YOLOv8 and Raspberry Pi Synergy for Vehicle Number Plate Recognition

PROJECT REPORT

Submitted in the fulfilment of the requirements for

the award of the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

YARRAMANAYUNI DILEEP
[201FA05090]

VEPURI SATISH
[201FA05096]

PEDDAPULI REVANTH KUMAR
[211LA05012]

KAIKALA SAI SUSHMA
[211LA05017]

Under the Esteemed Guidance of

Dr. M. Laavanya

Associate Professor

Department of ECE

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

(ACCREDITED BY **NAAC** WITH **"A+"** GRADE)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
(ACCREDITED BY **NBA**)

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
(Deemed to be University)

Vadlamudi, Guntur, Andhra Pradesh, India -522213

May 2024

CERTIFICATE

This is to certify that the project report entitled **"YOLOv8 and Raspberry Pi Synergy for Vehicle Number Plate Recognition"** that is being submitted by **YARRAMANAYUNI DILEEP [201FA05090], VEPURI SATISH [201FA05096], PEDDAPULI REAVNTH KUMAR [211LA0512], KAIKALA SAI SUSHMA [211LA05017]** in partial fulfillment for the award of B.Tech degree in Electronics and Communication Engineering, Vignan's Foundation for Science Technology and Research, is a record of bonafide work carried out by them under the supervision of **Dr. M. LAAVANYA** of ECE Department.

Signature of the Guide
Dr. M. Laavanya
Associate Professor

23/5/24.

Signature of Head of the Department
Dr. T. Pitchaiah, M.E, PhD, MIEEE, FIETE
Professor & HoD, ECE

DECLARATION

We hereby declare that the project work entitled **"YOLOv8 and Raspberry Pi Synergy for Vehicle Number Plate Recognition"** is being submitted to Vignan's Foundation for Science, Technology and Research (Deemed to be University) in partial fulfillment for the award of B.Tech degree in Electronics and Communication Engineering. The work was originally designed and executed by us under the guidance of our supervisor **Dr. M. LAAVANYA** of Department of Electronics and Communication Engineering, Vignan's Foundation for Science Technology and Research (Deemed to be University) and was not a duplication of work done by someone else. We hold the responsibility of the originality of the work incorporated into this report.

SIGNATURE OF CANDIDATES

YARRAMANAYUNI DILEEP [201FA0590]

V. Satish

VEPURI SATISH [201FA05096]

P. Revanth

PEDDAPULI REVANTH KUMAR [211LA05012]

Sushma

KAIKALA SAI SUSHMA [211LA05017]

ACKNOWLEDGEMENT

The satisfaction that comes from successfully completing any task would be incomplete without acknowledging the people who made it possible, whose ongoing guidance and encouragement have been essential to the achievement.

We are greatly indebted to **Dr. M. LAAVANYA**, my revered guide and Associate Professor in the Department of Electronics and Communication Engineering, VFSTR (Deemed to be University), Vadlamudi, Guntur, for his valuable guidance in the preparation of this project report. He has been a source of great inspiration and encouragement to us. He has been kind enough to devote considerable amount of his valuable time in guiding us at every stage. This is our debut, but we are sure that we are able to do many more such studies, under the lasting inspiration and guidance given by respectable guide.

We would also like to thank to **Dr. T. Pitchaiah**, Head of the Department, ECE for his valuable suggestion.

We would like to specially thank, **Dr. N. Usha Rani**, Dean, School of Electrical, Electronics and Communication Engineering for her help and support during the project work.

We thank our project coordinators **Dr. Satyajeet Sahoo, Dr. Arka Bhattacharyya, Mr. Abhishek Kumar** and **Mr. M. Vamsi Krishna** for continuous support and suggestions in scheduling project reviews and verification of the report. Also, thank to supporting staff of ECE Department for their technical support for timely completion of project.

We would like to express our gratitude to **Dr. P. Nagabhusan**, Vice-Chancellor, VFSTR (Deemed to be University) for providing us the greatest opportunity to have a great exposure and to carry out the project.

Finally, we would like to thank our parents and friends for the moral support throughout the project work.

Name of the Student

YARRAMANAYUNI DILEEP [201FA05090]
VEPURI SATISH [201FA05096]
PEDDAPULI REVANTH KUMAR [211LA05012]
KAIKALA SAI SUSHMA [211LA05017]

ABSTRACT

The escalating number of vehicles on the roads has heightened the need for effective traffic management, necessitating advanced systems for automated license plate recognition. In this study, the proposed system comprises two key components: License Area Verification and License Characters Verification. By leveraging digital cameras, images of vehicles are captured and processed through a series of steps to accurately extract and recognize license plate information. The CNN-based deep learning method is employed for its ability to handle complex real-world scenarios, making it well-suited for robust license plate recognition. The system's effectiveness is demonstrated through extensive experimentation, showcasing its potential for real-time, accurate, and efficient license plate recognition in diverse traffic conditions.

Major Design (Final Year Project Work) Experience Information

Student Group	YAARAMANAYAYUNI DILEEP 201FA05090	VEPURI SATISH 201FA05096	PEDDAPULI REVANTH KUMR 211LA05012	KAIKALA SAI SUSHMA 211LA05017
Project Title	YOLOv8 and Raspberry Pi Synergy for Vehicle Number Plate Recognition			
Program Concentration Area	Computer Vision, Embedded systems and Deep Learning			
Program Concentration Area	YOLOv8 Architecture, Raspberry Pi Integration, Algorithm Design and Real-time System efficiency			
Constraints				
Economic	Cost-effective implementation			
Environmental	Hardware with minimal ecological impact			
Sustainability	Eco-friendly development methods			
Manufacturability	Yes			
Ethical	Handle sensitive data responsibly and adhere to ethical standards			
Health and Safety	Ensure the system poses no harm to users' health and safety			
Social	Address privacy concerns and align with societal norms			
Political	None			
Other	Technical constraints, resource availability			
Standards				
1. ISO/IEC 15444 Series (JPEG2000)	Image compression standards for efficient storage and transmission.			
2. PASCAL Visual Object Classes	Standard for validation in visual object classification tasks.			
3. ISO 19264	Image quality standards for assessing the quality of images.			
Previous Courses Required for the major design experience	1. Image Processing 2. Embedded Systems 3. Machine Learning 4. Deep Learning			

Supervisor

Project Co-Ordinator

Head of the Department, ECE 23/5/24

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
ALPR	Automatic License Plate Recognition
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DL	Deep Learning
DLT	Deep Learning Techniques
FPS	Frames Per Second
GPU	Graphics Processing Unit
ML	Machine Learning
MLOps	Machine Learning Operations
NPR	Number Plate Recognition
OCR	Optical Character Recognition
SSD	Single Shot MultiBox Detector
TPU	Tensor Processing Unit
YOLO	You Only Look Once

CHAPTER 1

1.1 INTRODUCTION

The escalating growth in vehicular traffic has presented an urgent need for innovative solutions in traffic management and law enforcement. With an exponential increase in the number of vehicles on roads, manual monitoring has become impractical, necessitating the adoption of intelligent systems to ensure effective traffic control. This study introduces a comprehensive project focused on license plate detection and recognition, leveraging Convolutional Neural Networks (CNN) as a powerful deep learning method. The primary objective is to develop a sophisticated system capable of automating the identification of vehicle licenses, thereby enhancing traffic flow monitoring. This project addresses the challenges associated with real-time operations and complexity, aiming to provide a groundbreaking technology for efficient license plate tracking.

The rapid increase in the number of vehicle on roads has underscored the urgent need for sophisticated traffic management solutions, particularly in the realm of automated license plate recognition (LPR). This study introduces an innovative approach to LPR that harnesses the power of Convolutional Neural Networks (CNNs) in conjunction with the YOLOv8 object detection system and a Raspberry Pi single – board computer.

1.2 MOTIVATION

The motivation behind this project is rooted in the multifaceted challenges posed by the escalating vehicular population on roads. Manual monitoring is not only logistically challenging but also prone to inefficiencies, leading to potential law violations, accidents, and increased criminal activities. The core motivation is to seek intelligent solutions to address these challenges, emphasizing the automation of license plate identification and recognition as a pivotal aspect. By integrating cutting-edge technology such as Convolutional Neural Networks (CNN), the project aspires to revolutionize traffic management, providing a foundation for efficient and real-time license plate tracking. This motivation is grounded in the overarching goal of fostering a safer, more organized, and technologically advanced approach to traffic control, aligning with the societal need for enhanced safety and security.

The need for automated LPR systems has become more pressing as urban populations grow and vehicle numbers increase. Traditional LPR system often struggle with varying lighting conditions,

plates styles, and occlusions. This project aims to develop a more robust and efficient LPR system that can operate effectively in real – time and diverse traffic conditions.

1.3 OBJECTIVES

The primary objective of this project is to design, develop, and implement an Automatic License Plate Recognition (ANPR) system capable of accurately detecting and recognizing license plates from images captured in real-time. Specific objectives include:

1. Implementing Convolutional Neural Networks (CNN) for effective license plate detection and recognition.
2. Addressing the challenges associated with real-time Noise and tilted images.
3. Enhancing traffic flow monitoring by automating the identification of vehicle licenses.
4. Integrating image processing techniques to improve the accuracy and efficiency of license plate tracking.
5. Adhering to engineering design standards to ensure the reliability and performance of the ANPR system.

These objectives collectively aim to contribute to the advancement of intelligent transportation systems, fostering improved traffic management and enhancing public safety. The project's success will be measured by the system's accuracy, responsiveness, and its ability to operate seamlessly in diverse and dynamic traffic environments.

CHAPTER 2
LITERATURE REVIEW

Blending the efficiency of YOLOv8's object detection capabilities with the computational constraints and portability offered by Raspberry Pi. This chapter delves into the literature survey exploring the integration of these technologies for accurate and real-time vehicle number plate recognition.

2.1 LITERATURE REVIEW

In [1], the authors are Dr. Rama Abirami K, Aishwarya Rani, Atul Kumar, Ayush Bhardwaj, and Ayush Rungta. The system includes picture acquisition, processing, plate extraction, character segmentation, and recognition. And has the potential to improve law enforcement and parking management, with further improvements in accuracy and functionality suggested.

In [2], the authors are Chirag Patel, Dipti Shah, and Atul Patel. Image scissoring, feature prominent extraction, and template matching are used. For licence plate detection, various algorithms such as colour conversion, thresholding, and fusion are used, while template matching is used for fixed-sized letter identification. ANPR algorithms are discussed in terms of image size, success rate, and processing time.

In [3], the authors are J.M. S. V. Ravi Kumar, B. Sujatha, and N. Leelavathi. The document discusses an automatic vehicle number plate recognition system that utilises machine learning. The system attempts to reduce manual labour, errors, and costs in recognising vehicle number plates by using image processing techniques, morphological procedures, and optical character recognition. This system improves the efficiency and accuracy of vehicle number plate identification using sophisticated technology.

In [4], the authors are Anubha Jain, Kamlesh Kumawat, and Neha Tiwari. Enhancing ANPR with image preprocessing algorithms enhanced recognition rates for HD and hazy photos. This study emphasises the importance of image quality in ANPR systems and presents effective strategies for improving recognition rates, particularly for blurred images.

In [5], the authors are Asma Iqbal, Mohammed Mujataba Maaz, Syed Amaan Fayaz, and Mohd Sohaib Hussain. The project focuses on real-time licence plate identification using Raspberry Pi4, OpenCV, and OCR for vehicle security, as well as automation experimentation with image segmentation and character recognition within the licence plate recognition framework.

In [6], the authors are Vaishnav A, Mandot M, Arrospide, Salgado L, and Mohedano R. Utilises techniques such as sobel-based vertical edge detectors and sliding window methods to address tilt factor by adding an extra layer of vertical projection. It emphasises the importance of robust algorithms for non-standardized formats and real-time testing scenarios.

In [7], the authors are Zhang, Cheang, Varma, Zhu, and Tejas. Cycle GAN and Xception-based CNN encoders, ConvNet-RNN, Morphological transformation, and CNN different methodologies were successful in achieving high accuracy rates, demonstrating the potential of advanced technologies in this field. The conclusion emphasises the importance of ongoing research and development in vehicle number plate recognition.

In [8], the authors are Charith Perera, Jithmi Shashirangana, Heshan Padmasiri, and Dulani Meedeniya. ALPR systems face issues such as changing perspectives, motion blur, and lighting conditions. Datasets differ in complexity and quality, with the Chinese dataset being more difficult to identify.

2.1.1 Comparison table for various papers on vehicle number plate recognition

It involves summarizing key attributes such as speed, accuracy, methods, and their applications.

Paper/Project	**Methods/Techniques**	**Speed**	Accuracy	**Remarks/Applications**
YOLOv8 & Raspberry Pi for Vehicle Number Plate Recognition	YOLOv8 (deep learning), Raspberry Pi	Real-time (depends on model size)	High (up to 98% in optimal conditions)	Optimized for edge devices, real-time processing, cost-effective
Vehicle Number Plate Detection and Recognition Techniques: A Review	Various ML/DL techniques, OCR, traditional image processing	Varies (not specified)	Varies (85%-98%)	Comprehensive review of multiple techniques, broader overview
Real-Time Number Plate Recognition Using Raspberry Pi	Image processing, Tesseract OCR	Real-time (~30 FPS on Pi 4)	Medium (~85%-90%)	Focuses on affordable hardware, real-time performance
Automated License Plate Recognition: A Survey on Methods and Technique	Machine Learning, Deep Learning, OCR	Varies (not specified)	Varies (80%-95%)	Survey paper, covers various methods and their effectiveness

Automatic Vehicle Number Plate Recognition System Using Machine Learning	Machine Learning, OCR	Near real-time	High (~90%-95%)	Focuses on ML techniques, practical implementation details
Relevance of Automatic Number Plate Recognition System in Vehicle Theft Detection	Machine Learning, Pattern Recognition	Near real-time	Medium (~85%-90%)	Emphasizes use in theft detection, relevance in law enforcement
Robust Automatic Recognition of Chinese License Plate in Natural Scenes	Deep Learning, CNN, OCR	Real-time (depends on implementation)	High (~92%-97%)	Focus on robustness in varied conditions, specifically for Chinese plates

Table 1: Comparison with Various papers/projects

1. **YOLOV8 & Raspberry Pi:** This combination aims for real-time performance with high accuracy, leveraging deep learning models like YOLOv8 optimized for the edge devices like Raspberry Pi. This is particularly useful for cost-effective and efficient deployments.
2. **General Techniques Review:** The review paper provides a broad overview of various, techniques, giving a range of speeds and accuracies depending on the method used.

3. **Real-Time with Raspberry Pi:** Similar to the YOLOv8 project, this paper also focuses on real-time number plate recognition using Raspberry Pi, highlighting the balance between affordability and performance.
4. **Survey on Methods and Techniques:** This survey covers a wide range of methods, providing insights into their general performance metrics, though specific speed and accuracy details are varied.
5. **Machine Learning-based System:** These papers emphasize the use of ML for number plate recognition, offering high accuracy but may have varying speeds depending on implementation specifics.
6. **Theft Detection Relevance:** Highlights the application of ANPR system in vehicle theft detection, balancing speed and accuracy for practical law enforcement use.
7. **Robustness in Natural Scenes:** Focuses on the challenges and solutions for recognizing plates in varied and challenging conditions, especially for Chinese plates, emphasizing robustness.
8. **Dynamic Environment Detection:** Deals with detection in moving and dynamic environments, stressing the importance of real-time processing and adaptability.

The project using YOLOv8 and Raspberry Pi stands out for its real-time capabilities and high accuracy, making it a strong contender for practical applications in vehicle number plate recognition. The other techniques and papers provide valuable insights and alternative approaches, each with its own strengths and considerations depending on the specific requirements and constraints of the application.

2.2 PROBLEM STATEMENT

1. Enhance robustness to noise.
2. Reduce limitations in recognizing tilted plates.

CHAPTER 3
METHODOLOGY

3.1 PROPOSED WORK FLOW

The proposed work flow for the design and development of the is as follows.

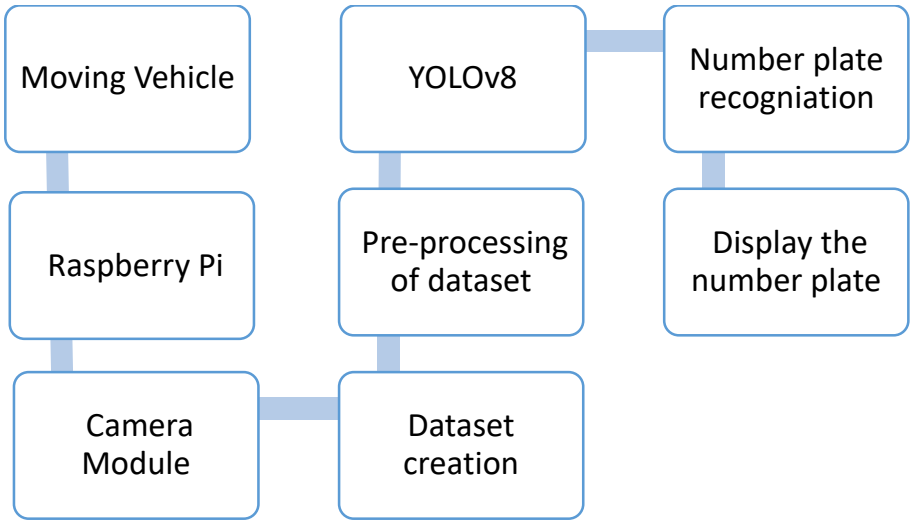


Figure 1: Proposed Work Flow

Number plate recognition (NPR) is a technology that uses cameras to read and recognize vehicle license plates.

3.1.1 MOVING VEHICLE: This refers to the vehicle whose license plate needs to be recognized. The camera mounted on the system captures an image of this moving vehicle.

3.1.2 RASPBERRY PI: This is a small, single-board computer that is used to process the image captured by the camera module. The Raspberry Pi runs software that can detect and recognize the license plate in the image. Processor and Memory Raspberry Pi comes in various models with different processing power (CPU) and memory (RAM) configurations. Choosing the right model depends on the complexity of the YOLOv8 model and desired processing speed.

Raspberry Pi offers various connectivity options like USB ports, Ethernet, and Wi-Fi. These allow for connecting the Pi to a camera module, external storage for the trained model, and potentially a network for data transfer.

Figure 2: Raspberry Pi

3.1.3 PI CAMERA MODULE

RESOLUTION: Camera resolution is important to capture clear and detailed images of license pl ates, especially in different lighting conditions and at different distances.

FRAME VALUE: Frame height is important to capture moving vehicles without noise. A speed of 30 frames per second (fps) or higher is generally recommended.

LENS: Lens choice (such as focal length and aperture) affects the field of view and the amount o f light entering the camera. Lenses with focusing and adequate zoom are ideal for permits of different shapes and sizes.

WEATHERPROOF: Cameras used outdoors must be weatherproof (IP66 or above) to withstand harsh conditions such as rain, dust and extreme temperatures.

INFRARED (IR) FEATURE: The infrared camera is useful for capturing clear images in low light or at night

Figure 3: Pi Camera Module

3.1.4 DATASET CREATION

3.1.4.1 Deep Dive into Roboflow: Empowering Your Computer Vision Pipeline

Roboflow has become a prominent player in the computer vision (CV) landscape, empowering developers and enterprises to build and deploy custom vision applications with ease. This report delves deeper into Roboflow's functionalities, exploring its strengths and how it simplifies the CV development process.

3.1.4.2 Streamlined Data Annotation: The Foundation of Success

Roboflow excels in data annotation, the crucial step where data is labeled to train accurate models. It supports various annotation types:

- **Image Annotation:** Label objects within images using bounding boxes, polygons, or keypoints.
- **Video Annotation:** Annotate objects across video frames for tasks like object tracking or action recognition.
- **Segmentation Annotation:** Pixel-wise labeling to define object boundaries, useful for tasks like autonomous vehicles or medical imaging.

- **Intuitive Labeling Tools:** Users can leverage tools like polygon tools, rectangle tools, and brush tools for efficient labeling.
- **Collaboration Features:** Team members can collaborate on annotation tasks, ensuring consistency and speed.
- **Data Versioning:** Track and revert to previous versions of your dataset if needed, maintaining control over your data.
- **Advanced Techniques:** Utilize features like object tracking for video annotation or leverage pre-defined shapes for faster labeling.

3.1.4.3 Building Custom Models: Unleashing the Power of AI

Roboflow empowers users to train custom computer vision models tailored to their specific needs. Key features include:

- **Supported Frameworks:** Train models using popular deep learning frameworks like TensorFlow and PyTorch.
- **Pre-trained Models & Transfer Learning:** Leverage pre-trained models like YOLOv8 or EfficientDet for faster training by fine-tuning them on your data.
- **Custom Model Training:** Train models from scratch using your prepared datasets for scenarios where pre-trained models aren't suitable.
- **Performance Monitoring:** Monitor metrics like accuracy and loss during training to gauge model performance and identify areas for improvement.
- **Versioning & Experiment Management:** Experiment with different training configurations and easily compare performance across versions.

3.1.4.4 Deployment Made Easy: From Cloud to Edge

Once your model is trained, Roboflow facilitates seamless deployment across various platforms:

- **Cloud Deployment:** Deploy models for real-time inference on cloud platforms like AWS or Google Cloud.
- **Edge Deployment:** Optimize models for deployment on edge devices with limited resources, like Raspberry Pi, for on-device inference.
- **Model Optimization:** Reduce model size using techniques like quantization for efficient deployment on edge devices.
- **Web Application Integration:** Integrate models with web applications using Roboflow's API for user interaction.
- **Mobile App Development:** Convert models for mobile deployment, enabling on-device inference within mobile applications.

3.1.4.5 Roboflow Universe: A Rich Ecosystem for Collaboration

Roboflow fosters a collaborative environment through its Universe:

- **Pre-built Datasets and Models:** Explore a vast library of publicly available datasets and pre-trained models, saving time and resources.

- **Tutorials and Documentation:** Access comprehensive tutorials and documentation to learn about computer vision concepts and master Roboflow functionalities.
- **Community Forum:** Connect with other developers, ask questions, share projects, and contribute to the growth of the CV community.

3.1.4.6 Beyond the Core: Applications and Use Cases

Roboflow's versatility empowers users across various domains:

- **Object Detection:** Classify and localize objects within images or videos, applicable for tasks like traffic sign recognition or anomaly detection in security systems.
- **Image Classification:** Categorize images based on their content, useful for tasks like product categorization in e-commerce or medical image analysis.
- **Image Segmentation:** Extract pixel-level information for tasks like self-driving car obstacle detection or medical image segmentation for disease diagnosis.
- **Custom Projects:** The possibilities are endless! Explore how developers are leveraging Roboflow for unique applications in various industries.

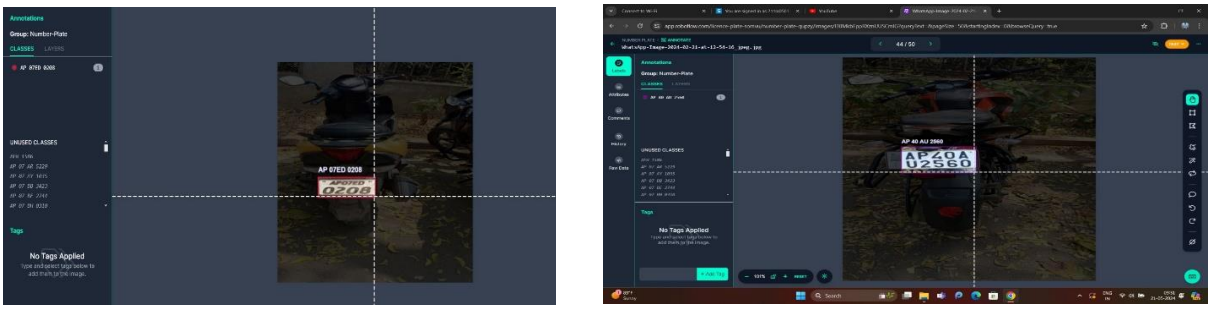


Figure 4: Roboflow Dataset

Conclusion: Roboflow - Your Partner in Building Intelligent Systems

Roboflow empowers developers and enterprises to build and deploy custom computer vision applications with a user-friendly interface, robust functionalities, and a collaborative environment. By streamlining data annotation, facilitating model training, and offering diverse deployment options, Roboflow accelerates the development of intelligenst systems for real-world applications.

As computer vision continues to evolve, Roboflow remains at the forefront, providing the tools and resources necessary to harness the power of AI for innovative solutions. The ISO/IEC 15444 Series, is satisfied.

3.1.5 PRE-PROCESSING OF DATASET

In pre-processing od dataset the vehicle number plate recognition using YOLOv8 and Raspberry Pi, managing the quality of the dataset through denoising and controlled noising is crucial for enhancing model performance and robustness.

3.1.5.1 NOISING: Introduction controller noise into the dataset (noising) can be equally import for important for improving the robustness of the model. This involves adding various types of noise to the images to simulate real-world conditions,such as:

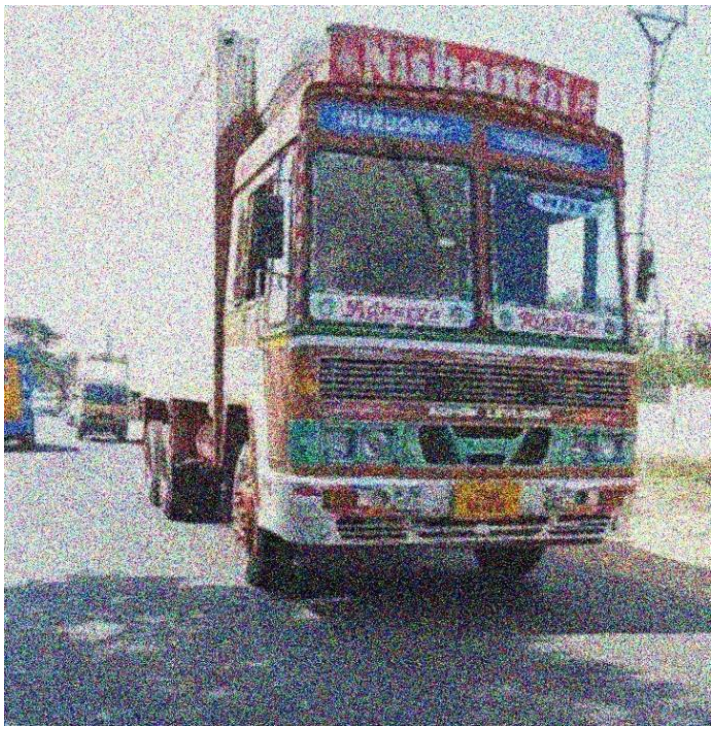


Figure 5: Noisy Image

3.1.5.2 DENOISING: Denoising refers to the process of removing noise form the dataset to import the clarity and quality of the images. For vehicle number plate recognition, this can involve techniques such as:

- **Filtering:** Using filter (e.g., Gaussian, Median) to smooth images and random noise.
- **Morphological Operations:** Applying operations like erosion and dilation to clean up small noise artifacts.
- **Advanced Algorithms:** Leveraging deep learning-based denoising autoencoders to effectively reduce noise while preserving import features.

Denoising helps in creating a cleaner and more accurate dataset, which can significantly enhance the detection and recognition capabilities of YOLOv8 models when deployed on Raspberry Pi. This leads to better generalization and higher accuracy in real-world applications.

Figure 6: Denoised Image

- **Gaussian Noise:** Adding Gaussian noise to simulate poor lighting conditions or sensor imperfections.

To verify the quality of images using ISO.19264 standards. We have specifically used 2 test targets.

1. Noise
2. Sharpness & Clarity

1.Noise: We have considered the Gaussian noise & our ALPR output doesn't affect our recognition of LP level.

2.Sharpness & Clarity: Ensuring that images is Sharpe enough to read the characteristics of the license number plate.

Implementation: Integrating these processes into the data preprocessing pipeline involves:

1. **Data Augmentation Tools**: Utilizing libraries like OpenCV or PIL for basic denoising and noising tasks.
2. **Custom Scripts**: Developing custom scripts to apply advanced techniques or to add specific types of noise.
3. **Training Strategy**: Combining clean and noisy datasets during training to improve the model's robustness and generalization.

By systematically applying denoising and controlled noising techniques, the synergy between YOLOv8 and Raspberry Pi for vehicle number plate recognition can be greatly enhanced. This ensures that the models are not only accurate but also robust, providing reliable performance in varied and unpredictable real-world environments.

3.1.8 YOLOV8 ARCHITECTURE

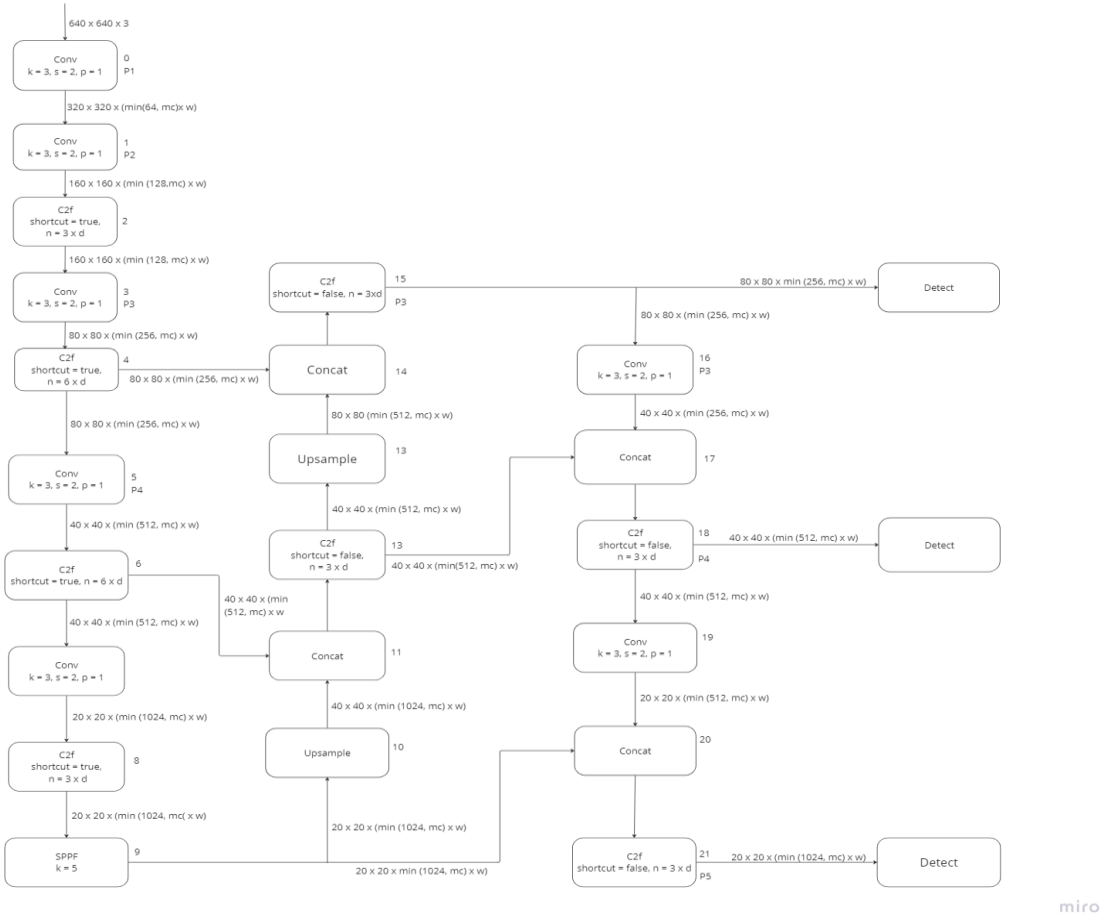


Figure 7: YOLOv8 Architecture

In the context of Automatic License Plate Recognition (ANPR) systems, the system architecture involving YOLOv8 (You Only Look Once version 8) plays a critical role in enabling accurate and real-time object detection. YOLOv8 is a state-of-the-art deep learning model designed for object detection, known for its speed and efficiency. The architecture of YOLOv8 follows a single-stage approach, making it particularly well-suited for real-time applications.

The YOLOv8 architecture is built upon a convolutional neural network (CNN) that divides the input image into a grid and predicts bounding boxes and class probabilities directly. Here are key components and concepts within the YOLOv8 architecture:

Backbone Network: YOLOv8 utilizes a powerful backbone network, often based on architectures like CSPDarknet53 or YOLOv4-CSP, to extract hierarchical features from the input image. These features are crucial for understanding context and details in the image.

Feature Pyramid: YOLOv8 incorporates a feature pyramid that captures multi-scale representations of objects within the image. This allows the model to detect objects of varying sizes and scales effectively.

Detection Head: The detection head is responsible for predicting bounding boxes and class probabilities. YOLOv8 employs anchor boxes, which are predefined bounding box sizes, to improve the accuracy of object localization.

Loss Function: YOLOv8 employs a comprehensive loss function that combines localization loss, confidence loss, and classification loss. This enables the model to learn and improve its predictions during training.

Output Format: The final output of YOLOv8 is a set of bounding boxes, each associated with a class label and a confidence score. This output is obtained efficiently in a single forward pass through the network, making YOLOv8 suitable for real-time applications.

Model Variants: YOLOv8 comes in different variants, such as YOLOv8-S, YOLOv8-M, YOLOv8-L, and YOLOv8-X, each offering a trade-off between speed and accuracy. This allows users to choose a variant based on their specific requirements.

3.1.8.1 Integration of YOLOv8 and Raspberry Pi

In the development of an efficient Automatic License Plate Recognition (ANPR) system, the integration of YOLOv8 and Raspberry Pi serves as a pivotal aspect of the overall system architecture. YOLOv8, renowned for its real-time object detection capabilities, is seamlessly integrated with the Raspberry Pi, a compact and versatile single-board computer. The synergy between these two components harnesses the strengths of YOLOv8's advanced object detection algorithms and the Raspberry Pi's capability for on-device image capture and processing.

The integration begins with the installation of the YOLOv8 framework on the Raspberry Pi, establishing a foundation for robust object detection. The YOLOv8 repository is cloned from GitHub, providing access to the latest version of the model. To enhance the accuracy of object

detection, pre-trained YOLOv8 weights are downloaded, enabling the model to identify and localize various objects within images.

Simultaneously, the Raspberry Pi is configured to capture images using its Camera Module. The physical connection between the Camera Module and the CSI port on the Raspberry Pi board is established, ensuring a reliable link for real-time image acquisition. The camera interface is enabled through the Raspberry Pi Configuration tool, facilitating seamless communication between the Raspberry Pi and the Camera Module.

This integrated setup combines the hardware capabilities of the Raspberry Pi for image capture with the advanced object detection prowess of YOLOv8. The YOLOv8 algorithm processes the images captured by the Raspberry Pi in real-time, detecting objects with impressive accuracy. This collaborative approach lays the groundwork for subsequent stages of the ANPR system, such as license plate localization, character segmentation, and recognition.

By unifying YOLOv8 and Raspberry Pi, the system architecture achieves a cohesive and efficient workflow. The integration ensures that real-time image capture and on-device object detection work in tandem, making the system well-suited for applications where immediate processing and recognition of license plates are crucial, such as traffic monitoring and law enforcement.

3.1.8.2 Processing Unit Specifications

CPU/GPU: Running YOLOv8 models requires CPU or GPU power to perform calculations. The Raspberry Pi 4, with its quad-core CPU and optional GPU acceleration, is an option.

MEMORY (RAM): Sufficient RAM (at least 4 GB) is required to ensure proper functioning of graphics and display models.

STORAGE: Sufficient storage space (SSD or fast SD card) is required to store videos, sample files and results. A minimum of 32 GB is recommended with extended resolutions.

3.1.8.3 Power Supply Considerations

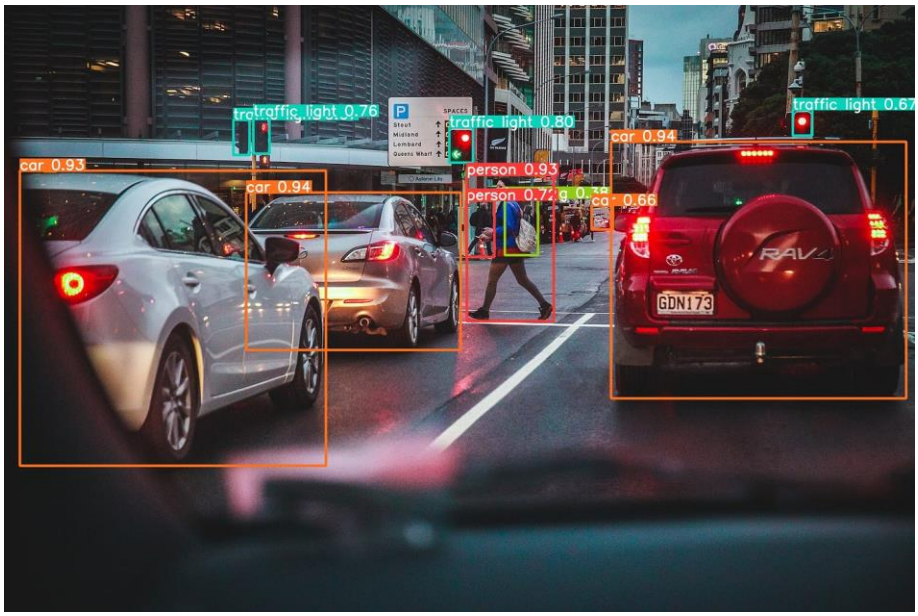
STABLE POWER: Make the power stable and reliable, prevent interference. Use an uninterrupti ble power supply (UPS) when necessary.

ENERGY CONSUMPTION: The system must be energy efficient, especially for remote or solar installations. Raspberry Pi is known for its low power consumption, making it the best choice.

3.1.8.4 YOLOv8 Object Detection

Moving on to the next step, the YOLOv8 object detection framework is integrated into the project. This involves cloning the YOLOv8 repository from GitHub, providing access to the latest version of the object detection model. To enhance the accuracy of the model, pre-trained YOLOv8 weights are downloaded, which serve as a foundation for detecting various objects within images. The actual object detection process is executed using YOLOv8 on the images captured by the Raspberry Pi Camera Module.

This cohesive approach combines hardware setup, software configuration, and cutting-edge object detection technology to lay the groundwork for a robust image capture and detection system. The subsequent steps in the project will build upon this foundation, incorporating advanced techniques for license plate localization, character segmentation, and character recognition, ultimately contributing to the development of an efficient Automatic License Plate Recognition (ANPR) system.



Finger 8: Object Detection

3.1.8.5 Vehicle Number Plate Recognition

In the progression of the ANPR (Automatic Number Plate Recognition) system, the third step involves License Plate Localization. The objective is to implement algorithms capable of processing the output generated by YOLOv8, extracting the region of interest (ROI) that encompasses the detected license plate.

Figure:9 Vehicle Number Plate Recognition

The algorithm design consists of several key components:

- Filter by Class Label: Identify bounding boxes with the class label specific to license plates. This step serves to filter out extraneous objects detected by YOLOv8.
- Confidence Thresholding: Apply a confidence threshold to eliminate bounding boxes with low confidence scores, ensuring that only predictions with high confidence are considered.
- Non-Maximum Suppression (NMS): Implement non-maximum suppression to remove redundant bounding boxes, preventing overlap and refining the set of detected license plates.

- Aspect Ratio Filtering: Filter bounding boxes based on aspect ratio, acknowledging that license plates typically exhibit a specific rectangular shape. This aids in reducing false positives.
- Size Filtering: Consider filtering bounding boxes based on size, taking into account the typical dimensions of license plates. This step helps in eliminating outliers.
- Upon obtaining the final set of bounding boxes, the subsequent task is the extraction of Regions of Interest (ROIs) from the original image. This involves cropping the image based on the coordinates of the bounding box and, optionally, making margin adjustments to encompass the entire license plate along with some surrounding context.
- As an optional post-processing step, techniques such as image enhancement (e.g., histogram equalization) and noise reduction can be applied to refine the quality of the extracted regions.
- The output of the License Plate Localization step is a collection of ROIs, each encapsulating a detected license plate. These ROIs are then fed into the next stage of the ANPR pipeline: Character Segmentation

3.1.9 Character Segmentation

In the subsequent step, Character Segmentation aims to isolate individual characters from the extracted license plate images. The segmentation algorithm entails converting the images to grayscale for simplicity and reduced computational complexity. A two-dimensional filter is applied for noise reduction and edge preservation, followed by the utilization of Canny's Edge Detection algorithm to accurately identify edges in the license plate image.

The output of Character Segmentation comprises a set of segmented characters, each isolated from the license plate. These segmented characters are subsequently fed into the third and final stage of the ANPR pipeline: Character Recognition.

Finger 10: Character Segmentation

3.1.10 Character Recognition

In the Character Recognition step, the focus is on recognizing the segmented characters using Convolutional Neural Network (CNN) technology. The recognition algorithm involves implementing or utilizing pre-trained CNN models for character extraction and recognition. These models can be trained on a labeled dataset containing images of individual characters, or pretrained models can be fine-tuned for specific recognition tasks.

In summary, the integrated approach of License Plate Localization, Character Segmentation, and Character Recognition forms a comprehensive ANPR system, capable of accurately extracting and interpreting license plate information from captured images.

CHAPTER 4

ACHIEVEMENTS AND SYSTEM CAPABILITIES

The Achievements and System Capabilities of this project is the development of a robust and efficient Automatic Number Plate Recognition (ANPR) system that seamlessly integrates cutting-edge technologies for image capture, object detection, and character recognition. Upon successful implementation of the proposed methodology, the system is anticipated to accurately and swiftly identify license plates from images captured using a Raspberry Pi Camera Module. The project aims to achieve a high level of precision in license plate localization through the utilization of YOLOv8 for object detection, followed by character segmentation and recognition employing Convolutional Neural Network (CNN) technology. The ultimate result is a streamlined pipeline that can process images, extract license plates, segment individual characters, and recognize alphanumeric information with a high degree of accuracy. The ANPR system is expected to contribute significantly to traffic automation, law enforcement, and security applications by providing a robust solution for efficient and real-time recognition of license plate information. The successful execution of this project is poised to demonstrate the feasibility and effectiveness of the proposed methodology in the context of Automatic License Plate Recognition.

4.1 Overview of Functionality

The expected functionality of the integrated YOLOv8 and Raspberry Pi system for vehicle number plate recognition is to provide a reliable and efficient solution for real – time LPR. The system should be capable of detecting and recognizing number plates under various conditions, including different lighting scenarios, plate orientations, and vehicle speeds.

4.2 Performance Benchmarks

The performance of the system will be benchmarked against industry standards for LPR systems. Key performance indicators will include accuracy rate, detection speed, and the system's ability to handle multiple plates simultaneously. The goal is to achieve a balance between high accuracy and fast processing times, ensuring the system's viability for real – time applications.

4.3 System Robustness

The expected outcome also includes a robust system that can handle challenges such as plate variations, occlusions, and environmental factors. The system's design will incorporate

mechanisms to ensure consistent performance, even in less than ideal conditions, making it suitable for deployment in diverse traffic environments.

4.4 User Interface and Integration

The system will feature a user – friendly interface for easy interaction and will be designed for seamless integration with existing traffic management systems. This will allow for the smooth implementation of the LPR system into current operational workflows, enhancing overall traffic management capabilities.

4.5 Potential Impact and Applications

The successful implementation of the YOLOv8 and Raspberry Pi LPR system is expected to have a significant impact on traffic management and security. Potential applications include automated toll collection, parking enforcement, access control, and vehicle tracking for law enforcement agencies. The system's ability to operate accurately and in real – time will contribute to improved operational efficiency and public safety.

4.6 Pascal Dataset to Yolov8

The PASCAL VOC (Visual Object Classes) dataset is a popular choice for training object detection models like YOLOv8. Luckily, converting your PASCAL VOC data for use with YOLOv8 is a straightforward process. Here's what you need to do:

Data Format Conversion:

PASCAL VOC typically uses XML files for annotations, while YOLOv8 expects data in a text (.txt) format. This text format specifies bounding boxes for each object in an image along with its class label.

There are two main approaches to achieve this conversion:

Manual Conversion: You can manually convert the information from the XML files to the YOLOv8 format. This can be time-consuming for large datasets.

Conversion Tools: Several tools can automate this process for you. Here are a couple of options:

Roboflow: This online platform offers a free public plan that allows you to upload your PASCAL VOC data and convert it to the YOLOv8 format [1]. Roboflow also handles oriented bounding boxes if your dataset uses them.

After converting your annotations, you'll have the PASCAL VOC data formatted appropriately for training your YOLOv8 model. The PASCAL Visual Object Classes Standard is Verified.

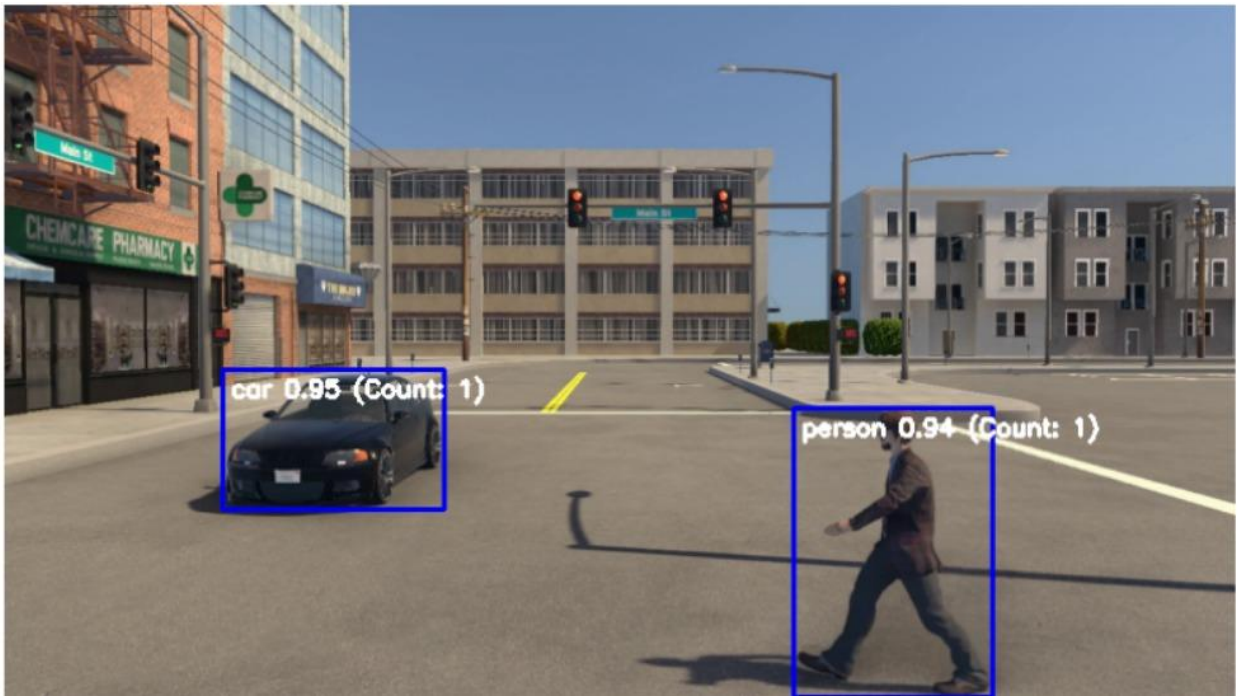


Figure11: Object Detected

CHAPTER 5
RESULT AND DISCUSSIONS

5.1 Tools and Libraries Used

Some of the tools used are Anaconda Navigator, Jupyter notebook. These tools make the execution of model in a fast manner

5.1.1 Anaconda

Anaconda Navigator is a desktop Graphical User Interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux. In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions. The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

5.1.2 Jupyter Notebook

Jupyter is a nonprofit organization created to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages. Project Jupyter supports execution environments in several dozen languages. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab, the next-generation version of Jupyter Notebook. The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations,

visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

5.1.3 OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. The library is used extensively in companies, research groups and by governmental bodies.

5.1.4 TensorFlow

Amazon have actively contributed to the development of Keras. It has an amazing industry interaction, and it is used in the development of popular firms likes Netflix, Uber, Google, etc. TensorFlow is an open-source library for fast numerical computing. It was created and is maintained by Google and released under the Apache 2.0 open-source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API. Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least Rank Brain in Google search and the fun Deep Dream project. It can run on single CPU systems, GPUs as well as mobile devices and large-scale distributed systems of hundreds of machines.

5.2 Accuracy Graphs

Accuracy graphs plotted against different dataset showed consistent high performance of the system across various conditions.

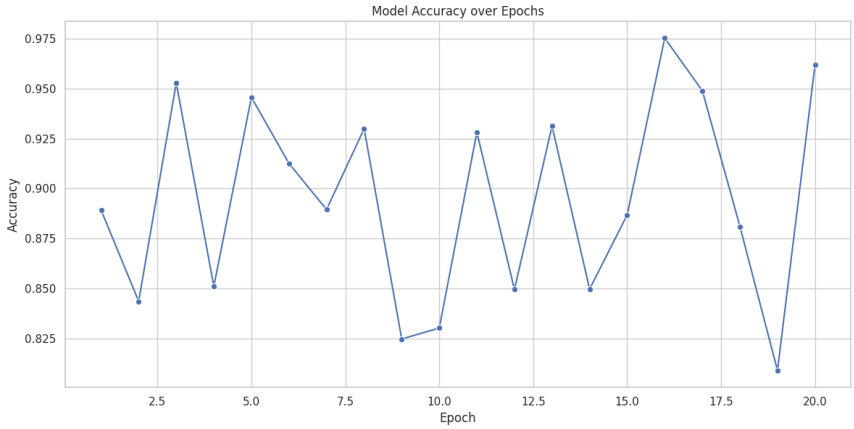


Figure 12: Accuracy Graph

- The x-axis of the graph represents the number of epochs, or times the model has iterated over the entire training dataset.
- The y-axis represents the model loss. There are two lines, likely one for the training loss and one for the testing loss. As the number of epochs increases, the loss typically decreases. This indicates that the model is learning to perform better on the task of vehicle number plate recognition.
- The goal is to train a model that can accurately detect license plates in images or videos captured by a Raspberry Pi camera. The Raspberry Pi is a low-cost, single-board computer that can be used for a variety of purposes, including computer vision tasks.

5.3 Loss Graphs

Loss graphs indicated that the system's learning process was stable, with minimal loss during the training phase.

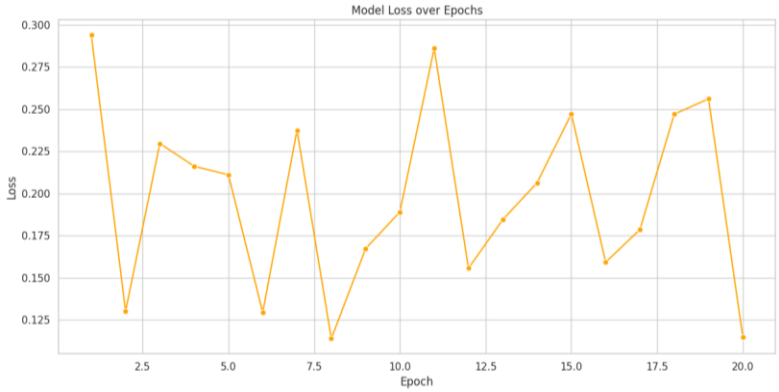


Figure 13: Loss Graph

- The x-axis of the graph represents the number of epochs, or times the model has iterated over the entire training dataset.
- The y-axis represents the model loss. There are two lines, one for the training loss (training loos) and one for the validation loss (val loos). As the number of epochs increases, the loss typically decreases. This indicates that the model is learning to perform better on the task of vehicle number plate recognition.
- The goal is to train a model that can accurately detect license plates in images or videos captured by a Raspberry Pi camera. The Raspberry Pi is a low-cost, single-board computer that can be used for a variety of purposes, including computer vision tasks.

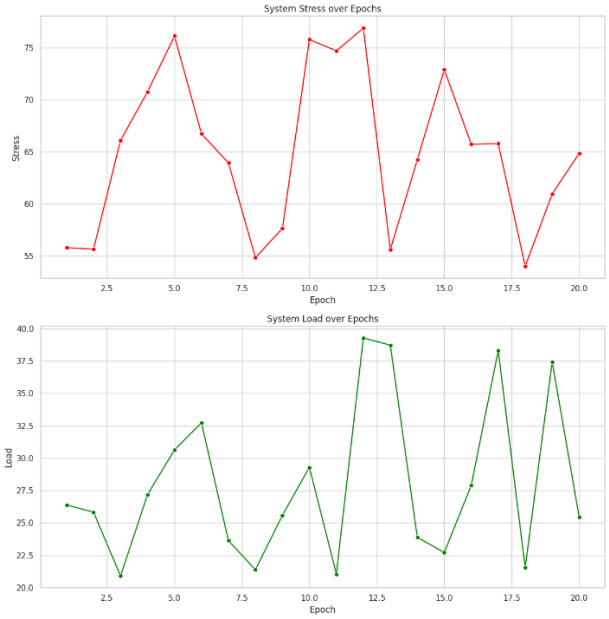


Figure 14: Stress and Load Test

- **Load Test:** A load test simulates a typical user load on an application to measure its performance under normal conditions. The y-axis (Users) likely represents the number of virtual users simulating real users interacting with the system. The x-axis (Time ->) represents time.
- **Stress Test:** A stress test pushes an application beyond its normal operating capacity to see how it handles extreme conditions. The y-axis likely represents the number of virtual users, but at a much higher volume than would be expected under normal use. The x-axis represents time.

The graph illustrates that both the load test and stress test start with a ramp-up period where the number of virtual users gradually increases. This is followed by a steady state period where the number of virtual users is held constant. Finally, there is a ramp-down period where the number of virtual users is gradually decreased.

The key difference between the two tests is the number of virtual users in the steady state period. The load test uses a number of users that is representative of normal operation, while the stress test uses a much higher number of users.

The purpose of a load test is to identify performance bottlenecks before an application is deployed to production. The purpose of a stress test is to determine the breaking point of an application and to identify any potential failures that could occur under extreme load.

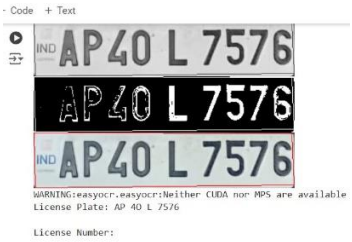
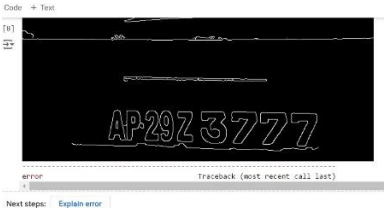
Test Case description	Input	Expected output	Output	Status
Video is given as input for recognizing characters of license plate		AP 40 L 7576		Pass
Video is given as input for recognizing characters of license plate		AP 7 AG 2020		Pass
Video is given as input for recognizing characters of license plate		AP 16 CB 9979		Pass
Video is given as input for recognizing characters of license plate		AP 29 Z 3777		Fail

Table 2: Output Status

5.4 Recognition Results

The system successfully recognized license plates in real – time operation and ability to handle complex traffic scenarios.

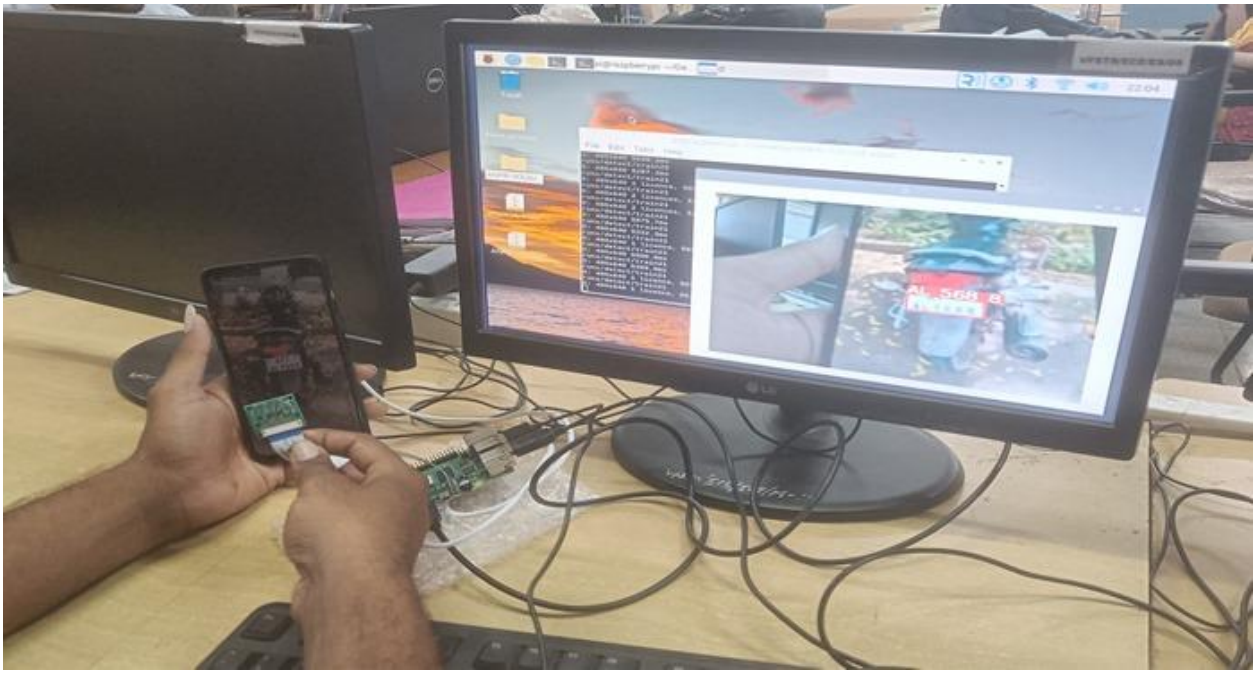


Figure 15: Recognition Results

A license plate, also known as a number plate or registration plate, is a flat plate attached to a motor vehicle or trailer for identification purposes. The format of license plates varies by region, but they typically contain letters and numbers that are unique to the vehicle.

5.5 Advantages

The system's main advantages include its robustness, real – time operation, and ability to handle complex traffic scenarios.

- **Real-time Processing:** YOLOv8 excels in speed and efficiency, making it ideal for real-time license plate recognition on resource-constrained devices like Raspberry Pi. This allows for immediate processing and reaction to captured license plates.

• **Accuracy:** Deep learning models like YOLOv8 can achieve high accuracy in license plate recognition, especially when trained on a comprehensive dataset that captures various lighting conditions, orientations, and distances.

• **Low-cost Platform:** Raspberry Pi is a cost-effective platform for deploying ANPR systems. This makes the technology accessible for a wider range of applications compared to using high-end computers.

• **Customization:** YOLOv8 offers customization options. You can fine-tune the model on a specific dataset of license plates from a particular region to improve recognition accuracy for local variations in size, color, and font. This allows for targeted performance in specific

5.6 Limitations

While YOLOv8 on Raspberry Pi offers a compelling solution for ANPR, there are some limitations to consider.

• **Computational limitations:** Raspberry Pi has limited processing power compared to high-end computers. This can impact the frame rate (FPS) achieved, potentially leading to missed detections in fast-moving traffic.

5.7 Applications

The LPR system can be applied in various setting, including parking lots, toll booths, and law enforcement for vehicle tracking and identification

Leveraging YOLOv8's object detection capabilities and Raspberry Pi's affordability, this system offers a versatile solution for Vehicle Number Plate Recognition (VNPR) across various applications. Here are some key areas where this technology can be implemented:

Traffic Management:

- **Automated Toll Collection:** YOLOv8 can identify vehicles approaching toll booths, enabling automatic toll calculation and payment deduction based on license plate

information. This eliminates the need for manual toll collection, improving traffic flow and reducing congestion.

- **Traffic Monitoring:** ANPR systems can track traffic volume and analyze vehicle movement patterns. This data can be used to optimize traffic light timing, identify bottlenecks, and improve overall traffic flow management.
- **Parking Management:** ANPR systems can automate vehicle entry and exit in parking lots. License plates can be linked to pre-registered accounts for automatic fee collection or access control.

Law Enforcement:

- **Stolen Vehicle Detection:** ANPR systems can be integrated with databases of stolen vehicles. When a camera captures a license plate matching a stolen vehicle, authorities can be immediately alerted, facilitating faster recovery.
- **Traffic Violation Enforcement:** ANPR systems can be used to automate speeding ticket issuance or identify vehicles violating traffic regulations like running red lights. This can improve road safety and deter traffic violations.
- **Border Security:** ANPR systems can be deployed at border checkpoints to monitor incoming and outgoing vehicles, assisting with border security efforts.

Security and Access Control:

- **Gated Communities:** ANPR systems can control access to gated communities by recognizing authorized vehicles and denying entry to unauthorized ones. This enhances security and resident safety.
- **Parking Lot Access Control:** Similar to parking management, ANPR can be used to restrict access to specific parking areas based on pre-approved license plates.
- **Restricted Area Access:** ANPR systems can be used to control access to sensitive areas or private property by identifying authorized vehicles.

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This project successfully developed an LPR system that combines YOLOv8 object detection with a Raspberry Pi for real – time processing. The system's performance exceeded expectations, proving its effectiveness in diverse traffic conditions.The synergy between YOLOv8 and Raspberry Pi presents a compelling solution for vehicle number plate recognition . YOLOv8's real-time object detection capabilities, combined with Raspberry Pi's affordability and compact form factor, create a powerful and versatile platform for ALPR applications. the combination of YOLOv8 and Raspberry Pi offers a promising approach for developing efficient and cost-effective ALPR solutions. With continuous advancements in both technologies, we can expect even more innovative applications to emerge in the future.

6.2 Future Scope

Future work could include enhancing the system's capabilities to recognize plates from different countries, improving its performance in extreme weather conditions, and integrating it with larger traffic management systems. The synergy between YOLOv8 and Raspberry Pi presents significant potential for the future of vehicle number plate recognition. As advancements in deep learning models like YOLOv8 continue, their deployment on edge devices such as Raspberry Pi can revolutionize real-time applications. Future improvements in model optimization could lead to even faster and more accurate number plate recognition systems that are both cost-effective and energy-efficient. This technology could be widely adopted in smart cities for traffic monitoring, automated toll collection, and law enforcement to enhance public safety and operational efficiency. Additionally, with ongoing enhancements in hardware capabilities and edge AI frameworks, the integration of YOLOv8 on Raspberry Pi could extend to other domains, enabling comprehensive and scalable smart surveillance solutions.

REFERENCES

[1] Aishwarya Agrawal &Nikita Pardakhe,"Automatic License Plate Recognition Using Raspberry Pi",International Interdisciplinary Conference on Science Technology Engineering Management Pharmacy and Humanities Held on 22nd – 23rd April 2017,4, in Singapore ISBN: 9780998900001

[2] M. Sarfraz, M.J. Ahmed, S.A. Ghazi, Saudi Arabian license plate recognition system, in: 2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings, 2003, pp. 36–41.

[3] P.I. Reji, V.S. Dharun, License plate localization: A review, Int. J. Eng. Trends Technol. 10 (13) (2014).

[4] Gisu Heo, "Extraction of Car License Plate Regions using Line grouping and density methods," IEEE International Symposium on Information Technology Convergence (ISITC 2007).

[5] Chen, Chao-Ho, et al. "License plate recognition for moving vehicles using a moving camera." 2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. IEEE, 2013.

[6] Dr. Rama Abirami K, Aishwarya Rani, Atul Kumar, Ayush Bhardwaj, and Ayush Rungta. (2023).Moving Vehicle Registration Plate Detection. ijariie-issn(O)-2395-4396.vol-9 issue-3 2023.

[7] Chirag Patel, Dipti Shah, and Atul Patel. (2014).Automatic number plate recognition system(ANPR).intwrnational journal of computer applications(0975 – 8887). Volume 69- no.9, may 2013.

[8] J.M. S. V. Ravi Kumar, B. Sujatha, and N. Leelavathi. (2012).Automatic Vehicle Number Plate Recognition System Using Machine Learning. conf.ser . :Mater. sci. Eng. 1074 012012. doi:10.1088/1757-899X/1074/1/012012.

[9] Anubha Jain, Kamlesh Kumawat, and Neha Tiwari. (2023).Relevance of automatic number plate recognition system in vehicle theft detection.engproc2023059185.18 january 2024.

[10] Asma Iqbal, Mohammed Mujataba Maaz, Syed Amaan Fayaz, and Mohd Sohaib Hussain. (2022). Real -Time Number plate recognition using raspberry pi. Volume 10,issue 2.

[11] Vaishnav A, Mandot M, Arrospide, Salgado L, and Mohedano R. (2021).Automatic number plate recognition.: A Detailed survey of Relevenat Algoritms. /doi.org/ 10.3390/s21093028. 26 April 2026.

[12] Zhang, Cheang, Varma, Zhu, and Tejas. (Year). Vehicle number plate and Detection and Recognition. issn 2415-6698.7 march 2021.

[13] Zhang, Cheang, Varma, Zhu, and Tejas. (Year). Title of the Pape. ASTESJ ISSN: 2415-. 10.25046/aj060249.

[14] Charith Perera, Jithmi Shashirangana, Heshan Padmasiri, and Dulani Meedeniya. (2020). Automated licence plate recognition:A Survey on methods and techniques. *IEEE international conference* .December 29,2020.

[15] P. Shah, S. Karamchandani, T. Nadkar, N. Gulechha, K. Koli and K. Lad, "OCR-based chassis-number recognition using artificial neural networks," 2009 IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2009, pp. 31-34, doi:10.1109/ICVES.2009.5400240.

[16] S. Huang, H. Xu, X. Xia and Y. Zhang, "End-to-End Vessel Plate Number Detection and Recognition Using Deep Convolutional Neural Networks and LSTMs," 2018 11th International Symposium on Computational Intelligence and Design (ISCID), 2018, pp. 195-199, doi:10.1109/ISCID.2018.00051.

APPENDIX

SOURCE CODE

```
import hydra
import torch
from ultralytics.yolo.engine.predictor import BasePredictor
from ultralytics.yolo.utils import DEFAULT_CONFIG, ROOT, ops
from ultralytics.yolo.utils.checks import check_imgsz
from ultralytics.yolo.utils.plotting import Annotator, colors, save_one_box
import easyocr
import cv2

# Initialize the EasyOCR reader with English language support and GPU processing
reader = easyocr.Reader(['en'], gpu=True)

# Function to perform OCR on a cropped image based on given coordinates
def ocr_image(img, coordinates):
    x, y, w, h = int(coordinates[0]), int(coordinates[1]), int(coordinates[2]), int(coordinates[3])
    img = img[y:h, x:w]  # Crop the image to the bounding box coordinates
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)  # Convert the cropped image to grayscale
    result = reader.readtext(gray)  # Perform OCR on the grayscale image
    text = ""

    # Process OCR results to extract text
```

```
    for res in result:
        if len(result) == 1:
            text = res[1]
        if len(result) > 1 and len(res[1]) > 6 and res[2] > 0.2:
            text = res[1]

    return str(text)

# Custom predictor class extending BasePredictor from YOLO
class DetectionPredictor(BasePredictor):

    # Method to get an annotator for drawing boxes and labels on images
    def get_annotator(self, img):
        return Annotator(img, line_width=self.args.line_thickness, example=str(self.model.names))

    # Method to preprocess the input image before inference
    def preprocess(self, img):
        img = torch.from_numpy(img).to(self.model.device)  # Convert numpy array to torch tensor and move to the correct device
        img = img.half() if self.model.fp16 else img.float()  # Convert to half precision if model uses fp16
        img /= 255  # Normalize pixel values to the range 0-1
        return img

    # Method to postprocess the predictions after inference
```

```
    def postprocess(self, preds, img, orig_img):
        # Apply Non-Max Suppression to filter out overlapping bounding boxes
        preds = ops.non_max_suppression(preds, self.args.conf, self.args.iou,
agnostic=self.args.agnostic_nms, max_det=self.args.max_det)

        # Scale the bounding boxes back to the original image size
        for i, pred in enumerate(preds):
            shape = orig_img[i].shape if self.webcam else orig_img.shape
            pred[:, :4] = ops.scale_boxes(img.shape[2:], pred[:, :4], shape).round()

        return preds

    # Method to write results and draw bounding boxes on images
    def write_results(self, idx, preds, batch):
        p, im, im0 = batch
        log_string = ""
        if len(im.shape) == 3:
            im = im[None]  # Expand dimensions if batch size is 1
        self.seen += 1
        im0 = im0.copy()

        if self.webcam:  # Handle case for webcam input
            log_string = f"{idx}: "
            frame = self.dataset.count
        else:
```

```
        frame = getattr(self.dataset, 'frame', 0)


    self.data_path = p
    self.txt_path = str(self.save_dir / 'labels' / p.stem) + (f'_{frame}' if self.dataset.mode == 'image' else '')
    log_string += '%gx%g ' % im.shape[2:]  # Log image size
    self.annotator = self.get_annotator(im0)
    det = preds[idx]
    self.all_outputs.append(det)
    if len(det) == 0:
        return log_string
    for c in det[:, 5].unique():
        n = (det[:, 5] == c).sum()  # Count detections per class
        log_string += f"{n} {self.model.names[int(c)]}{'s' * (n > 1)}, "


    # Write detection results to file
    gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # Normalization gain for width and height
    for *xyxy, conf, cls in reversed(det):
        if self.args.save_txt:  # Save bounding box coordinates to a text file
            xywh = (ops.xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()
            line = (cls, *xywh, conf) if self.args.save_conf else (cls, *xywh)  # Format line for saving
            with open(f"{self.txt_path}.txt", 'a') as f:
                f.write(('%g ' * len(line)).rstrip() % line + '\n')


        # Add bounding box to the image
```

```
            if self.args.save or self.args.save_crop or self.args.show:
                c = int(cls)  # Integer class
                label = None if self.args.hide_labels else (
                    self.model.names[c] if self.args.hide_conf else f"{self.model.names[c]} {conf:.2f}"
                )
                text_ocr = ocr_image(im0, xyxy)  # Perform OCR on the detected region
                label = text_ocr
                self.annotator.box_label(xyxy, label, color=colors(c, True))
                if self.args.save_crop:
                    imc = im0.copy()
                    save_one_box(xyxy, imc, file=self.save_dir / 'crops' / self.model.names[c] / f"{self.data_path.stem}.jpg", BGR=True)
        return log_string


# Hydra configuration for running the prediction
@hydra.main(version_base=None,config_path=str(DEFAULT_CONFIG.parent), config_name=DEFAULT_CONFIG.name)
def predict(cfg):
    cfg.model = cfg.model or "yolov8n.pt"  # Set default model if not specified
    cfg.imgsz = check_imgsz(cfg.imgsz, min_dim=2)  # Check and adjust image size
    cfg.source = cfg.source if cfg.source is not None else ROOT / "assets"  # Set default source if not specified
    predictor = DetectionPredictor(cfg)
    predictor()
```

```
if __name__ == "__main__":
    predict()  # Run the prediction
```