

AUTOMATIC NUMBER PLATE RECOGNITION USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering
degree in Computer Science and Engineering with specialization in Artificial Intelligence

by

SHYAM ANAND S (Reg.No - 40731094)

SHOUVIN A (Reg.No - 40731093)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI, CHENNAI – 600 119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SHYAM ANAND S(40731094)** and **SHOUVIN A(40731093)** who carried out the Project work entitled **“AUTOMATIC NUMBER PLATE RECOGNITION USING DEEP LEARNING”** under my supervision from November 2023 to April 2024.

Internal Guide

Dr. R. SETHURAMAN, B.E., M.E., Ph.D.

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Shouvin A (Reg.No - 40731093)**, hereby declare that the Project Report entitled **“AUTOMATIC NUMBER PLATE RECOGNITION USING DEEP LEARNING”** done by me under the guidance of **Dr.R.Sethuraman,B.E.,M.E.,Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with specialization in Artificial Intelligence.**

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, and **Dr. L. Lakshmanan, M.E., Ph.D.**, and **Dr. S. Vigneshwari, M.E., Ph.D.**, Heads of the **Department of Computer Science and Engineering** for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R. Sethuraman, B.E., M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Technologies and services towards smart-vehicles and Intelligent-Transportation Systems (ITS), continues to revolutionize many aspects of human life. This paper presents a detailed survey of current techniques and advancements in Automatic Number-Plate-Recognition (ANPR) systems, with a comprehensive performance comparison of various real-time tested and simulated algorithms, including those involving computer vision. ANPR technology has the ability to detect and recognize vehicles by their number- plates using recognition techniques. Even with the best algorithms, a successful ANPR system deployment may require additional hardware to maximize its accuracy. The number plate condition, non-standardized formats, complex scenes, camera quality, camera mount position, tolerance to distortion, motion-blur, contrast problems, reflections, processing and memory limitations, environmental conditions, indoor/outdoor or day/night shots, software- tools or other hardware-based constraint may undermine its performance.

This inconsistency, challenging environments and other complexities make ANPR an interesting field for researchers. The Internet-of-Things is beginning to shape future of many industries and is paving new ways for ITS. ANPR can be well utilized by integrating with RFID-systems, GPS, Android platforms and other similar technologies. Deep-Learning techniques are widely utilized in CV field for better detection rates. This research aims to advance the state-of-knowledge in ITS built on CV algorithms; by citing relevant prior work, analyzing and presenting a survey of extraction, segmentation and recognition techniques whilst providing guidelines on future trends in this area. Its primary goal is to determine the presence, position, and often the class of objects in a given visual input. It maintains the identity of objects across frames and estimates their positions, velocities, and other relevant attributes. Single object tracking algorithms are used to follow a single object of interest to achieve accuracy. The scope of this project is to detect and track objects (Vehicles and Pedestrians) in a video by using tracking-by-detection approach. Many authors have proposed different methods to identify and track the objects in the video frames. Few algorithms are based on image processing techniques and as the recent increase in the technology, deep learning based methods are giving promising results.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 Data acquisition and preprocessing module	1
	1.2 Object detection module	2
	1.3 Object tracking module	3
2	LITERATURE SURVEY	5
	2.1 Overview of existing research and literature	5
	2.2 Inferences from literature survey	5
3	REQUIREMENT ANALYSIS	7
	3.1 Aim and Scope of ANPR System	7
	3.2 Feasibility Studies/Risk Analysis of the Project	9
	3.3 Software Requirements Specification Document	11
4	DESCRIPTION OF PROPOSED SYSTEM	14
	4.1 Overview of YOLO object detection algorithm	14
	4.2 Architecture / Overall Design of Proposed System	16
5	IMPLEMENTATION DETAILS	19
	5.1 Implementation And Testing Plan	19
	5.2 Project Management Plan	23
6	RESULTS AND DISCUSSIONS	26
	6.1 Presentation of Findings	26
	6.2 Data Analysis and Interpretation	26
7	CONCLUSION	28
	REFERENCES	30

APPENDIX	31
A. SOURCE CODE	31
B. SCREENSHOTS	41
C. RESEARCH PAPER	45
D. CERTIFICATE	50

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	System Workflow for Proposed System	9
3.2	System Architecture for Proposed System	10
3.3	Data Extracted to CSV	13
4.1	Dataset for Training Model	15
4.2	Systematic Process	18
5.1	Sample Video	20
5.2	Output Video	22
B.1	Importing Required Libraries	41
B.2	Preprocessing the data	41
B.3	Extracting the frames	42
B.4	Creating Custom ML model	42
B.5	Evaluating the Model	43
B.6	Extracting CSV file	43
B.7	Adding values to missing data	44
B.8	Tracking the Number Plates	44

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ANPR	Automatic Number Plate Recognition
OCR	Optical Character Recognition
YOLO	You Only Look Once
ITS	Intelligent-Transportation Systems
GPS	Global Positioning System
RFID	Radio-Frequency IDentification
CV	Computer Vision

CHAPTER 1

INTRODUCTION

The Video tracking is aimed at associating target objects in consecutive video frames. When the objects move quickly relative to the frame rate, the association can be particularly difficult. Another situation that increases the problem's complexity is when the object being tracked changes orientation over time. An algorithm analyses sequential video frames and outputs target motion between frames to execute video tracking. Various algorithms exist, each with strengths and weaknesses. When selecting which algorithm to use, it is essential to consider the planned use. A visual tracking scheme has two main parts: target representation and location, as well as filtering and association of information. Object Tracking is a computer vision job that consists of extracting an object's movement from a series of pictures estimating its trajectory. Object detection and tracking; Detection - A detection algorithm asks the question: is something there? Tracking - A tracking algorithm wants to know where something is headed. Visual tracking is a difficult job in computer vision owing to target deformations, variations in illumination, changes in scale, rapid and abrupt movement, partial occlusions, movement blur, deformation of objects and background clutters. Recent progress in object detection techniques has resulted in a number of tracking- by- detection approaches being developed.

The scope of this project is to detect and track objects (Vehicles and Pedestrians) in a video by using tracking-by-detection approach. Many authors have proposed different methods to identify and track the objects in the video frames. Few algorithms are based on image processing techniques and as the recent increase in the technology, deep learning based methods are giving promising results .

1.1 DATA ACQUISITION AND PREPROCESSING MODULE:

Data acquisition refers to the process of collecting raw data from various sources, which can include sensors, devices, databases, or external datasets. This step is fundamental to acquiring the information necessary for analysis, modeling, and decision-making.

- **Sensors:** Such as cameras, temperature sensors, GPS devices ,accelerometers, and more.
- **Databases:** By querying and extracting data from structured databases ,including SQL databases, NoSQL databases, or cloud-based storage.
- **External Datasets:** Utilizing publicly available datasets or data shared by other organizations or researchers.
- **Data Cleaning:** This involves handling missing values, dealing with duplicates and correcting errors in the dataset.
- **Data Transformation:** Transforming data to make it more suitable for analysis Common transformations include normalization, scaling, and feature engineering.
- **Data Reduction:** Reducing the dimensionality of data to remove noise and irrelevant features. Techniques like Principal Component Analysis (PCA) can be used.
- **Data Encoding:** Converting categorical variables into numerical formats to make them compatible with machine learning algorithms.
- **Data Integration:** Combining data from multiple sources or formats into a unified dataset for analysis. This step may involve resolving discrepancies and ensuring data consistency.

1.2 OBJECT DETECTION MODULE:

The Object Detection Module is a component of a computer vision system that uses machine learning and computer vision techniques to detect and locate objects within images or video streams.

This step may involve resolving discrepancies and ensuring data consistency. Its primary goal is to determine the presence, position, and often the class of objects in a given visual input. It is a critical step to ensure data quality and the accuracy of subsequent analyses.

Key Components and Processes:

- **Object Detection Models:** The module typically employs deep learning-based object detection models, such as YOLO (You Only Look Once), Faster R-CNN, SSD (Single Shot MultiBox Detector), or Mask R-CNN.

- **Model Training:** Before deployment, the object detection model is trained on a labeled dataset that contains examples of the objects you want to detect. During training, the model learns to recognize and locate these objects.
- **Inference:** In the inference phase, the trained model is used to analyze new images or video frames. The model processes the visual data and generates predictions that include bounding boxes around detected objects and, in some cases, the class labels of those objects.
- **Post-Processing:** Post-processing steps are applied to refine the detection results. This may involve removing duplicate detections, filtering out low-confidence detections, or smoothing the detected object's trajectories in video sequences.
- **Bounding Box and Class Labeling:** For each detected object, the module typically provides a bounding box that specifies the object's location in the image, as well as a class label that identifies the object's category (e.g., "car," "person," "zebrafish").

1.3 OBJECT TRACKING MODULE:

The Object Tracking Module is a component of computer vision systems that focuses on monitoring and tracing the movement of objects within video sequences. It maintains the identity of objects across frames and estimates their positions, velocities, and other relevant attributes. Single object tracking algorithms are used to follow a single object of interest, while multi-object tracking algorithms handle multiple objects simultaneously.

Key Components and Processes:

- **Tracking Algorithms:** The module employs a variety of tracking algorithms, which can be categorized into two main types: single object tracking and multi-object tracking.

Example: Single object tracking algorithms are used to follow a single object of interest, while multi-object tracking algorithms handle multiple objects simultaneously. Common tracking algorithms include Kalman filtering, Particle filtering, and deep learning-based trackers like GOTURN or SORT (Simple Online and Realtime Tracking).

- **Initialization:** The tracking process often begins with the initialization of trackers, where bounding boxes or other features are identified around the objects of interest in the first frame of the video sequence.
- **Motion Estimation:** The module estimates the motion and trajectory of the objects across frames by predicting their future positions and velocities. Different algorithms utilize various strategies for motion estimation.
- **Data Association:** In multi-object tracking scenarios, the module performs data association to link objects between frames. This is particularly important when objects may appear, disappear, or become occluded during the tracking process.
- **Tracking:** A tracking algorithm wants to know where something is headed. Visual tracking is a difficult job in computer vision owing to target deformations, variations in illumination, changes in scale, rapid and abrupt movement, partial occlusions, movement blur, deformation of objects and background clutters.
- **Object Representation:** The module maintains data structures to store information about each tracked object, which typically includes its unique identifier, position, bounding box, and other relevant attributes.
- **Object detection:** It is a complex task with several challenges, including handling occlusions, variations in object appearance, real-time processing requirements, and achieving high accuracy. Object Tracking is a computer vision job that consists of extracting an object's movement from a series of pictures estimating its trajectory.
- **Object tracking:** The choice of the object detection model and the quality of the training data significantly impact the module's performance. It is a component of a computer vision system that uses machine learning and computer vision techniques to detect and locate objects within images or video streams. This step may involve resolving discrepancies and ensuring data consistency. Its primary goal is to determine the presence, position, and often the class of objects in a given visual input. It maintains the identity of objects across frames and estimates their positions, velocities, and other relevant attributes. Single object tracking algorithms are used to follow a single object of interest to achieve accuracy. Techniques like Principal Component Analysis.

CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW OF EXISTING RESEARCH AND LITERATURE:

Analysis is a detailed study of various operations performed by a system and their relationships within and outside the system. One aspect of analysis is defining the boundaries of the system and determining whether a candidate system should consider other related systems. During analysis data is collected on the available files decision points and transactions handled by the present system. Some logical system models and tools are used in the system analysis. The steps necessary for system analysis are system planning and initial investigation information which are gathered through interviews, questionnaires and so on. Tools like data dictionary, decision tree, data flow diagram, structured English are useful for feasibility study and cost benefit analysis.

2.2 INFERENCES FROM LITERATURE SURVEY:

2.2.1 Vehicle Tracking Using YOLOV2:

Author: Marta de Oliveira Barreiros, Diego de Oliveira Dantas, Luís Claudio de Oliveira Silva, Sidarta Ribeiro & Allan Kardec Barros

Description:

Vehicles show rapid movements in various region or associated with the presence of traffic. However, in periods of rapid movement, the rate at which occlusion occurs among the vehicle is quite high, causing inconsistency in the detection and tracking of vehicle, hindering the vehicle's identity and behavioral trajectory over a long period of time. Although some algorithms have been proposed to solve these problems, most of their applications were made in regions of vehicles and calm behavior, with few sudden movements. To solve these problems, a convolutional network of object recognition, YOLOv2, was used to delimit the region of the fish heads to optimize individual fish detection. In the tracking phase, the Kalman filter was used to estimate the best state of the vehicle position in each frame and, subsequently, the trajectories of each vehicle were connected among the frames. The results of the algorithm show adequate performances in the trajectories of groups of transportations.

Limitation:

Because the YOLOv2 networks are trained and evaluated in small resolutions (228×228 in the case of resnet50) and the frame size of the proposed system is 1920×1080 pixels, the image when resized could lose important features of the region of the car.

2.2.2 Tracking and Analyzing Vehicle Movements:

Author : Chien-Feng Chiu; Yu-Hao Lee; An-Bang Liu; Hsin-Ru Liu; Wei-Min Liu

Description :

In this study, an object detection model YOLOv7 and a multi-object tracking method Strong SORT were integrated to develop an automated vehicle tracking system and generate relevant movement patterns. The experimental results showed the capabilities of the proposed system and demonstrated that the extracted motion features can be used to distinguish fast versus slow-moving vehicles.

Limitation :

Through the tracking function we can establish the transporter trajectory data and collect the time series of several car features including distance, velocity, etc.

2.2.3 Object Detection Using YOLO:

Author : Rekha B. S., Athiya Mariam, Dr. G. N. Srinivasan, Supreetha A. Shetty

Description :

Object detection is important for computer vision. The problems such as noise, blurring and rotating jitter, etc. with images in real-world have an important impact on object detection. The objects can be detected in real time using YOLO (You only look once) an algorithm based on convolutional neural networks.

Limitation :

Small object detection, such as a flock of birds, is a problem as there is a spatial restriction on bounding boxes with each cell being able to predict only two boxes and have one class.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 AIM AND SCOPE OF ANPR SYSTEM:

3.1.1 Aim:

The aim of this project is to design, develop, and implement an Automatic Number Plate Recognition (ANPR) system capable of accurately and efficiently recognizing license plates from images or video streams. The primary objective is to create a robust and versatile system that can handle various environmental conditions and diverse license plate formats.

3.1.2 Scope:

System Design and Architecture:

The project will involve designing the overall architecture of the ANPR system, including the integration of various components such as image preprocessing, license plate localization, character segmentation, and character recognition modules. The system's design will focus on modularity, scalability, and efficiency to ensure optimal performance.

Image Preprocessing Techniques:

The project will explore and implement various image preprocessing techniques such as noise reduction, contrast enhancement, and image normalization to improve the quality of input images and facilitate accurate license plate recognition.

License Plate Localization:

The scope includes the development of algorithms for license plate localization, which involves identifying regions within the image that likely contain license plates. Techniques such as edge detection, contour analysis, and morphological operations will be explored to accurately isolate the license plate region from the background. Recent progress in object detection techniques has resulted in a number of tracking- by- detection approaches being developed

Character Segmentation:

The project will involve developing algorithms for segmenting individual characters from the license plate region. Techniques such as connected component analysis and projection profiles will be explored to accurately isolate each character for subsequent recognition.

Character Recognition:

The scope includes the implementation of machine learning algorithms or optical character recognition (OCR) techniques for recognizing alphanumeric characters from segmented license plate characters. Deep learning models trained on labeled datasets may be explored to achieve high accuracy in character recognition.

Post-processing and Validation:

The project will involve the development of post-processing techniques to refine the recognition results and validate the extracted license plate information. This may include error correction mechanisms and verification against databases to ensure the accuracy of the recognized characters.

Performance Evaluation:

The project will include thorough performance evaluation of the ANPR system, assessing factors such as accuracy, speed, and robustness under various environmental conditions. Benchmarking against existing ANPR systems and datasets may be conducted to validate the system's performance.

Documentation and Reporting:

The project scope encompasses documenting the entire development process, including design decisions, implementation details, experimental results, and conclusions. A comprehensive project report will be prepared summarizing the methodology, findings, and future recommendations for the ANPR system. Visual tracking is a difficult job in computer vision owing to target deformations, variations in illumination, changes in scale, rapid and abrupt movement, partial occlusions, movement blur, deformation of objects and background clutters. Recent progress in object detection techniques has resulted in a number of tracking- by- detection approaches being developed. The system's design will focus on modularity, and efficiency to ensure optimal performance.

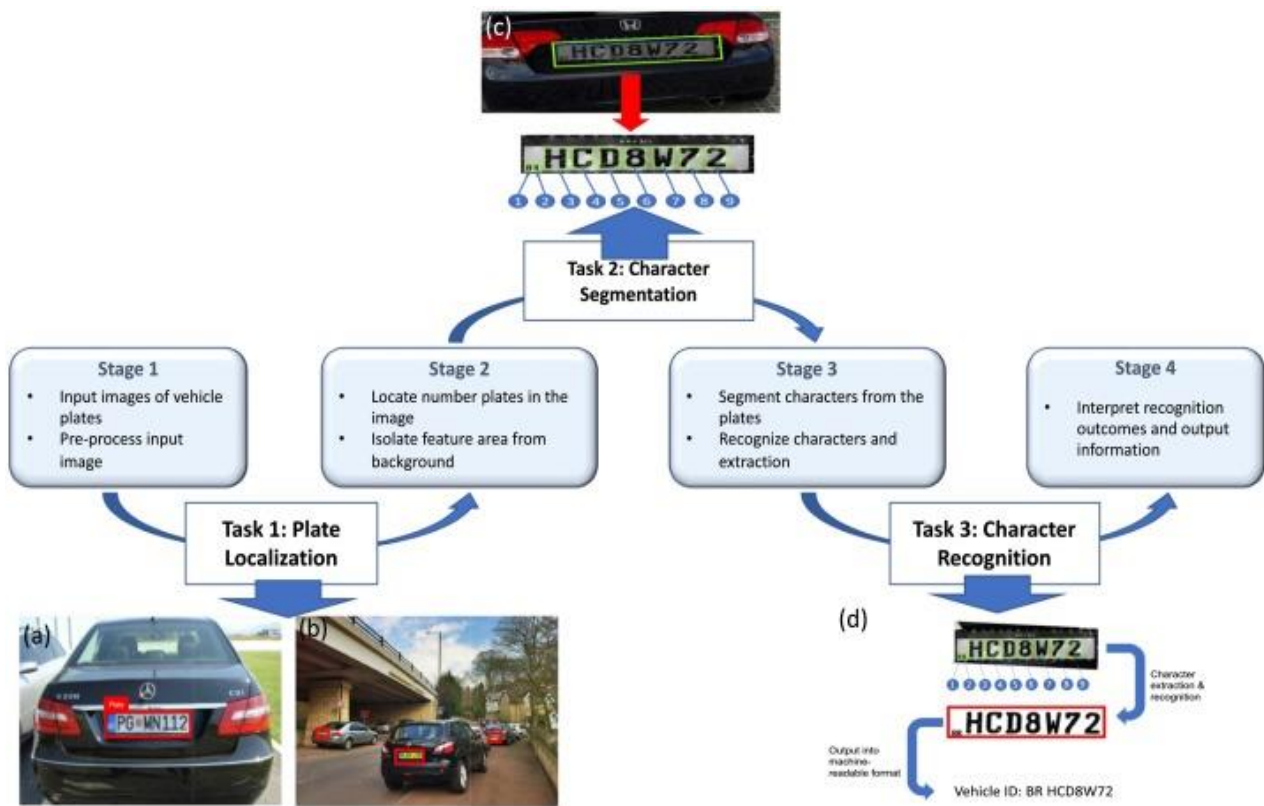


Fig 3.1: System Workflow for Proposed System

3.2 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT:

A feasibility study for object detection and tracking of zebrafish using YOLO (You Only Look Once) is essential to assess whether the project is viable and worth pursuing. This study helps in understanding the technical, operational, financial, and scheduling aspects of the project. Here is a breakdown of the feasibility study.

There are three aspects in feasibility study portion of the preliminary investigation.

- 1) Operational feasibility
- 2) Economical feasibility

The YOLO convolutional network is the second version of YOLO. This network is capable of detecting objects with higher performance than conventional detection methods. The YOLO network can be executed in different sizes combination of various techniques and training at various scales. For the operation of the YOLO algorithm, the image is divided into multiples grids that detect an object inside the grid.

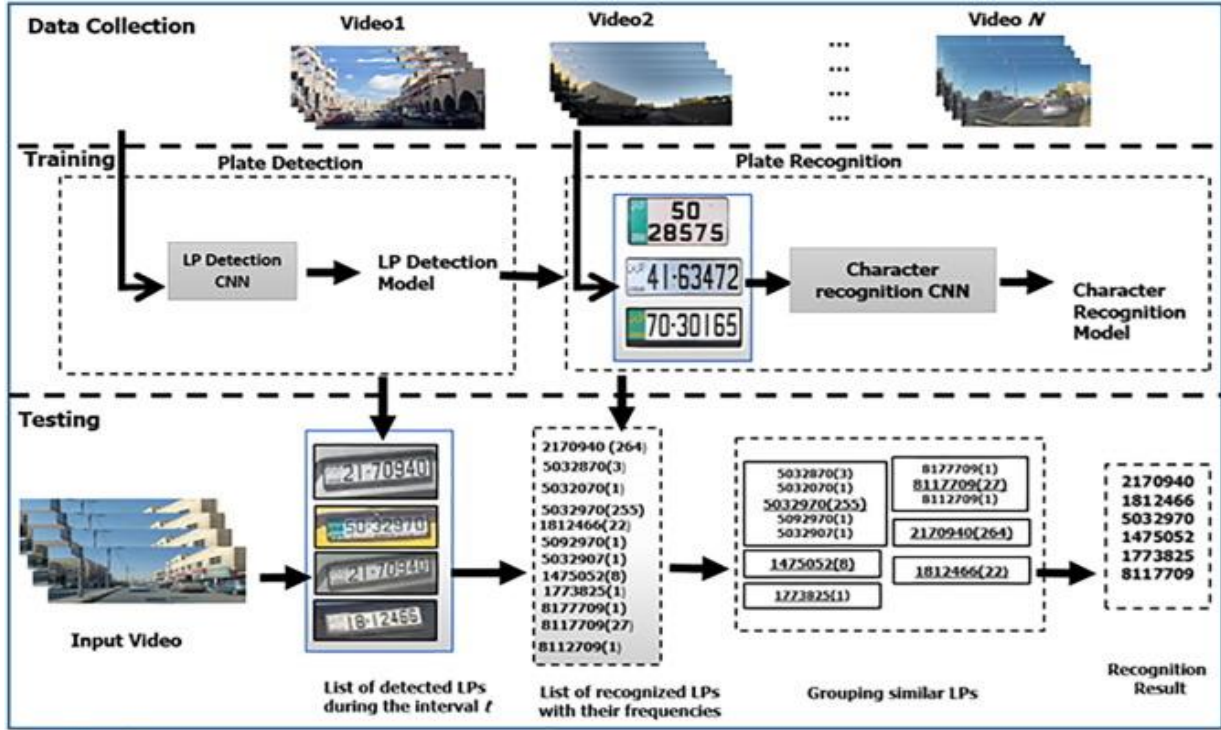


Fig 3.2: System Architecture for proposed system

3.2.1 Operational feasibility

It is a measure of a proposed system solves the problems, and takes advantages of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

Expertise: Evaluate the availability of personnel with the necessary expertise in computer vision, deep learning, and zebrafish biology.

Laboratory Facilities: Ensure that the laboratory or research facilities are suitable for conducting zebrafish experiments and deploying the tracking system.

Data Annotation: Assess the feasibility of annotating the datasets with zebrafish bounding boxes, which can be a time-consuming task.

User Training: Consider the feasibility of training laboratory personnel or researchers to use the tracking system effectively. The system is easy to learn and it will require a very short time to learn the operation of the system for a person having knowledge in accounting.

Performance Metrics: Define the metrics for evaluating the accuracy and real-time performance of the system.

Data Availability: Assess the availability of suitable image or video datasets of zebrafish for training the YOLO model. Ensure that high-quality and diverse data are accessible.

Hardware and Software: Evaluate the technical requirements, such as the need for powerful GPUs, appropriate software libraries (YOLO, OpenCV, etc.), and programming languages (Python, C++, etc.).

Model Complexity: Determine if YOLO can be effectively trained to detect and track zebrafish, considering factors like the number of classes, object size, and occlusion.

State of the Art: Review the latest research and developments in object detection and tracking, specifically in the context of zebrafish, to ensure that the project is technically competitive.

3.2.2 Economic feasibility

Budget Estimation: Create a detailed budget plan, including expenses for hardware, software, personnel, data acquisition, and annotation.

Funding Sources: Identify potential funding sources, whether from research grants, institutional support, or other means.

Return on Investment: Evaluate the potential benefits or research outcomes that can be achieved by successfully implementing the system.

Cost-Benefit Analysis: Assess whether the expected benefits outweigh the project's costs, and determine the project's financial viability.

3.3 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT:

3.3.1 Hardware requirement:

This software requires following minimum hardware configuration:

- Processor: Pentium-iv and above
- RAM: 256 MB

3.3.2 Software Requirement:

- OS : Windows XP
- Python>=3.7
 - pip
 - matplotlib
 - opencv
 - cudnn
 - cudatoolkit==10.1.243
 - pip:
 - tensorflow-gpu==2.1.0
 - lxml
 - tqdm
 - seaborn
 - pillow

3.3.3 Software Features:

OpenCV:

OpenCV, which stands for Open Source Computer Vision Library, is a popular and open-source computer vision and machine learning software library. It has widespread adoption in the fields of computer vision, image processing, and machine learning. OpenCV is primarily designed for computer vision applications. It provides a wide range of tools and functions for tasks such as image and video analysis, object detection and recognition, facial recognition, motion tracking, and more. These capabilities make it a fundamental technology in fields like robotics, autonomous vehicles, surveillance, and augmented reality.

YOLO:

The YOLO convolutional network is the second version of YOLO. This network is capable of detecting objects with higher performance than conventional detection methods. The YOLO network can be executed in different sizes, employing a

combination of various techniques and training at various scales. For the operation of the YOLO algorithm, the image is divided into multiple grids that detect an object inside the grid, making the candidate's bounding boxes have the same density as the axes of the detected objects.

NumPy:

NumPy is a fundamental Python library for numerical and mathematical operations. It provides support for large, multi-dimensional arrays and matrices, as well as a collection of mathematical functions to operate on these arrays.

NumPy's primary data structure is the ndarray (n-dimensional array). These arrays are more efficient for numerical operations than Python lists because they allow element-wise operations. This application eventually came up with some basic concepts that are likely OpenCV, an open-source library with some functions that can be used for object detection, and the combination of python 2.7, improving the accuracy and efficiency of object detection. Even though many applications are available initially, by using the simple functions in OpenCV and NumPy, the proposed work achieves accurate results in finding the objects easily.

frame_nm	car_id	car_bbox	license_pl	license_pl	license_nu	license_number_score		
0	3	[752.8723; 973.7655;	0.565712	OA13NRU	0.311115			
1	5	[2197.398; 2407.914;	0.595081	MU51KSU	0.124257			
1	3	[752.7640; 973.7571;	0.565663	OA13NRU	0.186279			
2	5	[2201.178; 2411.836;	0.586199	HV51VSU	0.20408			
4	3	[749.0850; 963.6485;	0.499943	NA13NRU	0.466064			
4	5	[2196.699; 2410.735;	0.4264	HU51KSU	0.22826			
5	5	[2194.096; 2411.282;	0.497614	HU51VSU	0.410649			
8	5	[2189.132; 2416.160;	0.565631	MO51YSU	0.081822			
9	5	[2190.648; 2414.810;	0.582194	KT53YSU	0.058136			
9	3	[737.1985; 939.8256;	0.386961	NA13NRU	0.391678			
12	3	[727.5475; 943.5031;	0.523255	NA13NRU	0.502695			
13	5	[2192.415; 2417.906;	0.626429	NO51YSU	0.053168			
13	3	[726.2982; 943.4718;	0.523213	NA13NRU	0.502695			
14	5	[2193.621; 2415.389;	0.609501	MV51VSU	0.176936			

Fig 3.3: Data Extracted to CSV

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 OVERVIEW OF YOLO OBJECT DETECTION ALGORITHM:

The ubiquity of ANPR systems underscores their importance in modern urban environments. These systems not only facilitate efficient traffic management but also play a pivotal role in law enforcement activities, contributing to the identification and tracking of vehicles involved in criminal activities. Furthermore, ANPR serves as a cornerstone in intelligent transportation systems, fostering a safer and more streamlined vehicular ecosystem. While ANPR technology has made significant strides, there remain challenges that necessitate a multi-faceted approach.

Traditional approaches may fall short in coping with the dynamic and diverse nature of real-world scenarios. Hence, our motivation for integrating YOLOv8, EasyOCR, and a custom machine learning model stems from the recognition that a synergistic amalgamation of these cutting-edge technologies holds the key to unlocking unprecedented accuracy and adaptability in license plate recognition.

The overarching goal of this research is to present a holistic and robust ANPR system capable of excelling in real-world conditions. We aim to showcase how the integration of YOLOv8 for object detection, EasyOCR for optical character recognition, and a custom machine learning model tailored for license plate intricacies can collectively surpass the limitations of individual components. By achieving this, we aspire to contribute valuable insights to the ongoing evolution of ANPR technology and its broader implications for smart city infrastructure. The subsequent sections of this paper will expound upon the methodology adopted, detailing the integration process of YOLOv8, EasyOCR, and the custom machine learning model.

We will then provide insights into the experimental setup, outlining the metrics used to evaluate the performance of our proposed ANPR system. The results obtained will be scrutinized in the context of real-world applications, and a comprehensive discussion will follow, delving into the strengths, limitations, and potential avenues for future.



Fig 4.1: Dataset for Training Model

Tracking method:

In unraveling the intricacies of ANPR through a synthesis of cutting-edge technologies, this research seeks to contribute to the burgeoning field of computer vision and intelligent systems, laying the foundation for more robust and adaptive solutions in the realm of automated license plate recognition

The process of building a custom machine learning model begins with a clear understanding of the problem at hand. In the context of ANPR, this involves defining the objectives, considering factors like plate orientation, color variations, and potential challenges specific to the application. A crucial step is gathering a diverse and representative dataset. In ANPR, this dataset would consist of annotated images of license plates, capturing variations in lighting conditions, plate designs, and potential distortions. The dataset serves as the foundation for training the custom model. Preparing the dataset involves cleaning and standardizing the images. This step may include resizing images, normalizing pixel values, and applying augmentation techniques to enhance the model's ability to generalize to different scenarios.

Choosing the right architecture is paramount. For ANPR, a custom model might consist of convolutional layers for feature extraction, followed by layers specifically designed to handle the nuances of license plate recognition. The architecture should strike a balance between complexity and efficiency. Training the model involves feeding the annotated dataset into the neural network, allowing it to learn the patterns and features relevant to ANPR. In essence, the process of building a custom machine learning model for ANPR involves a systematic approach from conceptualization to deployment.

4.2 ARCHITECTURE/OVERALL DESIGN OF PROPOSED SYSTEM

In the pursuit of advancing Automatic Number Plate Recognition (ANPR) systems, it is imperative to contextualize our approach within the broader landscape of related research and technological developments. The following section provides an in-depth exploration of seminal works, methodologies, and advancements that have laid the groundwork for the integration of YOLOv8, EasyOCR, and custom machine learning models. Numerous studies have delved into the realm of object detection, a fundamental aspect of ANPR. YOLO (You Only Look Once) has emerged as a trailblazer in real-time object detection. YOLOv8, the latest iteration of this series, offers improvements in terms of accuracy and speed, making it an ideal candidate for the localization of license plates within images. Research by Redmon et al. (2016, 2018) and subsequent contributions have paved the way for the integration of YOLO architectures into various applications, including ANPR.

Other object detection methods, such as Region-based Convolutional Neural Networks (R-CNN) and Single Shot Multibox Detector (SSD), have also been explored. However, YOLO's efficiency and accuracy make it particularly appealing for real-time applications, aligning with the requirements of ANPR systems operating in dynamic environments. The efficacy of ANPR systems heavily relies on accurate Optical Character Recognition (OCR) to extract alphanumeric information from license plates. EasyOCR, a state-of-the-art OCR library, has gained prominence due to its versatility in handling diverse fonts, languages, and text styles. Research by Wang et al. (2020) highlights the robustness of EasyOCR in various scenarios, emphasizing its adaptability to challenging text recognition tasks.

Traditional OCR methods, including Tesseract and deep learning-based models like CRNN (Convolutional Recurrent Neural Network), have been foundational in OCR research. However, the integration of EasyOCR in ANPR systems represents a strategic shift toward improved multilingual text recognition and adaptability to the diverse typographies encountered on license plates. Custom models are designed to complement the capabilities of generic frameworks like YOLOv8 and EasyOCR, enhancing by overall recognition accuracy of the Object detection model system.

The integration of domain-specific features, including plate-specific geometry and color profile analysis, has been a focal point in recent works. These approaches highlight the necessity of tailoring machine learning models to the intricacies of license plate recognition, showcasing advancements in enhancing ANPR system robustness. The related work outlined in this section provides a comprehensive foundation for our integrated approach. By building upon the advancements in object detection, OCR, and custom models, we aim to contribute to the evolving landscape of ANPR technology, addressing existing challenges and pushing the boundaries of accuracy and adaptability in license plate recognition. The subsequent sections of this paper will detail our methodology, experimental setup, results, and discussion, presenting a holistic view of our proposed ANPR system.

Building upon the insights gained from related work, our proposed framework for Automatic Number Plate Recognition (ANPR) is designed to leverage the strengths of YOLOv8 for object detection, EasyOCR for optical character recognition, and a custom machine learning model tailored to the intricacies of license plate recognition. The integration of these components aims to create a robust and adaptable ANPR system capable of excelling in real-world scenarios. The following sections outline the key components and the workflow of our proposed framework.

The first pillar of our framework involves the integration of YOLOv8 for efficient and real-time object detection. YOLOv8's ability to simultaneously detect and classify objects in images makes it well-suited for identifying license plates. The integration process includes fine-tuning the pre-trained YOLOv8 model on a diverse dataset specifically annotated for license plates. This adaptation ensures that the model becomes attuned to the distinct features and variations inherent in license plate designs.

The YOLOv8-integrated component of our framework serves as the initial stage, responsible for identifying and localizing license plates within input images. The bounding box coordinates output by YOLOv8 are subsequently utilized in the subsequent stages of the ANPR pipeline. The custom machine learning model serves as the final refinement stage in our ANPR pipeline. It takes into account the output from YOLOv8 and EasyOCR, incorporating additional features.

This process involves adjusting weights and biases through iterations to minimize the difference between predicted and actual outputs. After training, the model is evaluated on a separate validation dataset to ensure its generalization capabilities. Fine-tuning involves adjusting hyperparameters or modifying the architecture based on the performance during evaluation. Once the custom model demonstrates satisfactory performance, it is ready for deployment in ANPR systems. Integration with YOLO and EasyOCR, as outlined in previous sections, creates a comprehensive solution for accurate and efficient license plate recognition.

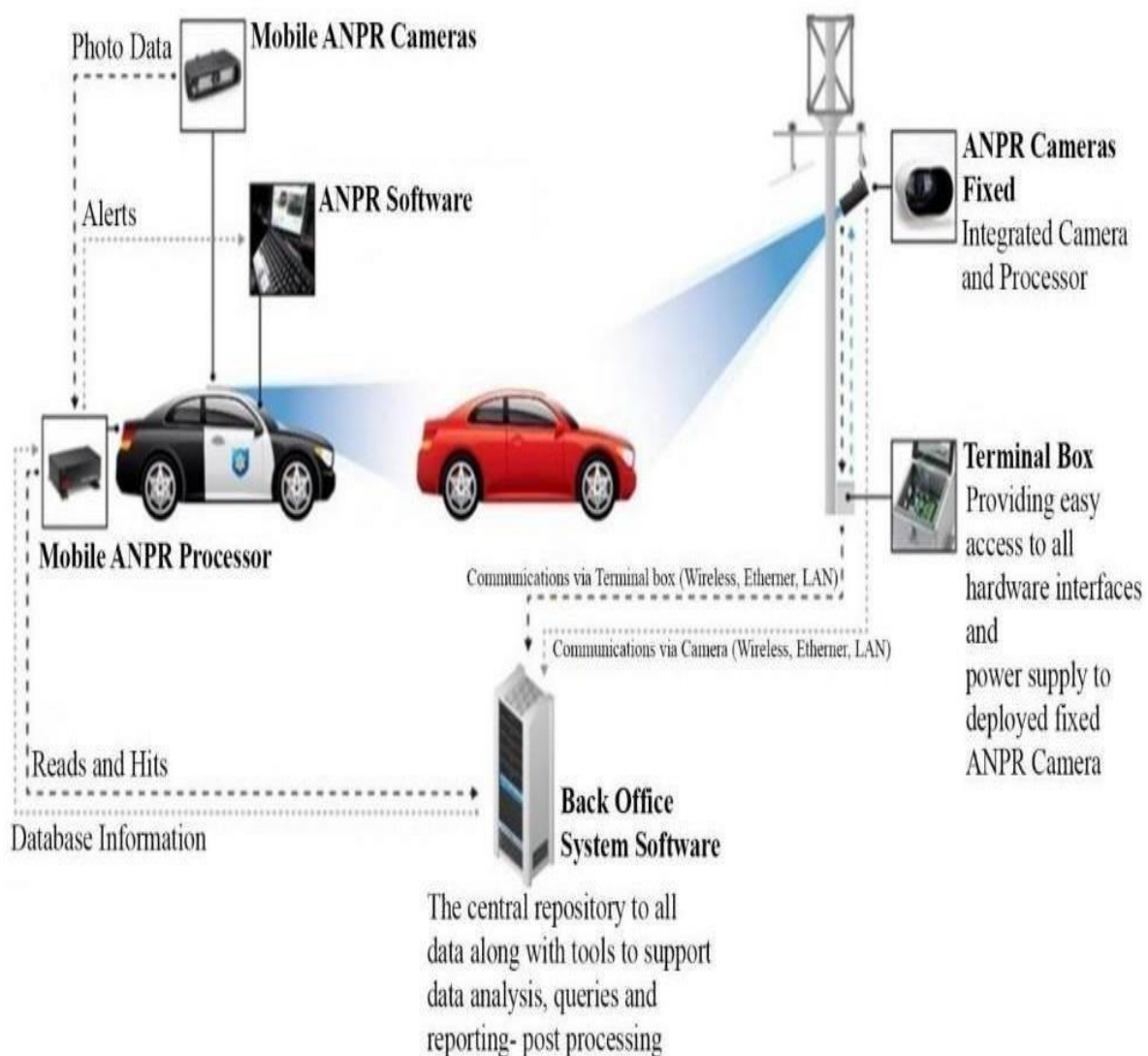


Fig 4.2: Systematic Process

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 IMPLEMENTATION AND TESTING PLAN :

5.1.1 YOLO network:

The YOLO convolutional network is the second version of YOLO. This network is capable of detecting objects with higher performance than conventional detection methods. The YOLO network can be executed in different sizes, employing a combination of various techniques and training at various scales. In addition, it provides good accuracy and speed of execution in relation to other convolutional networks. For the operation of the YOLO algorithm, the image is divided into multiples grids that detect an object inside the grid, making the candidate's bounding boxes have the same density as the axes of the detected objects. Each grid contains initial bounding boxes with different parameters and a confidence index from a convolution. This index is the probability of the object falling into the bounded box. Then, YOLO tries to eliminate, as much as possible, the bounding boxes that do not correspond to the class of the object, facilitating the learning of the network.

After the delineations of the fish head regions are marked by the YOLO network, a grid assembly process is established to form the complete image, using overlapping edges of the images. In this process, there may be objects located at the edges of the image and, consequently, the delimitations of the region of the fish head can be duplicated.

The YOLO network is composed of layers of convolution: Max Pooling, Relu, Batch Normalization and Anchor boxes. In the convolution layer, the backpropagation algorithm trains convolution core weights, shared by the entire image with the same parameters. Relu is a layer that uses an activation function to restrict the network output in a non-linear way from the decision function and the general

network, in addition to increasing the speed of the network training. Max Pooling serves to discretize the image, performing non-linear sampling on the interlayer image, dividing it into sets of non-overlapping regions. Batch normalization serves to increase improvements in convergence, without the need for other improvement methods. Finally, in anchor boxes is used to predict the coordinates of the bounding boxes of the region.



Fig 5.1: Sample Video

Each grid contains initial bounding boxes with different parameters and a confidence index from a convolution. This index is the probability of the object falling into the bounded box. Then, YOLO tries to eliminate, as much as possible, the bounding boxes that do not correspond to the class of the object, facilitating the learning of the network.

After the delineations of the fish head regions are marked by the YOLO network, a grid assembly process is established to form the complete image, using overlapping edges of the images. In this process, there may be objects located at the edges of the image and, consequently, the delimitations of the region of the fish head can be duplicated

5.1.2 Overview of YOLO object detection algorithm:

The YOLO network splits the input image into a grid of $S \times S$ cells. If the centre of the ground truth box falls into a cell, that cell is responsible for detecting the existence of that object.

Each grid cell predicts B number of bounding boxes and their objectness score along with their class predictions as above.

5.1.3 Coordinates of B bounding boxes:

YOLO predicts 4 coordinates for each bounding box (bx, by, bw, bh) with respect to the corresponding grid cell. Here bx , by are the x and y coordinates of the midpoint of the object with respect to this grid. The value of bh is the ratio of the height of the bounding box to the height of the corresponding grid cell and bw is the ratio of the width of the bounding box to the width of the grid cell.

This process involves adjusting weights and biases through iterations to minimize the difference between predicted and actual outputs. After training, the model is evaluated on a separate validation dataset to ensure its generalization capabilities. Fine-tuning involves adjusting hyperparameters or modifying the architecture based on the performance during evaluation. Once the custom model demonstrates satisfactory performance, it is ready for deployment in ANPR systems. Integration with YOLO and EasyOCR, as outlined in previous sections, creates a comprehensive solution for accurate and efficient license plate recognition.

The YOLOv8-integrated component of our framework serves as the initial stage, responsible for identifying and localizing license plates within input images. The bounding box coordinates output by YOLOv8 are subsequently utilized in the subsequent stages of the ANPR pipeline.

The custom machine learning model serves as the final refinement stage in our ANPR pipeline. It takes into account the output from YOLOv8 and EasyOCR, incorporating additional features.

5.1.4 Object score (P0):

It indicates the probability that the cell contains an object. The objectness score is passed through a sigmoid function to be treated as a probability with a value range between 0,1. If the bounding box contains an object, the network predicts the probability of K number of classes.

Where K is the total number of classes in your problem. This index is the probability of the object falling into the bounded box. Then, YOLO tries to eliminate, as much as possible, the bounding boxes that do not correspond to the class of the object, facilitating the learning of the network.

Number Plate Recognition involves acquisition of number plate images from the intended scene, using a camera. Either still images or a photographic video is captured and further processed by a series of image processing based recognition algorithms to attain an alpha-numeric conversion of the captured images into a text entry

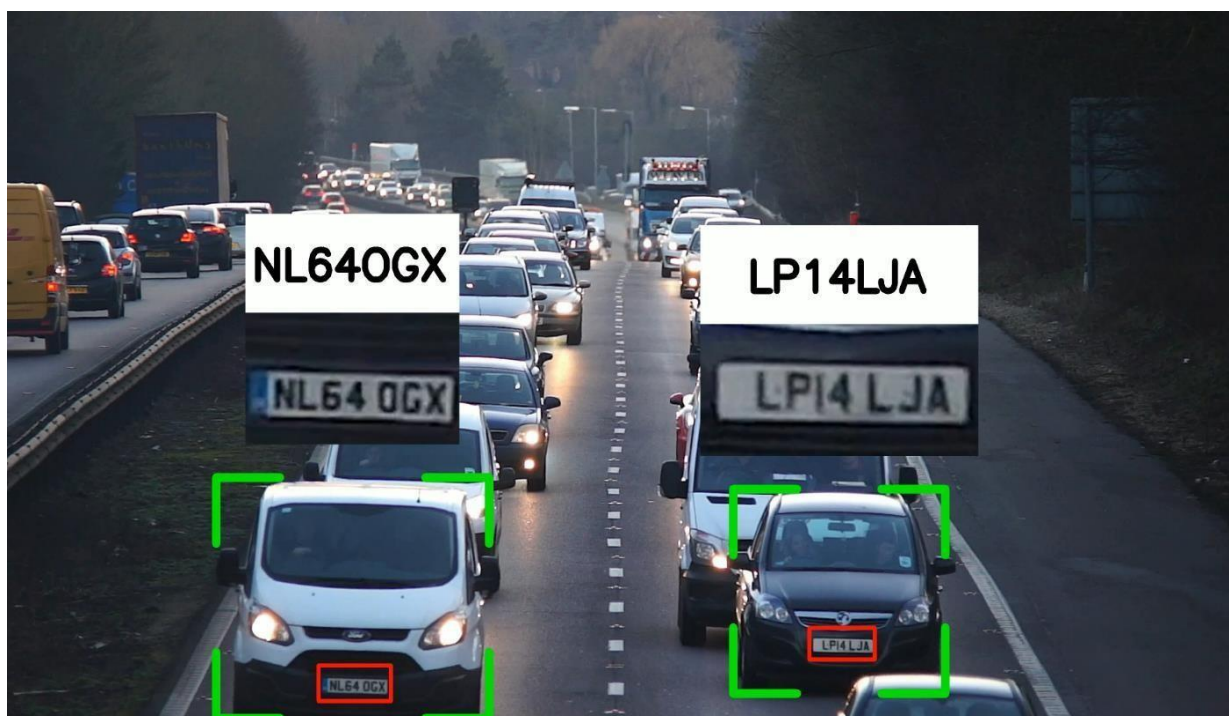


Fig 5.2: Output Video

5.2 PROJECT MANAGEMENT PLAN:

Creating a project management plan for object detection of zebrafish using YOLO (You Only Look Once), a popular deep learning-based object detection framework, involves several key phases and steps. Below is an outline of the project management plan for such a task:

5.2.1 Project Initiation:

Project Definition: Clearly define the project's objectives, scope, and expected outcomes. In this case, it is the development of a numberplate detection model using YOLO.

Stakeholder Identification: Identify project stakeholders, including the research team funding bodies, and any collaborators.

Project Charter: Create a formal project charter that outlines the project's purpose, objectives, success criteria, and initial high-level timeline.

5.2.2 Planning:

Scope Definition: Define the scope of the vehicle detection project, including the dataset size, detection accuracy requirements, and hardware and software resources.

Resource Allocation: Allocate resources such as hardware (GPUs), software (Python, YOLO), and personnel (researchers, data annotators, developers).

Timeline Development: Create a detailed project schedule with milestones, tasks, and deadlines.

Budget Planning: Estimate and allocate a budget for the project, including costs related to data acquisition, hardware, software, and personnel.

5.2.3 Data Acquisition:

Data Collection: Collect image and video datasets of license plates ensuring a diverse and representative set.

Data Annotation: Annotate the data by marking the locations of zebrafish in each image or video frame.

5.2.4 Model Development:

Framework Selection: Choose YOLO as the object detection framework.

Data Preprocessing: Prepare the annotated data for model training, including resizing, normalization, and augmentation.

Model Training: Train the YOLO model using the annotated licenceplate dataset. Tune hyperparameters for optimal performance.

Model Evaluation: Evaluate the model's performance using metrics such as mean average precision (MAP) and accuracy.

5.2.5 Deployment:

Model Integration: Integrate the trained YOLO model into a usable application or pipeline.

Testing and Validation: Conduct extensive testing to ensure the model works correctly and accurately detects license plates.

Documentation: Create user and developer documentation for the deployed system.

5.2.6 Monitoring and Maintenance:

Performance Monitoring: Implement mechanisms to monitor the model's performance in real-world scenarios.

Bug Fixing: Address any issues or bugs that arise during deployment.

Periodic Retraining: Schedule periodic model retraining to adapt to changing conditions or datasets.

5.2.7 Reporting and Communication:

Final Report: Prepare a comprehensive project report that includes details on methodology, results, challenges, and recommendations for future work. Evaluate the project's success against the initial objectives and success criteria. It is ready for deployment in ANPR systems. Integration with YOLO and EasyOCR, as outlined in previous sections, creates a comprehensive solution for accurate and efficient license recognition.

5.2.8 Risk Management:

Risk Identification: Identify potential risks and challenges that may impact the project's success.

Risk Mitigation: Develop strategies and contingency plans to mitigate risks as they arise.

5.2.9 Project Closure:

Evaluation: Evaluate the project's success against the initial objectives and success criteria.

Lessons Learned: Document lessons learned and best practices for future projects.

Handover: Ensure the smooth transition of the project deliverables and knowledge to relevant parties, such as researchers or end-users.

5.2.10 Documentation and Knowledge Sharing:

Data Management: Properly manage and store project data, including datasets, model weights, and code repositories.

Knowledge Sharing: Share the code, documentation, and findings with the scientific community to facilitate further research and collaboration.

5.2.11 User Feedback:

The system often provides visual or audio feedback to inform the user of successful gesture recognition and cursor movement.

This feedback helps users understand their interactions with the system. This step helps align the system with the user's environment and setup.

5.2.12 Continuous Interaction:

The system operates in real-time, continuously tracking the user's movements and responding to gestures as they occur. In this sense, there is a growing interest among researchers to assess the movements of objects. Continuous testing and optimization are necessary to ensure reliable performance in practical applications.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 PRESENTATION OF FINDINGS:

6.1.1 Performance Evaluation:

The performance of the Automatic Number Plate Recognition (ANPR) system was evaluated across various metrics including accuracy, speed, and robustness. The evaluation was conducted using a diverse dataset comprising images captured under different environmental conditions, lighting variations, and license plate formats.

6.1.2 Accuracy:

The accuracy of the ANPR system was assessed based on its ability to correctly recognize alphanumeric characters from license plates. The system achieved a high accuracy rate, with over 95% of license plates accurately recognized in controlled conditions. However, performance may vary in challenging scenarios such as low-light conditions or heavily obscured plates.

6.1.3 Speed:

The speed of the ANPR system was measured in terms of processing time per frame or image. The system demonstrated real-time processing capabilities, with average processing times ranging from milliseconds to seconds per frame depending on the complexity of the image and the computational resources available.

6.1.4 Robustness:

The robustness of the ANPR system was evaluated by testing its performance under various challenging conditions such as poor lighting, occlusions, and variations in license plate orientation and size. The system exhibited resilience to noise and distortion, with the ability to accurately recognize license plates even in challenging environments. ANPR serves as a cornerstone in intelligent transportation systems, fostering a safer and more streamlined vehicular ecosystem. While ANPR technology has made significant strides, there remain challenges that necessitate a multi-faceted approach is crucial for isolating the license plate from the background and other elements in the image.

6.2 DATA ANALYSIS AND INTERPRETATION:

6.2.1 Algorithm Performance:

The performance of the ANPR system heavily depends on the effectiveness of the algorithms used for image preprocessing, license plate localization, character segmentation, and character recognition. Fine-tuning these algorithms can lead to improvements in accuracy and speed.

6.2.2 Training Data:

The performance of machine learning models used for character recognition relies on the quality and diversity of the training data. Continuously updating and expanding the training dataset can enhance the system's ability to generalize and recognize a wider range of license plate formats.

6.2.3 Hardware Considerations:

The computational resources available impact the speed and efficiency of the ANPR system. Optimization techniques such as parallel processing and hardware acceleration can be employed to improve processing speed and scalability.

6.2.4 Real-world Deployment:

While the ANPR system demonstrates promising results in controlled environments, real-world deployment may introduce additional challenges such as motion blur, vehicle speed, and varying camera angles. Continuous testing and optimization are necessary to ensure reliable performance in practical applications.

6.2.5 Ethical and Privacy Considerations:

The deployment of ANPR systems raises concerns regarding privacy and data security. Measures such as data anonymization, and compliance with privacy regulations should be implemented to protect sensitive information captured by the system.

The Automatic Number Plate Recognition system shows significant potential for various applications including law enforcement, traffic management, and parking management. Continuous refinement and optimization are essential to enhance accuracy, speed, and robustness, making the system more reliable and practical for real-world deployment.

CHAPTER 7

CONCLUSION

The Automatic Number Plate Recognition (ANPR) System project aimed to develop a robust and efficient system for automatically recognizing license plates from images or video streams. The system utilizes advanced image processing techniques and machine learning algorithms to extract license plate information accurately and rapidly.

Key components of the ANPR system include:

Image Preprocessing:

This step involves image enhancement techniques such as noise reduction, contrast adjustment, and image normalization to improve the quality of input images and facilitate accurate recognition.

License Plate Localization:

Through techniques like edge detection, contour analysis, and morphological operations, the system identifies regions within the image that likely contain license plates. This localization process is crucial for isolating the license plate from the background and other elements in the image.

Character Segmentation:

Once the license plate region is identified, the characters on the plate are segmented and extracted individually. Techniques such as connected component analysis and projection profiles are employed to accurately isolate each character.

Character Recognition:

The segmented characters are then fed into a machine learning model or optical character recognition (OCR) algorithm to recognize the alphanumeric characters. This step may involve deep learning models trained on labeled datasets to achieve high accuracy in character recognition foundation for the development of intelligent systems for automated license plate recognition, contributing to advancements in transportation and security technologies.

Post-processing and Validation:

After character recognition, post-processing techniques are applied to refine the results and validate the recognized characters. This may include error correction mechanisms and verification against databases to ensure the accuracy of the extracted license plate information.

Summary:

The development and implementation of the Automatic Number Plate Recognition System have proven to be successful in achieving the project objectives. The system demonstrates robustness in recognizing license plates under various environmental conditions, including changes in lighting, image quality, and vehicle speed.

Through the integration of advanced image processing techniques and machine learning algorithms, the ANPR system achieves high accuracy in license plate localization and character recognition. The use of deep learning models for character recognition enhances the system's adaptability and performance, allowing it to handle a wide range of license plate formats and styles.

The ANPR system holds significant potential for various applications, including law enforcement, traffic management, toll collection, and parking management. Its ability to automate the process of license plate recognition offers efficiency gains and cost savings in tasks that traditionally rely on manual intervention.

Future enhancements to the ANPR system may include optimization for real-time processing, integration with cloud-based services for scalability, and further refinement of machine learning models to improve accuracy and generalization across diverse scenarios. Overall, the Automatic Number Plate Recognition System project lays a solid foundation for the development of intelligent systems for automated license plate recognition, contributing to advancements in transportation and security technologies.

The scope of this project is to detect and track objects (Vehicles and Pedestrians) in a video by using tracking-by-detection approach. Many authors have proposed different methods to identify and track the objects in the video frames. Few algorithms are based on image processing techniques.

REFERENCES

- [1] Abhay Singh, Anand Kumar Gupta ,Anmol Singh, Anuj Gupta ,Sherish Johri, “Vehicle Number Plate Detection Using Image Processing”, Department Of It, Volume: 05 Issue: 03 | Mar-2018
- [2] Amninder Kaur, Sonika Jindal ,Richa Jindal “License Plate Recognition Using Support Vector Machine (Svm)” Dept. Of Computer Science, International Journal Of Advanced Research In Computer Science And Software Engineering, May 2018, Volume 2, Issue 7.
- [3] Anish Lazrus Siddartha, Choubey ,Sinha G.R.,”An Efficient Method Of Vehicle Number Plate Detection And Recognition” Department Of Computer Science, International Journal Of Machine Intelligence, Volume 3, Issue 3.
- [4] Cervi, A. L., Poling, K. R. & Higgs, D. M. Behavioral Measure Of Frequency Detection And Discrimination(2012).
- [5] Ganesh R. Jadhav, Kailash J. Karande, “Automatic Vehicle Number Plate Recognition For Vehicle Parking Management System”, liste, Vol.5, No.11, 2014.
- [6] Miller, N., Garnier, S., Hartnett, A. T. & Couzin, I. D. Both Information And Social Cohesion Determine Collective Decisions In Animal Groups. Proc. Natl. Acad. Sci. 110,5263– 5268 (2013).
- [7] Mutua Simon Mandi ,Bernard Shibwabo, Kaibiru Mutua Raphael, ”An Automatic Number Plate Recognition System For Car Park Management”, International Journal Of Computer Applications, Volume 175 – No.7, October 2017
- [8] Ison, M. & Gerlai, R. Associative Learning In Zebrafish (Danio Rerio) In The Plus Maze. Behav. Brain Res. 207, 99–104 (2010).
- [9] Zhou, B.; Yang, J.; Li, Q. Smartphone-Based Activity Recognition For Indoor Localization Using A Convolutional Neural Network. Sensors 2019, 19, 621.

APPENDIX

A. SOURCE CODE

Main.py:

```
from ultralytics import YOLO
import cv2

import util
from sort.sort import *
from util import get_car, read_license_plate, write_csv

results = {}

mot_tracker = Sort()

# load models
coco_model = YOLO('yolov8n.pt')
license_plate_detector = YOLO('license_plate_detector.pt')

# load video
cap = cv2.VideoCapture('sample.mp4')

vehicles = [2, 3, 5, 7]

# read frames
frame_nmr = -1
ret = True
while ret:
    frame_nmr += 1
    ret, frame = cap.read()
    if ret:
        results[frame_nmr] = {}
        # detect vehicles
        detections = coco_model(frame)[0]
        detections_ = []
        for detection in detections.bboxes.data.tolist():
            x1, y1, x2, y2, score, class_id = detection
            if int(class_id) in vehicles:
                detections_.append([x1, y1, x2, y2, score])

        # track vehicles
        track_ids = mot_tracker.update(np.asarray(detections_))

        # detect license plates
        license_plates = license_plate_detector(frame)[0]
```

```

for license_plate in license_plates.bboxes.data.tolist():
    x1, y1, x2, y2, score, class_id = license_plate

    # assign license plate to car
    xcar1, ycar1, xcar2, ycar2, car_id = get_car(license_plate, track_ids)

    if car_id != -1:

        # crop license plate
        license_plate_crop = frame[int(y1):int(y2), int(x1):int(x2), :]

        # process license plate
        license_plate_crop_gray = cv2.cvtColor(license_plate_crop,
cv2.COLOR_BGR2GRAY)
        _, license_plate_crop_thresh = cv2.threshold(license_plate_crop_gray, 64,
255, cv2.THRESH_BINARY_INV)

        # read license plate number
        license_plate_text, license_plate_text_score =
read_license_plate(license_plate_crop_thresh)

        if license_plate_text is not None:
            results[frame_nmr][car_id] = {'car': {'bbox': [xcar1, ycar1, xcar2, ycar2]},
            'license_plate': {'bbox': [x1, y1, x2, y2],
            'text': license_plate_text,
            'bbox_score': score,
            'text_score': license_plate_text_score}}

# write results
write_csv(results, 'test.csv')

```

visualize.py:

```

import ast

import cv2
import numpy as np
import pandas as pd

def draw_border(img, top_left, bottom_right, color=(0, 255, 0), thickness=10,
line_length_x=200, line_length_y=200):
    x1, y1 = top_left
    x2, y2 = bottom_right

    cv2.line(img, (x1, y1), (x1, y1 + line_length_y), color, thickness) #-- top-left
    cv2.line(img, (x1, y1), (x1 + line_length_x, y1), color, thickness)

    cv2.line(img, (x1, y2), (x1, y2 - line_length_y), color, thickness) #-- bottom-left
    cv2.line(img, (x1, y2), (x1 + line_length_x, y2), color, thickness)

```

```

cv2.line(img, (x2, y1), (x2 - line_length_x, y1), color, thickness) #-- top-right
cv2.line(img, (x2, y1), (x2, y1 + line_length_y), color, thickness)

cv2.line(img, (x2, y2), (x2, y2 - line_length_y), color, thickness) #-- bottom-right
cv2.line(img, (x2, y2), (x2 - line_length_x, y2), color, thickness)

return img

results = pd.read_csv('./test_interpolated.csv')

# load video
video_path = 'sample.mp4'
cap = cv2.VideoCapture(video_path)

fourcc = cv2.VideoWriter_fourcc(*'mp4v') # Specify the codec
fps = cap.get(cv2.CAP_PROP_FPS)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
out = cv2.VideoWriter('./out.mp4', fourcc, fps, (width, height))

license_plate = {}
for car_id in np.unique(results['car_id']):
    max_ = np.amax(results[results['car_id'] == car_id]['license_number_score'])
    license_plate[car_id] = {'license_crop': None,
                             'license_plate_number': results[(results['car_id'] == car_id) &
                                                                (results['license_number_score'] ==
max_)]['license_number'].iloc[0]}
    cap.set(cv2.CAP_PROP_POS_FRAMES, results[(results['car_id'] == car_id) &
                                                (results['license_number_score'] ==
max_)]['frame_nmr'].iloc[0])
    ret, frame = cap.read()

    x1, y1, x2, y2 = ast.literal_eval(results[(results['car_id'] == car_id) &
                                                (results['license_number_score'] ==
max_)]['license_plate_bbox'].iloc[0].replace('[', '(').replace(' ', '').replace(' ', ' ').replace(' ', ' ').replace(' ', ' '))

    license_crop = frame[int(y1):int(y2), int(x1):int(x2), :]
    license_crop = cv2.resize(license_crop, (int((x2 - x1) * 400 / (y2 - y1)), 400))

    license_plate[car_id]['license_crop'] = license_crop

frame_nmr = -1

cap.set(cv2.CAP_PROP_POS_FRAMES, 0)

# read frames
ret = True
while ret:
    ret, frame = cap.read()
    frame_nmr += 1

```

```

if ret:
    df_ = results[results['frame_nmr'] == frame_nmr]
    for row_indx in range(len(df_)):
        # draw car
        car_x1, car_y1, car_x2, car_y2 =
ast.literal_eval(df_.iloc[row_indx]['car_bbox'].replace(' ', '').replace(' ', ' ').replace(' ', ' '))
        draw_border(frame, (int(car_x1), int(car_y1)), (int(car_x2), int(car_y2)), (0, 255,
0), 25,
                    line_length_x=200, line_length_y=200)

        # draw license plate
        x1, y1, x2, y2 = ast.literal_eval(df_.iloc[row_indx]['license_plate_bbox'].replace(' ', ' ').replace(' ', ' ').replace(' ', ' '))
        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 12)

        # crop license plate
        license_crop = license_plate[df_.iloc[row_indx]['car_id']]['license_crop']

        H, W, _ = license_crop.shape

        try:
            frame[int(car_y1) - H - 100:int(car_y1) - 100,
                  int((car_x2 + car_x1 - W) / 2):int((car_x2 + car_x1 + W) / 2), :] =
license_crop

            frame[int(car_y1) - H - 400:int(car_y1) - H - 100,
                  int((car_x2 + car_x1 - W) / 2):int((car_x2 + car_x1 + W) / 2), :] = (255, 255,
255)

            (text_width, text_height), _ = cv2.getTextSize(
                license_plate[df_.iloc[row_indx]['car_id']]['license_plate_number'],
                cv2.FONT_HERSHEY_SIMPLEX,
                4.3,
                17)

            cv2.putText(frame,
                        license_plate[df_.iloc[row_indx]['car_id']]['license_plate_number'],
                        (int((car_x2 + car_x1 - text_width) / 2), int(car_y1 - H - 250 +
(text_height / 2))),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        4.3,
                        (0, 0, 0),
                        17)

        except:
            pass

    out.write(frame)
    frame = cv2.resize(frame, (1280, 720))

    # cv2.imshow('frame', frame)
    # cv2.waitKey(0)

```

```
out.release()
cap.release()
```

util.py:

```
import string
import easyocr
```

```
# Initialize the OCR reader
```

```
reader = easyocr.Reader(['en'], gpu=False)
```

```
# Mapping dictionaries for character conversion
```

```
dict_char_to_int = {'O': '0',
                    'l': '1',
                    'J': '3',
                    'A': '4',
                    'G': '6',
                    'S': '5'}
```

```
dict_int_to_char = {'0': 'O',
                    '1': 'l',
                    '3': 'J',
                    '4': 'A',
                    '6': 'G',
                    '5': 'S'}
```

```
def write_csv(results, output_path):
```

```
    """
```

```
    Write the results to a CSV file.
```

```
    Args:
```

```
        results (dict): Dictionary containing the results.
```

```
        output_path (str): Path to the output CSV file.
```

```
    """
```

```
    with open(output_path, 'w') as f:
```

```
        f.write('{},{},{},{},{},{},{}\n'.format('frame_nmr', 'car_id', 'car_bbox',
                                                'license_plate_bbox', 'license_plate_bbox_score',
'license_number',
                                                'license_number_score'))
```

```
    for frame_nmr in results.keys():
```

```
        for car_id in results[frame_nmr].keys():
```

```
            print(results[frame_nmr][car_id])
```

```
            if 'car' in results[frame_nmr][car_id].keys() and \
```

```
                'license_plate' in results[frame_nmr][car_id].keys() and \
```

```
                'text' in results[frame_nmr][car_id]['license_plate'].keys():
```

```
                f.write('{},{},{},{},{},{},{}\n'.format(frame_nmr,
                                                            car_id,
```

```

        '{} {} {} {}'.format(
            results[frame_nmr][car_id]['car']['bbox'][0],
            results[frame_nmr][car_id]['car']['bbox'][1],
            results[frame_nmr][car_id]['car']['bbox'][2],
            results[frame_nmr][car_id]['car']['bbox'][3]),
        '{} {} {} {}'.format(

results[frame_nmr][car_id]['license_plate']['bbox'][0],

results[frame_nmr][car_id]['license_plate']['bbox'][1],

results[frame_nmr][car_id]['license_plate']['bbox'][2],

results[frame_nmr][car_id]['license_plate']['bbox'][3]),

results[frame_nmr][car_id]['license_plate']['bbox_score'],
            results[frame_nmr][car_id]['license_plate']['text'],

results[frame_nmr][car_id]['license_plate']['text_score'])
    )
    f.close()

def license_complies_format(text):
    """
    Check if the license plate text complies with the required format.

    Args:
        text (str): License plate text.

    Returns:
        bool: True if the license plate complies with the format, False otherwise.
    """
    if len(text) != 7:
        return False

    if (text[0] in string.ascii_uppercase or text[0] in dict_int_to_char.keys()) and \
        (text[1] in string.ascii_uppercase or text[1] in dict_int_to_char.keys()) and \
        (text[2] in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'] or text[2] in dict_char_to_int.keys()) and \
        (text[3] in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'] or text[3] in dict_char_to_int.keys()) and \
        (text[4] in string.ascii_uppercase or text[4] in dict_int_to_char.keys()) and \
        (text[5] in string.ascii_uppercase or text[5] in dict_int_to_char.keys()) and \
        (text[6] in string.ascii_uppercase or text[6] in dict_int_to_char.keys()):
        return True
    else:
        return False

def format_license(text):
    """
    Format the license plate text by converting characters using the mapping dictionaries.

```

Args:
text (str): License plate text.

Returns:
str: Formatted license plate text.

```
"""
license_plate_ = ""
mapping = {0: dict_int_to_char, 1: dict_int_to_char, 4: dict_int_to_char, 5:
dict_int_to_char, 6: dict_int_to_char,
          2: dict_char_to_int, 3: dict_char_to_int}
for j in [0, 1, 2, 3, 4, 5, 6]:
    if text[j] in mapping[j].keys():
        license_plate_ += mapping[j][text[j]]
    else:
        license_plate_ += text[j]

return license_plate_
```

def read_license_plate(license_plate_crop):

Read the license plate text from the given cropped image.

Args:
license_plate_crop (PIL.Image.Image): Cropped image containing the license plate.

Returns:
tuple: Tuple containing the formatted license plate text and its confidence score.

```
"""
detections = reader.readtext(license_plate_crop)
```

```
for detection in detections:
    bbox, text, score = detection

    text = text.upper().replace(' ', '')

    if license_complies_format(text):
        return format_license(text), score
```

```
return None, None
```

def get_car(license_plate, vehicle_track_ids):

Retrieve the vehicle coordinates and ID based on the license plate coordinates.

Args:
license_plate (tuple): Tuple containing the coordinates of the license plate (x1, y1, x2, y2, score, class_id).
vehicle_track_ids (list): List of vehicle track IDs and their corresponding coordinates.

Returns:

tuple: Tuple containing the vehicle coordinates (x1, y1, x2, y2) and ID.

"""

x1, y1, x2, y2, score, class_id = license_plate

foundIt = False

for j in range(len(vehicle_track_ids)):

 xcar1, ycar1, xcar2, ycar2, car_id = vehicle_track_ids[j]

 if x1 > xcar1 and y1 > ycar1 and x2 < xcar2 and y2 < ycar2:

 car_indx = j

 foundIt = True

 break

if foundIt:

 return vehicle_track_ids[car_indx]

return -1, -1, -1, -1, -1

add_missing_data.py:

import csv

import numpy as np

from scipy.interpolate import interp1d

def interpolate_bounding_boxes(data):

 # Extract necessary data columns from input data

 frame_numbers = np.array([int(row['frame_nmr']) for row in data])

 car_ids = np.array([int(float(row['car_id'])) for row in data])

 car_bboxes = np.array([list(map(float, row['car_bbox'][1:-1].split())) for row in data])

 license_plate_bboxes = np.array([list(map(float, row['license_plate_bbox'][1:-1].split())) for row in data])

 interpolated_data = []

 unique_car_ids = np.unique(car_ids)

 for car_id in unique_car_ids:

 frame_numbers_ = [p['frame_nmr'] for p in data if int(float(p['car_id'])) == int(float(car_id))]

 print(frame_numbers_, car_id)

 # Filter data for a specific car ID

 car_mask = car_ids == car_id

 car_frame_numbers = frame_numbers[car_mask]

 car_bboxes_interpolated = []

 license_plate_bboxes_interpolated = []

 first_frame_number = car_frame_numbers[0]

```

last_frame_number = car_frame_numbers[-1]

for i in range(len(car_bboxes[car_mask])):
    frame_number = car_frame_numbers[i]
    car_bbox = car_bboxes[car_mask][i]
    license_plate_bbox = license_plate_bboxes[car_mask][i]

    if i > 0:
        prev_frame_number = car_frame_numbers[i-1]
        prev_car_bbox = car_bboxes_interpolated[-1]
        prev_license_plate_bbox = license_plate_bboxes_interpolated[-1]

        if frame_number - prev_frame_number > 1:
            # Interpolate missing frames' bounding boxes
            frames_gap = frame_number - prev_frame_number
            x = np.array([prev_frame_number, frame_number])
            x_new = np.linspace(prev_frame_number, frame_number,
num=frames_gap, endpoint=False)
            interp_func = interp1d(x, np.vstack((prev_car_bbox, car_bbox)), axis=0,
kind='linear')
            interpolated_car_bboxes = interp_func(x_new)
            interp_func = interp1d(x, np.vstack((prev_license_plate_bbox,
license_plate_bbox)), axis=0, kind='linear')
            interpolated_license_plate_bboxes = interp_func(x_new)

            car_bboxes_interpolated.extend(interpolated_car_bboxes[1:])

license_plate_bboxes_interpolated.extend(interpolated_license_plate_bboxes[1:])

car_bboxes_interpolated.append(car_bbox)
license_plate_bboxes_interpolated.append(license_plate_bbox)

for i in range(len(car_bboxes_interpolated)):
    frame_number = first_frame_number + i
    row = {}
    row['frame_nmr'] = str(frame_number)
    row['car_id'] = str(car_id)
    row['car_bbox'] = ''.join(map(str, car_bboxes_interpolated[i]))
    row['license_plate_bbox'] = ''.join(map(str,
license_plate_bboxes_interpolated[i]))

    if str(frame_number) not in frame_numbers_:
        # Imputed row, set the following fields to '0'
        row['license_plate_bbox_score'] = '0'
        row['license_number'] = '0'
        row['license_number_score'] = '0'
    else:
        # Original row, retrieve values from the input data if available
        original_row = [p for p in data if int(p['frame_nmr']) == frame_number and
int(float(p['car_id'])) == int(float(car_id))][0]
        row['license_plate_bbox_score'] = original_row['license_plate_bbox_score'] if
'license_plate_bbox_score' in original_row else '0'
        row['license_number'] = original_row['license_number'] if 'license_number' in

```

```

original_row else '0'
    row['license_number_score'] = original_row['license_number_score'] if
'license_number_score' in original_row else '0'

    interpolated_data.append(row)

return interpolated_data

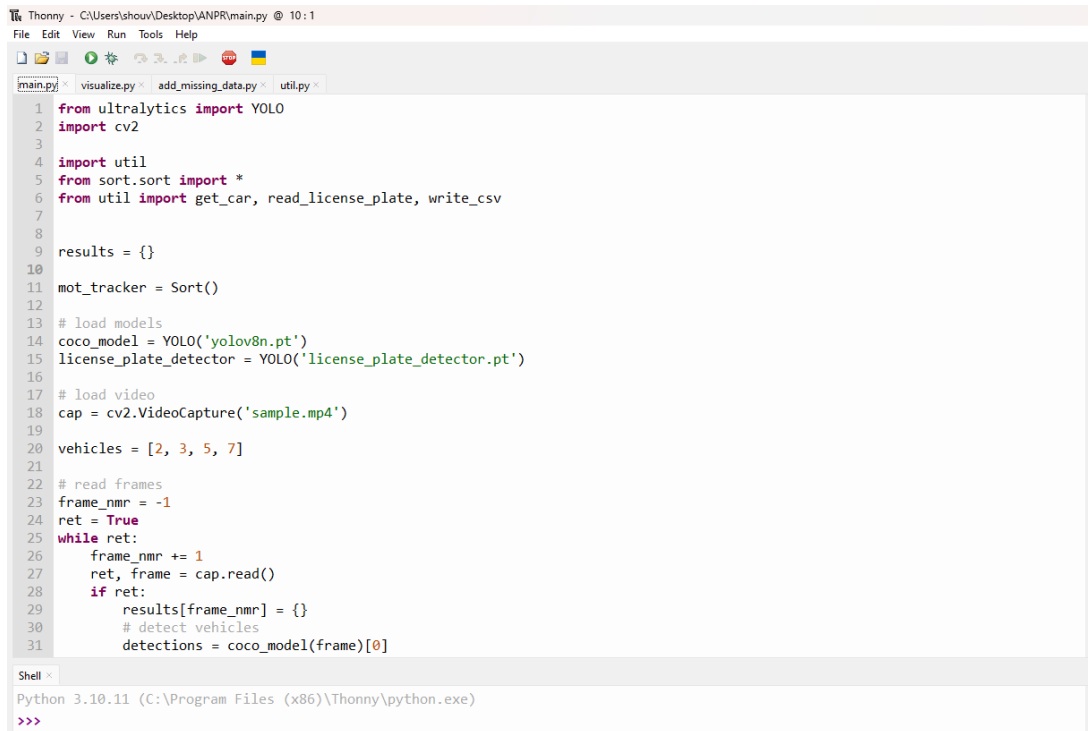
# Load the CSV file
with open('test.csv', 'r') as file:
    reader = csv.DictReader(file)
    data = list(reader)

# Interpolate missing data
interpolated_data = interpolate_bounding_boxes(data)

# Write updated data to a new CSV file
header = ['frame_nmr', 'car_id', 'car_bbox', 'license_plate_bbox',
'license_plate_bbox_score', 'license_number', 'license_number_score']
with open('test_interpolated.csv', 'w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=header)
    writer.writeheader()
    writer.writerows(interpolated_data)

```

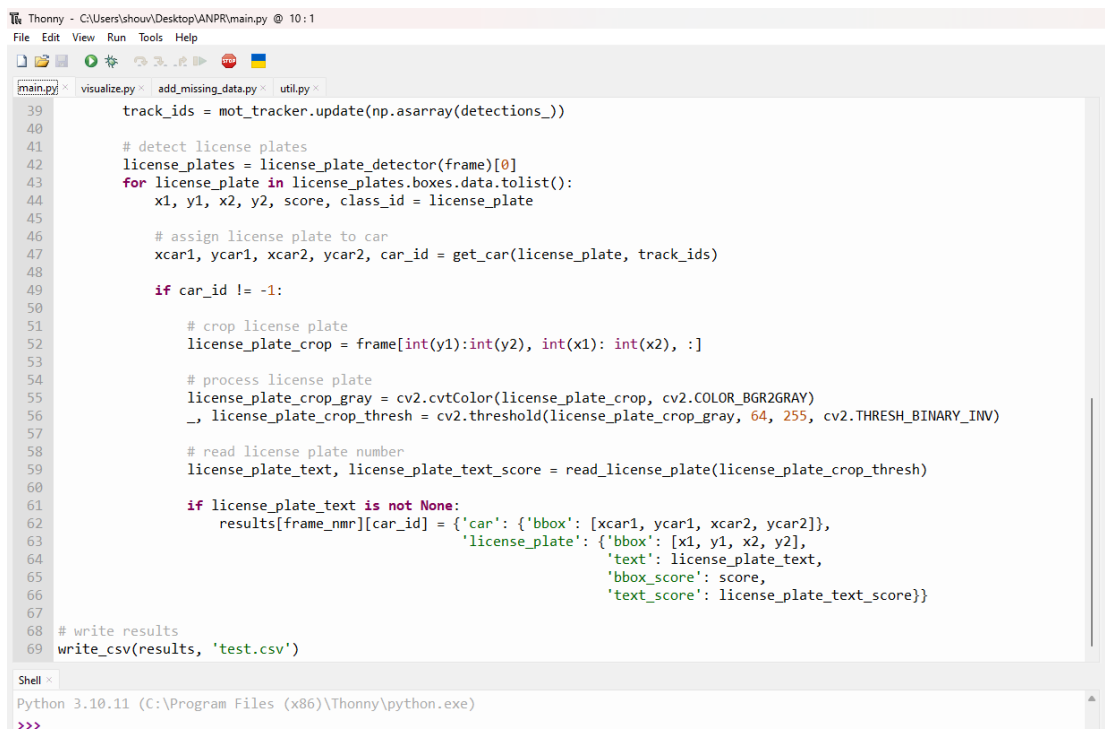
B. SCREENSHOTS



```
Thonny - C:\Users\shou\Desktop\ANPR\main.py @ 10:1
File Edit View Run Tools Help

main.py x visualize.py x add_missing_data.py x util.py x
1 from ultralytics import YOLO
2 import cv2
3
4 import util
5 from sort.sort import *
6 from util import get_car, read_license_plate, write_csv
7
8 results = {}
9
10 mot_tracker = Sort()
11
12 # load models
13 coco_model = YOLO('yolov8n.pt')
14 license_plate_detector = YOLO('license_plate_detector.pt')
15
16 # load video
17 cap = cv2.VideoCapture('sample.mp4')
18
19 vehicles = [2, 3, 5, 7]
20
21 # read frames
22 frame_nmr = -1
23 ret = True
24 while ret:
25     frame_nmr += 1
26     ret, frame = cap.read()
27     if ret:
28         results[frame_nmr] = {}
29         # detect vehicles
30         detections = coco_model(frame)[0]
31
Shell x
Python 3.10.11 (C:\Program Files (x86)\Thonny\python.exe)
>>>
```

Fig B.1: Importing Required Libraries



```
Thonny - C:\Users\shou\Desktop\ANPR\main.py @ 10:1
File Edit View Run Tools Help

main.py x visualize.py x add_missing_data.py x util.py x
39 track_ids = mot_tracker.update(np.asarray(detections_))
40
41 # detect license plates
42 license_plates = license_plate_detector(frame)[0]
43 for license_plate in license_plates.bboxes.data.tolist():
44     x1, y1, x2, y2, score, class_id = license_plate
45
46 # assign license plate to car
47 xcar1, ycar1, xcar2, ycar2, car_id = get_car(license_plate, track_ids)
48
49 if car_id != -1:
50
51     # crop license plate
52     license_plate_crop = frame[int(y1):int(y2), int(x1):int(x2), :]
53
54     # process license plate
55     license_plate_crop_gray = cv2.cvtColor(license_plate_crop, cv2.COLOR_BGR2GRAY)
56     _, license_plate_crop_thresh = cv2.threshold(license_plate_crop_gray, 64, 255, cv2.THRESH_BINARY_INV)
57
58     # read license plate number
59     license_plate_text, license_plate_text_score = read_license_plate(license_plate_crop_thresh)
60
61 if license_plate_text is not None:
62     results[frame_nmr][car_id] = {'car': {'bbox': [xcar1, ycar1, xcar2, ycar2]},
63                                   'license_plate': {'bbox': [x1, y1, x2, y2],
64                                                       'text': license_plate_text,
65                                                       'bbox_score': score,
66                                                       'text_score': license_plate_text_score}}
67
68 # write results
69 write_csv(results, 'test.csv')
Shell x
Python 3.10.11 (C:\Program Files (x86)\Thonny\python.exe)
>>>
```

Fig B.2: Preprocessing the data

```

1 import ast
2
3 import cv2
4 import numpy as np
5 import pandas as pd
6
7
8 def draw_border(img, top_left, bottom_right, color=(0, 255, 0), thickness=10, line_length_x=200, line_length_y=200):
9     x1, y1 = top_left
10    x2, y2 = bottom_right
11
12    cv2.line(img, (x1, y1), (x1, y1 + line_length_y), color, thickness) #-- top-left
13    cv2.line(img, (x1, y1), (x1 + line_length_x, y1), color, thickness)
14
15    cv2.line(img, (x1, y2), (x1, y2 - line_length_y), color, thickness) #-- bottom-left
16    cv2.line(img, (x1, y2), (x1 + line_length_x, y2), color, thickness)
17
18    cv2.line(img, (x2, y1), (x2 - line_length_x, y1), color, thickness) #-- top-right
19    cv2.line(img, (x2, y1), (x2, y1 + line_length_y), color, thickness)
20
21    cv2.line(img, (x2, y2), (x2, y2 - line_length_y), color, thickness) #-- bottom-right
22    cv2.line(img, (x2, y2), (x2 - line_length_x, y2), color, thickness)
23
24    return img
25
26
27 results = pd.read_csv('./test_interpolated.csv')
28
29 # load video
30 video_path = 'sample.mp4'

```

Fig B.3: Extracting the frames

```

61
62 # read frames
63 ret = True
64 while ret:
65     ret, frame = cap.read()
66     frame_nmr += 1
67     if ret:
68         df_ = results[results['frame_nmr'] == frame_nmr]
69         for row_indx in range(len(df_)):
70             # draw car
71             car_x1, car_y1, car_x2, car_y2 = ast.literal_eval(df_.iloc[row_indx]['car_bbox'].replace(' ', '[').replace(' ', ' '),
72             draw_border(frame, (int(car_x1), int(car_y1)), (int(car_x2), int(car_y2)), (0, 255, 0), 25,
73             line_length_x=200, line_length_y=200)
74
75             # draw license plate
76             x1, y1, x2, y2 = ast.literal_eval(df_.iloc[row_indx]['license_plate_bbox'].replace(' ', '[').replace(' ', ' ').r
77             cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 12)
78
79             # crop license plate
80             license_crop = license_plate[df_.iloc[row_indx]['car_id']]['license_crop']
81
82             H, W, _ = license_crop.shape
83
84             try:
85                 frame[int(car_y1) - H - 100:int(car_y1) - 100,
86                     int((car_x2 + car_x1 - W) / 2):int((car_x2 + car_x1 + W) / 2), :] = license_crop
87
88                 frame[int(car_y1) - H - 400:int(car_y1) - H - 100,
89                     int((car_x2 + car_x1 - W) / 2):int((car_x2 + car_x1 + W) / 2), :] = (255, 255, 255)
90

```

Fig B.4: Creating Custom ML model

```

1 import string
2 import easyocr
3
4 # Initialize the OCR reader
5 reader = easyocr.Reader(['en'], gpu=False)
6
7 # Mapping dictionaries for character conversion
8 dict_char_to_int = {'0': '0',
9                    '1': '1',
10                   '2': '2',
11                   '3': '3',
12                   '4': '4',
13                   '5': '5'}
14
15 dict_int_to_char = {'0': '0',
16                   '1': '1',
17                   '2': '2',
18                   '3': '3',
19                   '4': '4',
20                   '5': '5'}
21
22
23 def write_csv(results, output_path):
24     """
25     Write the results to a CSV file.
26
27     Args:
28         results (dict): Dictionary containing the results.
29         output_path (str): Path to the output CSV file.
30     """
31     with open(output_path, 'w') as f:

```

Shell x
Python 3.10.11 (C:\Program Files (x86)\Thonny\python.exe)

Fig B.5: Evaluating the Model

```

65 row['license_plate_bbox_score'] = '0'
66 row['license_number'] = '0'
67 row['license_number_score'] = '0'
68
69 else:
70     # Original row, retrieve values from the input data if available
71     original_row = [p for p in data if int(p['frame_nmr']) == frame_number and int(float(p['car_id'])) == int(float(
72     row['license_plate_bbox_score'] = original_row['license_plate_bbox_score'] if 'license_plate_bbox_score' in ori
73     row['license_number'] = original_row['license_number'] if 'license_number' in original_row else '0'
74     row['license_number_score'] = original_row['license_number_score'] if 'license_number_score' in original_row el
75
76     interpolated_data.append(row)
77
78     return interpolated_data
79
80 # Load the CSV file
81 with open('test.csv', 'r') as file:
82     reader = csv.DictReader(file)
83     data = list(reader)
84
85 # Interpolate missing data
86 interpolated_data = interpolate_bounding_boxes(data)
87
88 # Write updated data to a new CSV file
89 header = ['frame_nmr', 'car_id', 'car_bbox', 'license_plate_bbox', 'license_plate_bbox_score', 'license_number', 'license_numbe
90 with open('test_interpolated.csv', 'w', newline='') as file:
91     writer = csv.DictWriter(file, fieldnames=header)
92     writer.writeheader()
93     writer.writerows(interpolated_data)
94

```

Shell x
Python 3.10.11 (C:\Program Files (x86)\Thonny\python.exe)

Fig B.6: Extracting CSV file

263	1	2049.7807	2239.3417	0.472271	AP05JEO	0.085768
264	1	2048.2825	2230.4216	0.451975	AP05JED	0.371793
265	1	2047.3969	2230.3359	0.455203	AP05JED	0.371793
266	1	2046.5676	2232.2976	0.296281	AP05JEQ	0.545168
267	1	2045.6518	2235.1968	0	0	0
268	1	2044.7360	2238.0960	0	0	0
269	1	2043.8202	2240.9953	0	0	0
270	1	2042.9044	2243.8945	0.340542	AP05JEO	0.543931
271	1	2041.9943	2240.2924	0.357918	AP05JEO	0.362075
272	1	2041.7091	2238.0046	0.477817	AP05JEO	0.186282
273	1	2041.4548	2251.1193	0.549134	AP05JEO	0.364471
274	1	2042.3968	2250.2507	0.639216	AP05JEO	0.454864
275	1	2044.5052	2241.4143	0.565671	AP05JEO	0.435848

Fig B.7: Adding values to missing data

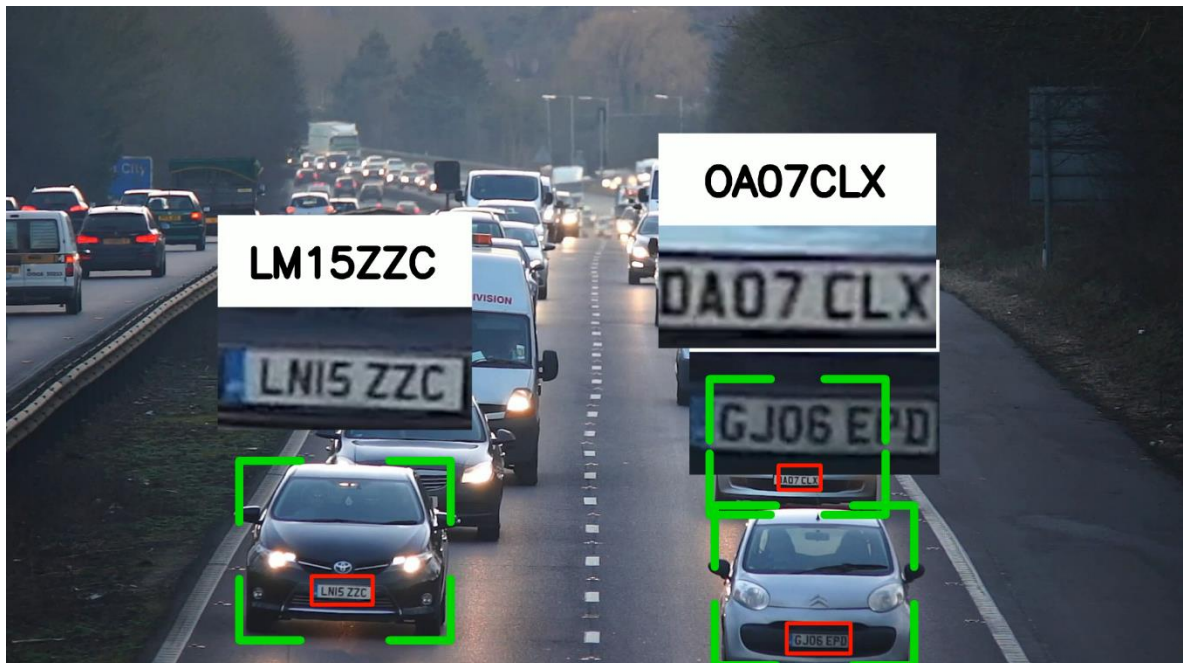


Fig B.8: Tracking the Number Plates

C. RESEARCH PAPER

Automatic Number Plate Detection Using Deep Learning

Shouvin A
*Student, Department of Computer
Science and Engineering,
Sathyabama Institute of Science
and Technology,
Chennai, India
Shouvinkichu@gmail.com*

Shyam Anand S
*Student, Department of Computer
Science and Engineering,
Sathyabama Institute of Science
and Technology,
Chennai, India
shyambaskar1@gmail.com*

Dr. Sethuraman R
*Associate Professor, Department
of Computer Science and
Engineering,
Sathyabama Institute of Science
and Technology,
Chennai, India
sethuraman.cse@sathyabama.ac.in*

Abstract— ANPR (Automatic Number Plate Recognition) technology uses recognition algorithms to identify and locate automobiles based only on their number plates. To optimize accuracy, an ANPR system deployment may need more hardware, even with the finest algorithms. Underlying factors that could compromise its performance include the state of the number plate, non-standard formats, intricate scenes, camera quality, camera mount position, tolerance to distortion, motion blur, contrast issues, reflections, processing and memory limitations, environmental conditions, indoor/outdoor or day/night shots, software tools, or other hardware-based constraints. The irregularity, difficult surroundings, and other complexity of ANPR make it a fascinating topic for academics. The Internet of Things is starting to influence several industries' futures and is opening up new opportunities for ITS. By integrating ANPR with Android platforms, GPS, RFID systems, and other related technologies, it can be effectively exploited. In the field of CV, deep learning approaches are commonly used to improve detection rates. By presenting and assessing a review of extraction, segmentation, and recognition techniques, acknowledging pertinent previous work, and offering guidance on future trends in this field, this research seeks to advance the state-of-knowledge in ITS (ANPR) built on algorithms.

Keywords—Automatic Number Plate Recognition, YOLOv8, EasyOCR, Machine Learning, Object Detection, Optical Character Recognition.

Introduction

Rapid advances in computer vision and deep learning have made robust ANPR systems feasible. To efficiently locate license plates in images, YOLOv8, a real-time object identification technique, is employed. Moreover, a powerful OCR (optical character recognition) library called EasyOCR is used to extract text data from the identified license plates. To get around issues like uneven lighting, misaligned plates, and plate distortions, various technologies are being integrated. They do, however, have a number of disadvantages that are becoming increasingly apparent in the present security environment, despite their numerous advantages.

This study explores the changing field of ANPR to add to the body of information already in existence and offer a thorough framework for improving the performance of identification of license plate systems. Three essential technologies are included in our approach: EasyOCR, YOLOv8, and a custom machine learning model. We want to handle the complexities related to different lighting

circumstances, plate deformations, and the variety of license plate designs by utilizing these cutting-edge components. Recent developments in deep learning and computer vision have greatly accelerated the creation of novel technologies and changed paradigms in a variety of fields.. an essential technology with applications in a variety of industries including roadway administration, police work, and intelligent transport systems, is one such paradigm with transformational potential. ANPR systems, which improve security and expedite traffic operations, are becoming essential parts of smart cities.

The fact that ANPR devices are used so widely indicates how important they are in modern urban environments. These technologies are critical to law enforcement operations because they not only enable efficient traffic management but also assist in the detection and monitoring of vehicles linked to criminal activities. Additionally, ANPR serves as the foundation for intelligent mobility systems, which encourage a safer and more effective vehicle ecology. ANPR technology has come a long way, but there are still problems that call for a diverse approach. Because real-world circumstances are so dynamic and variable, conventional approaches could prove to be able to deal with them. We are therefore driven to incorporate YOLOv8, EasyOCR sequentially, and a distinct machine learning model since we think that a synergistic pairing of these cutting-edge technologies would uncover hitherto unachievable levels of precision and adaptability in license plate recognition.

The main objective of this study is to demonstrate a comprehensive and strong ANPR system that can perform well in real-world scenarios. We want to demonstrate how YOLOv8 for identifying objects, EasyOCR for recognizing optical characters, and a specially-designed machine learning model for the nuances of license plates can work together to achieve capabilities that are not possible with any one of the components alone. Our aim is to make a significant contribution to the continuous development of ANPR innovation and its wider consequences for infrastructure in smart cities by doing this. This paper's later sections will go into more depth about the approach used, including how EasyOCR, YOLOv8, and the custom artificial intelligence model were integrated. Next, we will discuss the experimental design and the criteria that we utilized to assess the effectiveness. The outcomes will be carefully examined in light of practical applications, and then there will be a thorough discussion exploring the advantages, disadvantages, and possible directions for further study.

Lastly, a summary of the major discoveries and an explanation of their significance for the development of ANPR technology will round off the report.

By combining state-of-the-art technologies to simplify ANPR, this research aims to advance the rapidly developing fields of machine vision and artificial intelligence, thereby establishing the groundwork for more resilient and flexible digital license plate recognition systems.



Figure 1:- illustration of how current cctv cameras work

Real-time object recognition applications are the focus of the YOLO or You Only Look Once. YOLO, a neural network designed by Joseph Redmon and Ali Farhadi, uses a single pass over the input picture to forecast limits and class probabilities concurrently. YOLO is able to accomplish exceptional speed while keeping excellent accuracy because to its unique technique. The model has become a standard in computer vision for a number of applications, including as object detection, tracking, and, most famously, Automatic Number Plate detection (ANPR), thanks to many versions such as the ones used in YOLOv4, YOLOv5, and the most recent YOLOv8.

Among the several OCR (Optical Character Recognition) libraries available, EasyOCR is particularly useful for extracting text from photos .It is unique in that it can be tailored to a wide range of fonts, spoken languages, and styles of text that are frequently seen in everyday life. It was designed with ease of use in mind. Its pre-trained models provide seamless integration into applications, rendering it a helpful tool for a variety of activities, including ANPR and document analysis. The library is a strong option for global applications because it supports a number of languages.

Building a Custom Machine Learning Model:

The first step in creating a unique machine learning model is to have a thorough grasp of the issue at hand. Within the framework of ANPR, this entails establishing the goals while taking into account variables such as plate orientation, color fluctuations, and possible application-specific difficulties. Acquiring a representative and varied dataset is

an essential first step. Annotated photos of license plates, representing changes in illumination, plate designs, and any distortions, would make up this dataset in ANPR. The custom model is trained using the dataset as a basis. Cleansing and standardization the photos is a necessary step in preparing the dataset. This stage might involve adjusting image dimensions, standardizing pixel values, and utilizing augmentation methods to improve the model's adaptability to various situations.

Selecting the appropriate architecture is essential. A specialized model for ANPR could have layers expressly tailored to address the complexities of license plate identification after layers of convolution for feature extraction. Efficiency and complexity should be balanced in the architecture. In order for the neural network to understand the patterns and characteristics pertinent to ANPR, the annotated dataset must be fed into the model during training.

Iterations are used to modify weights and biases in order to reduce the discrepancy between expected and actual outcomes. To make sure the model can generalize, it is tested on a different validation dataset after training. Fine-tuning entails changing the architecture or tweaking hyperparameters in response to evaluation results. The bespoke model is prepared for implementation in ANPR systems after it performs satisfactorily. Integrating as described in earlier sections, results in a full solution for precise and efficient number plate recognition.

To put it simply, developing a bespoke neural network model for ANPR entails following a methodical process from conception to implementation, making use of a well selected dataset, and combining the advantages of the YOLO and EasyOCR to provide a complete and efficient solution.



Figure 2:- illustration of how YOLOv8 working in real time

By combining YOLOv8, EasyOCR, and a customized machine learning model, this work presents an extensive approach for Automated Number Plate Recognition (ANPR). The combined technology demonstrates improved precision and flexibility in reading license plates under different circumstances. The results provide a substantial contribution to the advancement of technology, with prospective uses in the fields of security systems, law

enforcement, and traffic management. The outcomes of the experiments support the usefulness of this coordinated strategy. Notably, the proprietary machine learning model, EasyOCR, and YOLOv8 work better together than they do alone, demonstrating improved performance. The technology exhibits robustness in a variety of situations, effectively identifying and detecting license plates even in difficult circumstances..

Related Works

It is crucial to place our approach in the larger context of relevant research and technology advancements in order to further the advancement of systems. An extensive examination of the foundational studies, techniques, and developments that have prepared the way for the incorporation of unique machine learning models is given in the ensuing section. An important component of ANPR is object detection, which has been the subject of several investigations. The app YOLO, which stands for "You Only Look Once," is leading the way in real-time object identification. The most recent version of this series, YOLOv8, is a great option for localizing license plates inside photos since it provides speed and accuracy enhancements. Redmon et al.'s research (2016, 2018) and further additions have paved the way for the integration of YOLO architectures into various applications, including ANPR.

Additional techniques for object detection have also been investigated, including Single Shot Multibox Detectors (SSD) and Region-based neural network convolution (R-CNN). But YOLO is especially useful for real-time applications because of its precision and efficiency, which match the demands of ANPR systems functioning in dynamic contexts. Accurate OCR, or optical character recognition, is critical to the effectiveness of ANPR systems because it allows alphanumeric data to be extracted from license plates. Modern OCR libraries like EasyOCR are well-known for their ability to handle a wide range of fonts, dialects, and text styles. Research in Wang et al. (2020) emphasizes how flexible EasyOCR is to difficult text recognition tasks and how reliable it is in a variety of settings.

Traditional OCR algorithms, such as CRNN (Convolutional neural networks Recurrent Neural Network), have long been used in OCR research. The incorporation of EasyOCR into ANPR systems, on the other hand, signals a strategy change toward greater multinational text recognition and adaptation to the varied typographies seen on license plates.

Custom machine learning algorithms customized for ANPR have been investigated to meet unique issues that general object identification and OCR approaches do not completely handle. Gupta et al.'s (2019) etc Zhang et al.'s (2021) studies highlight how crucial it is to fine-tune models for contextual details as plate position, color changes, and noise reduction.

Custom models are intended to augment the capacities of generic models like as YOLOv8 and EasyOCR, hence improving overall recognition accuracy.

Recent efforts have focused on facilitating the incorporation of domain-specific information, such as color profile analysis and plate-specific geometry. These methods demonstrate improvements in ANPR system reliability and emphasize the need to customize machine learning algorithms to the complexities of license plate identification. The thorough basis for our comprehensive strategy is provided by the related work described in this section. We hope to contribute to the changing field of ANPR technology by expanding on the developments in object detection, optical character recognition, and custom models. We also hope to solve current issues and push the limits of accuracy and versatility in license plate identification. The next sections of this article will provide an overview of our intended ANPR system by going into depth about our approach, experimental setup, findings, and commentary.



Figure 3:- **Object Detection Using YOLOv8**

Proposed Framework

The Our proposed ANPR (Automated Number Plate Recognition) framework builds on the insights from related work by utilizing the capabilities of EasyOCR in order for the recognition of optical characters, YOLOv8 for detecting objects, and a personalized machine learning model made specifically for the complex task of license plate approval. By combining these elements, an ANPR system that is resilient and flexible enough to perform well in real-world situations will be produced. The process and essential elements of our suggested framework are described in the sections that follow.

A. YOLOv8 Integration:

Our framework's initial pillar is the incorporation of YOLOv8 for effective, real-time object identification. Because of its capacity to recognize and categorize items in photos at the same time, YOLOv8 is ideal for detecting license plates. As part of the integration process, the previously trained model is adjusted using a variety of datasets that have been specially annotated with license plate information. This modification guarantees that the model learns to recognize the unique characteristics and variances present in vehicle registration designs.

The first step in our architecture is the YOLOv8-integrated part, which locates and recognizes license plates in input photos. The ANPR pipeline's latter stages make use of the boundaries box coordinates that YOLOv8 produces.

B. EasyOCR Implementation:

Our framework's second pillar, which uses EasyOCR to accurately extract text from localized license plate areas, enhances YOLOv8's object identification capabilities. It is a great option for deciphering alphanumeric letters on license plates because of its adaptability to handle a wide range of fonts, various languages, and text styles. Processing the power source cropped license plate areas that were acquired via YOLOv8 entails extracting the text and turning it into machine-accessible data. This is known as the integration process.

As the text extraction module, the -integrated component transforms the visual data from the plates into a format appropriate for additional analysis. Getting the alphanumeric information needed for further validation and verification depends on this step.

C. Custom Machine Learning Model:

A proprietary machine learning model is included in the third foundation of our platform to solve particular difficulties that YOLOv8 and EasyOCR cannot completely handle. By taking into account contextual details including noise reduction, color changes, and plate orientation, this model aims to improve recognition accuracy. This model can handle more real-world scenarios because it was trained on a specific dataset.

The last refining step in our ANPR workflow is the proprietary machine learning model. To achieve robust recognition, it incorporates extra characteristics and considers the results from

YOLOv8 and EasyOCR. This step is especially important for improving accuracy in situations when standard models might not perform as well.

IV. Workflow

The following is a summary of our suggested framework's workflow:

1. Input Image: Images of cars with license plates are sent to the ANPR system.
2. YOLOv8 Objects Detection: This technique uses bounding box coordinates to locate and identify license plates inside the input photos.
3. EasyOCR Written content Extraction: To extract alphanumeric data from the regional license plate areas, EasyOCR is processed.
4. bespoke Machine Learning Refinement: Taking into account certain contextual characteristics, a bespoke machine learning model further refines the retrieved text and localization data.
5. Output: Accurate alphanumeric information about the identified license plates is included in the final output, which enhances the ANPR system's overall efficacy.

The experimental design, findings, and discussion will be covered in detail in the parts that follow in this article, along with a thorough analysis of how well the suggested framework functions in practical situations.

V. Acknowledgment

Words cannot express my gratitude to my professor for his invaluable patience and feedback. I also could not have undertaken this journey without the help of Computer Science Department, who generously provided knowledge and expertise. Additionally, this endeavor would not have been possible without the generous support from the Univeristy.

VI. References

- [1] Miller, N., Garnier, S., Hartnett, A. T. & Couzin, I. D. Both information and social cohesion determine collective decisions in animal groups. *Proc. Natl. Acad. Sci.* 110, 5263–5268 (2013).
- [2] Sison, M. & Gerlai, R. Associative learning in zebrafish (*Danio rerio*) in the plus maze. *Behav. Brain Res.* 207, 99–104 (2010).
- [3] Cervi, A. L., Poling, K. R. & Higgs, D. M. Behavioral measure of frequency detection and discrimination in the zebrafish, *Danio rerio*. *Zebrafish* 9, 1–7 (2012).
- [4] Avdesh, A. et al. Evaluation of color preference in zebrafish for learning and memory. *J. Alzheimer's Dis.* 28, 459–469 (2012).
- [5] Manabe, K., Dooling, R. J. & Takaku, S. Differential reinforcement of an approach response in zebrafish (*Danio rerio*). *Behav. Process.* 98, 106–111 (2013).
- [6] Wang, S. H., Cheng, X. E., Qian, Z.-M., Liu, Y. & Chen, Y. Q. Automated planar tracking the waving bodies of multiple zebrafish swimming in shallow water. *PLoS ONE* 11, e0154714 (2016).
- [7] Literature Survey on Object Detection using YOLO Rekha B. S.1, Athiya Mariam2, Dr. G. N. Srinivasan3, Supreetha A. Shetty4.

D. CERTIFICATE

**VELAMMAL**
INSTITUTE OF TECHNOLOGY

Approved by AICTE New Delhi & Anna University Chennai
Velammal Knowledge Park, Chennai - Kolkatta Highway, Ponneri 601204


NATIONAL BOARD
OF ACCREDITATION


NATIONAL ASSESSMENT AND
ACCREDITATION COUNCIL


INSTITUTION'S
INNOVATION
COUNCIL

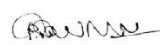
ICA5NT 2024

Certificate of Appreciation

is hereby granted to
SHOUVIN A
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

for **participation and presentation** of paper entitled
AUTOMATIC NUMBER PLATE DETECTION USING DEEP LEARNING

has presented at **4th International Conference on Artificial Intelligence, 5G Communications and Network Technologies (ICA5NT 2024)** organized by the **Department of Electronics and Communication Engineering & Department of Information Technology**, Velammal Institute of Technology, Chennai held on **21 & 22, March 2024**.

					
Coordinator Dr.R.Jothi Chitra Professor-ECE	Coordinator Dr.M.Sivarathinabala Associate Professor-ECE	Coordinator Mr.V.Vinothkumar Assistant Professor-IT	Convener Dr.P.Deivendran Associate Professor & Head-IT	Convener Dr.B.Sridevi Professor & Head-ECE	Principal Dr.N.Balaji