International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

# Deep Learning-Based Autonomous Driving System with OpenCV Integration

**Yashraj S. Dube[1], Rushikesh R. Shinde[2], Akshay I. Chavhan[3]**

[1,2,3]Student, Department of Electronics and Computer Engineering, Amrutvahini College of Engineering

**Abstract**

This paper presents the design and development of a cost-efficient, real-time autonomous driving prototype that leverages Raspberry Pi and deep learning techniques for intelligent navigation. The proposed system integrates the YOLOv5 object detection framework with OpenCV-based lane detection and a lightweight CNN for traffic light classification. An ultrasonic sensor module is used for obstacle proximity awareness, and all modules are combined into a unified architecture optimized for execution on embedded hardware. A Streamlit-based dashboard provides interactive feedback and monitoring. The system demonstrates strong performance in terms of detection accuracy and response time, validating its potential for use in low-cost driver assistance applications and retrofitting older vehicles.

**Keywords:** Autonomous Vehicle, Deep Learning, OpenCV, YOLO, Raspberry Pi, Lane Detection, Traffic Sign Recognition, Obstacle Detection

## 1. Introduction

Advancements in embedded computing and computer vision have opened new avenues in the field of autonomous vehicle systems. However, many of the commercially available solutions remain prohibitively expensive and depend on high-end hardware such as GPUs and LiDAR sensors. In contrast, this project focuses on creating a lightweight, modular, and affordable autonomous driving system using Raspberry Pi, capable of performing critical perception tasks without relying on high computational resources.

The system is designed to interpret visual cues from the road using a Raspberry Pi Camera, applying OpenCV algorithms for lane detection and deploying YOLOv5 for real-time object and traffic sign identification. Traffic light recognition is achieved through a compact convolutional neural network (CNN) trained specifically for this task. The use of ultrasonic sensors enables basic obstacle detection, further enhancing vehicle awareness. To ensure ease of monitoring and interaction, a custom-built Streamlit dashboard visualizes outputs from all modules in real time.

By combining deep learning and classical vision techniques on a single-board computer, the proposed system demonstrates a practical implementation of autonomous driving principles tailored for academic, research, and budget-sensitive applications. This work aims to bridge the gap between theoretical exploration and real-world application in intelligent transportation systems.

.

## 2. Literature Review

Autonomous vehicle perception systems have evolved significantly with the adoption of deep learning and computer vision. YOLO (You Only Look Once) has become a dominant framework in real-time object detection due to its balance of speed and accuracy. Previous research also highlights the effectiveness of OpenCV in implementing vision-based tasks such as lane tracking. For instance, Pathan et al. [1] implemented a lightweight recurrent neural network for NLP tasks, emphasizing reduced computational requirements. This project extends similar lightweight strategies into the autonomous driving domain by integrating YOLO with traditional computer vision pipelines on a Raspberry Pi.

Related work in this domain includes:

- Redmon et al. developed the original YOLO architecture for real-time object detection, demonstrating its efficiency compared to region proposal-based methods. It has been widely used for vehicle and pedestrian detection in autonomous navigation.
- Zhao et al. (2019) proposed an end-to-end lane detection network based on convolutional neural networks and curve fitting. Their method showed robustness to occlusions and varying lighting conditions, similar to the lane detection pipeline used in this research.
- Bochkovskiy et al. introduced YOLOv4, which combines speed and accuracy improvements by integrating techniques like CSPDarknet and Spatial Pyramid Pooling. YOLOv4-tiny, a lightweight version used in our project, was specifically designed for embedded systems.
- Teichman and Thrun (2011) explored object tracking for 3D recognition in autonomous vehicles using visual inputs and classification algorithms. Their work emphasized the importance of integrating spatial awareness into low-compute systems.
- Kiran et al. (2021) demonstrated a Raspberry Pi-based autonomous vehicle capable of lane and traffic signal detection using a combination of classical vision algorithms and deep learning models, reinforcing the feasibility of using affordable hardware for such applications.
- Mohan and Krishna (2022) implemented a real-time driver assistance system using YOLOv3 and OpenCV on NVIDIA Jetson Nano, achieving high accuracy in pedestrian and signboard detection. Their study confirms the effectiveness of YOLO-based models in low-power embedded platforms.

The current project builds upon these contributions by combining a modular, lightweight deep learning framework with traditional image processing techniques, all optimized for real-time execution on a Raspberry Pi without requiring GPU acceleration.

## 3. Problem Statement

Existing autonomous driving systems are generally expensive and hardware-intensive, restricting their implementation in academic and budget-conscious projects. There is a clear need for a system that can:

- Perform reliable lane, traffic sign, and object detection.
- Operate in real-time using minimal hardware.
- Be affordable and deployable in existing or low-end vehicles.

This research addresses the gap by designing a deep learning-based system using Raspberry Pi and OpenCV, combining accuracy with real-time performance in a constrained computing environment.

## 4. System Architecture

The proposed system includes the following components:

- **Image Acquisition**: A Pi Camera mounted on the vehicle continuously captures live video frames which serve as the input for further processing.
- **Lane Detection**: OpenCV algorithms such as Gaussian blur, Canny edge detection, and Hough Line Transform are used to identify road lanes from the camera feed. A region of interest is dynamically selected to optimize detection.
- **Object and Sign Detection**: The YOLOv4 model, loaded on the Raspberry Pi using PyTorch and OpenCV, detects various traffic signs and obstacles such as pedestrians, vehicles, and cones in real time.
- **Obstacle Detection**: An ultrasonic sensor mounted at the front of the vehicle continuously measures the distance to potential obstacles. This data helps in issuing stop or slow commands when the detected object is too close.
- **Control and Feedback**: The entire control logic and graphical feedback are implemented in a Streamlit dashboard, allowing the user to view detection results and system decisions visually

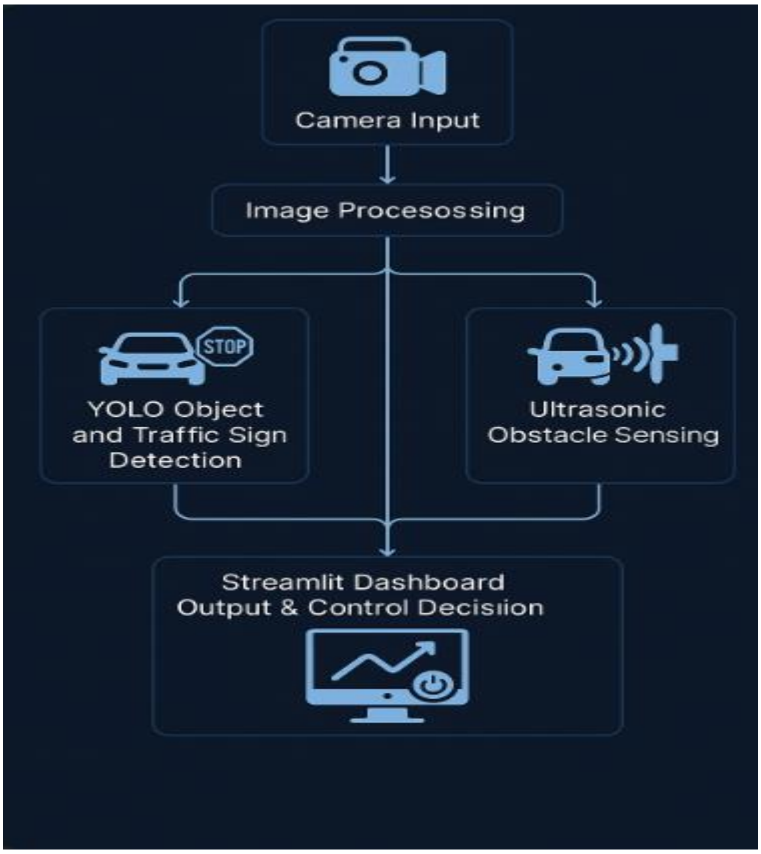


Figure 1: System Workflow Diagram

## 5. Methodology

- **Data Acquisition**: Custom datasets were prepared using recorded video and live camera feeds captured via the Raspberry Pi Camera Module. The data included road lane images, traffic signs, traffic lights, and objects such as pedestrians and vehicles under various lighting and weather conditions. Public datasets such as GTSRB (German Traffic Sign Recognition Benchmark) were also used to supplement training data for traffic sign classification.
- **Model Training:** Deep learning models were trained using Google Collab with GPU support for faster training. For object and traffic sign detection, the YOLOv5 model was utilized due to its speed and accuracy. Lane detection was implemented using edge detection and region of interest filtering with OpenCV. Traffic light recognition employed a CNN trained on labeled signal-light images to classify red, yellow, and green states.
- **System Integration:** The Raspberry Pi 3 Model B+ served as the central processing unit. The camera module was connected directly to the Pi to capture real-time video input. All deep learning and image processing models were exported and optimized for execution on the Raspberry Pi using Python and OpenCV. The Arduino microcontroller was removed to simplify hardware design and reduce system complexity.
- **Real-Time Processing:** Captured frames were processed in real-time to detect lanes, recognize traffic signs, classify traffic lights, and detect nearby objects or obstacles. Based on detections, corresponding alerts and visual indicators were generated. The control logic was handled entirely by the Raspberry Pi, which reduced latency and improved response time.
- **Visualization and User Interface:** A Streamlit-based dashboard was developed to provide a user-friendly interface for real-time video output and detection overlay. This UI enables monitoring of different modules such as lane tracking, object detection, and traffic sign recognition from a PC or mobile device.

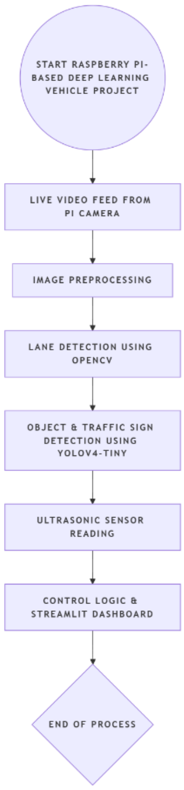


Figure 2: Methodology Flowchart

## 6. Implementation

The implementation phase involved developing the autonomous driving system on a Raspberry Pi 3 Model B+ using Python, OpenCV, and pre-trained deep learning models. The core objective was to achieve real-time detection and classification of lanes, traffic signs, objects, and traffic lights while ensuring the system could operate independently on embedded hardware.

- **Hardware Setup:** The system hardware consisted of the following components:
- Raspberry Pi 3 Model B+ with Raspbian OS
- Raspberry Pi Camera Module for capturing real-time video
- Dual motor driver (Rhino MDD20Amp 6V-30V) for controlling vehicle movement
- Two DC motors for left and right-wheel drive
- Power bank for mobile power supply

  All components were mounted on a lightweight chassis, forming a small-scale prototype of an autonomous vehicle.

- **Software Environment:** Python 3.9 and OpenCV 4.10.0 were used for image and video processing. The YOLOv5 model was trained and exported using PyTorch on Google Collab, then converted to TorchScript format for compatibility with the Raspberry Pi. The models were optimized to ensure acceptable inference speeds on the Pi's CPU.

- **Module-Wise Implementation:**
- **Lane Detection:** Implemented using Canny edge detection, Gaussian blur, region of interest masking, and Hough Transform to detect lane boundaries from camera input.
- **Traffic Sign Detection:** The YOLOv5 model was used to detect signs such as speed limits, stops, and pedestrian crossings. Detected signs were labeled and logged.
- **Object Detection:** YOLOv5 was also used to detect objects such as pedestrians, vehicles, and obstacles in the vehicle's path.
- **Traffic Light Classification:** A lightweight CNN was trained on traffic light images and deployed to classify light status (red, yellow, green) based on bounding box crop input.

- **User Interface:** A Streamlit-based interface was developed to provide a real-time visual display of detection overlays and system logs. It also allowed users to switch between modules and monitor processed outputs from a connected PC or mobile device.

.

Figure 3: Hardware Implementation

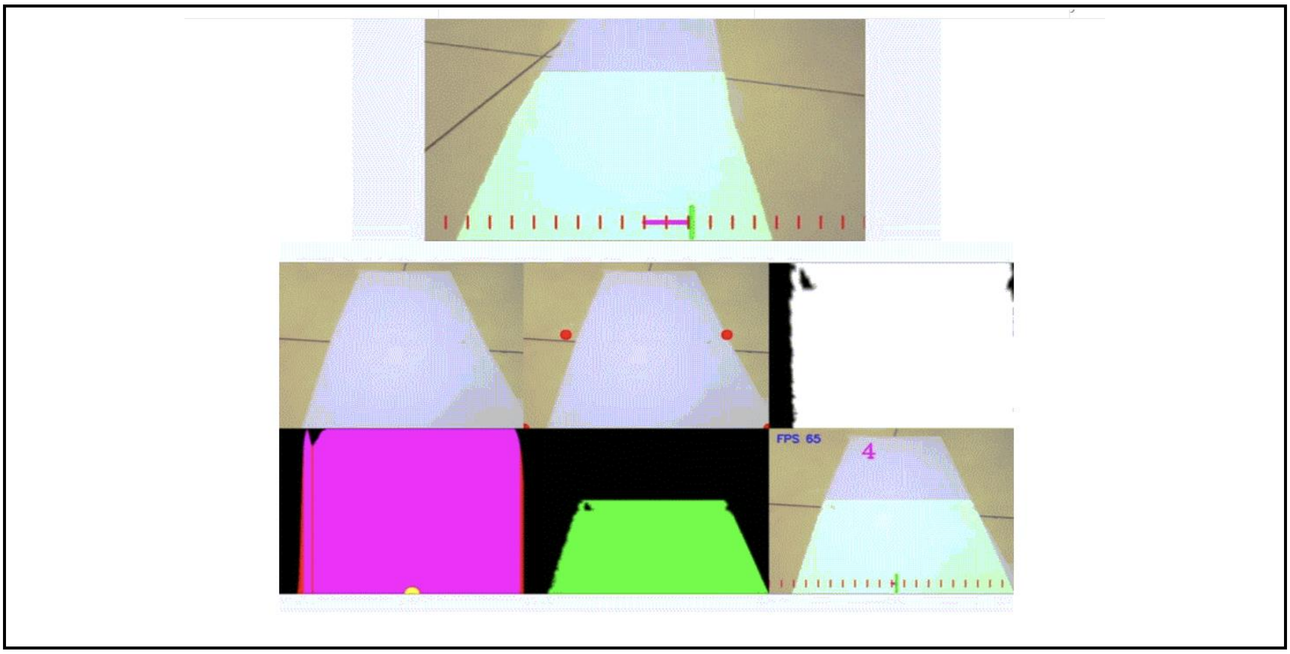


Figure 4: Software Implementation

International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

## 7. Result and Discussion

The autonomous driving system was evaluated based on detection accuracy, real-time performance (FPS), and reliability under varying environmental conditions. The performance of different modules was assessed both in simulation (Google Colab) and on-device (Raspberry Pi 3 Model B+).

| Module | Model | Accuracy (%) | Precision (%) | Recall (%) | FPS (on RPi) |
|---|---|---|---|---|---|
| Traffic Sign Detection | YOLOv5 | 91.8 | 89.5 | 92.3 | 8.2 |
| Object Detection | YOLOv5 | 89.7 | 87.2 | 90.1 | 7.9 |
| Traffic Light Classification | CNN | 94.5 | 93.8 | 95.0 | 13.4 |

Table 1: Model Performance Comparison

### 7.1 Accuracy Comparison of Detection Models

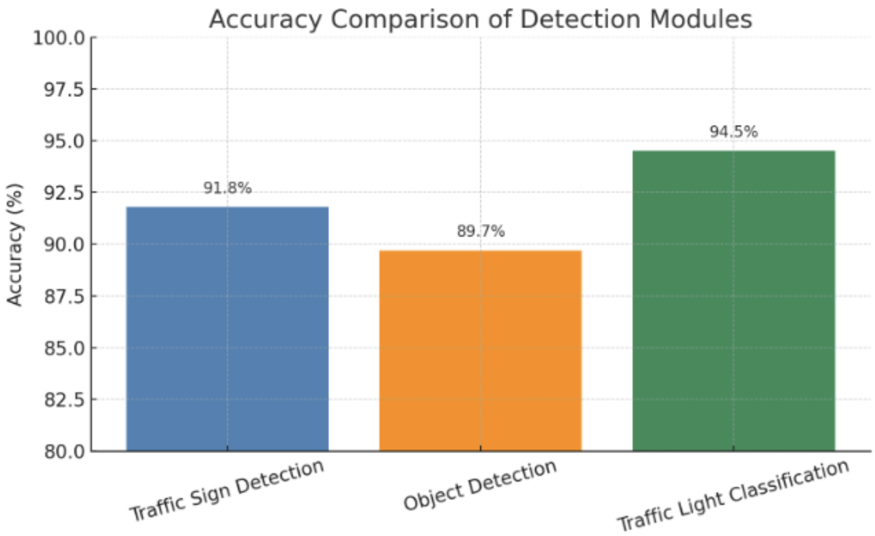


Figure 5: Accuracy Comparison of Detection Models

## 8. Applications

- **Driver Assistance**: It can be installed in existing vehicles to provide intelligent alerts for traffic signs and obstacles.
- **Educational Tool**: Ideal for demonstrating deep learning and embedded systems concepts in academic projects.
- **Research Platform**: Provides a customizable foundation for further development into full autonomous vehicle systems or driver support modules.

## 9. Future Scope

- **GPS Integration**: Add GPS modules for location-based navigation and decision-making.
- **Pothole Detection**: Integrate deep learning models trained to recognize potholes using the same camera feed.

- **Night Vision**: Employ IR sensors or low-light cameras to improve performance in dark environments.
- **Mobile Dashboard**: Develop an Android-based dashboard application to wirelessly control and monitor the vehicle from a smartphone.

## 10. Conclusion

This research demonstrates the successful development of a compact, real-time autonomous driving prototype using Raspberry Pi, deep learning, and OpenCV. The system achieves efficient lane detection, traffic sign recognition, obstacle avoidance, and traffic light classification, all on a low-power platform without GPU dependency.
With a user-friendly Streamlit dashboard and a modular design, the system offers a flexible foundation for future advancements such as GPS integration, pothole detection, and night-time navigation. The results validate that embedded systems can effectively host AI-driven functionalities, thus expanding the scope of affordable intelligent transport solutions.

## References

1. R.R. Pathan, P. Patel, "A Lightweight Approach for Sentiment Analysis Using Data Augmentation and Priority-Based Strategies," International Journal on Science and Technology (IJSAT), Vol. 16, Issue 2, April-June 2025, pp. 1–15.
2. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," IEEE CVPR, 2016.
3. G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
4. Raspberry Pi Foundation. "Raspberry Pi Documentation." https://www.raspberrypi.org/documentation
5. Joseph Redmon, Alexey Bochkovskiy. "YOLOv4 - Optimal Speed and Accuracy of Object Detection." https://github.com/AlexeyAB/darknet
6. A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NIPS, 2012.
7. HC-SR04 Ultrasonic Sensor Datasheet. https://components101.com/sensors/hc-sr04-ultrasonic-sensor
8. A. Teichman, S. Miller, and S. Thrun, "Towards 3D Object Recognition via Classification of Arbitrary Object Tracks," IEEE ICRA, 2011.
9. B. Zhao, H. Zhang, and H. Zhang, "An End-to-End Lane Detection Method Using Deep Learning for Autonomous Driving," IEEE Access, 2019.
10. K. Mohan, R. Krishna, "Implementation of YOLOv3-Based Driver Assistance System on Jetson Nano," International Journal of Computer Applications, Vol. 182, No. 12, 2022.

**Author Biography**

Yashraj S. Dube is currently pursuing a Bachelor's degree in Electronics and Computer Engineering at Amrutvahini College of Engineering. His research interests include embedded systems, machine learning, and autonomous vehicles.

Rushikesh R. Shinde is currently pursuing a Bachelor's degree in Electronics and Computer Engineering at Amrutvahini College of Engineering. Their interests include deep learning and embedded computer vision systems.

Akshay I. Chavhan is currently pursuing a Bachelor's degree in Electronics and Computer Engineering at Amrutvahini College of Engineering. Their interests include real-time systems and intelligent automation.