Object Detection and Self-Driving Bot Using Deep Learning

T S Preethivardhan[1] Department of ECE,JNN college of engirnering ,
Sharanya Y S [2] Department of ECE , JNN college of engirnering ,
Sangam S S[3] Department of ECE , JNN college of engirnering,
Darshan Gurunath Madival [4] Department of ECE , JNN college of engirnering,
Prema K N[5] Department of ECE , JNN college of engirnering

***Abstract*—This paper presents the implementation of an autonomous self-driving bot integrated with deep learning for object detection, showcasing its potential in real-world scenarios. The system employs cutting-edge deep learning models such as YOLO (You Only Look Once) to perform real-time object recognition, enabling the bot to detect vehicles and obstacles with high precision. The proposed solution incorporates sensor fusion, combining camera inputs with data from ultrasonic sensors to ensure robust perception and environmental awareness.Experimental results demonstrate the system's ability to detect objects with over 93 accuracy.**

I. Introduction

In the realm of modern technology, object detection plays a pivotal role in enhancing automation and safety in diverse applications, including autonomous driving, surveillance, and robotics. This project, titled "Object Detection and SelfDriving Bot Using Deep Learning," focuses on harnessing advanced algorithms to process real-time video data for precise object detection and classification. Traditional object detection methods often struggle with the high-speed demands of real-world scenarios, such as autonomous navigation through dynamic environments. These systems require accuracy, efficiency, and real-time processing, which conventional methods lack. The You Only Look Once (YOLO) algorithm addresses these challenges by treating object detection as a single regression problem, making it well-suited for time-sensitive tasks. Autonomous systems, particularly self-driving vehicles, require robust object detection to navigate safely and efficiently. Existing methods face challenges in achieving realtime performance without compromising accuracy, especially in detecting smaller or overlapping objects. These limitations necessitate a more optimized and scalable approach to ensure seamless operation. This project aims to design a self-driving bot capable of detecting and avoiding obstacles in real-time using the YOLO algorithm integrated with deep learning. Objectives include optimizing object detection processes, minimizing latency, and ensuring reliable operation in dynamic scenarios.

II. Related Work

A. Object Detection in Autonomous Systems

Object detection is a critical component of autonomous navigation systems, allowing robots and self-driving cars to perceive and react to their surroundings. Early object detection techniques were based on handcrafted features such as Haar cascades and HOG (Histogram of Oriented Gradients) features. However, these methods struggled with complex environments and varied object shapes. The introduction of Convolutional Neural Networks (CNNs) has revolutionized this field by learning robust feature representations directly from raw image data (Krizhevsky et al., 2012). One of the earliest and most influential works in deep learning-based object detection is R-CNN (Regions with CNN features) by Girshick et al. (2014), which applied CNNs to regions of interest in an image, achieving significant improvements over traditional methods. The development of Fast R-CNN and Faster R-CNN (Ren et al., 2015) further improved the speed and accuracy of object detection by integrating region proposals directly into the CNN architecture. While effective, these methods were still relatively slow for real-time applications, prompting the development of Single Shot Multibox Detectors (SSD) (Liu et al., 2016) and You Only Look Once (YOLO) (Redmon et al., 2016), both of which provide a more efficient and faster solution for realtime object detection. YOLO, in particular, has gained popularity due to its ability to simultaneously predict multiple bounding boxes and class labels in a single forward pass, making it suitable for resource-constrained environments such as autonomous vehicles and robots.

B. Self-Driving Bots and Autonomous Vehicles

The application of deep learning to self-driving cars has been a primary focus in the field of robotics. End-to-end learning systems, such as NVIDIA's PilotNet (Bojarski et al., 2016), have shown promise in autonomously navigating vehicles by directly mapping images from cameras to control commands (e.g., steering, throttle, and brake). These systems leverage CNNs trained on large datasets of driving scenarios, enabling vehicles to learn complex driving behaviors without explicit programming for every situation. In addition to CNNs for perception, reinforcement learning (RL) has also been applied to self-driving bots to enable them to make decisions through trial and error. Deep QLearning and Proximal Policy Optimization (PPO) have been used to train agents in simulators to learn optimal driving policies (Mnih et al., 2015; Schulman et al., 2017). RL-based methods allow for the continuous improvement of decisionmaking systems through interaction with the environment, which is crucial for adapting to dynamic, real-world driving conditions.

C. *Sensor Fusion and Real-Time Navigation*

A key challenge in autonomous driving is effectively integrating information from multiple sensors (e.g., cameras, LiDAR, radar) to improve the reliability and robustness of object detection and navigation. Sensor fusion techniques combine data from these sensors to create a more comprehensive understanding of the environment, overcoming the individual limitations of each sensor type. Several works have explored the combination of visual data with LiDAR point clouds to improve depth perception and obstacle detection (Geiger et al., 2012; Zhang et al., 2017). For real-time autonomous navigation, approaches like Simultaneous Localization and Mapping (SLAM) have been essential for enabling autonomous robots and vehicles to navigate unknown environments. DeepSLAM (Zhang et al., 2017), for instance, combines deep learning and SLAM techniques to simultaneously detect objects and map the environment, significantly improving the robot's ability to navigate complex scenes without relying on pre-built maps.

III. METHODOLOGY

The project methodology integrates live video processing with object detection using deep learning techniques. Initially, video frames are captured in real-time using a camera or other video source and are preprocessed to meet the input requirements of the YOLO (You Only Look Once) model. Preprocessing involves resizing frames to standard dimensions (e.g., 416x416 pixels), normalizing pixel values, and formatting channels to ensure compatibility with the model. The frames are then passed through a Convolutional Neural Network (CNN) for feature extraction, where convolutional and pooling layers identify significant image features like edges and textures, enabling efficient object recognition. The YOLO algorithm divides each frame into a grid, predicts bounding boxes, and classifies objects in real-time. Post-processing techniques, such as Non-Max Suppression (NMS), refine the predictions by eliminating redundant bounding boxes. Finally, the detected objects are displayed on the original frames with bounding boxes and labels, enabling continuous real-time tracking. Hardware integration, including microcontrollers, sensors, and motor drivers, allows the self-driving bot to navigate based on detected objects, avoiding obstacles and adjusting its movements accordingly

A. *Dataset*

The dataset utilized in this project, obtained from Kaggle, represents a comprehensive and well-structured collection curated to support real-time object detection and recognition.The chosen dataset for this project likely contained annotated images of various objects across diverse categories, complete with bounding box coordinates and class labels. Such datasets are indispensable for training, validating, and testing machine learning models, particularly for real-time applications like the YOLO architecture.The dataset consisted of thousands of images capturing objects in diverse environments, ensuring the model's adaptability to realworld scenarios. These images were annotated with precise bounding boxes, indicating the location of objects, and class labels to define the objects' categories (e.g., car, person, bicycle, etc.).

B. *Feature Extraction*

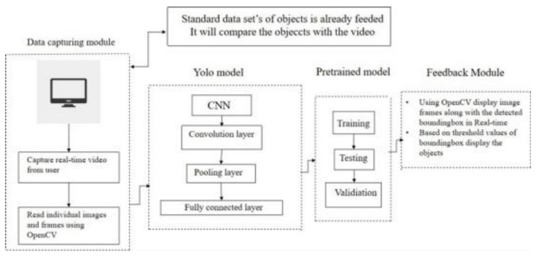
Standard data set's of objects is already feeded
It will compare the objeccts with the video
Data capturing module
Capture real-time video from user
Read individual images and frames using OpenCV
Yolo model
CNN
Convolution layer
Pooling layer
Fully connected layer
Pretrained model
Training
Testing
Validiation
Feedback Module
• Using OpenCV display image frames along with the detected boundingbox in Real-time
• Based on threshold values of boundingbox display the objects

Fig. 1. Block diagram

The diagram illustrates a real-time object detection and recognition system that integrates four primary components: a data capturing module, the YOLO (You Only Look Once) model, a pretrained model, and a feedback module. This system is designed to detect and recognize objects in video feeds efficiently, making it suitable for applications such as surveillance, autonomous vehicles, and real-time monitoring. The workflow begins with data capturing, followed by object detection and recognition, and ends with real-time feedback to the user. The data capturing module collects real-time video input from the user, typically through a camera or another video source. This video feed is processed into individual frames using OpenCV, a powerful open-source library for computer vision tasks. These frames serve as the raw input for the object detection pipeline, enabling the system to analyze each frame in detail. By leveraging OpenCV, the system ensures seamless integration with various video sources, making it highly adaptable for different environments. The core of the system is the YOLO model, a state-of-the-art algorithm for real-time object detection. Unlike traditional methods that process regions of an image separately, YOLO operates as a single neural network, analyzing the entire image in one pass. Its architecture includes three key layers. The convolution layer extracts features such as edges, textures, and patterns from the image, while the pooling layer reduces the size of the feature maps to retain important information and optimize computational efficiency. The fully connected layer then combines the extracted features to make predictions, including object classifications and their exact locations, represented as bounding boxes. YOLO's efficiency and ability to detect multiple objects simultaneously make it ideal for real-time applications. A pretrained model is incorporated into the system to enhance accuracy and robustness. This model is trained on a large, standardized dataset of objects and undergoes three stages: training, testing, and validation. By using the pretrained model as a reference, the system can compare detected objects in the video frames with known objects, ensuring reliable and precise results. The feedback module provides real-time interaction with the user. Using OpenCV, it overlays bounding boxes on the video feed to visually highlight detected objects. To ensure accuracy, the system incorporates confidence threshold values, displaying only those objects with a high detection confidence. This feedback mechanism not only provides clear visualization but also enhances user trust by filtering out low-confidence detections. In summary, the system combines data capturing, YOLO-based object detection, pretrained model validation, and real-time feedback to deliver accurate and efficient object recognition. Its modular design and adaptability make it suitable for a wide range of applications, including video surveillance, autonomous navigation, and industrial automation. This integration of speed, accuracy, and real-time feedback ensures the system is both practical and reliable for diverse operational needs.

C. Convolutional Neural Networks

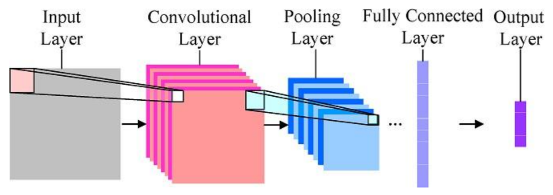

Fig. 2. Layers of CNN

Convolutional Neural Networks (CNNs) are a type of deep learning model specifically designed for analyzing grid-like data such as images or video frames. They are composed of multiple layers that work together to automatically and efficiently learn spatial hierarchies of features from the input data, which makes them highly effective for tasks like image classification, object detection, and segmentation. The core building block of a CNN is the convolutional layer, which applies filters (also called kernels) to the input image. These filters slide over the image, performing element-wise multiplications and summations to extract key features like edges, textures, and patterns. The output of these operations is a feature map that highlights specific patterns in the image. Following the convolutional layers are pooling layers, which reduce the dimensionality of the feature maps by downsampling, typically using operations like max-pooling. This helps retain important features while reducing computational complexity. After the convolution and pooling layers, the CNN typically includes one or more fully connected layers, where the features learned from the previous layers are combined and used for classification or regression tasks. The fully connected layers output the final prediction, such as the class label in image classification. CNNs are particularly powerful because they automatically learn relevant features from data, eliminating the need for manual feature extraction, and their ability to detect patterns at different levels of abstraction makes them suitable for complex tasks in computer vision and other fields.

D. Hardware Module

The hardware module for the object detection and self-driving bot integrates essential components for capturing, processing, and executing autonomous movements based on detected objects. The system is built around a microcontroller, such as an Arduino Uno, which acts as the brain, processing sensor inputs and controlling outputs. To enhance capabilities, an ESP32-CAM module is employed for real-time video capture and wireless connectivity. This module facilitates streaming video frames for object detection using the YOLO deep learning model. For obstacle detection and navigation, ultrasonic sensors are strategically mounted on the bot's front and sides. These sensors use sound waves to measure distances, effectively detecting obstacles up to a range of 400 cm, and perform well in various lighting conditions compared to infrared sensors. The movement of the bot is driven by DC motors controlled by a motor driver module like the L298N, which allows bidirectional control for forward and reverse movements. The motor driver separates the power for the motors from the logic circuit, ensuring stable operation and protecting the microcontroller. Wheels and a durable chassis are mounted to provide stable support and efficient movement. The chassis is pre-drilled for easy assembly and alignment of sensors and other components.

A compact power supply, typically a rechargeable lithiumion or lithium-polymer battery, powers the entire system. Voltage regulators are incorporated to provide consistent and stable power delivery to all components, ensuring uninterrupted operation. A breadboard and jumper wires are used for prototyping and connecting various components without soldering, making the system modular and easy to debug. The hardware module works in tandem with the software, where video frames captured by the ESP32-CAM are processed using YOLO on a compatible GPU or edge computing device. Based on the detected objects, commands are sent to the microcontroller to control the motors, enabling the bot to navigate autonomously while avoiding obstacles. The system's modular design makes it scalable for future enhancements, such as integrating more advanced sensors, improving navigation algorithms, or adding features like live monitoring or voice commands. Together, these components form a cohesive and efficient hardware platform for real-time object detection and autonomous navigation.

IV. Results and Discussion

The Obtained results of object detection in live video streams using YOLO include real-time detection of multiple objects with bounding boxes and labels, maintain ing high accuracy and frame rates (30 FPS on GPUs). Objects are continuously tracked across frames, and Non-Max Suppression (NMS) ensures minimal over lap in detected bounding boxes. In addition to real-time detection and tracking, YOLO's ability to process the entire image in a single pass ensures fast and

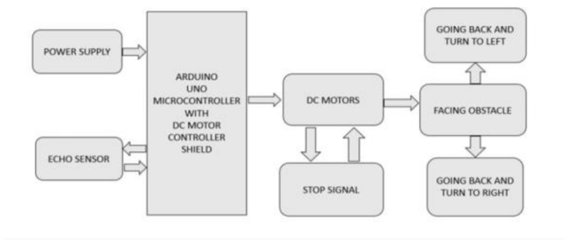
POWER SUPPLY
ECHO SENSOR
ARDUINO UNO MICROCONTROLLER WITH DC MOTOR CONTROLLER SHIELD
DC MOTORS
STOP SIGNAL
FACING OBSTACLE
GOING BACK AND TURN TO LEFT
GOING BACK AND TURN TO RIGHT

Fig. 3. System architecture of the self-driving bot.

efficient object recognition. Its speed and scalability make it suitable for various real-world applications. The Obtained results is a functional, autonomous robot that moves forward while detecting and avoiding obstacles in its path by changing direction. It should be able to navigate around objects without human intervention, demon strating basic obstacle avoidance behavior.

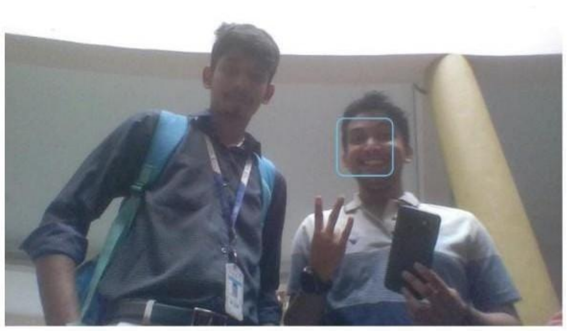

Fig. 4. Results obtained (before running the model)

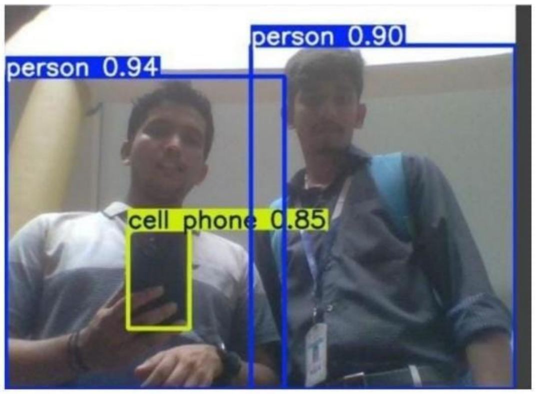

Fig. 5. Results obtained (After running the model)

A. Comparision Table

The table highlights the comparison of different hardware components used with various YOLO (You Only Look Once) versions for object detection and their cor responding accuracy levels. Among the listed devices, the ESP32-CAM module is chosen for its affordability and simplicity. This module is a low-cost microcon troller with an integrated camera

Different components for interface	Yolo version	Output Accuracy %
ESP32 CAM	Yolo v3	89-91%
Raspberry Pi 4	Yolo v5	93-96%
NVIDIA Jetson NANO	Yolo v5 And above	94-99%

Fig. 6. Comparision Table

and Wi-Fi connectivity, making it an attractive choice for budget-conscious projects. Paired with YOLO v3, the ESP32-CAM achieves an output accuracy of 89-91 percent, which is sufficient for basic appli cations like home surveillance, IoT monitoring, or simple image recognition tasks. Its lightweight design and low power consumption make it especially suit-able for small-scale, portable, or lowresource systems. In contrast, more advanced devices like the Raspberry Pi 4 and NVIDIA Jetson Nano support the more computationally intensive YOLO v5 and above, achieving significantly higher accuracy (up to 98 percent). However, these devices come at a higher cost and are better suited for applications requiring real-time process ing, such as robotics or autonomous systems. The choice to use the ESP32-CAM demonstrates a practical tradeoff between cost and performance, showcasing its value as an efficient and economical solution for simpler object detection needs. Its integration flexibility and ease of use further solidify its appeal for beginner and intermediate-level projects.

V. Conclusion and Future Work

In this study , object detection and self-driving bots using deep learning techniques, specifically YOLO and Convo-lutional Neural Networks (CNNs), demonstrates substantial potential in real-time video analysis and autonomous nav-igation. By leveraging the power of YOLO, the system is able to efficiently detect and classify objects in live video streams with high accuracy, making it suitable for dynamic and time-sensitive applications such as autonomous driving, surveillance, and industrial automation. The integration of CNNs allows for hierarchical feature extraction from images, enabling the model to learn complex patterns and improve ob-ject detection robustness in various environmental conditions. Looking to the future, the scope of this project is vast. As advancements in deep learning and computer vision continue to evolve, the incorporation of techniques such as multi-modal sensing, reinforcement learning, and realtime 3D object de-tection could significantly improve the model's performance. Further, the optimization of hardware acceleration, particularly through GPU and FPGA, will enhance the speed and efficiency of real-time detection systems. Moreover, integrating more sophisticated algorithms could enable the development of fully autonomous systems capable of adapting to complex environments, ranging from smart cities to advanced robotics. This project lays the foundation for the next generation of intelligent, real-time decisionmaking systems with the poten-tial to impact a wide array of industries, from autonomous transportation to smart surveillance.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to all those who have contributed to the successful completion of the ”Object Detection and Self-Driving Bot using Deep Learning” project. First and foremost, I would like to thank my project guide,Prema K N, for their invaluable guidance, constant support, and expert insights throughout the course of this project. Their encouragement and knowledge have been a significant factor in making this project a success. I would also like to extend my heartfelt thanks to the Electronics and communication department, JNN college of Enginerring Shimoga, for providing the resources and infrastructure necessary for the development and implementation of the project. I am also grateful to my peers and team members for their collaboration and efforts in brainstorming ideas, testing models, and troubleshooting challenges encountered during the project. Their enthusiasm and teamwork were essential in achieving our goals. Finally, I would like to acknowledge the various online resources, tutorials, and documentation that were immensely helpful in enhancing my understanding of deep learning algorithms and self-driving technologies. Thank you to everyone who has supported me in any way to bring this project to fruition. References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

REFERENCES

[1] Bojarski, M., et al. (2016). End to End Learning for Self-Driving Cars. arXiv preprint arXiv:1604.07316.

[2] Geiger, A., et al. (2012). Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[3] Girshick, R., et al. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[4] Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

[5] Liu, W., et al. (2016). SSD: Single Shot Multibox Detector. European Conference on Computer Vision (ECCV).

[6] Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. Nature.

[7] Redmon, J., et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv preprint arXiv:1506.02640.

[8] Ren, S., et al. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Neural Information Processing Systems (NeurIPS)

[9] Schulman, J., et al. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347

[10] Zhang, L., et al. (2017). DeepSLAM: A Deep Learning Approach to Simultaneous Localization and Mapping. IEEE Transactions on Robotics

[11] D. Padilla Carrasco, H. A. Rashwan, M. A. Garc ́ ıa, and D. Puig, ”TYOLO: Tiny Vehicle Detection Based on YOLO and Multi-Scale Convolutional Neural Net works,” IEEE Access, vol. 9, pp. 48983–48995, 2021, doi: 10.1109/ACCESS.2021.3064438.

[12] X. Qian, X. Wang, S. Yang, and J. Lei, ”LFF-YOLO: A YOLO Algorithm for Detecting Large-Scale Construction Site Safety Hazards,” Automation in Construc tion, vol. 131, p. 103946, 2022, doi: 10.1016/j.autcon.2021.103946.

[13] L. Falaschetti, L. Manoni, L. Palma, P. Pierleoni, and C. Turchetti, ”Embedded Real-Time Vehicle and Pedestrian Detection Using a Compressed Tiny YOLO v3 Architecture,” IEEE Trans. Intell. Transp. Syst., vol. 24, no. 1, pp. 120-130, Jan. 2024, doi: 10.1109/TITS.2023.3290901.

[14] Q. Xu, R. Lin, H. Yue, H. Huang, Y. Yang, and Z. Yao, ”Research on Small Target Detection in Driving Scenarios Based on Improved YOLO Network,” IEEE Access, vol. 8, pp. 182228–182237, 2020, doi: 10.1109/ACCESS.2020.3029297

[15] Y. Cao, C. Li, Y. Peng, and H. Ru, ”MCS-YOLO: A Multiscale Object Detection Method for Autonomous Driving Road Environment Recognition,” IEEE Access, vol. 11, pp. 89745-89755, 2023, doi: 10.1109/ACCESS.2023.3312387.