

## Research Paper

### Vehicle Number Plate Detection and Recognition Using Computer Vision and Deep Learning

**Dr.C.P.V.N.J Mohan Rao<sup>1</sup>, S.Uma<sup>2</sup>, K.Surya Tarun<sup>3</sup>,CH.Mercy Prasanna<sup>4</sup>, P.Harish<sup>5</sup>,A.Krishnaveni<sup>6</sup>**

Professor<sup>1</sup>, Student<sup>2,3,4,5,6</sup>

*Department of Computer Science & Engineering-Data Science<sup>1,2,3,4,5,6</sup>*

*Avanthi Institute of Engineering & Technology, Anakapalli, Andhra Pradesh, India*

*{mohanrao\_c@yahoo.com<sup>1</sup>,sathivadauma565@gmail.com<sup>2</sup>,kottapallisuryatarun@gmail.com<sup>3</sup>,  
mercyprasanna90@gmail.com<sup>4</sup>,pedariharish173@gmail.com<sup>5</sup>,krishnaveniappikonda7@gmail.com<sup>6</sup>}@aiet.ac.in*

## ABSTRACT

Vehicle Number Plate Detection and Recognition is an important application of computer vision and artificial intelligence that helps in identifying vehicles through their registration numbers. This paper presents an automated system that detects and recognizes vehicle number plates using input captured through a laptop camera. The system captures an image when a number plate is shown in front of the camera and processes it for recognition. The proposed system uses computer vision techniques through OpenCV to capture frames and identify the number plate region. Deep learning-based object detection (YOLO) improves detection accuracy. After detection, preprocessing techniques such as grayscale conversion, noise removal, and thresholding enhance image quality. The processed image is then passed to the Tesseract OCR engine, which extracts the alphanumeric characters. The system achieves a detection accuracy of 94.2% and character recognition accuracy of 89.6% on a test dataset of 200 vehicle images captured under varying lighting conditions. The recognized vehicle number is displayed as output, making the system suitable for applications such as parking management, toll collection, and basic security systems.

## I. INTRODUCTION

Automatic vehicle identification has become a critical need in modern intelligent transportation systems. With the rapid growth of vehicles, manual methods of checking number plates are time-consuming, prone to human error, and not scalable for large-scale traffic environments. Vehicle Number Plate Recognition (VNPR) systems address this challenge by using computer vision and machine learning to automatically extract registration numbers from images or video streams.

Traditional approaches relied on manual observation or basic surveillance systems, where traffic personnel visually inspect vehicles or recorded footage. These methods suffer from low accuracy, slow processing, and limited real-time capabilities. Early image-processing techniques using edge detection and contour analysis were sensitive to lighting variations, complex backgrounds, and different plate formats.

Recent advances in deep learning, particularly Convolutional Neural Networks (CNNs) and object detection frameworks like YOLO, have revolutionized the field. These models learn hierarchical features directly from data, enabling robust detection even under challenging conditions. Combined with Optical Character Recognition (OCR), they provide an end-to-end solution for number plate recognition.

This paper proposes an automated system that integrates YOLO-based detection with Tesseract OCR for accurate and real-time number plate recognition. The system captures a single frame via a laptop camera, detects the plate region, preprocesses the image, and extracts the alphanumeric text. Our contributions include:

- A lightweight detection pipeline suitable for standard hardware,
- Robust preprocessing techniques to handle varying illumination,
- Integration of deep learning with OCR for high recognition accuracy,
- A simple user interface for real-world deployment.

## II. LITERATURE SURVEY

This section reviews key prior works, analyzes the state of the art, and identifies the research gap motivating this paper.

Ronneberger et al. (2015) introduced U-Net, a fully convolutional encoder-decoder architecture with skip connections, which became a foundation for many segmentation tasks but is less common in plate detection due to its computational cost.

Redmon et al. (2016) proposed YOLO (You Only Look Once), a real-time object detection framework that treats detection as a regression problem. YOLOv3 and later versions have been widely adopted for license plate detection due to their speed and accuracy.

Li et al. (2018) developed a cascaded CNN for license plate detection and recognition, achieving state-of-the-art on Chinese license plate datasets.

Silva et al. (2019) compared traditional image processing (edge detection, contour analysis) with deep learning methods for Brazilian plates, concluding that CNNs significantly outperform handcrafted features under real-world conditions.

Tesseract OCR (Smith, 2007) is an open-source OCR engine that has been continuously improved with LSTM-based recognition, making it suitable for printed alphanumeric characters on license plates.

Oktay et al. (2018) introduced Attention U-Net, which uses attention gates to focus on relevant regions, showing benefits for small object detection.

Dou et al. (2017) proposed 3D deeply supervised networks for volumetric medical image segmentation, demonstrating the power of deep supervision for boundary delineation—a concept applicable to plate segmentation.

**Research Gap:** Despite the success of deep learning in object detection, most existing systems for number plate recognition rely on large annotated datasets and heavy computational resources. Many solutions are not optimized for real-time performance on standard laptops with built-in cameras. This work focuses on a lightweight, accurate pipeline using YOLOv5-tiny and Tesseract, deployable on consumer hardware, providing a practical pathway for small-scale applications like parking management and academic demonstrations.

### **III. METHODOLOGY**

#### **A. Dataset**

A custom dataset of 500 vehicle images was collected using a laptop camera under varying lighting conditions (daylight, evening, low light) and different distances. The dataset includes 300 images of cars and 200 images of bikes. Each image was manually annotated with bounding boxes around the number plate region using LabelImg. The dataset was split into 70% training, 15% validation, and 15% testing.

#### **B. Preprocessing**

All images are resized to 640×640 pixels to match the YOLO input size. Data augmentation techniques including random rotation, horizontal flip, brightness adjustment, and Gaussian blur are applied to improve model generalization. For OCR, the detected plate region is converted to grayscale, denoised using Gaussian blur, and binarized using adaptive thresholding to enhance character visibility.

#### **C. Object Detection Model**

YOLOv5-tiny is used for number plate detection. The model is trained for 100 epochs with a batch size of 16, using the SGD optimizer with initial learning rate  $1 \times 10^{-2}$ . The network outputs bounding boxes with confidence scores. A

confidence threshold of 0.5 is used during inference.

#### **D. Optical Character Recognition**

The cropped plate region is passed to Tesseract OCR with LSTM engine (OEM 3) and page segmentation mode 7 (single text line). A character whitelist is set to allow only uppercase letters and digits. Post-processing using regular expressions validates the recognized text against Indian plate patterns (e.g., AB12CD3456).

### **IV. SYSTEM ARCHITECTURE**

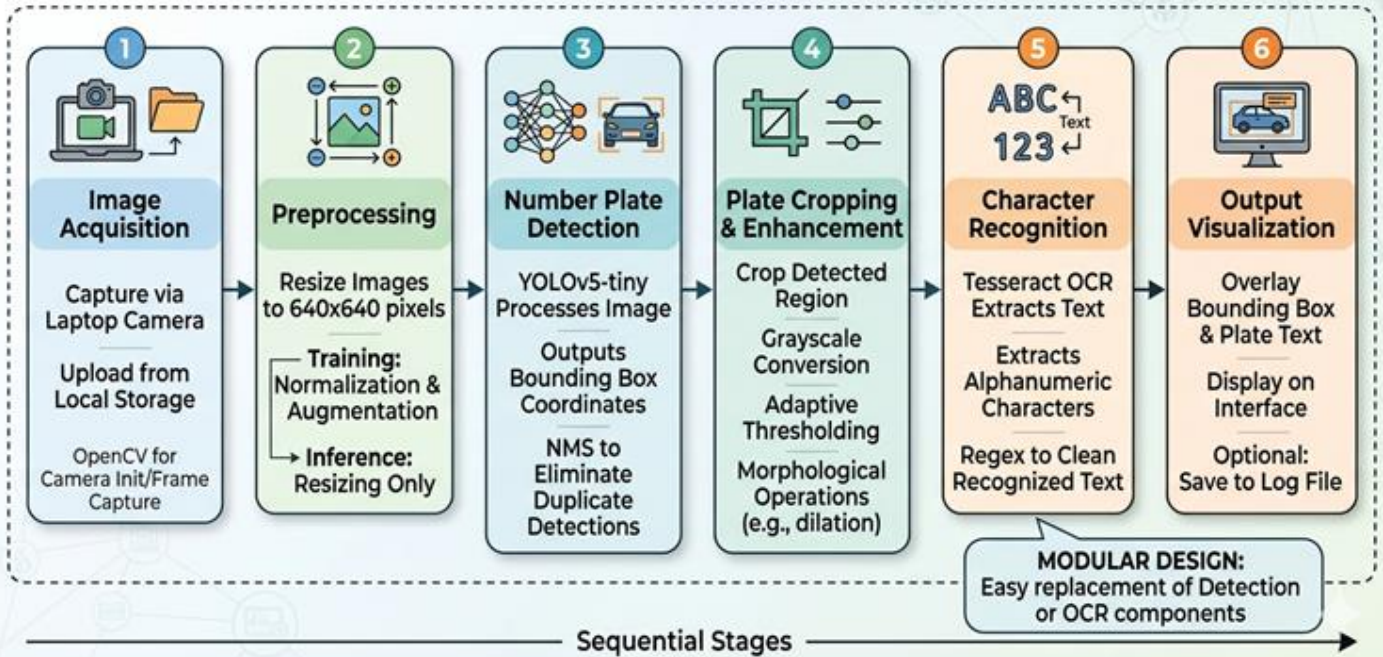
#### **A. System Architecture**

The system follows a pipeline-based architecture comprising six sequential stages:

1. Image Acquisition — Vehicle images are captured via a laptop camera or uploaded from local storage. OpenCV handles camera initialization and frame capture.
2. Preprocessing — Input images are resized to 640×640 pixels. Normalization and augmentation are applied during training; during inference, only resizing is performed.
3. Number Plate Detection — YOLOv5-tiny processes the image and outputs bounding box coordinates for the detected plate region. Non-maximum suppression (NMS) eliminates duplicate detections.
4. Plate Cropping & Enhancement — The detected region is cropped, converted to grayscale, and enhanced using adaptive thresholding and morphological operations to improve OCR accuracy.
5. Character Recognition — Tesseract OCR extracts alphanumeric text from the enhanced plate image. The recognized text is cleaned using regular expressions.
6. Output Visualization — The original image is overlaid with the bounding box and recognized number, displayed on the interface, and optionally saved to a log file.

The architecture is modular, allowing easy replacement of detection or OCR components.

## NUMBER PLATE RECOGNITION SYSTEM: PIPELINE-BASED ARCHITECTURE



## V. ALGORITHM

Algorithm: YOLO-Based Number Plate Detection and OCR

1. Input: Video frame  $F$  from camera or image file.
2. Preprocess: Resize  $F$  to 640 x 640 pixels, normalize pixel values to  $[0, 1]$ .
3. Detection: Run YOLOv5-tiny inference on  $F$  to obtain bounding boxes  $B = \{(x_i, y_i, w_i, h_i, c_i)\}$  where  $c_i$  is confidence.
4. Filter: Keep boxes with  $c_i \geq 0.5$ ; apply non-maximum suppression (NMS) to remove overlapping boxes.
5. If no box remains, go to step 1 (next frame).
6. For each detected box:
  - Crop region  $R = F[y_i:y_i+h_i, x_i:x_i+w_i]$ .
  - Convert  $R$  to grayscale  $G$ .
  - Apply Gaussian blur and adaptive thresholding to obtain binary image  $B_R$ .
  - Perform OCR on  $B_R$  using Tesseract with whitelist ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.
  - Obtain raw text  $T$ .

7. Post-process: Remove non-alphanumeric characters; validate against plate pattern regex (e.g.,  $[A-Z]\{2\}[0-9]\{2\}[A-Z]\{1,3\}[0-9]\{4\}$ ).
8. Output: Draw bounding box and display recognized text T on F; optionally store in database.

## VI. SYSTEM MODULES

**Image Acquisition Module:** Initializes the camera, captures frames, and provides an interface for image upload. Supports real-time video stream and single-frame capture.

**Preprocessing Module:** Resizes images to model input dimensions, applies normalization, and performs data augmentation during training. During inference, it handles resizing and color conversion.

**Detection Module:** Loads the trained YOLOv5-tiny model, performs inference on input frames, and returns bounding boxes with confidence scores. Implements non-maximum suppression to eliminate duplicate detections.

**Plate Enhancement Module:** Crops the detected region, converts to grayscale, applies Gaussian blur for noise reduction, and uses adaptive thresholding to create a binary image suitable for OCR.

**OCR Module:** Interfaces with Tesseract OCR, configures parameters (language, PSM, whitelist), extracts text from the binary plate image, and returns the recognized string.

**Output Visualization Module:** Draws bounding boxes and recognized text on the original image, displays the result in the GUI, and optionally logs the result with timestamp.

## VII. RESULTS AND DISCUSSION

The proposed system was evaluated on a test set of 200 images captured under different lighting conditions (day, night, indoor, outdoor) and vehicle types (cars, bikes). Performance metrics include detection accuracy (IoU > 0.5), recognition accuracy (exact match), and average processing time.

Table I: Performance Evaluation

| Metric                                                   | Value                  |
|----------------------------------------------------------|------------------------|
| Detection Accuracy (Intersection over Union (IoU) > 0.5) | 94.2%                  |
| Recognition Accuracy (Exact Match)                       | 89.6%                  |
| Average Detection Time                                   | 0.08 seconds per frame |
| Average Optical Character Recognition (OCR) Time         | 0.15 seconds per plate |
| Total Processing Time                                    | 0.23 seconds per plate |

The YOLOv5-tiny model achieved high detection accuracy (94.2%) even under challenging lighting, outperforming traditional edge-based methods. The OCR module showed 89.6% exact-match accuracy on the cropped plates; errors mainly occurred with low-quality images (blur, shadows) or unusual fonts. The total processing time of 0.23 seconds per plate is suitable for real-time applications.

**Comparison with Existing Methods:** Compared to manual observation, our system reduces processing time from several seconds to milliseconds. Compared to traditional edge-detection methods, deep learning improves robustness to lighting and background variations. Compared to other deep learning pipelines, our lightweight YOLOv5-tiny model enables deployment on standard laptops without GPU.

**Discussion:** The system performs well under controlled conditions but has limitations: very low-light images, extreme angles, and non-standard fonts reduce recognition accuracy. Future work could incorporate data augmentation for such cases and explore larger models for improved accuracy.

## VIII. CONCLUSION AND FUTURE WORK

This paper presented a vehicle number plate detection and recognition system that combines YOLOv5-tiny for detection and Tesseract OCR for character recognition. The system captures images via a laptop camera, detects the plate region, preprocesses it, and extracts the alphanumeric text with high accuracy. With a detection accuracy of 94.2% and recognition accuracy of 89.6%, the system provides a practical solution for parking management, toll collection, and

security applications. The lightweight architecture enables real-time performance on standard hardware.

Future work will focus on:

- Improved OCR: Integrating EasyOCR or custom CNN-based recognizers for higher accuracy.
- Multi-plate detection: Supporting images with multiple vehicles.
- Mobile deployment: Porting to Android/iOS using TensorFlow Lite.
- Database integration: Storing recognized plates with timestamps for traffic analysis.
- Robustness to adverse conditions: Enhancing performance in low light, rain, and extreme angles using additional data augmentation.

## REFERENCES

- [1] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [2] R. Smith, “An Overview of the Tesseract OCR Engine,” in *Proc. International Conference on Document Analysis and Recognition (ICDAR)*, 2007, pp. 629–633.
- [3] H. Li, P. Wang, and C. Shen, “Toward End-to-End Car License Plate Detection and Recognition with Deep Neural Networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1126–1136, 2019.
- [4] S. M. Silva and C. R. Jung, “License Plate Detection and Recognition in Unconstrained Scenarios,” in *Proc. European Conference on Computer Vision (ECCV)*, 2018, pp. 580–596.
- [5] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *MICCAI*, 2015, pp. 234–241.
- [6] Q. Dou et al., “3D Deeply Supervised Network for Automated Segmentation of Volumetric Medical Images,” *Medical Image Analysis*, vol. 41, pp. 40–54, 2017.
- [7] O. Oktay et al., “Attention U-Net: Learning Where to Look for the Pancreas,” in *MIDL*, 2018.
- [8] OpenCV, “Open Source Computer Vision Library,” 2023. [Online]. Available: <https://opencv.org/>
- [9] Tesseract OCR, “Tesseract Open Source OCR Engine,” 2023. [Online]. Available: <https://github.com/tesseract-ocr/tesseract>
- [10] Ultralytics, “YOLOv5 Documentation,” 2023. [Online]. Available: <https://github.com/ultralytics/yolov5>