame, then runs plate detection on that smaller region. This two-step approach significantly improved detection accuracy.

B. Text Recognition (OCR)

Extracting readable text from plates was one of the hardest parts of the project. Standard OCR tools failed because local plates use different fonts and character styles, images have low resolution due to distance, and blur or noise further degrades image quality.

PaddleOCR was selected as the recognition engine due to its flexibility and strong performance on custom character sets. A training dataset was built from local plate images, and a character dictionary restricted to A–Z and 0–9 was used during fine-tuning. Training the model on local data gave a clear improvement in accuracy over using a generic pre-trained OCR model.

C. Helmet Detection

Helmet detection is difficult because the helmet region is a small part of the overall frame, local helmet styles and colors vary widely, and the camera angle can cause partial occlusion. A dataset from Kaggle was used because no local dataset was available. While this dataset covers international helmet styles, it does not fully represent local variations, so accuracy was limited.

To compensate, the system first crops the rider region from the frame before running the helmet classifier. This reduces background noise and focuses the model on the relevant area, which improved detection performance despite the dataset limitation.

D. Traffic Signal Color Detection

Detecting the correct state of a traffic light (red, yellow, or green) from live video is challenging. Lighting conditions change throughout the day, the signal object is small in the frame, and background elements such as trees and leaves can produce false matches.

An initial approach using HSV color thresholds worked reasonably well in daylight but failed at night and in low-light conditions, when colors become inconsistent. A second, more robust approach was developed that does not depend on color alone. The steps are as follows:

- A Region of Interest (ROI) is defined in the upper part of the frame to reduce unnecessary computation.
- YOLOv2-6m detects the traffic light object, and a CSRT tracker maintains its position across frames.
- The ROI is converted to grayscale, blurred, and processed with adaptive thresholding to find bright blobs.
- Morphological operations clean up noise and connect fragmented regions.
- Convex hull analysis smooths blob shapes for more reliable measurements.
- Contour analysis filters candidates using geometric features: area, convexity, smoothness, and circularity. This step separates true signal lights from irregular shapes like leaves.
- The final signal state is determined by vertical position: top blob = red, middle blob = yellow, bottom blob = green. This position-based rule makes the system robust to color distortions.

E. Zebra Crossing Detection

Zebra crossings must be identified in the road frame before violations can be detected. White stripe patterns are hard to locate reliably because road markings fade over time, shadows and changing light alter appearance, and other road surfaces can produce similar patterns.

Edge detection was used to find strong boundaries, and contour-based filtering was applied to select candidate regions. A scoring system evaluated each candidate based on shape regularity and stripe alignment. Only regions that passed the score threshold were accepted as valid zebra crossings.

A zebra crossing violation is logged only when the traffic signal is red and a vehicle is detected occupying the crossing area. This conditional logic avoids false positives during normal green-light traffic flow.

V. RESULTS AND DISCUSSION

The system was evaluated against real-world footage captured from a single rear-mounted camera. Key accuracy metrics are summarized in Table I.

Component	Accuracy (%)
Zebra Crossing & Traffic Signal Violation	100
Traffic Light Detection	92
Number Plate Detection	91.5
OCR Character Recognition	67.23

TABLE I. SYSTEM ACCURACY METRICS

Number plate detection accuracy improved significantly after fine-tuning on local data and applying the two-step crop-then-detect pipeline. OCR accuracy on local plates was better with the trained PaddleOCR model than with generic pre-trained alternatives, particularly for plates with non-standard fonts.

Traffic signal detection using the position-based blob method was more stable across lighting conditions than the initial HSV color approach. The CSRT tracker maintained signal position reliably across frames, reducing flickering in the detected state. Helmet detection accuracy was acceptable but not optimal due to the reliance on an international dataset, and local helmet variety was a clear gap. One-way violation detection using StrongSORT and angle-based trajectory analysis produced low false-positive rates because of the multi-frame confirmation requirement. Zebra crossing detection performance was

dependent on the quality of the road marking; well-maintained markings were detected reliably, while heavily faded markings were sometimes missed.

VI. LIMITATIONS AND CHALLENGES

Several practical constraints affected the system during development and testing. First, the absence of local GPU resources required all training and fine-tuning to be carried out in cloud environments, which introduced limitations on iteration speed and dataset size. Second, a wired power source is required to operate a standard CCTV camera continuously in the field; to work around this during testing, a mobile camera was used as a substitute input device.

VII. CONCLUSION

This paper presented an AI-based traffic violation detection system built to run on a single, low-cost CCTV camera. The system detects four types of violations — red-light crossing, wrong-way driving, zebra crossing violations, and helmet absence — and reads vehicle number plates using a fine-tuned OCR model.

The main contribution of this work is practical: it provides a cheap, deployable tool for temporary violation monitoring that generates hotspot statistics. These statistics give traffic authorities a factual basis for deciding where to place wardens and where to build permanent camera infrastructure. This reduces overall enforcement costs while improving coverage in high-risk areas.

The biggest limitation is helmet detection accuracy, which can be improved by building a local dataset. Future work should also look at multi-camera setups to improve plate image quality and cover wider road sections. Extending the hotspot analysis with time-based patterns and violation type breakdowns would further increase the system's value for traffic management planning.

REFERENCES

[1] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," GitHub repository, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[2] PaddlePaddle Authors, "PaddleOCR: Awesome multilingual OCR toolkits based on PaddlePaddle," GitHub repository, 2020. [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR

[3] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "FairMOT: On the Fairness of Detection and Re-identification in Multiple Object Tracking," International Journal of Computer Vision, vol. 129, pp. 3069–3087, 2021.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," in Proc. IEEE ICIP, 2017, pp. 3645–3649.

[5] L. Du, Y. Zhao, Q. Zhang, and H. Yin, "StrongSORT: Make DeepSORT Great Again," IEEE Transactions on Multimedia, 2023.

[6] D. Bradski and A. Kaehler, Learning OpenCV 4: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, 2019.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.

[8] R. Laroca et al., "A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector," in Proc. IJCNN, 2018.