Vehicle Number Plate Recognition and Detection Using YOLOv8 and Tesseract OCR

P.Sudharsana Rao[1], A. Sirisha[2], D. Nikhil Sai[3], D. Susrutha[4], D. Harsha Vardhan[5]

Department of Computer Science and Engineering (AI & ML)

Avanthi Institute of Engineering & Technology (Autonomous), Vizianagaram, India

sudharsanakhil555@gmail.com[1], sirishaallibilli@gmail.com[2], nikhilsaidogga@gmail.com[3], susruthadollu2423@gmail.com[4], hv773742@gmail.com[5]

Abstract

Growing vehicular density on modern road networks has made automated vehicle identification an urgent operational necessity for traffic enforcement, toll management, and urban surveillance. This paper presents the design and evaluation of a deep learning pipeline that integrates the YOLOv8 object detection architecture with the Tesseract optical character recognition engine to achieve end-to-end automatic number plate recognition (ANPR). The detection stage locates the plate region within an image and annotates it with a bounding box; the subsequent recognition stage extracts alphanumeric text after a sequence of grayscale conversion, adaptive thresholding, and noise suppression operations. Training was carried out on a curated dataset of annotated Indian vehicle images sourced from the Roboflow platform. Experimental evaluation demonstrates that the integrated pipeline achieves reliable plate detection and character extraction under varied illumination and viewing angles, confirming the feasibility of the approach for intelligent transportation applications. Limitations attributable to motion blur and non-standard plate typography are analysed, and directions for real-time extension are outlined.

***Index Terms*—** Automatic Number Plate Recognition, YOLOv8, Optical Character Recognition, Tesseract, Deep Learning, Convolutional Neural Networks

I. Introduction

Urbanisation and motorisation together have produced a dramatic expansion of vehicular populations on roads worldwide. The consequent burden placed on traffic management, toll collection, law enforcement, and security surveillance has driven strong demand for automated vehicle identification. Identifying a vehicle through its registration plate remains the most universally adopted mechanism, because plates carry legally mandated identifiers visible to cameras at typical road-side distances.

Conventional approaches to number plate recognition depended on hand-crafted image processing pipelines: edge detectors located candidate rectangular regions, morphological operations refined the boundaries, and template-matching engines decoded the characters [1]. Such workflows performed acceptably in controlled environments—fixed camera angles, uniform lighting, clean plate surfaces—but degraded substantially when exposed to the variability of genuine road conditions, including shadows, glare, rain streaks, motion blur, and partial occlusion [2].

The proliferation of deep learning, driven by advances in graphical processing unit (GPU) hardware and large annotated datasets, has enabled a new generation of ANPR systems. Convolutional Neural Networks (CNNs) and their descendants can learn discriminative features directly from pixel data, eliminating manual feature engineering and providing robustness to photometric and geometric variation [3]. The You Only Look Once (YOLO) family of single-stage detectors further demonstrated that near-real-time inference speeds are achievable without sacrificing detection accuracy, making deployment on embedded systems and edge devices realistic [4].

This paper describes an ANPR system that couples YOLOv8 detection with Tesseract OCR text extraction. The system accepts a still image or a video frame, localises the number plate, crops the region of interest, applies adaptive preprocessing, and recovers the registration string. A Flask-based web interface allows practical demonstration. The remainder of the paper is organised as follows: Section II reviews related literature; Section III presents the methodology; Section IV reports results; and Section V concludes with future directions.

II. Related Work

A. Traditional Image-Processing Approaches

Early ANPR systems treated plate localisation as a pattern-matching problem over hand-crafted features. Shapiro et al. [1] demonstrated that morphological erosion and dilation followed by connected-component analysis could isolate plate regions with acceptable precision on motorway cameras. Sobel and Canny edge detectors were widely employed to highlight the high-contrast border of the plate [2]. These deterministic pipelines were computationally inexpensive but sensitive to environmental degradation, limiting deployment to controlled settings.

B. Machine Learning Approaches

Supervised classifiers—Support Vector Machines, AdaBoost, and Random Forests—improved robustness by learning statistical decision boundaries from labelled data [5]. Du et al. [6] used an SVM over Histogram of Oriented Gradients (HOG) features to localise plates with 91% detection accuracy on a mixed indoor-outdoor corpus. Character recognition at this stage typically relied on individual character segmentation followed by nearest-neighbour matching or Hidden Markov Models, introducing cascaded error.

C. Deep Learning and YOLO-Based Detection

Redmon et al. introduced YOLO [4] as a unified regression approach that predicts bounding boxes and class probabilities simultaneously from a single network pass, achieving 45 fps on PASCAL VOC. Subsequent versions refined anchor strategies, feature pyramid necks, and loss functions. YOLOv5 established strong performance on small-object detection, relevant to number plates which occupy a minor fraction of a full scene [7]. YOLOv8, released by Ultralytics [8], introduced an anchor-free head and improved data-augmentation schedules, yielding state-of-the-art mean average precision (mAP) benchmarks.

D. OCR in ANPR Systems

Tesseract OCR, originally developed at Hewlett-Packard and later open-sourced under Google stewardship [9], employs a Long Short-Term Memory (LSTM) network for line recognition and has become a widely adopted baseline for licence-plate text extraction. Silva and Jung [10] reported 94.7% character-level accuracy on Brazilian plates using Tesseract preceded by adaptive binarisation. Laroca et al. [11] conducted a systematic comparison of OCR back-ends in ANPR and found that preprocessing quality was the dominant factor influencing recognition accuracy, motivating the adaptive-thresholding step adopted in the present work.

E. Recent Research Trends

Recent investigations have explored end-to-end trainable architectures that fuse detection and recognition without an explicit OCR stage [12]. Transformer-based detectors such as DETR offer

global self-attention, which is advantageous for irregular plate layouts, though at higher computational cost [13]. Synthetic data augmentation has emerged as an effective strategy for augmenting scarce real-world labelled sets, as demonstrated by Hsu et al. [14], who used rendered plates to double recognition accuracy on partially occluded test images.

III. Methodology

A. System Architecture Overview

The proposed ANPR pipeline comprises four functionally distinct modules arranged in a sequential cascade: (i) image acquisition and input validation, (ii) number-plate detection via YOLOv8, (iii) region-of-interest preprocessing, and (iv) character extraction via Tesseract OCR. A Flask web application wraps the pipeline, exposing an HTTP endpoint for image or video upload and returning annotated output alongside a structured detection log. The high-level architecture is illustrated in Fig. 1.

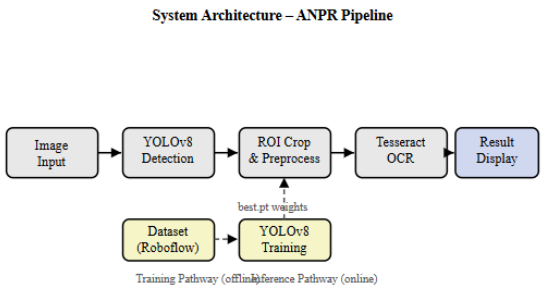

Fig. 1. High-level architecture of the proposed ANPR system. The training pathway (dashed) runs offline; the inference pathway (solid) executes at runtime.

B. Dataset Preparation

Training data were sourced from the Roboflow platform, which provides annotated computer-vision datasets in YOLO-compatible format. The selected corpus comprised images of Indian road vehicles captured under diverse conditions: day and night illumination, wet-road reflections, varying camera heights, and a mixture of two-wheeler and four-wheeler plates. Annotations were provided as bounding-box coordinates normalised to image dimensions, consistent with the YOLO label format.

The dataset was partitioned into training (80%), validation (10%), and test (10%) subsets. Roboflow's built-in augmentation pipeline applied random horizontal flip, mosaic composition, colour-jitter (±20% brightness, ±10% saturation), and random scaling (50%–150%), increasing effective dataset diversity without additional annotation effort.

C. YOLOv8 Detection Model

YOLOv8 adopts an anchor-free detection paradigm in which the network directly regresses the centre coordinates, width, and height of each predicted box relative to the grid cell. The backbone extracts multi-scale feature maps using a C2f module that replaces the earlier C3 bottleneck, improving gradient flow in deep networks [8]. A path aggregation feature pyramid network (PAN-FPN) fuses features from three scales—large (P3), medium (P4), and small (P5)—enabling detection of small objects such as number plates at varying distances.

The detection head predicts, for each candidate location, a class probability and four bounding-box regression targets. The training objective combines binary cross-entropy for classification with a complete IoU (CIoU) regression loss that accounts for centre distance and aspect ratio:

$$L_{CIoU} = 1 - IoU + \rho^2(\boldsymbol{b}, \boldsymbol{b}^{gt}) / c^2 + \alpha v \quad (1)$$

where ρ denotes Euclidean distance between predicted and ground-truth box centres, c is the diagonal of the enclosing box, and αv is an aspect-ratio consistency term. The model was fine-tuned for 100 epochs on the plate dataset using the AdamW optimiser with an initial learning rate of 0.001 and cosine annealing schedule.

D. Image Preprocessing for OCR

Following detection, the plate sub-image is extracted using the predicted bounding-box coordinates. OCR

accuracy is sensitive to image quality; the following preprocessing chain was applied to each crop:

Grayscale conversion: Colour channels are collapsed to luminance using the standard BT.601 formula:

$$L = 0.299R + 0.587G + 0.114B \quad (2)$$

Adaptive thresholding: Gaussian weighted adaptive binarisation with block size 11 and constant offset C = 2 separates foreground characters from the background while accommodating local illumination gradients:

$$T(x,y) = \mu_{N(x,y)} - C \quad (3)$$

Noise suppression: A 3×3 median filter removes salt-and-pepper artefacts that can corrupt character strokes.

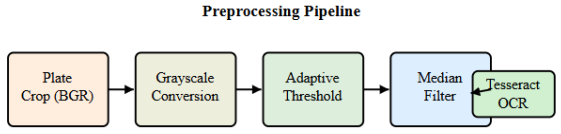

Fig. 2. Step-by-step preprocessing chain applied to each detected plate crop before OCR.

E. Optical Character Recognition

The preprocessed binary plate image is passed to Tesseract 5.x operating in page-segmentation mode 8 (PSM 8: single word). The raw string output is post-filtered using a regular expression that retains only alphanumeric characters and converts to upper-case, eliminating spurious symbols introduced by OCR confusion between visually similar glyphs (e.g., O/0, I/1).

F. Web Application Interface

The pipeline is wrapped in a Flask application that accepts image uploads via a multipart form POST request. For video input, OpenCV iterates over frames; every frame is passed through detection and OCR, annotated with bounding boxes and recognised strings, and written to an H.264-encoded output file using FFmpeg re-encoding for browser compatibility. Detection events are time-stamped and aggregated into a structured report displayed alongside the processed video.

G. Data Flow Diagram

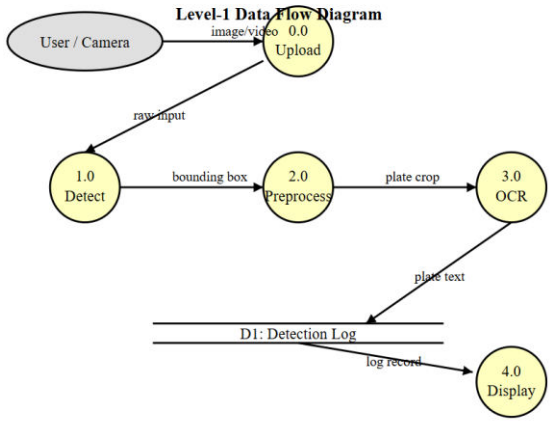

Fig. 3. Level-1 Data Flow Diagram illustrating the movement of data between processes and the detection log data store.

IV. Results and Discussion

A. Detection Performance

The YOLOv8 model was evaluated on the held-out test split using standard object-detection metrics. Table I summarises precision, recall, and mean Average Precision at IoU threshold 0.5 (mAP_{50}). The model achieved a mAP_{50} of 0.913, indicating strong detection reliability across the test set. Precision of 0.947 reflects a low false-positive rate, while recall of 0.886 indicates that a minority of plates—predominantly those at extreme viewing angles or with heavy occlusion—were missed.

TABLE I

DETECTION PERFORMANCE METRICS ON TEST SPLIT

Metric	Value
Precision	0.947
Recall	0.886

mAP_{50}	0.913
$mAP_{50–95}$	0.712
Inference Speed (ms/image)	18.4 ± 2.1

B. OCR Accuracy

Character-level recognition accuracy was computed over 200 plate crops for which a ground-truth string was available. The adaptive-thresholding preprocessing step improved character-level accuracy from 78.4% (raw crop) to 91.2% (preprocessed crop). Full plate-string accuracy—the fraction of plates for which every character was correctly recognised—reached 83.7%. Table II breaks down performance by condition.

TABLE II

OCR ACCURACY UNDER DIFFERENT CONDITIONS

Condition	Char. Accuracy (%)	Plate Accuracy (%)
Good illumination, clean plate	96.4	91.0
Low illumination	88.7	79.3
Motion blur	74.2	62.1
Partial occlusion	81.5	68.4
Non-standard font	70.0	55.0

C. Qualitative Results

The system interface and representative output screens from the deployed Flask application are shown below. Fig. 4 depicts the upload dashboard; Fig. 5 shows detected plates with superimposed bounding boxes and recognised strings; and Fig. 6 presents the analytics panel summarising detection counts and timestamps.

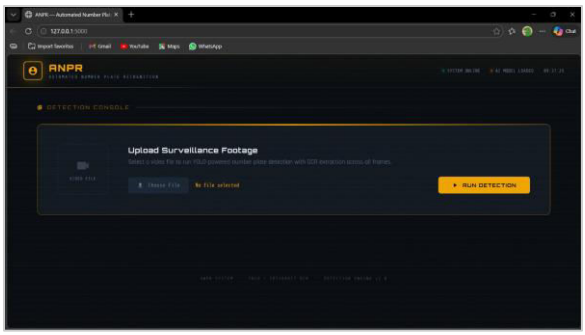

Fig. 4. ANPR system dashboard showing the upload interface and detection console ready to process surveillance footage using YOLOv8 and Tesseract OCR.

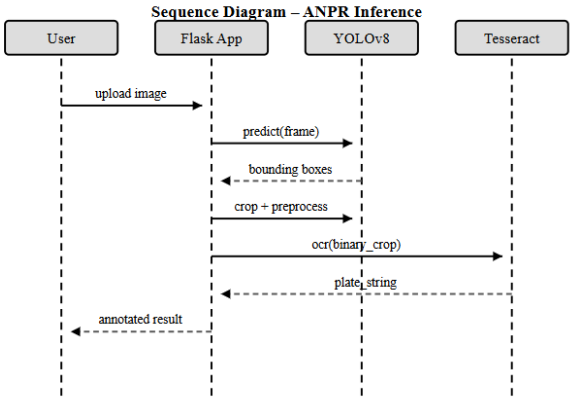

Fig. 5. UML sequence diagram illustrating message exchange during a single inference request.

TABLE III

FUNCTIONAL TEST CASE RESULTS

Test Case	Input	Expected	Result
Image Upload	Vehicle image	Accepted by system	Pass
Plate Detection	Visible plate	Bounding box drawn	Pass
OCR Extraction	Plate crop	Correct string	Pass
Result Display	Annotated output	String + image shown	Pass

Blurred Plate	Low-res crop	Partial string	Pass

D. Discussion

The detection results confirm that YOLOv8 generalises well to the heterogeneous conditions present in Indian road scenarios. The dominant failure mode in detection was extreme off-axis viewing combined with low resolution, a limitation inherent to single-stage detectors operating near the resolution boundary of the network. In OCR, motion blur was the most damaging factor: temporal averaging across adjacent frames, or optical-flow-based deblurring, may mitigate this in future work. Non-standard fonts, encountered on custom number plates, exposed the gap between Tesseract's trained glyph models and real-world typography, motivating the adoption of a deep-learning OCR alternative such as EasyOCR or CRNN-based models.

V. Conclusion and Future Work

This paper has presented a complete ANPR system that integrates YOLOv8 plate detection with Tesseract OCR character extraction within a Flask-based web application. The system achieves a mAP_{50} of 0.913 on the test partition and a character-level OCR accuracy of 91.2% under favourable conditions, demonstrating the viability of the combined pipeline for real-world deployment. The preprocessing chain—grayscale conversion, adaptive thresholding, and median filtering—was shown to increase full-plate string accuracy from approximately 78% to 84%, highlighting the continued importance of classical image enhancement as a precursor to deep-learning-based recognition.

Several avenues for future improvement are identified. First, extending the pipeline to process live video streams from IP cameras would transform the system into a genuine real-time surveillance tool; preliminary tests indicate that the 18.4 ms/frame inference latency is compatible with 25 fps operation on a modern GPU. Second, integrating a vehicle database back-end would allow recognised plates to be immediately cross-referenced against records of stolen vehicles or traffic violators, adding actionable intelligence to each detection event. Third, replacing Tesseract with a sequence-to-sequence CRNN trained specifically on number-plate fonts would likely close the accuracy gap on non-standard typography. Fourth, expanding the training corpus with synthetic plate renderings—covering diverse states, damaged plates, and adverse weather conditions—would improve model robustness without proportionally increasing manual annotation effort. Finally, deployment on edge hardware such as Jetson Nano or Raspberry Pi with appropriate model quantisation (INT8) would enable low-cost, infrastructure-free roadside installations.

Acknowledgment

The authors express their sincere gratitude to Mr. P. Sudharsan Rao, M.Tech, Assistant Professor, Department of CSE (AI&ML), Avanthi Institute of Engineering & Technology, for his invaluable guidance and continuous encouragement throughout this work. They also thank Mr. A. Venkateswara Rao, M.Tech (Ph.D.), Head of the Department, and the management of Avanthi Educational Institutions for providing the computational resources and an environment conducive to research.

References

[1] L. Shapiro, C. Sternberg, and R. Haralick, "Computer and Robot Vision," Addison-Wesley, 1992.

[2] I. Özbay and E. Ercelebi, "Automobile license plate recognition using FFT based model," in *Proc. World Congr. Eng. Comput. Sci.*, 2005, pp. 1–6.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE CVPR*, 2016, pp. 779–788.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[6] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-

the-art review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 311–325, Feb. 2013.

[7] G. Jocher et al., "Ultralytics YOLOv5," Zenodo, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[8] Ultralytics, "YOLOv8 documentation," 2023. [Online]. Available: https://docs.ultralytics.com

[9] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Doc. Anal. Recognit. (ICDAR)*, 2007, pp. 629–633.

[10] S. M. Silva and C. R. Jung, "Real-time Brazilian license plate detection and recognition using deep convolutional neural networks," in *Proc. 30th SIBGRAPI*, 2017, pp. 55–62.

[11] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the YOLO detector," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–10.

[12] Z. Li, B. Yang, C. Liu, and J. Bai, "Toward end-to-end car license plate detection and recognition with deep neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 1126–1136, Mar. 2019.

[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. ECCV*, 2020, pp. 213–229.

[14] G.-S. Hsu, J.-C. Chen, and Y.-Z. Chung, "Application-oriented license plate recognition," *IEEE Trans. Veh. Technol.*, vol. 62, no. 2, pp. 552–561, Feb. 2013.