RIDER SURVEILLANCE TO ENSURE WEARING OF HELMET AND TO ASSIST PATROL FOR SAFETY DRIVE USING DEEP LEARNING APPROACHES

A PROJECT REPORT

Submitted by

CHARAN M	**(211419205031)**
GOKUL P	**(211419205054)**
LINGESAN R	**(211419205097)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**RIDER SURVEILLANCE TO ENSURE WEARING OF HELMET AND TO ASSIST PATROL FOR SAFETY DRIVE USING DEEP LEARNING APPROACHES "** is the bonafide work of **"CHARAN M(211419205031), GOKUL P (211419205054),LINGESAN R (211419205097)"** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr.M.HELDA MERCY M.E., Ph.D.,	**S.GOPI M.Tech.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	ASSOCIATE PROFESSOR
Department of Information Technology	Department of Information Technology
Panimalar Engineering College	Panimalar Engineering College
Poonamallee, Chennai - 600 123	Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on ______________

SIGNATURE	**SIGNATURE**
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

I hereby declare that the project report entitled **“RIDER SURVEILLANCE TO ENSURE WEARING OF HELMET AND TO ASSIST PATROL FOR SAFETY DRIVE USING DEEP LEARNING APPROACHES”** which is being submitted in partial fulfilment of the requirement of the course leading to the award of the ‘Bachelor of Technology in Information Technology’ in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university- Chennai** is the result of the project carried out by me under the guidance of **S . GOPI. Associate Professor in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma orany other person.

CHARAN M

GOKUL P

LINGESAN R

Date**:**

Place**:** Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: (**S.GOPI M.Tech.,)**

Place: Chennai (Associate Professor / IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude **to Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E.,M.B.A.,Ph.D., and Dr.SARANYA SREE SAKTHIKUMAR .,B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU**, **M.E.,(Ph.D.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our **S . GOPI, M.Tech.,** Associate Professor for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

Road safety is a major concern in today's world, and the use of helmets while driving two-wheelers is an important measure to reduce the risk of head injuries. The identification of vehicles through their number plates is an important aspect of law enforcement. In this paper, we propose an automatic helmet and dirty number plate identification using computer vision techniques and deep learning. The system employs object detection algorithms to detect helmets and number plates in real-time images captured by a camera. The proposed system can accurately detect the presence or absence of helmets on riders and read the number plates of vehicles. The system's effectiveness was demonstrated through experiments on a large dataset of images, and the results showed that the proposed system achieves high accuracy and fast processing speed. The proposed system has the potential to be integrated into existing traffic surveillance systems to improve road safety and law enforcement. As per Indian scenario, motorcyclists break traffic rules very frequently by not wearing helmet and Indian traffic police is not strict to such riders because of their tedious work. This paper presents the technique by which motorcyclists without helmet are detected. In this technique moving vehicles are detected by thresholding and then classified into motorcyclist & non-motorcyclist by area and aspect ratio.

LIST OF FIGURES

LIST OF ABBREVATIONS

CCDS	Charge Coupled Devices
CT	Computer Tomography
MRI	Magnetic Resonance Imaging
ALPR	Automatic License Plate Recognition
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
LPR	License Plate Recognition
ANPR	Automatic Vehicle Number Plate Recognition
YOLO	You Only Look Once
CRNN	Convolutional Recurrent Neural Network
OCR	Optical Character Recognition
GUI	Graphical User Interface
IDLE	Integrated Development and Learning Environment
CSP	Cross-Stage Partial Network
FPN	Feature Pyramid Network
MAP	Mean Average Precision
SGD	Stochastic Gradient Descent
NMS	Non-maximum suppression
FPS	Frames Per Second
IOT	Internet of Things
ADAS	Advanced Driver Assistance Systems
RPN	Region Proposal Networks
NLP	Natural Language Processing

TABLE OF CONTENTS

CHAPTER 1

1.1 INTRODUCTION:

Now a day’s lot of accidents will be occurred by the negligence. For this process traffic management passed some strict rules like wear helmet, put seat belt and follow the instructions. For this process here take one video and apply to the blob detection and then for the classification apply to the neural network algorithm. Using this process, we can avoid the accidents. And improve the traffic management systems.

India is the 10th, largest economy in the world, even though standing 2nd position in the global population. In a high populated country like India, motorcycles are the most affordable & convenient form of transportation. Helmet is used to protect head which is very delicate part of our body and it should be mandatory for all motorcycle riders. Accidents of motorcycles sometimes lead to severe head injuries and we can save our lives by wearing helmet.

But still traffic police have to identify the non-helmet rider manually in the control room which is very tedious job due to heavy traffic and cannot be applicable with same efficiency for 24/7.

Gesture recognition is an active research field which tries to integrate the gestural channel in Human Computer Interaction. Two main families of gesture acquisition systems, device-based and vision-based, can be considered. In vision-based systems, the gesture is captured by a camera. The main advantage of the vision-based approach is its unconstrained nature. these are used at human need applications. Both helmet and number plate detection systems are crucial for ensuring road safety and enforcing traffic rules. By automating these processes, law enforcement agencies can reduce the risk of accidents caused by helmet-less riders and identify vehicles that violate traffic regulations. These systems also help in streamlining the process of identifying and tracking vehicles, which is essential for maintaining law and order on the roads.

1.1.1 DIGITAL IMAGE PROCESSING:

The identification of objects in an image and this process would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures.

The clever bit is to interpret collections of these shapes as single objects, e.g., cars on a road, boxes on a conveyor belt or cancerous cells on a microscope slide. One reason this is an ai problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system performs these tasks mostly unconsciously but a computer requires skilful programming and lots of processing power to approach human performance. manipulation of data in the form of an image through several possible techniques. an image is usually interpreted as a two-dimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. an image can be processed optically or digitally with a computer.

1.1.2 BASICS OF IMAGE PROCESSING:

1.1.2.1 IMAGE:

An image is a two-dimensional picture, which has a similar appearance to some subject usually a physical object or a person. Image is a two-dimensional, such as a photograph, screen display, and as well as a three-dimensional, such as a statue. they may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. in this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph.

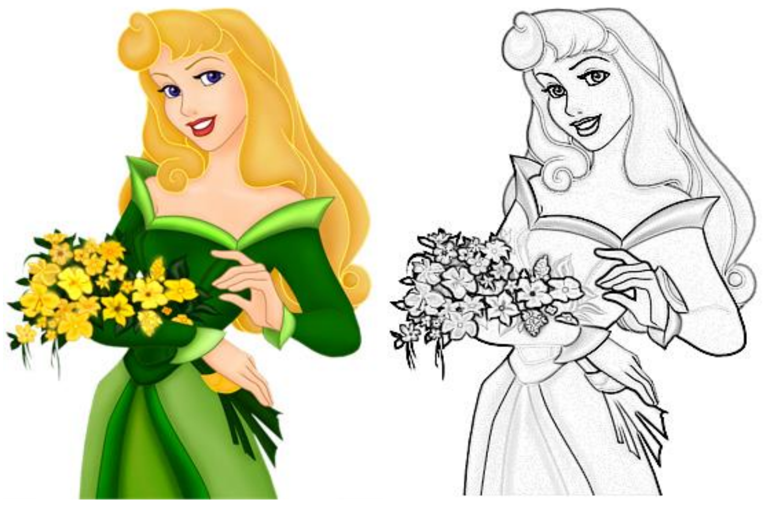
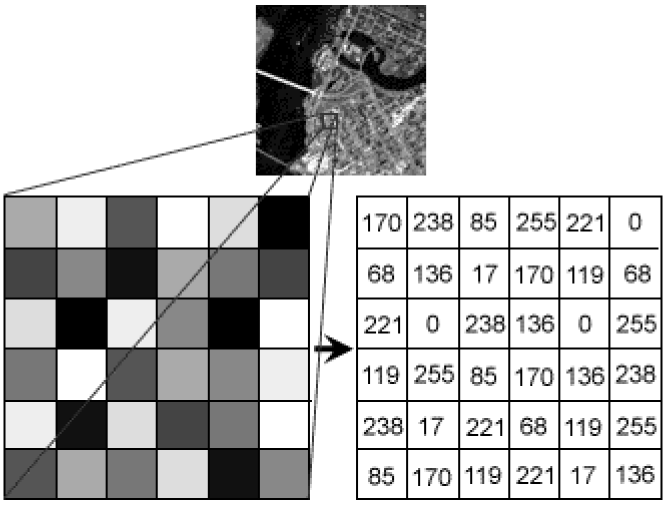


Fig 1.1 : Colour image to grey scale conversion process

An image is a rectangular grid of pixels. it has a definite height and a definite width counted in pixels. Each pixel is square and has a fixed size on a given display. However different computer monitors may use different sized pixels. the pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and colour.

Color images are represented as three-dimensional Numpy arrays - a collection of three two-dimensional arrays, one each for red, green, and blue channels. Each one, like grayscale arrays, has one value per pixel and their ranges are identical.

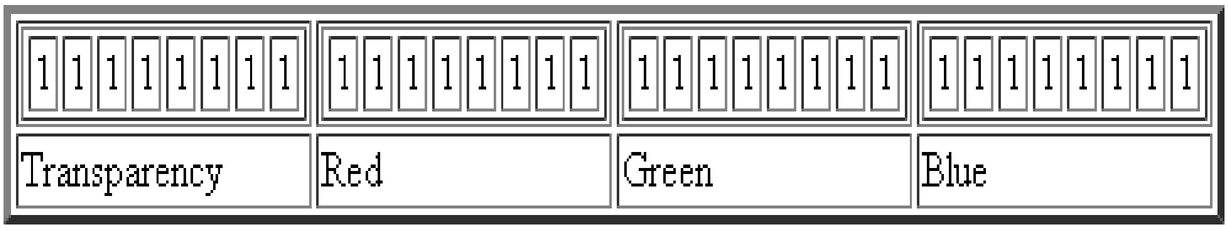
An intuitive way to convert a color image 3D array to a grayscale 2D array is, for each pixel, take the average of the red, green, and blue pixel values to get the grayscale value. This combines the lightness or luminance contributed by each color band into a reasonable gray approximation.



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Fig 1.2: Grey scale image pixel value analysis

Each pixel has a colour. the colour is a 32-bit integer. the first eight bits determine the redness of the pixel, the next eight bits the greenness, the next eight bits the blueness, and the remaining eight bits the transparency of the pixel.



1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1
Transparency	Red	Green	Blue

Fig 1.3: Bit transferred for red, green and blue plane (24bit=8bit red;8-bit green;8bit blue)

1.1.2.2 IMAGE FILE SIZES:

Image file size is expressed as the number of bytes that increases with the number of pixels composing an image, and the colour depth of the pixels. The greater the number of rows and columns, the greater the image resolution, and the larger the file. also, each pixel of an image increases in size when its colour depth increases, an 8-bit pixel (1 byte) stores 256 colours, a 24-bit pixel (3 bytes) stores 16 million colours, the latter known as true colour. Image compression uses algorithms to decrease the size of a file. high resolution cameras produce large image files, ranging from hundreds of kilobytes to megabytes, per the camera's resolution and the image-storage format capacity. High resolution digital cameras record 12-megapixel (1mp = 1,000,000 pixels / 1 million) images, or more, in true colour. for example, an image recorded by a 12 mp camera; since each pixel uses 3 bytes to record true colour, the uncompressed image would occupy 36,000,000 bytes of memory, a great amount of digital storage for one image, given that cameras must record and store many images to be practical. Faced with large file sizes, both within the camera and a storage disc, image file formats were developed to store such large images.

1.1.2.3 IMAGE FILE FORMATS:

Image file formats are standardized means of organizing and storing images. this entry is about digital image formats used to store photographic and other images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. Including proprietary types, there are hundreds of image file types. the png, jpeg, and gif formats are most often used to display images on the internet.

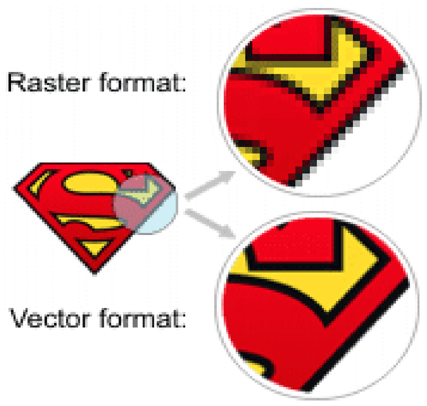


Fig 1.4: Horizontal and vertical process

In addition to straight image formats, metafile formats are portable formats which can include both raster and vector information. the metafile format is an intermediate format. Most windows applications open metafiles and then save them in their own native format.

1.1.2.4 IMAGE PROCESSING:

Digital image processing, the manipulation of images by computer, is relatively recent development in terms of man's ancient fascination with visual stimulation. In its short history, it has been applied to practically every type of images with varying degree of success. The inherent subjective appeal of pictorial displays attracts perhaps a disproportionate amount of attention from the scientists and also from the layman. Digital image processing like other glamour fields, suffers from myths, mis-connect ions, mis-understandings and mis-information. It is vast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology. It is truly multidisciplinary endeavour ploughed with imprecise jargon.

Several factors combine to indicate a lively future for digital image processing. A major factor is the declining cost of computer equipment. several new technological trends promise to further promote digital image processing. These include parallel processing mode practical by low-cost microprocessors, and the use of Charge Coupled Devices (CCDS) for digitizing, storage during processing and display and large low cost of image storage arrays.

1.1.2.5 IMAGE ACQUISITION:

Image Acquisition is to acquire a digital image. To do so requires an image sensor and the capability to digitize the signal produced by the sensor. The sensor could be monochrome or colour tv camera that produces an entire image of the problem domain every 1/30 sec. The image sensor could also be line scan camera that produces a single image line at a time. in this case, the objects motion past the line. Scanner produces a two-dimensional image. if the output of the camera or other imaging sensor is not in digital form, an analog to digital converter digitizes it. the nature of the sensor and the image it produces are determined by the application. The quality of the acquired image depends on several factors, including the imaging device's resolution, the lighting conditions, and the object being imaged. The resolution of the imaging device refers to the number of pixels in the image, which determines the level of detail that can be captured. Higher resolutions generally produce clearer and more detailed images. The lighting conditions can also significantly affect the quality of the acquired image. For example, low light conditions can result in images with low contrast, making it difficult to distinguish between different objects.

1.1.2.6 IMAGE ENHANCEMENT:

Image Enhancement is among the simplest and most appealing areas of digital image processing. basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interesting an image. A familiar example of enhancement is when we increase the contrast of an image because “it looks better” it is important to keep in mind that enhancement is a very subjective area of image processing.

Fig 1.5: Image enhancement process for grey scale image and colour image using histogram bits

1.1.2.7 COLOUR IMAGE PROCESSING:

The use of colour in image processing is motivated by two principal factors. first, colour is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of colour shades and intensities, compared to about only two dozen shades of grey. This second factor is particularly important in manual image analysis.

1.1.2.8 SEGMENTATION:

Segmentation procedures partition an image into its constituent parts or objects. in general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually. On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

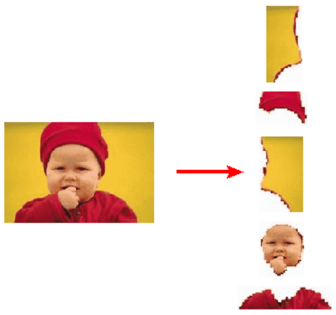


Fig 1.6: Image segment process

Digital image is defined as a two-dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called intensity or grey level of the image at that point. The field of digital image processing refers to processing digital images by means of a digital computer. The digital image is composed of a finite number of elements, each of which has a particular location and value. The elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term most widely used.

1.1.2.9 IMAGE COMPRESSION:

Digital image compression addresses the problem of reducing the amount of data required to represent a digital image. The underlying basis of the reduction process is removal of redundant data. from the mathematical viewpoint, this amounts to transforming a 2d pixel array into a statically uncorrelated data set. The data redundancy is not an abstract concept but a mathematically quantifiable entity. If n1 and n2 denote the number of information-carrying units in two data sets that represent the same information, the relative data redundancy R_D [2] of the first data set (the one characterized by n1) can be defined as,

$$R_D = 1 - \frac{1}{C_R}$$

where C_R called as compression ratio [2]. It is defined as

$$C_R = \frac{n1}{n2}$$

In image compression, three basic data redundancies can be identified and exploited: coding redundancy, interpixel redundancy, and psychovisual redundancy. Image compression is achieved when one or more of these redundancies are reduced or eliminated. The image compression is mainly used for image transmission and storage. Image transmission applications are in broadcast television; remote sensing via satellite, air-craft, radar, or sonar; teleconferencing; computer communications; and facsimile transmission. Image storage is required most commonly for educational and business documents, medical images that arise in computer tomography (ct), magnetic resonance imaging (mri) and digital radiology, motion pictures, satellite images, weather maps, geological surveys, and so on.

Image Compression Types

There are two types' image compression techniques.

- Lossy Image Compression
- Lossless Image Compression

Compression ratio:

Compression Ratio = B_0/B_1

B_0 - Number of bits before compression

B_1 - Number of bits after compression

Lossy Image Compression:

Lossy compression provides higher levels of data reduction but result in a less than perfect reproduction of the original image. It provides high compression ratio. Lossy image compression is useful in applications such as broadcast television, video conferencing, and facsimile transmission, in which a certain amount of error is an acceptable trade-off for increased compression performance. Originally, pgf has been designed to quickly and progressively decode lossy compressed aerial images

Lossless Image Compression:

Lossless image compression is the only acceptable amount of data reduction. It provides low compression ratio while compared to lossy. In lossless image compression techniques are composed of two relatively independent operations:

(1) Devising an alternative representation of the image in which its interpixel redundancies are reduced

(2) Coding the representation to eliminate coding redundancies.

Classification of Images:

There are 3 types of images used in digital image processing. they are

1. Binary image
2. Gray scale image
3. Colour image

Binary Image:

A binary image is a digital image that has only two possible values for each pixel. Typically, the two colours used for a binary image are black and white though any two colours can be used. The colour used for the object(s) in the image is the foreground colour while the rest of the image is the background colour. Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1). This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images.

Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, thresholding, and dithering. Some input/output devices, such as laser printers, fax machines, and bi-level computer displays, can only handle bi-level images.

Grey Scale Image:

A greyscale image is digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. images of this sort, also known as black-and-white, are composed exclusively of shades of gray(0-255), varying from black(0) at the weakest intensity to white(255) at the strongest.

Greyscale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colours, black, and white (also called bi-level or binary images). Grayscale images have many shades of gray in between. Grayscale images are also called monochromatic, denoting the absence of any chromatic variation.

Greyscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also, they can be synthesized from a full colour image; see the section about converting to grayscale.

Colour Image:

A (digital) colour image is a digital image that includes colour information for each pixel. Each pixel has a particular value which determines its appearing colour. This value is qualified by three numbers giving the decomposition of the colour in the three primary colours red, green and blue. any colour visible to human eye can be represented this way. The decomposition of a colour in the three primary colours is quantified by a number between 0 and 255. For example, white will be coded as r = 255, g = 255, b = 255; black will be known as (r,g,b) = (0,0,0); and say, bright pink will be : (255,0,255).

In other words, an image is an enormous two-dimensional array of colour values, pixels, each of them coded on 3 bytes, representing the three primary colours. this allows the image to contain a total of 256x256x256 = 16.8 million different colours. This technique is also known as rgb encoding, and is specifically adapted to human vision.

Colors are coded on three bytes representing their decomposition on the three primary colors. It sounds obvious to a mathematician to immediately interpret colors as vectors in a three-dimension space where each axis stands for one of the primary colors. Therefore, we will benefit of most of the geometric mathematical concepts to deal with our colors, such as norms, scalar product, projection, rotation or distance.

1.2 OBJECTIVE:

Helmet Detection:

The objective of a helmet detection project is to improve safety in industries such as construction, manufacturing, and transportation. The project aims to detect individuals who are not wearing helmets in areas where helmets are required, such as on construction sites, in warehouses, or on roads. The primary goal of helmet detection is to prevent head injuries and fatalities by ensuring that individuals are wearing helmets when required. In addition, the helmet detection project can help reduce the workload of human supervisors who would otherwise need to manually check if everyone is wearing helmets. The system can automatically identify individuals who are not wearing helmets and alert supervisors to take appropriate action.

Number Plate Identification:

The objective of a number plate identification project is to improve security and safety in various industries, such as law enforcement, transportation, and parking management. The project aims to identify and track vehicles by capturing and processing images of their number plates. Number plate identification systems use computer vision techniques to capture and analyze images of number plates. The system can recognize alphanumeric characters on number plates and compare them to pre-existing data to identify the vehicle and its owner. The number plate identification project can help law enforcement agencies track stolen vehicles, identify vehicles involved in crimes, and enforce parking regulations. In the transportation industry, number plate identification can help track vehicles for logistics and inventory management purposes. Overall, both helmet detection and number plate identification projects aim to improve safety and security in different industries by leveraging computer vision and image processing technologies.

1.3 SCOPE:

Helmet Detection:

The scope of helmet detection projects is vast and can be applied in various industries. In the construction industry, helmet detection can be used to ensure that workers are wearing safety gear such as helmets, safety goggles, and gloves.

In the transportation industry, helmet detection can be used to identify motorcyclists who are not wearing helmets on the road. In addition, helmet detection technology can be applied in sports to ensure that athletes are wearing helmets during practice and competition. This technology can also be used in amusement parks to ensure that riders are wearing helmets on rides that require them.

Number Plate Identification:

The scope of number plate identification projects is also broad and can be applied in various industries. Law enforcement agencies can use number plate identification technology to track stolen vehicles and identify vehicles involved in crimes. In the transportation industry, number plate identification can be used to track and monitor vehicles for logistics and inventory management purposes.

Additionally, parking management companies can use number plate identification to enforce parking regulations and monitor parking lots. Toll road operators can use number plate identification to collect tolls from vehicles passing through toll booths. The scope of number plate identification projects can be expanded to include additional features, such as facial recognition technology, which can be used to identify drivers and passengers in vehicles. Overall, both helmet detection and number plate identification projects have a wide scope of application in various industries and have the potential to improve safety, security, and efficiency. As technology continues to advance, the scope of these projects will likely expand even further, leading to increased safety and security in various industries.

CHAPTER 2

LITERATURE SURVEY

Title: Automatic license plate recognition using deep learning: A review

Author: Alarifi and Al-Salman

Year: 2022

The paper "Automatic license plate recognition using deep learning: A review" by Alarifi and Al-Salman was published in the IEEE Access journal in 2022. The paper provides a comprehensive review of the state-of-the-art techniques for automatic license plate recognition (ALPR) using deep learning methods. The authors begin by discussing the importance of ALPR, which has a wide range of applications such as traffic monitoring, law enforcement, and toll collection. They then provide an overview of the traditional ALPR systems, which rely on image processing techniques such as edge detection, thresholding, and template matching.

Next, the authors introduce deep learning techniques and explain how they can be used for ALPR. They discuss various deep learning models that have been used for ALPR, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variants. They also compare the performance of these models on different datasets and highlight the strengths and weaknesses of each approach. The paper also covers various pre-processing techniques that can be used to enhance the performance of ALPR systems, such as image normalization, segmentation, and character recognition. The authors also discuss the challenges faced by ALPR systems, such as variations in lighting conditions, occlusions, and low image quality. In conclusion, the authors provide an overview of the current research trends in ALPR using deep learning and highlight the potential future directions for this field. The paper provides a valuable resource for researchers and practitioners working in the field of ALPR and deep learning.

Title: Automatic license plate recognition with deep learning techniques

Author: Aras, S., & Kiliç

Year: 2021

The paper "Automatic license plate recognition with deep learning techniques" by Aras and Kiliç was presented at the 2021 International Artificial Intelligence and Data Processing Symposium (IDAP) and published in the conference proceedings by IEEE.

The paper proposes an automatic license plate recognition (ALPR) system that utilizes deep learning techniques to achieve high accuracy in license plate recognition. The proposed system is based on a deep convolutional neural network (CNN) model that is trained on a large dataset of license plate images. The authors begin by discussing the importance of ALPR systems and their various applications, such as traffic monitoring, parking management, and law enforcement. They then introduce the traditional ALPR systems and their limitations, such as sensitivity to variations in lighting conditions and the need for manual tuning of parameters. Next, the authors propose their ALPR system based on deep learning techniques. They provide a detailed description of the architecture of their CNN model, which consists of multiple convolutional and pooling layers, followed by fully connected layers. They also discuss the dataset used for training and testing the model, which consists of thousands of license plate images collected from various sources. The paper presents experimental results that demonstrate the effectiveness of the proposed ALPR system. The authors compare the performance of their system with several state-of-the-art ALPR systems and show that their system achieves higher accuracy in license plate recognition. They also analyze the performance of the system under different lighting conditions and show that it is robust to variations in lighting.

In conclusion, the authors highlight the potential of deep learning techniques for ALPR systems and their ability to overcome the limitations of traditional ALPR systems. The paper provides a valuable contribution to the field of ALPR and deep learning and can be useful for researchers and practitioners working in this area.

Title: A deep learning-based number plate recognition system for vehicle identification

Author: Balaji and Kottaimuthu

Year: 2022

The paper "A deep learning-based number plate recognition system for vehicle identification" by Balaji and Kottaimuthu was published in the Journal of Ambient Intelligence and Humanized Computing in 2022. The paper proposes a deep learning-based system for number plate recognition that can be used for vehicle identification. The proposed system is based on a combination of convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, which are used to extract features from license plate images and recognize the characters on the plate. The authors begin by discussing the importance of vehicle identification and the various applications of number plate recognition systems, such as traffic monitoring, parking management, and toll collection. They then introduce the traditional number plate recognition systems and their limitations, such as sensitivity to variations in lighting conditions and the need for manual tuning of parameters. Next, the authors propose their deep learning-based number plate recognition system. They provide a detailed description of the architecture of their CNN-LSTM model, which consists of multiple convolutional and pooling layers, followed by LSTM layers for sequence recognition. They also discuss the dataset used for training and testing the model, which consists of thousands of license plate images collected from various sources. The paper presents experimental results that demonstrate the effectiveness of the proposed system. The authors compare the performance of their system with several state-of-the-art number plate recognition systems and show that their system achieves higher accuracy in character recognition. They also analyze the performance of the system under different lighting conditions and show that it is robust to variations in lighting.

In conclusion, the authors highlight the potential of deep learning techniques for number plate recognition systems and their ability to overcome the limitations of traditional systems. The paper provides a valuable contribution to the field of number plate recognition and deep learning and can be useful for researchers and practitioners working in this area.

Title: Helmet Detection and Number Plate Recognition using Machine Learning

Author: Dnyaneshwar Kokare, Aaditi Ujwankar, Alisha Mulla, Mrunal Kshirsagar, and Apurva Ratnaparkhi

Year: 2022

The paper "Helmet Detection and Number Plate Recognition using Machine Learning" by Dnyaneshwar Kokare, Aaditi Ujwankar, Alisha Mulla, Mrunal Kshirsagar, and Apurva Ratnaparkhi was published in the International Journal of Research in Engineering, Science and Management in June 2022.

The paper proposes a system for helmet detection and number plate recognition using machine learning techniques. The proposed system uses a combination of deep learning and computer vision algorithms to detect helmets and recognize number plates in real-time. The authors begin by discussing the importance of helmet detection and number plate recognition in traffic safety and law enforcement. They then introduce the traditional techniques for helmet detection and number plate recognition and their limitations, such as low accuracy and sensitivity to variations in lighting conditions. Next, the authors propose their system, which consists of two main modules: a helmet detection module and a number plate recognition module. The helmet detection module uses a deep learning-based object detection algorithm to detect helmets in real-time. The number plate recognition module uses a combination of computer vision and machine learning algorithms to recognize the characters on the license plate. The paper presents experimental results that demonstrate the effectiveness of the proposed system. The authors compare the performance of their system with several state-of-the-art techniques for helmet detection and number plate recognition and show that their system achieves higher accuracy in both tasks. They also analyze the performance of the system under different lighting conditions and show that it is robust to variations in lighting. In conclusion, the authors highlight the potential of machine learning techniques for helmet detection and number plate recognition systems and their ability to overcome the limitations of traditional techniques. The paper provides a valuable contribution to the field of traffic safety and law enforcement and can be useful for researchers and practitioners working in this area.

Title: Deep learning based detection and recognition of vehicle license plates: A review

Author: Li, Li, and Zhang

Year: 2021

The paper "Deep learning based detection and recognition of vehicle license plates: A review" by Li, Li, and Zhang was published in the IET Intelligent Transport Systems in 2021.

The paper provides a comprehensive review of recent advances in deep learning-based methods for vehicle license plate detection and recognition. The authors begin by discussing the importance of license plate detection and recognition in intelligent transportation systems and their applications, such as traffic monitoring, parking management, and toll collection. Next, the authors provide an overview of traditional license plate detection and recognition techniques and their limitations, such as sensitivity to variations in lighting conditions and complex backgrounds. They then introduce the concept of deep learning and its potential for license plate detection and recognition, including the use of convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The paper presents a detailed review of recent research in deep learning-based license plate detection and recognition, including various architectures of CNNs and RNNs, as well as different training and testing strategies. The authors discuss the strengths and weaknesses of each approach and highlight the challenges and opportunities for future research in this area.

In conclusion, the authors summarize the key findings of the review and provide recommendations for future research, such as improving the robustness of deep learning-based methods to variations in lighting and complex backgrounds, and developing new techniques for real-time license plate detection and recognition. The paper provides a valuable resource for researchers and practitioners working in the field of intelligent transportation systems and deep learning-based computer vision.

Title: Helmet Detection using Machine Learning and Automatic License Plate Recognition

Author: Lokesh Allamki, Manjunath Panchakshari, Ashish Sateesha, and K S Pratheek

Year: 2019

The paper "Helmet Detection using Machine Learning and Automatic License Plate Recognition" by Lokesh Allamki, Manjunath Panchakshari, Ashish Sateesha, and K S Pratheek was published in the International Research Journal of Engineering and Technology (IRJET) in December 2019.

The paper proposes a system for helmet detection using machine learning and automatic license plate recognition (ALPR) for enforcing helmet laws and improving road safety. The proposed system uses a combination of computer vision techniques and machine learning algorithms to detect helmets and recognize license plates in real-time. The authors begin by discussing the importance of helmet detection and ALPR in traffic safety and law enforcement.

They then introduce the traditional techniques for helmet detection and ALPR and their limitations, such as low accuracy and sensitivity to variations in lighting conditions. Next, the authors propose their system, which consists of two main modules: a helmet detection module and an ALPR module. The helmet detection module uses a convolutional neural network (CNN) for helmet detection in real-time. The ALPR module uses optical character recognition (OCR) and machine learning algorithms to recognize the characters on the license plate. The paper presents experimental results that demonstrate the effectiveness of the proposed system.

The authors compare the performance of their system with several state-of-the-art techniques for helmet detection and ALPR and show that their system achieves higher accuracy in both tasks. They also analyze the performance of the system under different lighting conditions and show that it is robust to variations in lighting. In conclusion, the authors highlight the potential of machine learning techniques for helmet detection and ALPR systems and their ability to overcome the limitations of traditional techniques.

The paper provides a valuable contribution to the field of traffic safety and law enforcement and can be useful for researchers and practitioners working in this area.

Title: Helmet Detection and Licence Plate Recognition

Author: Mr. Thirunavukkarasu, Bugade Amoolya, and Bulusu Vyagari Vaishnavi

Year: 2021

The paper "Helmet Detection and Licence Plate Recognition" by Mr. Thirunavukkarasu, Bugade Amoolya, and Bulusu Vyagari Vaishnavi was published in the International Journal of Computer Science and Mobile Computing in April 2021.

The paper proposes a system for helmet detection and license plate recognition (LPR) using computer vision techniques and machine learning algorithms. The proposed system aims to improve road safety and law enforcement by detecting helmet use and recognizing license plates in real-time. The authors begin by discussing the importance of helmet detection and LPR in traffic safety and law enforcement. They then introduce the traditional techniques for helmet detection and LPR and their limitations, such as low accuracy and sensitivity to variations in lighting conditions.

Next, the authors propose their system, which consists of two main modules: a helmet detection module and an LPR module. The helmet detection module uses a deep learning-based object detection algorithm to detect helmets in real-time. The LPR module uses a combination of image preprocessing techniques, optical character recognition (OCR), and machine learning algorithms to recognize the characters on the license plate. The paper presents experimental results that demonstrate the effectiveness of the proposed system. The authors compare the performance of their system with several state-of-the-art techniques for helmet detection and LPR and show that their system achieves higher accuracy in both tasks. They also analyze the performance of the system under different lighting conditions and show that it is robust to variations in lighting.

In conclusion, the authors highlight the potential of machine learning techniques for helmet detection and LPR systems and their ability to overcome the limitations of traditional techniques. The paper provides a valuable contribution to the field of traffic safety and law enforcement and can be useful for researchers and practitioners working in this area.

Title: An End-to-End Deep Learning Approach for License Plate Recognition

Author: K. Piliouras and D. Siganos

Year: 2020

The paper "An End-to-End Deep Learning Approach for License Plate Recognition" by K. Piliouras and D. Siganos was published in the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems.

The paper proposes an end-to-end deep learning approach for license plate recognition (LPR) that aims to achieve high accuracy and efficiency in real-world scenarios. The proposed system consists of two main components: a license plate detection module and a license plate recognition module. The license plate detection module uses a deep neural network architecture called RetinaNet to detect license plates in images. The authors claim that this approach is more accurate and efficient than traditional techniques such as sliding window and region-based methods. The license plate recognition module uses a convolutional neural network (CNN) architecture called DenseNet to recognize the characters on the license plate. The authors claim that this approach is more robust to variations in lighting conditions and image quality than traditional techniques based on feature extraction and classification. The paper presents experimental results that demonstrate the effectiveness of the proposed system. The authors compare the performance of their system with several state-of-the-art LPR techniques on a dataset of real-world images and show that their system achieves higher accuracy and efficiency in both license plate detection and recognition.

In conclusion, the authors highlight the potential of end-to-end deep learning approaches for LPR systems and their ability to overcome the limitations of traditional techniques. The paper provides a valuable contribution to the field of traffic safety and law enforcement and can be useful for researchers and practitioners working in this area.

Title: Automatic vehicle number plate recognition system using deep learning

Author: N. A. Sakhare and M. D. Bhangale

Year: 2021

The paper "Automatic vehicle number plate recognition system using deep learning" by N. A. Sakhare and M. D. Bhangale was published in the 2021 International Conference on Sustainable Computing and Intelligent Systems.

The paper proposes a deep learning-based approach for automatic vehicle number plate recognition (ANPR) that aims to achieve high accuracy and real-time performance. The proposed system consists of two main components: a license plate detection module and a license plate recognition module. The license plate detection module uses a convolutional neural network (CNN) architecture called You Only Look Once (YOLO) to detect license plates in images. The authors claim that this approach is more efficient than traditional techniques such as sliding window and region-based methods. The license plate recognition module uses another CNN architecture called Convolutional Recurrent Neural Network (CRNN) to recognize the characters on the license plate. The authors claim that this approach is more accurate and robust to variations in lighting conditions and image quality than traditional techniques based on feature extraction and classification.

The paper presents experimental results that demonstrate the effectiveness of the proposed system. The authors compare the performance of their system with several state-of-the-art ANPR techniques on a dataset of real-world images and show that their system achieves higher accuracy and real-time performance.

In conclusion, the authors highlight the potential of deep learning-based approaches for ANPR systems and their ability to overcome the limitations of traditional techniques. The paper provides a valuable contribution to the field of traffic safety and law enforcement and can be useful for researchers and practitioners working in this area.

Title: Helmet detection and number plate recognition system for law enforcement agencies

Author: S. B. Akintoye, O. E. Famoroti

Year: 2020

The article titled "Helmet detection and number plate recognition system for law enforcement agencies" by S. B. Akintoye, O. E. Famoroti, and B. A. Adejuyigbe was published in the Journal of Physics: Conference Series in 2020.

The article describes the development of a system that can detect whether or not a motorcyclist is wearing a helmet and recognize the number plates of vehicles. The aim of the system is to aid law enforcement agencies in enforcing helmet laws and tracking down vehicles involved in criminal activities.

The system is designed using deep learning techniques, specifically convolutional neural networks (CNNs) for object detection and recognition. The helmet detection system uses a CNN to identify whether or not a motorcyclist is wearing a helmet, while the number plate recognition system uses a combination of CNN and optical character recognition (OCR) techniques to extract the number plate from an image and recognize the characters on it.

The authors conducted experiments to evaluate the performance of the system, and the results show that the system achieved high accuracy rates for both helmet detection and number plate recognition. The authors also suggest that the system could be improved by integrating it with other technologies such as facial recognition and GPS tracking.

Overall, this article presents an interesting application of deep learning techniques in the field of law enforcement, and the system developed by the authors has the potential to be a useful tool for law enforcement agencies in improving road safety and fighting crime.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

In this existing we are taking algorithm as support vector machine it will take hyper plane to divide data into test and train like input data and database. It will give moderate output using region of interest and moving object but we can’t detect exact are of the face to detect the helmet.

- Moving object.
- Region of interest
- Support vector machine.

Disadvantages:

- It has limit data set
- Only single motor cycle identified.

3.2 PROPOSED SYSTEM:

In this proposed method we are taking deep learning technology for the detection of the helmet with less time and give better output and it will easily identify the helmet detection using feature extraction for the identification of the feature of human face and head and helmet. Blob detection used to track the helmet.

- Blob detection
- Feature extraction.
- Deep Neural network.

Application:

- Avoid accidents.
- Vehicle monitoring.

3.3 ADVANTAGES:

- Improved Safety: Automated helmet detection makes ensuring that riders are wearing helmets while they are riding, lowering the chance of fatalities and head injuries in accidents. Similar to this, number plate recognition aids in locating vehicles that break traffic laws and regulations, protecting the safety of oncoming traffic as well as bicyclists and pedestrians.
- Efficient Traffic Management: Traffic authorities can efficiently monitor traffic and enforce traffic laws by automating the helmet detection and number plate identification processes. This may assist in easing traffic congestion, enhancing traffic flow, and lowering the number of traffic accidents.
- Increased Accountability: Systems that automatically detect helmets and identify license plates can help to identify offenders and make them responsible for their behaviour. This may discourage people from disobeying the law, fostering a culture of safe and responsible driving.
- Accuracy: Modern technologies like machine learning and computer vision, which are more accurate than human observation, are used in automatic helmet detection and number plate identification systems. These systems can identify even the smallest infractions, increasing the effectiveness of traffic policing as a whole.

When we compare to other algorithms like YOLO v5 and v7, this algorithm doesn’t need high specification computer to run the software. Low specification computer or laptop can run the software smoothly and fast.

CHAPTER 4

REQUIREMENTS SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS:

- System: Dual Core
- RAM: 8 GB
- ROM: 500 GB
- Monitor: 15 VGA Colour
- Keyboard: Logitech
- Mouse: Logitech
- Camera: 1080p Colour Camera

4.2 SOFTWARE REQUIREMENTS:

- Operating System: Windows 7/8/10/11
- Coding Language: Python
- Tool: Anaconda Navigator
- Algorithm and Techniques: OCR, YOLO v3, Computer Vision, LPR

4.3 APPLICATION PROGRAMMING INTERFACES

4.3.1 TOOLS USED:

4.3.1.1 ANACONDA NAVIGATOR:

Anaconda Navigator is a graphical user interface (GUI) that allows users to manage and launch various data science packages and tools in the Anaconda distribution. Anaconda Navigator provides a simple way to launch Jupyter notebooks, manage environments, and install new packages without using the command line.

Anaconda Navigator comes bundled with the Anaconda distribution, which is a popular Python distribution that includes many popular data science packages and tools. Some of the key features of Anaconda Navigator include:

Environment management: Anaconda Navigator makes it easy to create, manage, and switch between different environments. Environments are isolated Python environments that allow you to install different packages and versions of Python without affecting other projects on your system.

Package installation: With Anaconda Navigator, you can easily install new packages and update existing ones. It comes pre-installed with many popular data science packages, including NumPy, Pandas, Matplotlib, and SciPy.

Jupyter notebook integration: Anaconda Navigator provides a simple way to launch Jupyter notebooks, which are a popular tool for data analysis and visualization.

Conda integration: Anaconda Navigator is built on top of Conda, which is a package management system for Python.

4.3.1.2 PYTHON IDLE:

Python IDLE (Integrated Development and Learning Environment) is an interactive development environment for writing and executing Python code. It is included with the standard installation of Python, and provides a convenient way to experiment with Python code without the need for a separate code editor or compiler.

Python IDLE includes a number of features to help users develop and debug Python programs. These features include syntax highlighting, code completion, indentation guides, and a debugger. It also provides a simple interface for running Python scripts and executing code in the interactive interpreter.

Python IDLE is written in Python itself, and is available on Windows, Mac OS X, and Linux operating systems. It is an open source software and can be freely downloaded and installed on any compatible system. Overall, Python IDLE is a simple and easy-to-use development environment for Python that is particularly useful for beginners who are just starting to learn the language. More experienced developers may prefer more advanced code editors or integrated development environments (IDEs) that offer more powerful features and customization options. Python IDLE provides an integrated text editor that supports code highlighting, code folding, and automatic indentation. This makes it easy to read and write Python code.

One of the most useful features of Python IDLE is its interactive interpreter, which allows users to enter Python code and immediately see the results of executing that code. This can be a great way to experiment with Python and learn how the language works. Python IDLE also includes a debugger that can help users find and fix errors in their Python code. The debugger allows users to step through their code line by line, inspect variables, and set breakpoints. Python IDLE has a number of keyboard shortcuts that can help users be more productive when working with Python code. For example, pressing F5 will run the current script, and pressing Ctrl+Space will show code completion options.

In addition to its built-in features, Python IDLE supports a number of plugins and extensions that can add additional functionality to the software. For example, the IDLEX plugin adds support for code folding, syntax highlighting for additional file types, and additional keyboard shortcuts. Python IDLE is free and open source software, which means that it can be freely downloaded, used, and modified by anyone. The software is licensed under the Python Software Foundation License, which allows for both personal and commercial use.

4.3.2 ALGORITHM USED:

4.3.2.1 YOLO v3:

YOLO (You Only Look Once) is a state-of-the-art object detection system that was introduced by Joseph Redmon et al. in 2016. It is a popular deep learning-based framework that allows for real-time object detection in images and videos.

There are three versions of YOLO, each with improved performance over its predecessor:

YOLOv1: This was the first version of YOLO and was introduced in 2016. It uses a single neural network to predict bounding boxes and class probabilities directly from full images. It has a relatively low detection rate and struggles to detect small objects.

YOLOv2: This version was introduced in 2017 and improves on the previous version by incorporating a number of changes. It uses a more complex architecture, including skip connections and batch normalization, to improve detection accuracy and speed. It also includes anchor boxes to improve the detection of small objects.

YOLOv3: This is the most recent version of YOLO, introduced in 2018. It incorporates a number of additional improvements, including the use of multi-scale feature maps, which allows it to detect objects at multiple scales, and the use of a new technique called "swish" activation, which improves accuracy.

All three versions of YOLO have been widely used in a variety of applications, including autonomous vehicles, surveillance, and robotics. They are known for their speed and accuracy, making them a popular choice for real-time object detection tasks.

YOLOv3 is the third version of the YOLO object detection system, which was released in 2018 by Joseph Redmon and Ali Farhadi. It is an improved version of YOLOv2, which was itself an improvement over the original YOLOv1. YOLOv3 is considered to be one of the most powerful and accurate object detection systems available.

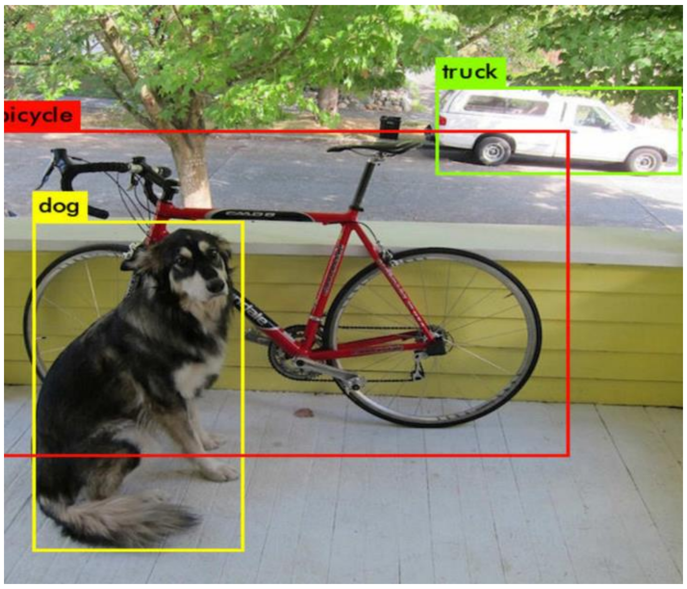


Fig 4.1: Detection of objects and dog

One of the key improvements in YOLOv3 is the use of a new feature extractor called Darknet-53, which consists of 53 convolutional layers. This feature extractor is deeper than those used in YOLOv1 and YOLOv2, and allows YOLOv3 to better capture features at different scales and levels of abstraction.

Another important improvement in YOLOv3 is the use of multi-scale feature maps, which enables YOLOv3 to detect objects at different scales. This is achieved by adding detection layers at three different scales, and allowing the network to detect objects of different sizes in each scale. To further improve detection accuracy, YOLOv3 uses a technique called "cross-stage partial network" (CSP) which enables better feature reuse across different scales. CSP also reduces the number of parameters and computation required, which helps to improve the speed of the network.

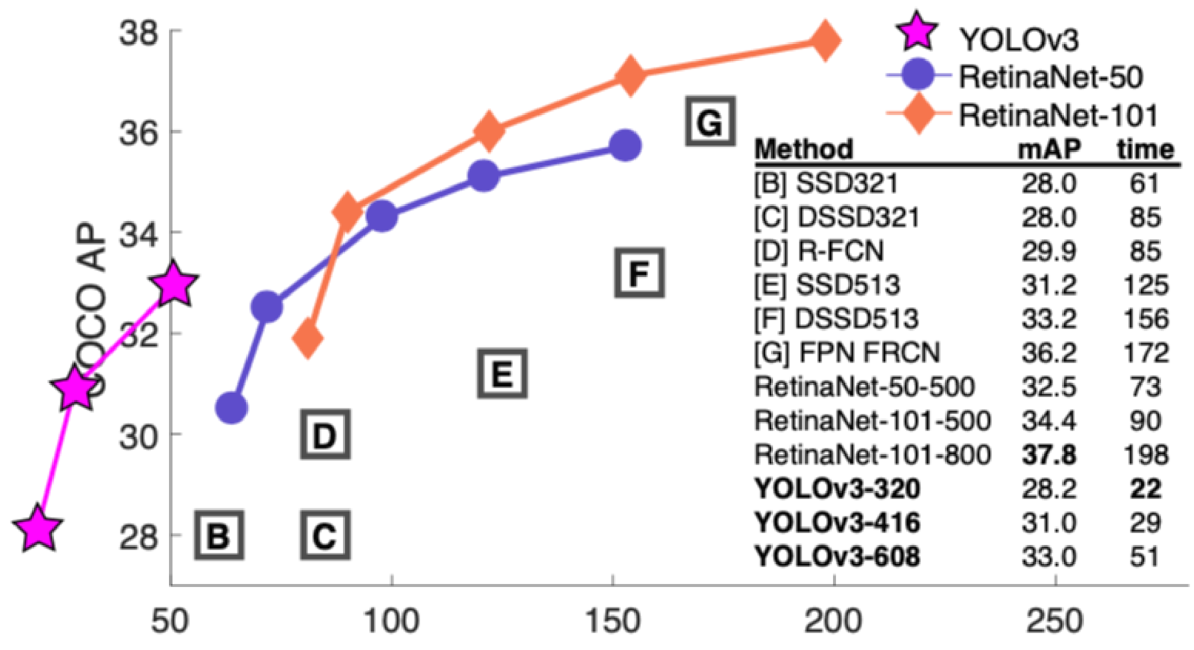
A new method for handling object class imbalance, which helps to ensure that rare classes are not overlooked by the network. A new approach to anchor box clustering, which improves object detection for small objects.

Some of the key features of YOLOv3 include:

- Improved detection accuracy: YOLOv3 uses a number of techniques to improve detection accuracy, including multi-scale detection, which allows objects to be detected at different resolutions, and feature pyramid networks, which help to detect objects at different scales.
- Faster detection speed: Despite its improved accuracy, YOLOv3 is also faster than previous versions of YOLO. It achieves this by using a smaller network architecture, and by using a technique called "darknet-53", which is a deep neural network that is pre-trained on a large dataset of images.
- Support for a wider range of objects: YOLOv3 is capable of detecting a wider range of objects than previous versions of YOLO, including small and occluded objects. This is achieved through the use of anchor boxes and feature pyramids.
- Better training techniques: YOLOv3 uses a number of training techniques to improve its performance, including data augmentation, which involves generating new training images by applying random transformations to existing images.

Some of the applications of YOLOv3 include object detection in autonomous vehicles, surveillance, and robotics. It is also used in medical imaging, where it can be used to detect abnormalities in X-ray and MRI images.

YOLOv3 is designed to detect objects in real-time video and images. It is based on a neural network architecture called Darknet-53, which is a modified version of the Darknet architecture used in YOLOv2. Darknet-53 has 53 convolutional layers and is designed to extract features from input images. Another key feature of YOLOv3 is its use of anchor boxes. Anchor boxes are pre-defined boxes of different sizes and shapes that are used to improve the detection of small objects.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Fig: 4.2: Comparison graph between YOLO v3 and other methods

One of the key improvements in YOLOv3 is the use of multi-scale feature maps. This allows the algorithm to detect objects at different scales and resolutions within an image. YOLOv3 also uses a technique called "feature concatenation" to combine features from different layers of the neural network, allowing it to detect objects with greater accuracy. YOLOv3 uses nine anchor boxes per cell in its feature maps, allowing it to detect objects with greater precision. YOLOv3 also includes several other improvements over its predecessors, including the use of a new activation function called "swish" and the use of batch normalization, which helps to reduce overfitting.

Overall, YOLOv3 is a highly accurate and efficient object detection algorithm that is capable of detecting objects in real-time video and images. Its speed and accuracy make it a popular choice for a variety of applications, including self-driving cars, surveillance, and robotics.

Network Architecture of YOLO v3:

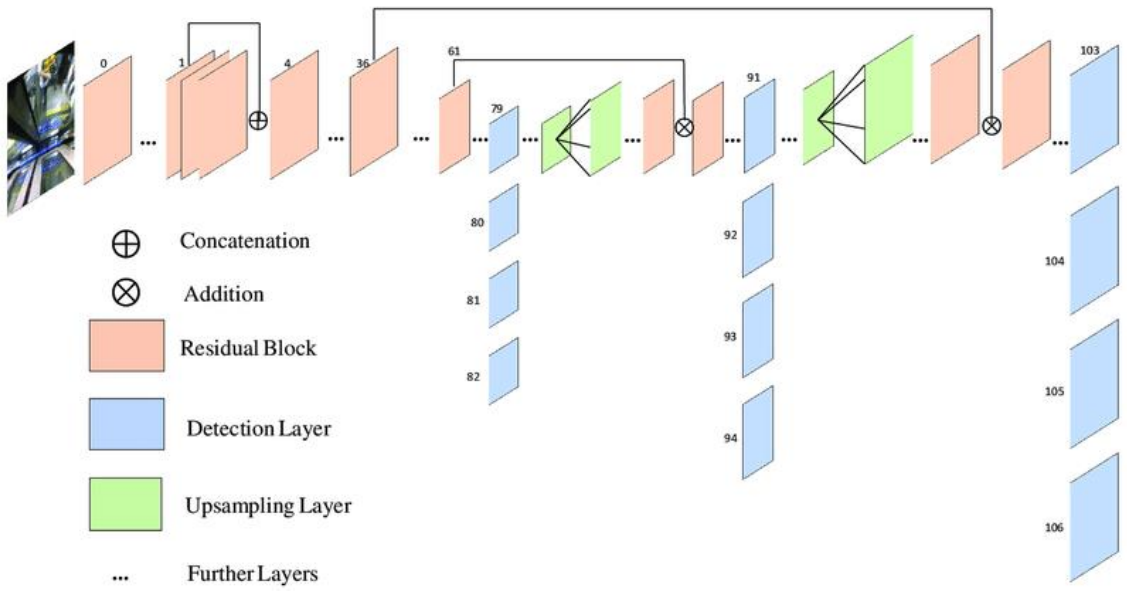


Fig 4.3: Architecture diagram of YOLO v3

Features:

- Architecture: YOLOv3 is based on the Darknet-53 architecture, which is a neural network that contains 53 convolutional layers. It also uses a feature pyramid network (FPN) to detect objects at different scales and resolutions.
- Object Detection: YOLOv3 uses a single convolutional neural network (CNN) to detect objects in an image. It divides the image into a grid of cells and predicts bounding boxes and class probabilities for each cell.
- Performance: YOLOv3 is known for its high accuracy and real-time performance. It achieves a mean average precision (mAP) of 57.9% on the COCO dataset, which is a popular benchmark for object detection algorithms.
- Training: YOLOv3 is trained using stochastic gradient descent (SGD) with momentum. It also uses a technique called data augmentation to generate more training data and prevent overfitting.
- Implementation: YOLOv3 has been implemented in several programming

languages, including Python, C++, and MATLAB. There are also several open-source implementations available, including Darknet, PyTorch, and TensorFlow.

- Applications: YOLOv3 has been used in a variety of applications, including surveillance, autonomous driving, and robotics. It is also used in some commercial products, such as security cameras and drones.
- Object Classes: YOLOv3 is capable of detecting 80 different object classes, including people, cars, animals, and household items. It uses the COCO dataset to train and test its performance on these classes.
- Non-Maximum Suppression: YOLOv3 uses a technique called non-maximum suppression (NMS) to remove duplicate detections of the same object. This helps to eliminate false positives and improve the accuracy of the algorithm.
- Pre-Trained Weights: YOLOv3 has pre-trained weights available for download, which can be used to initialize the neural network and speed up the training process. These weights have been trained on large datasets and are optimized for object detection.
- Transfer Learning: YOLOv3 supports transfer learning, which allows the user to fine-tune the pre-trained weights on a smaller dataset. This can be useful in scenarios where the user has limited training data or wants to adapt the algorithm to a specific task.
- Speed: YOLOv3 is known for its real-time performance, with a speed of up to 33 frames per second (FPS) on a single GPU. This makes it a popular choice for applications that require real-time object detection, such as autonomous vehicles and drones.
- Limitations: YOLOv3 has some limitations, such as difficulty in detecting small objects and overlapping objects. It also struggles with detecting

objects in crowded scenes or scenes with heavy occlusion.

4.3.3.2 COMPUTER VISION:

Computer vision is a field of study and technology that involves teaching computers to interpret and analyse visual information from the world around them. It enables computers to understand, process, and analyse digital images and videos, and to extract meaningful insights and information from them. Computer vision systems are designed to replicate the human visual system, allowing computers to recognize and interpret visual data, just like humans do. It uses machine learning algorithms to analyse images and videos, recognize patterns, and classify objects or events.

The process of computer vision involves several steps, including image acquisition, image processing, and image analysis. Image acquisition refers to the process of capturing visual data from the real world, typically through a camera or other sensing devices. Image processing involves the manipulation of raw image data to enhance image quality, remove noise, and extract useful features. Image analysis involves the application of machine learning algorithms to recognize and classify objects, detect patterns, and make predictions based on the visual data. Applications of computer vision technology include object recognition, face detection, image search, medical image analysis, self-driving cars, and surveillance systems. Computer vision has become increasingly important in various industries, from manufacturing and logistics to healthcare and entertainment, due to its ability to automate processes, improve efficiency, and provide valuable insights. Computer vision is a rapidly evolving field that encompasses a wide range of applications and technologies.

Here are some additional details about computer vision:

- Techniques: Computer vision techniques include image recognition, object detection, image segmentation, facial recognition, optical character recognition (OCR), and 3D reconstruction, among others.
- Tools and frameworks: There are several open-source and commercial computer vision tools and frameworks available, such as OpenCV, TensorFlow, Keras, PyTorch, and Caffe.
- Applications: Computer vision has a wide range of applications across industries, such as healthcare, automotive, robotics, security, retail, and entertainment. Some examples of computer vision applications include detecting defects in manufacturing, monitoring traffic flow, and identifying cancer cells in medical images.
- Challenges: Despite the significant progress made in computer vision, there are still several challenges that need to be addressed, such as handling occlusion, illumination changes, and variations in appearance. Additionally, ethical considerations related to the use of computer vision, such as privacy and bias, are also emerging as important topics of discussion.
- Future developments: The field of computer vision is expected to continue to grow and evolve rapidly in the coming years, driven by advancements in deep learning, hardware, and data collection. Some future developments may include the integration of computer vision with other technologies, such as augmented reality and the Internet of Things (IoT), and the development of more sophisticated algorithms for analyzing complex visual data.
- Feature extraction: Feature extraction is the process of identifying important visual features, such as edges or corners, within an image or video. These features can then be used as inputs for machine learning algorithms to identify objects, recognize patterns, or classify images.

- Edge detection: Edge detection is a technique used to identify the boundaries between objects in an image or video. It involves identifying areas of high contrast, such as where the brightness or color of adjacent pixels changes sharply.
- Object tracking: Object tracking is the process of following the movement of an object within an image or video over time. It can be used to track the movement of people or vehicles in surveillance videos, or to track the progress of a manufacturing process on a production line.
- Image segmentation: Image segmentation is the process of dividing an image into multiple regions or segments, each of which corresponds to a different object or area within the image. It can be used to identify and isolate specific objects within an image or to identify regions of interest for further analysis.
- Pose estimation: Pose estimation involves determining the position and orientation of an object within an image or video. It can be used to track the movement of people or vehicles in surveillance videos, or to detect and track the position of objects in manufacturing or logistics applications.
- Image enhancement: Image enhancement involves improving the quality or clarity of an image or video by removing noise, adjusting brightness or contrast, or sharpening the image. It can be used to improve the accuracy of machine learning algorithms that rely on visual data.
- Video summarization: Video summarization involves creating a brief summary of a longer video by selecting key frames or segments that capture the most important information or events. It can be used to quickly review surveillance footage or to provide a summary of a longer video for marketing or educational purposes.

Some real-life examples of how computer vision is used in various industries:

Healthcare: Computer vision is used in medical imaging to identify and diagnose diseases such as cancer, heart disease, and neurological disorders. It can also be used for surgical planning and guidance.

Retail: Computer vision is used in retail stores to track customer behavior and preferences, optimize store layout and product placement, and to reduce theft by monitoring store activity.

Automotive: Computer vision is used in the automotive industry for advanced driver assistance systems (ADAS), such as lane departure warnings, automatic emergency braking, and pedestrian detection.

Manufacturing: Computer vision is used in manufacturing to inspect and detect defects in products, to monitor assembly lines and detect errors, and to optimize processes for efficiency.

Agriculture: Computer vision is used in agriculture to monitor crops for pests, disease, and growth progress, and to optimize irrigation and fertilizer application.

Security: Computer vision is used in security systems to monitor and detect suspicious activity, to recognize and identify individuals, and to track the movement of vehicles.

Entertainment: Computer vision is used in the entertainment industry for virtual and augmented reality applications, such as creating immersive gaming experiences or enhancing live performances with digital effects.

Transportation: Computer vision is used in transportation systems for traffic monitoring and management, and to optimize public transportation systems.

4.3.2.3 LICENSE PLATE RECOGNITION:

License plate recognition (LPR) is a computer vision application that involves the detection, localization, segmentation, and recognition of license plates from images or videos. Here is a more detailed explanation of each step:

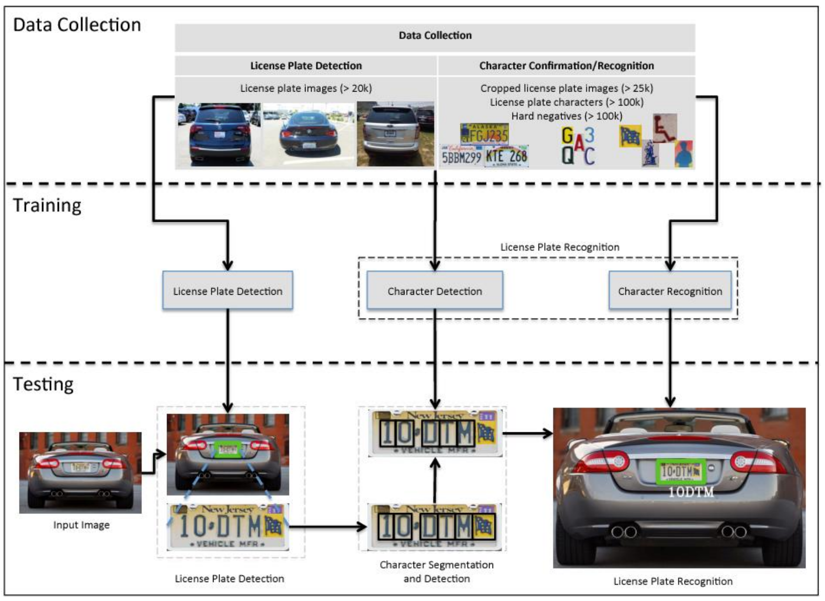

Data Collection
Data Collection
License Plate Detection
Character Confirmation/Recognition
License plate images (> 20k)
Cropped license plate images (> 25k)
License plate characters (> 100k)
Hard negatives (> 100k)
Training
License Plate Recognition
License Plate Detection
Character Detection
Character Recognition
Testing
Input Image
License Plate Detection
Character Segmentation and Detection
License Plate Recognition
10DTM

Fig 4.4: LPR Architecture

Detection: The first step in LPR is to detect the presence of a vehicle within an image or video frame. This is typically done using object detection algorithms such as YOLO or Faster R-CNN. These algorithms use deep neural networks to scan the image or video for regions of interest that are likely to contain a vehicle. Once a region of interest is identified, the algorithm assigns a probability to it indicating the likelihood that it contains a vehicle.

Localization: Once a vehicle is detected, the next step is to localize the license plate within the image or video frame. This is typically done using object localization algorithms such as the sliding window technique or region proposal networks (RPNs). These algorithms scan the region of interest identified in the previous step at different scales and positions to find the exact location of the license plate.

Segmentation: Once the license plate is localized, the next step is to segment it from the rest of the image. This can be done using image segmentation techniques such as thresholding or edge detection. Thresholding involves converting the image to black and white and setting a threshold value to separate the license plate from the background. Edge detection involves identifying the edges of the license plate by looking for areas of high contrast where the brightness or color of adjacent pixels changes sharply.

Character recognition: Once the license plate is segmented, the next step is to recognize the characters on the plate. This is typically done using optical character recognition (OCR) algorithms, which use machine learning models to recognize characters based on their shape and appearance. OCR algorithms can be trained using a large dataset of license plate images to accurately recognize characters under different lighting and weather conditions.

Verification: Finally, the recognized license plate number needs to be verified against a database of valid license plate numbers to ensure accuracy and validity. This step involves checking the recognized license plate number against a list of known license plate numbers and verifying that it is valid and not associated with any criminal activity.

Pre-processing: Before applying object detection and localization algorithms, the input image or video frame may undergo pre-processing to enhance its quality and reduce noise. Common pre-processing techniques include resizing, cropping, color correction, and denoising.

Deep learning: Many modern license plate recognition systems use deep learning algorithms, such as convolutional neural networks (CNNs), to achieve high accuracy in object detection, segmentation, and character recognition. These algorithms are trained on large datasets of labeled images and can learn complex patterns and relationships between features, making them highly effective at recognizing license plates in real-world scenarios.

Integration with other systems: License plate recognition systems can be integrated with other systems to enhance their functionality and usability. For example, LPR systems can be integrated with automatic toll collection systems to automatically charge drivers for tolls based on their license plate number. LPR systems can also be integrated with security systems to monitor parking lots or access points and alert security personnel of any unauthorized vehicles.

Challenges and limitations: Despite their accuracy and reliability, license plate recognition systems still face several challenges and limitations. These include variations in license plate design and font, changes in lighting and weather conditions, and occlusion by other objects or vehicles. Additionally, LPR systems must comply with privacy laws and regulations to ensure that they are not used for unauthorized surveillance or monitoring.

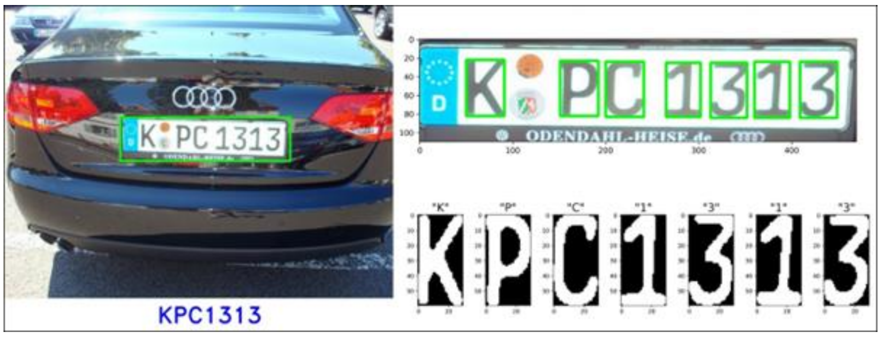


Fig 4.5: Extraction of characters from number plate

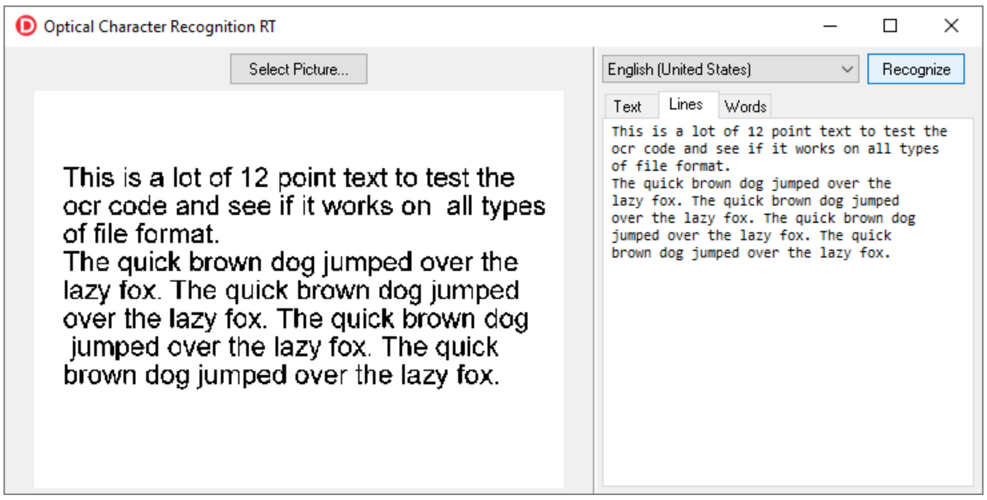
Future directions: License plate recognition is a rapidly evolving field, with ongoing research and development focused on improving accuracy, speed, and usability. Some future directions for LPR systems include the use of 3D imaging and lidar sensors to improve object detection and localization, the integration of LPR systems with autonomous vehicles for enhanced navigation and safety, and the development of more robust and efficient deep learning algorithms for license plate recognition.

Overall, license plate recognition is a complex computer vision application that requires a combination of object detection, localization, segmentation, and character recognition techniques. The accuracy of LPR systems depends on a variety of factors, such as the quality of the images or videos, the lighting and weather conditions, and the accuracy of the OCR algorithms. However, with advances in computer vision and machine learning, LPR systems have become increasingly accurate and reliable, making them a valuable tool for law enforcement, security, and transportation applications.

4.3.3 TECHNIQUE USED:

4.3.3.1 OCR:

OCR stands for Optical Character Recognition. It is a technology that is used to convert scanned images, PDFs, or other documents into editable and searchable text. OCR is an important tool for digitizing printed or handwritten text and making it accessible to computers. OCR technology works by analysing the scanned image of a document and recognizing individual characters. It then translates those characters into machine-readable text that can be edited, searched, and manipulated. OCR can also recognize and preserve formatting such as font size, style, and color. OCR is used in a variety of applications, including data entry, document management, and information retrieval. For example, OCR can be used to scan paper invoices and automatically extract data such as customer names, purchase orders, and billing amounts. It can also be used to digitize historical documents and make them available online for research and education purposes.



This is a lot of 12 point text to test the
ocr code and see if it works on all types
of file format.
The quick brown dog jumped over the
lazy fox. The quick brown dog jumped
over the lazy fox. The quick brown dog
 jumped over the lazy fox. The quick
brown dog jumped over the lazy fox.

```
This is a lot of 12 point text to test the
ocr code and see if it works on all types
of file format.
The quick brown dog jumped over the
lazy fox. The quick brown dog jumped
over the lazy fox. The quick brown dog
jumped over the lazy fox. The quick
brown dog jumped over the lazy fox.
```

Fig 4.6: Character extraction and printing

OCR technology has advanced significantly in recent years, with improved accuracy and the ability to recognize handwriting and even multi-lingual documents. OCR technology typically involves two steps: image pre-processing and character recognition. In the pre-processing step, the scanned image is cleaned up and enhanced to improve the accuracy of character recognition. This may involve removing noise or adjusting contrast and brightness. In the character recognition step, the OCR software analyses the pre-processed image and identifies individual characters using pattern recognition algorithms. OCR can be used to recognize both printed and handwritten text. However, recognizing handwriting can be more challenging than recognizing printed text, especially if the handwriting is messy or difficult to read. Some OCR software can recognize cursive handwriting and even different styles of handwriting, but the accuracy may not be as high as for printed text.

OCR technology is often integrated with other software applications, such as document management systems, content management systems, and electronic health record systems. This allows users to easily search for and retrieve information from scanned documents. There are many OCR software solutions available, ranging from free and open-source programs to commercial products. Some popular OCR software tools include Adobe Acrobat Pro, ABBYY FineReader, and Tesseract OCR. While OCR technology has many benefits, it is not always 100% accurate. Errors can occur if the scanned image is of poor quality, if the font is difficult to read, or if there are smudges or other markings on the page. In some cases, manual review and correction of OCR results may be necessary to ensure accuracy. OCR technology has been around since the 1930s, but it has become more sophisticated and widely used in recent years. Some of the factors driving the growth of OCR include the increasing digitization of information, the need for more efficient data entry and processing, and the desire to make historical documents more accessible to researchers and the general public. OCR can be used for a wide range of document types, including invoices, receipts, business cards, legal documents, and medical records. It can also be used to convert printed books and newspapers into electronic formats, which can be easily searched and accessed online. OCR software can be trained to recognize specific fonts or handwriting styles, which can improve the accuracy of character recognition. Some OCR software can also be customized to recognize specific types of documents or to extract specific types of data.

One of the challenges of OCR is dealing with variations in fonts, formatting, and language. For example, OCR software may have difficulty recognizing handwritten text that is stylized or has unusual letter shapes. It may also have difficulty recognizing text that is written in a language that is different from the language it was trained on. To improve the accuracy of OCR, it is important to ensure that the scanned image is of high quality and that the text is legible. This may involve using high-resolution scanners or adjusting the lighting and contrast of the scanned image. OCR technology relies on complex algorithms and artificial intelligence to analyze scanned images and recognize characters.

This process involves multiple steps, including image pre-processing, segmentation, feature extraction, and classification.

In the image pre-processing step, the scanned image is enhanced and cleaned up to remove any noise or distortions that could affect character recognition accuracy. This may involve adjusting the brightness and contrast of the image or removing background noise.

In the segmentation step, the image is divided into individual characters or words. This can be challenging if the text is touching or if there are other irregularities in the document.

In the feature extraction step, the software analyzes each character and extracts relevant features, such as the shape of the character, its height and width, and the presence of any lines or curves.

In the classification step, the software compares the extracted features to a set of predefined character models to determine the most likely character or word.

OCR software can be trained to recognize different fonts, languages, and writing styles. This typically involves feeding the software a large number of examples of the desired font or language and then using machine learning algorithms to identify patterns and improve recognition accuracy. OCR technology can be used in a variety of industries and applications, including finance, healthcare, legal, and education. For example, OCR can be used to extract information from invoices, medical records, or legal contracts, which can then be used for analysis or data entry.

OCR technology has some limitations and challenges. One of the main challenges is recognizing handwriting accurately, especially if the handwriting is cursive or hard to read. OCR software can also have difficulty with certain fonts, such as stylized or decorative fonts, and may require additional training or customization to recognize them accurately. OCR technology can be used to extract data from different types of documents, including text, images, and PDF files. It can also be used to recognize barcodes and other types of machine-readable codes. OCR software can be standalone programs that are installed on a computer or mobile device, or they can be cloud-based services that are accessed via the internet. Cloud-based OCR services can be particularly useful for businesses that need to process large volumes of documents quickly and efficiently. OCR software can also be used to convert scanned documents into different file formats, such as Microsoft Word or Excel, which can then be edited and manipulated like any other electronic document. OCR technology is constantly evolving and improving, thanks in part to advances in machine learning and artificial intelligence. For example, some OCR software can now recognize and analyze images and videos in addition to text, which has broadened its potential applications. OCR has also become more accessible and affordable in recent years, making it available to businesses and organizations of all sizes. Some OCR software is even available for free or as open-source projects, which can be particularly useful for smaller businesses or individuals.

Despite the advances in OCR technology, there are still some challenges and limitations to its use. For example, OCR may have difficulty recognizing text that is written in unusual fonts or languages, or that is distorted or blurry. Additionally, OCR may not be able to recognize handwritten text or characters that are not part of the standard alphabet or numerical system.

Here are some of the main advantages of using OCR:

- Increased efficiency: OCR can automate the process of data entry and document processing, which can save time and reduce errors. This can be particularly useful for businesses that need to process large volumes of documents quickly and efficiently.
- Improved accuracy: OCR technology has improved significantly in recent years, and it can now recognize text with high accuracy rates. This can reduce errors and ensure that data is entered correctly.
- Cost savings: By automating data entry and document processing, OCR can help businesses reduce the need for manual labor and other costly resources.
- Improved accessibility: OCR technology can be used to convert printed documents into electronic formats that are easier to search and access. This can be particularly useful for historical documents, rare books, and other materials that may not be widely available.

- Integration with other technologies: OCR can be integrated with other technologies, such as machine learning and artificial intelligence, to provide even more advanced data analysis and processing capabilities.
- Reduced storage space: OCR can convert large volumes of paper documents into digital formats, which can save physical storage space and reduce the need for paper-based storage.
- Environmental benefits: By reducing the need for paper-based document processing, OCR can help businesses reduce their environmental impact and promote sustainability.

Software used for OCR:

Tesseract OCR is an optical character reading engine developed by HP laboratories in 1985 and open sourced in 2005. Since 2006 it is developed by Google. Tesseract has Unicode (UTF-8) support and can recognize more than 100 languages “out of the box” and thus can be used for building different language scanning software also. Latest Tesseract version is Tesseract 4. It adds a new neural net (LSTM) based OCR engine which is focused on line recognition but also still supports the legacy Tesseract OCR engine which works by recognizing character patterns.

Generally, OCR works as follows:

- Pre-process image data, for example: convert to gray scale, smooth, de-skew, filter.
- Detect lines, words and characters.
- Produce ranked list of candidate characters based on trained data set. (Here the setDataPath() method is used for setting path of trainer data)
- Post process recognized characters, choose best characters based on confidence from previous step and language data. Language data includes dictionary, grammar rules, etc.

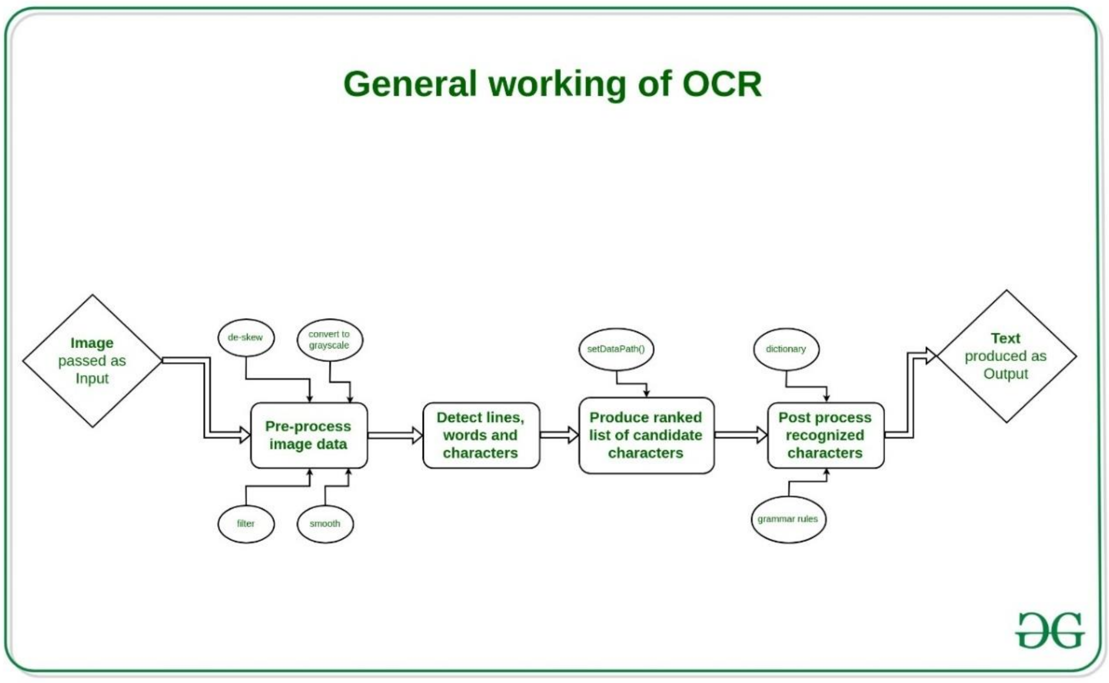


Fig 4.7: Architecture of OCR

DISADVANTAGES:

- The OCR is limited to language recognition.
- There is lot of effort that is required to make trainer data of different languages and implement that.
- One also need to do extra work on image processing as it is the most essential part that really matters when it comes to the performance of OCR.
- After doing such a great amount of work, no OCR can offer an accuracy of 100% and even after OCR we have to determine the unrecognized character by neighbouring methods of machine learning or manually correct it.

4.3.3.2 CNN:

Convolutional Neural Networks: These are commonly used in image and video processing tasks. They consist of convolutional layers that apply filters to the input data, pooling layers that downsample the data, and fully connected layers that produce the final output.

The main feature of a CNN is the convolutional layer. This layer applies a set of learnable filters to the input image or data, allowing the network to identify and extract important features from the image. The convolutional layer slides the filter over the input image, performing element-wise multiplication and summation to produce a feature map. CNNs also typically include pooling layers, which downsample the feature maps by taking the maximum value or average value of a set of adjacent pixels. This reduces the spatial size of the data, allowing the network to process larger images without running into memory issues. Finally, CNNs include fully connected layers, which perform the final classification or regression task. These layers take in the output from the previous layers and produce the final output.

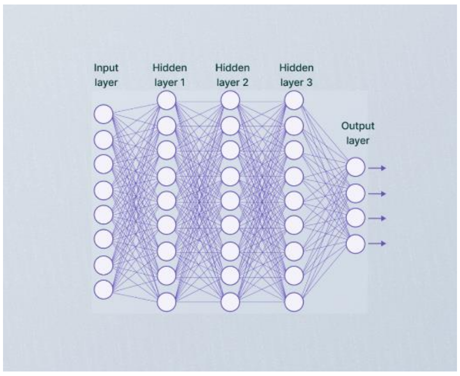


Fig 4.8: Processing of CNN

The advantage of using a CNN over a traditional neural network is that the convolutional layers are able to identify important spatial features in the data, such as edges and patterns, and are better suited for image and video processing tasks. CNNs have been used for a wide range of applications, including object recognition, image segmentation, and even natural language processing.

Convolutional Layers: As mentioned earlier, convolutional layers are the main feature of CNNs. The filters applied in the convolutional layers are also known as "kernels" or "feature detectors", and they are learned through the training process. The number of filters in a layer is determined by the hyperparameters of the network.

Activation Functions: Activation functions are applied after the convolutional layers to introduce nonlinearity into the network. The most common activation function used in CNNs is the Rectified Linear Unit (ReLU), which sets all negative values to zero.

Pooling Layers: Pooling layers are used to reduce the size of the feature maps while retaining the most important information. The most common type of pooling is max pooling, which takes the maximum value of a set of adjacent pixels. Other types of pooling include average pooling and L2 pooling.

Dropout: Dropout is a regularization technique used in CNNs to prevent overfitting. During training, random nodes in the network are dropped out with a specified probability, forcing the network to learn more robust features.

Transfer Learning: Transfer learning is a technique where a pre-trained CNN is used as a starting point for a new task. The pre-trained CNN is typically trained on a large dataset such as ImageNet, and the features learned in that network can be transferred to a new task with smaller amounts of training data.

Data Augmentation: Data augmentation is another technique used to prevent overfitting in CNNs. It involves applying random transformations to the input data such as rotation, translation, and scaling, effectively increasing the size of the training dataset.

Stride: The stride of a convolutional layer determines the step size of the filter as it moves across the input data. A stride of 1 means that the filter moves one pixel at a time, while a stride of 2 means that the filter skips every other pixel. Stride can be used to control the size of the output feature maps.

Padding: Padding can be used in convolutional layers to preserve the size of the input data. When the filter is applied to the input data, the edges of the input may be lost due to the filter size. Padding adds additional pixels around the edge of the input data, allowing the filter to move over the entire input without losing any pixels.

Backpropagation: CNNs are trained using backpropagation, which involves calculating the gradient of the loss function with respect to the weights of the network. The gradients are then used to update the weights in the opposite direction of the gradient, reducing the loss over time.

Architecture: There are many different architectures of CNNs, each with their own strengths and weaknesses. Some popular architectures include AlexNet, VGG, ResNet, and Inception. These architectures vary in terms of the number of layers, the number of filters, and the size of the filters used.

Interpretability: One of the challenges with CNNs is that they are often referred to as "black boxes", as it can be difficult to understand how the network arrived at its predictions. Recent research has focused on developing techniques for interpreting the features learned by CNNs, such as visualizing the activations of specific neurons in the network.

Convolutional Autoencoder: A convolutional autoencoder is a type of CNN that is used for unsupervised learning tasks such as image denoising or compression. It consists of an encoder network that compresses the input image into a lower-dimensional representation and a decoder network that reconstructs the original image from the compressed representation.

Object Detection: CNNs are commonly used for object detection tasks, where the goal is to identify the location and class of objects in an image. One popular approach is the region-based CNN (R-CNN), which uses a combination of region proposals and CNNs to classify objects in the proposed regions.

Semantic Segmentation: Semantic segmentation is the task of assigning a semantic label to each pixel in an image. CNNs have been used for semantic segmentation tasks by predicting a class label for each pixel in the image.

Style Transfer: Style transfer is the task of transferring the style of one image onto another image. CNNs have been used for this task by defining a loss function that encourages the output image to match the content of one image and the style of another image.

Recurrent CNNs: Recurrent CNNs (RCNNs) are a type of CNN that includes recurrent connections, allowing the network to process sequences of inputs such as time-series data or natural language text.

Adversarial Attacks: Adversarial attacks are a type of attack on CNNs that involves adding small, imperceptible perturbations to the input data, causing the network to misclassify the input. Adversarial attacks have been used to highlight the vulnerabilities of CNNs and to develop more robust networks.

Transfer Learning for Natural Language Processing: Transfer learning has also been applied to natural language processing (NLP) tasks using CNNs. Pre-trained CNNs are used as a starting point for new NLP tasks, allowing the network to learn important features of the input text.

Overall, CNNs are a powerful tool for machine learning tasks involving visual data, and they have been used for a wide range of applications in fields such as computer vision, natural language processing, and speech recognition. While there are many challenges associated with training and interpreting CNNs, their ability to automatically learn features from the input data has made them a key component of many state-of-the-art machine learning systems.

Examples:

Image Recognition: One of the most common applications of CNNs is in image recognition. For example, CNNs have been used to identify objects in photos for social media platforms, to recognize faces in security systems, and to assist in medical diagnosis by identifying anomalies in medical images. For instance, a CNN-based system called RetinaNet is used for object detection in autonomous driving, identifying pedestrians, vehicles, and other objects on the road.

Autonomous Driving: CNNs are also used in autonomous driving systems. They help in object detection, lane detection, and object tracking, and are used to train self-driving cars to make decisions based on the visual input received from cameras mounted on the car. For example, Tesla's Autopilot system uses a CNN-based object detection system for detecting pedestrians, cyclists, and other objects on the road.

Augmented Reality: CNNs have also been used in augmented reality (AR) applications, such as Snapchat filters and Pokemon Go. In these applications, CNNs are used to recognize and track the faces and body movements of users, enabling the system to apply AR effects in real-time.

Natural Language Processing: CNNs have also been applied to natural language processing (NLP) tasks, such as text classification and sentiment analysis. For example, CNNs can be used to classify text messages as spam or not spam, or to identify the sentiment expressed in a piece of text.

Healthcare: CNNs have been used in healthcare applications for tasks such as medical image analysis, disease diagnosis, and drug discovery. For example, CNNs can be used to analyze X-rays and CT scans to detect diseases such as pneumonia or cancer. Additionally, CNNs can be used to discover new drugs by predicting the activity of molecules and identifying potential drug targets.

Gaming: CNNs have also been used in gaming applications, such as creating agents that can play games like Atari or Go. For example, a CNN-based agent called AlphaGo was developed by Google's DeepMind to play the ancient Chinese board game Go, and it was able to beat the world's best human player in a series of matches.

Advantages:

Here are some advantages of using convolutional neural networks (CNNs):

High Accuracy: CNNs are highly accurate and have achieved state-of-the-art performance on a wide range of image and video recognition tasks. They are able to recognize patterns and features in visual data that are difficult for traditional machine learning algorithms to detect.

Robust to Noise: CNNs are robust to noise and variations in input data, making them suitable for use in real-world applications where data is often noisy or incomplete. They are able to learn invariant features that are robust to variations in lighting, angle, and other factors that may affect the appearance of visual data.

Feature Extraction: CNNs are able to extract useful features automatically from the input data, eliminating the need for manual feature engineering. This makes them highly effective at tasks such as object detection and recognition, where identifying and extracting relevant features is critical.

Transfer Learning: CNNs can be trained on large datasets and then fine-tuned for specific tasks, allowing for efficient transfer learning.

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 SYSTEM ARCHITECTURE:

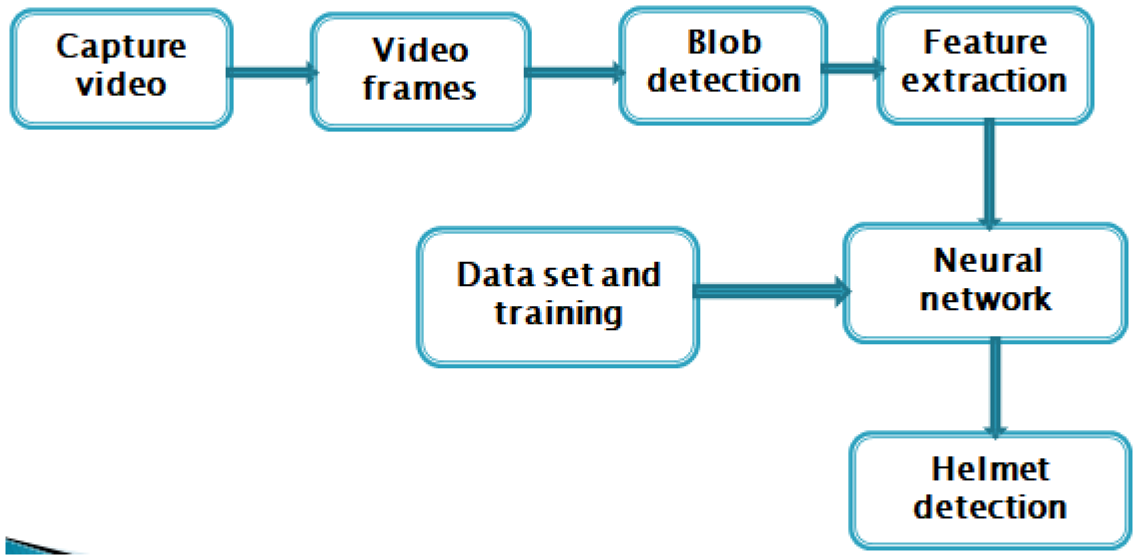


Fig 5.1: System architecture of Helmet and number plate detection

5.1.1: VIDEO STREAMING:

Video streaming technology is one way to deliver video over the Internet. Using streaming technologies, the delivery of audio and video over the Internet can reach many millions of customers using their personal computers, PDAs, mobile smartphones or other streaming devices. The reasons for video streaming technology growth are:

- Broadband networks are being deployed
- Video and audio compression techniques are more efficient
- Quality and variety of audio and video services over internet are increasing

There are two major ways for the transmission of video/audio information over the Internet:

Download mode: The content file is completely downloaded and then played. This mode requires long downloading time for the whole content file and requires hard disk space.

Streaming mode: The content file is not required to be downloaded completely and it is playing while parts of the content are being received and decoded.
By taking input from the video streaming given to pre-processing conversion. It converts RGB to GRAY scale image to Black and white image.
Pre-processing is a common name for operations with images at the lowest level of abstraction -- both input and output are intensity images.o The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important.

5.1.2 DATASET AND TRAINING:

Data set training is a process in which a machine learning model is trained on a set of labeled data to learn patterns and relationships between different features of the data. In the context of helmet and number plate detection, data set training involves feeding the machine learning model with a large dataset of images containing helmets and number plates along with their respective labels, indicating whether a helmet or number plate is present or not in each image.
The process of training a machine learning model typically involves several steps:
Data collection: Collect a large number of images of helmets and number plates in different conditions, such as varying lighting, angles, and distances.
Data labeling: Each image in the dataset should be labeled as either containing a helmet or a number plate, or not containing any of them.
Preprocessing: Images may be preprocessed to normalize the lighting, adjust the color balance, and remove noise or other artifacts that may affect the accuracy of the detection.
Training: The labeled data is fed into a machine learning algorithm, such as a convolutional neural network (CNN), which learns to detect helmets and number plates based on the features extracted from the images.
Testing and evaluation: The trained model is tested on a separate set of images to evaluate its accuracy in detecting helmets and number plates. The performance

of the model can be evaluated using metrics such as precision, recall, and F1 score.

Fine-tuning: The model may be further fine-tuned with additional data or changes to the model architecture to improve its performance.

Once the model is trained, it can be used to detect helmets and number plates in new images or videos by analyzing the features in the images and comparing them to the patterns learned during training.

5.1.3 HELMET DETECTION:

The process of detecting the helmet are as follows

Image acquisition: The first step is to acquire an image or video stream from a camera or other sensor.

Preprocessing: The image may be preprocessed to enhance the features of the helmet and remove any unwanted noise or artifacts. This may involve operations such as resizing, normalization, and filtering.

Object detection: The preprocessed image is analyzed using a machine learning algorithm, such as a convolutional neural network (CNN), to detect the presence and location of helmets. The CNN extracts features from the image and uses them to classify regions of the image as containing a helmet or not.

Non-maximum suppression: If multiple regions of the image are classified as containing a helmet, a non-maximum suppression algorithm may be used to eliminate redundant detections and select the most likely regions.

Display or action: The detected helmet regions can be displayed to the user or used to trigger an action, such as sounding an alarm or sending an alert.

5.2 UML DIAGRAMS:

5.2.1 USE CASE DIAGRAM:

Use case diagrams are considered for high level requirement analysis of a system. A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. Use cases once specified can be denoted both textual and visual representation (i.e., use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour.

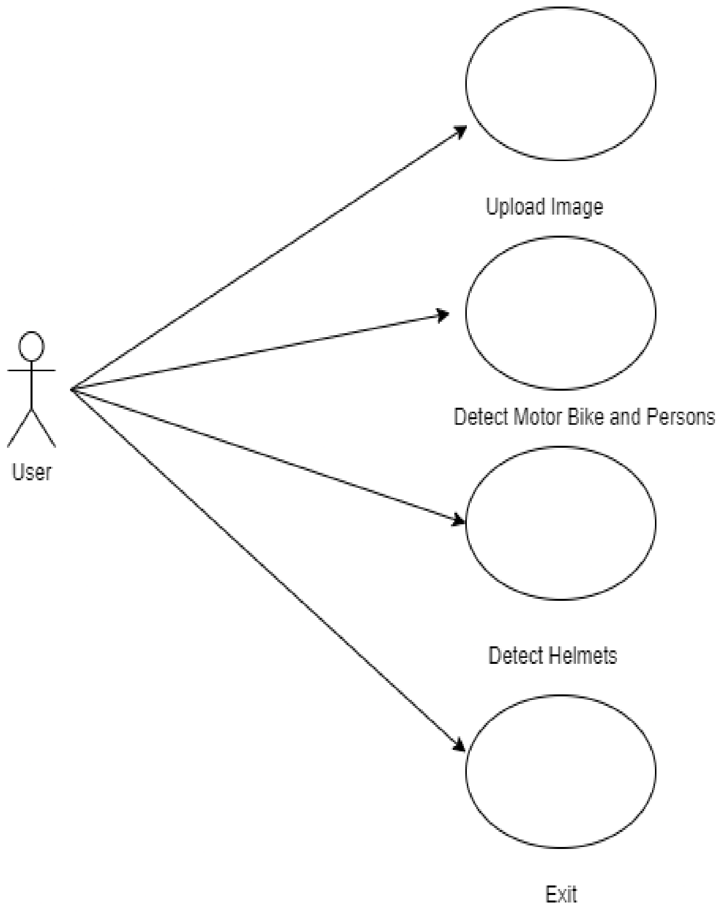


Fig.5.2 Use Case Diagram

5.2.2 CLASS DIAGRAM:

Class diagrams are one of the most useful types of diagrams in UML as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects. With our UML diagramming software, creating these diagrams is not as overwhelming as it might appear.

The system consists of four main classes: Patient, Doctor, Caregiver, and Assessment. The Patient class represents a person with dementia. It has attributes such as id, name, birthdate, gender, address, phone, diagnosis, and medications. It also has a method to calculate the patient's age based on their birthdate.

The Doctor class represents a medical professional who can diagnose and prescribe medication for patients. It has attributes such as id, name, and specialty. It has two methods: prescribeMedication() and diagnose(). The Caregiver class represents a person who provides support and care for a patient with dementia. It has attributes such as id, name, phone, email, relationship, and availability.

It has a method called provideSupport() that can be used to provide assistance to the patient. The Assessment class represents an evaluation of the patient's cognitive and functional abilities. It has attributes such as id, date, patient, doctor, score, and comments. It has a method to retrieve the score of the assessment. The Medication class represents a drug that can be prescribed to a patient. It has attributes such as id, name, dosage, frequency, and side effects. It has a method to retrieve the medication's name.

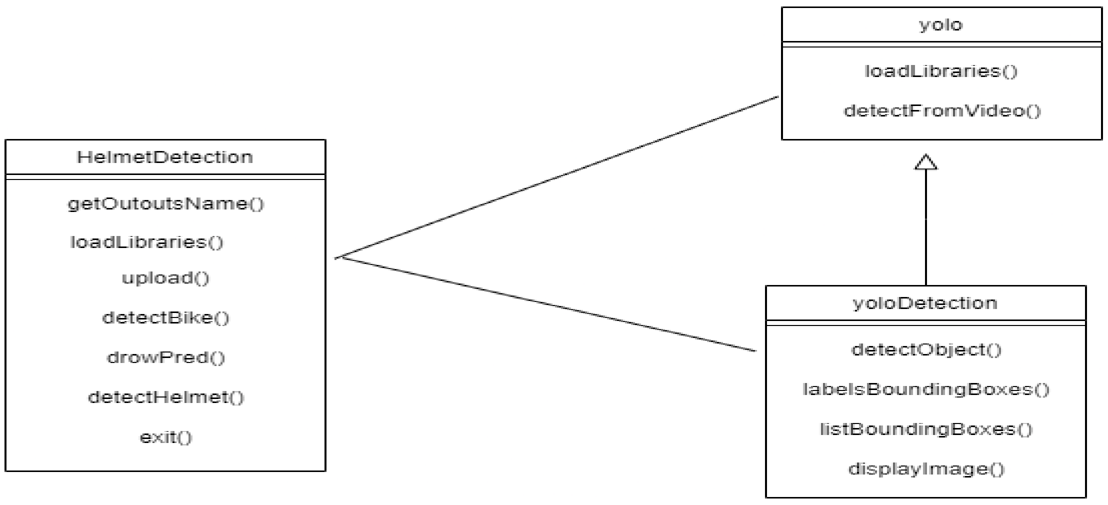


Fig.5.3 Class Diagram

5.2.3 ENTITY RELATIONSHIP DIAGRAM:

An entity relationship diagram (ERD) is a graphical representation of entities and their relationships to each other. Here's a short summary of an ERD for helmet and number plate recognition:

Entities:

1. Image: This entity represents the image captured by a camera that is used for helmet and number plate recognition.
2. Helmet: This entity represents the helmet worn by a person riding a two-wheeler.
3. Number Plate: This entity represents the number plate on a vehicle.
4. Camera: This entity represents the camera used for helmet and number plate recognition.
5. System: This entity represents the system used for processing the image captured by the camera.

Relationships:

1. An image is captured by a camera.
2. The system processes the image to identify the helmet and the number plate.
3. The helmet and the number plate are identified by the system based on the image.
4. The image, the helmet, and the number plate are related to each other through the camera and the system.

Overall, the ERD shows how the different entities are related to each other in the context of helmet and number plate recognition.

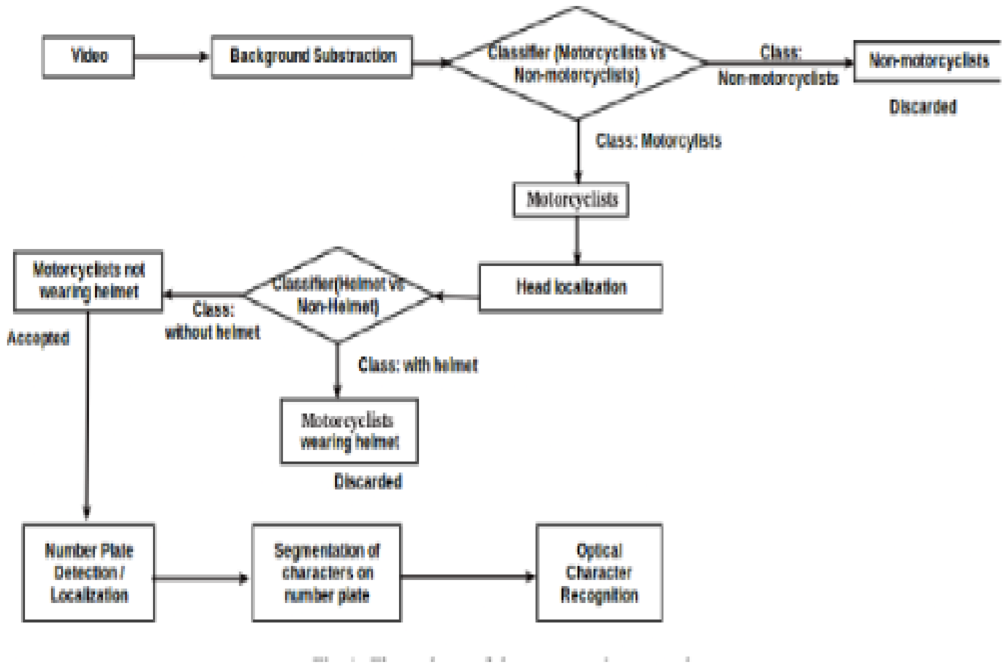

Video
Background Substraction
Classifier (Motorcyclists vs Non-motorcyclists)
Class: Non-motorcyclists
Non-motorcyclists
Discarded
Class: Motorcylists
Motorcyclists
Head localization
Classifier(Helmet vs Non-Helmet)
Class: without helmet
Motorcyclists not wearing helmet
Accepted
Class: with helmet
Motorcyclists wearing helmet
Discarded
Number Plate Detection / Localization
Segmentation of characters on number plate
Optical Character Recognition

Fig.5.4 Entity Relationship Diagram

5.2.4 DATA FLOW DIAGRAM:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

SYMBOLS AND NOTATIONS USED IN DFDS

Three common systems of symbols are named after their creators:

- Yourdon and Coad
- Yourdon and DeMarco
- Gane and Sarson

One main difference in their symbols is that Yourdon-Coad and Yourdon-DeMarco use circles for processes, while Gane and Sarson use rectangles with rounded corners, sometimes called lozenges. There are other symbol variations in use as well, so the important thing to keep in mind is to be clear and consistent in the shapes and notations you use to communicate and collaborate with others.

Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

- **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
- **Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as "Submit payment."
- **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."

- **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

LEVEL 0:

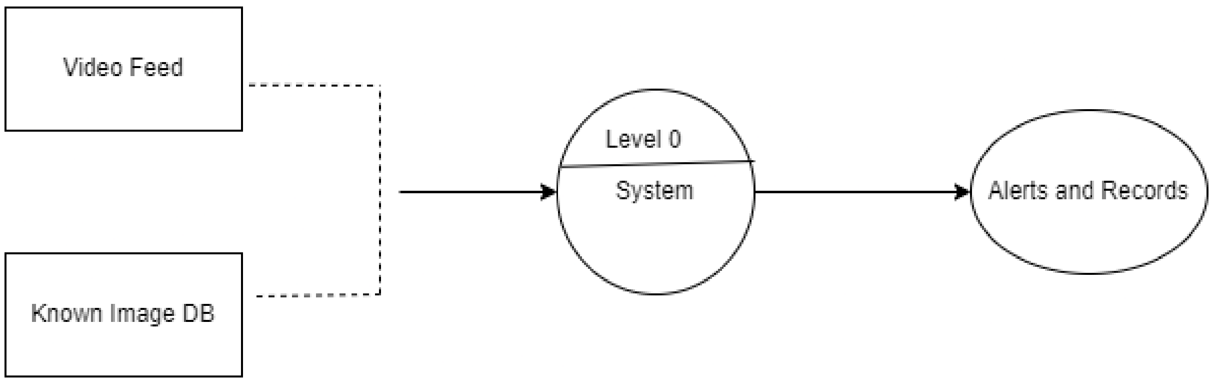


Fig.5.5 Level 0 DFD Diagram

LEVEL 1:

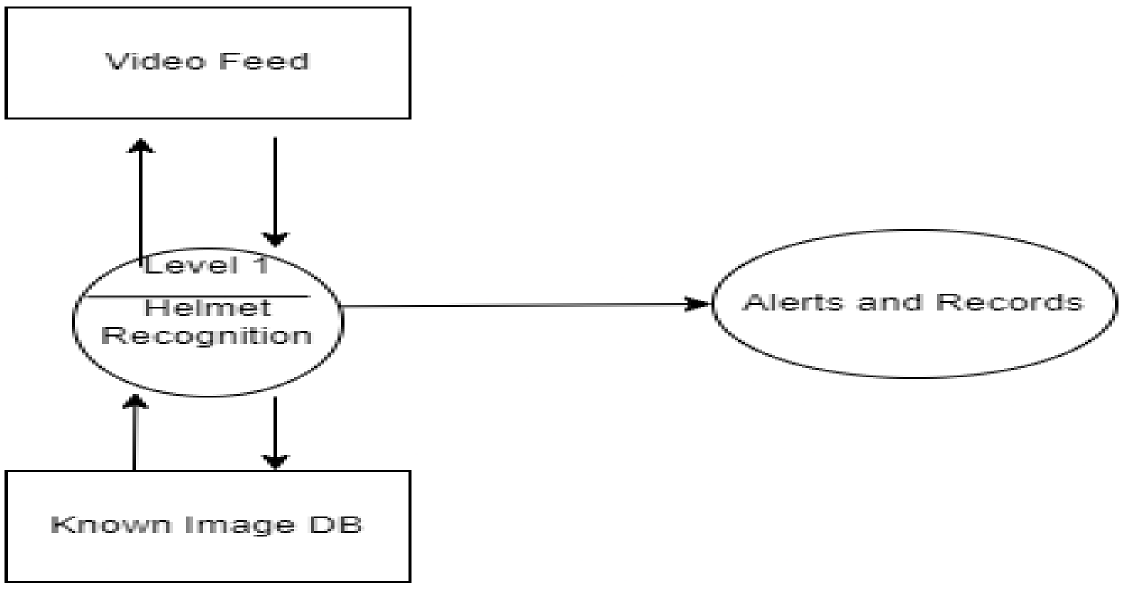


Fig.5.6 Level 1 DFD Diagram

LEVEL 2:

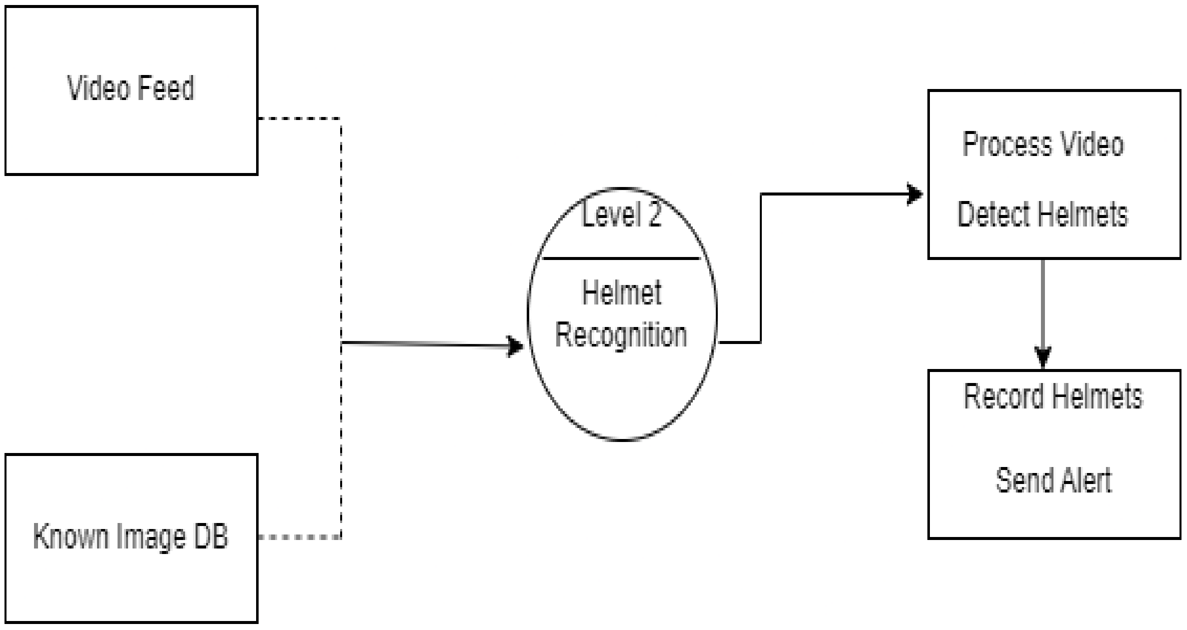


Fig.5.7 Level 2 DFD Diagram

5.3 FEASIBILITY

- **Social Feasibility:** One of the key factors affecting the social feasibility of these systems is the perception of privacy. Some people may feel that the constant monitoring of their vehicles and their activities on the road is an invasion of their privacy. This can lead to resistance and reluctance to accept these systems. Another factor that can affect the social feasibility of these systems is their accuracy and reliability. If the systems generate too many false positives or negatives, or if they malfunction frequently, they can lead to frustration and a loss of trust in the technology.
- **Technical Feasibility:** One of the key technical considerations for these systems is the accuracy of the detection algorithms. The system must be able to accurately identify the presence or absence of a helmet or number plate in various lighting and environmental conditions. This requires sophisticated computer vision algorithms and image processing techniques

that can analyze the images or videos captured by the cameras in real-time. Another technical consideration for these systems is the hardware and infrastructure requirements. This includes the cameras, processing hardware, and network connectivity needed to capture and analyze the images or videos. The system must be designed to handle large volumes of data and process it quickly to minimize delays in detecting violations.

- **Economic Feasibility:** One of the key economic considerations for these systems is the cost of hardware and infrastructure. This includes the cameras, processing hardware, and network connectivity needed to capture and analyze the images or videos. The cost of these components can vary significantly depending on the quality and specifications, and it is essential to select the appropriate hardware that meets the required standards and functionality while keeping costs reasonable.

CHAPTER 6

MODULES

The project consist of two modules. They are

- Helmet Detection
- Number plate identification

6.1 HELMET DETECTION:

The process of detection of helmet are as follows

- Data collection and labelling: The accuracy of the helmet detection system is highly dependent on the quality and quantity of the training data. The training dataset should contain a wide variety of images of helmets in different lighting conditions, camera angles, and backgrounds. Each image should be labelled to indicate the location of the helmet in the image.
- Machine learning algorithm: Convolutional neural networks (CNNs) are a popular choice for object detection tasks like helmet detection. A CNN is trained on the labelled dataset to learn features that are characteristic of helmets. During inference, the CNN analyses the features of an input image and outputs a set of bounding boxes and confidence scores, which indicate the location and probability of a helmet in the image.
- Pre-processing: The input images are pre-processed to enhance the quality of the image and reduce noise. This may involve operations such as resizing, cropping, and adjusting the contrast and brightness.
- Non-maximum suppression: Since a CNN can produce multiple bounding boxes for the same object, a non-maximum suppression algorithm is used to select the most likely detection and suppress redundant detections.
- Postprocessing: After the detection step, additional postprocessing steps may be applied to improve the accuracy of the detection. For example, morphological operations may be used to fill in gaps in the detection, or a

Kalman filter may be used to track the helmet over multiple frames of a video.

- Integration with other systems: The helmet detection system may be integrated with other systems such as a video recording system, an alarm system, or a notification system. For example, if the detection system detects a helmet violation, it can trigger an alarm and send a notification to the appropriate authorities.
- Training the CNN: The CNN is trained using a technique called backpropagation, which involves adjusting the weights of the network based on the difference between the predicted output and the true output. The training process typically involves several epochs of iteration over the entire training dataset.
- Data augmentation: To increase the size and diversity of the training dataset, data augmentation techniques such as flipping, rotating, and adding noise to the images can be used. This helps the CNN to learn features that are invariant to changes in the image, such as the orientation of the helmet.
- Hyperparameter tuning: The performance of the CNN is affected by several hyperparameters, such as the number of layers, the size of the filters, and the learning rate. These hyperparameters can be tuned using techniques such as grid search or Bayesian optimization to find the optimal set of parameters that maximize the accuracy of the helmet detection system.
- False positive and false negative reduction: Even with a well-trained CNN, there may still be cases where the system produces false positives (detecting a helmet where there is none) or false negatives (failing to detect a helmet that is present). These errors can be reduced by adjusting the threshold for the confidence score, applying postprocessing techniques, or

incorporating additional information such as the color or shape of the helmet.

- Real-time performance: In some applications, such as helmet detection for motorcycle riders, it is important for the detection system to operate in real-time. This requires optimizing the CNN and the postprocessing steps to minimize the computation time, as well as using specialized hardware such as GPUs or FPGAs to accelerate the processing.

Overall, the process of detecting helmets requires a combination of domain knowledge, machine learning expertise, and software engineering skills to build an accurate and reliable system that can operate in a variety of environments and conditions.

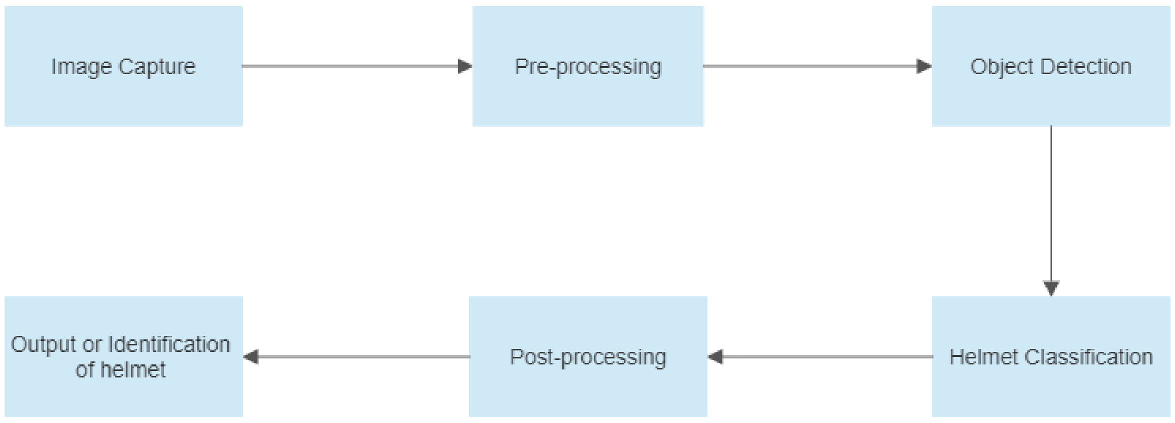


Fig 6.1: Architecture diagram of Helmet detection

Accuracy calculation of a person wearing helmet is as follows

In general, accuracy is a measure of how well a classification model correctly identifies positive and negative cases. In this case, positive cases could be defined as instances where a person is wearing a helmet, and negative cases could be defined as instances where a person is not wearing a helmet. Assuming you have data on the number of instances of wearing and not wearing a helmet, as well as the accuracy of a classification model that predicts whether a person is wearing a helmet or not, you could use the following formula to calculate accuracy

Accuracy = (True Positive + True Negative) / Total Population (1)

True Positive (TP) = the number of instances where the model correctly predicts that a person is wearing a helmet

True Negative (TN) = the number of instances where the model correctly predicts that a person is not wearing a helmet

Total Population = the total number of instances of wearing and not wearing a helmet

For example, if you have a dataset of 100 instances where 70 people are wearing a helmet and 30 people are not wearing a helmet, and your model correctly predicts 60 instances of wearing a helmet and 25 instances of not wearing a helmet, then you can calculate accuracy as:

Accuracy = (60 + 25) / 100 = 0.85 or 85% (2)

This means that your model is 85% accurate in predicting whether a person is wearing a helmet or not.

6.2 NUMBER PLATE IDENTIFICATION:

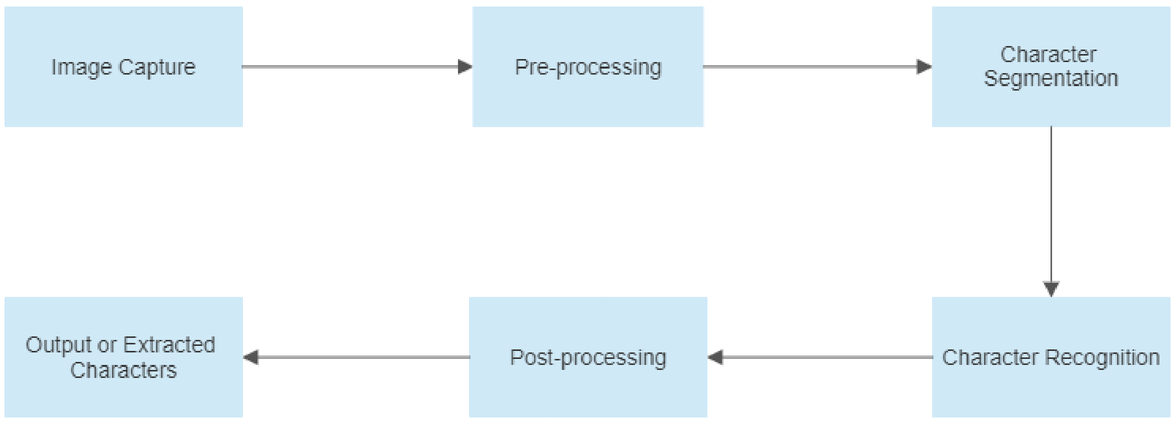


Fig 6.2: Architecture diagram of Number plate identification

The process of identifying a number plate in a vehicle typically involves the following steps:

- Image acquisition: The first step is to acquire an image of the vehicle's number plate. This may be done using a camera mounted on a fixed structure, such as a toll booth or parking lot entrance, or using a mobile device such as a smartphone.
- Pre-processing: The image is pre-processed to enhance the features of the number plate and remove any unwanted noise or artifacts. This may involve operations such as resizing, normalization, and filtering.
- Character segmentation: The pre-processed image is analysed using a segmentation algorithm to isolate each individual character in the number plate. This step is important because it allows the characters to be analysed separately, making the recognition task easier.
- Character recognition: The segmented characters are analysed using a character recognition algorithm to determine the identity of each character. This may involve using techniques such as optical character recognition (OCR), template matching, or neural networks.
- Postprocessing: After the character recognition step, additional postprocessing steps may be applied to improve the accuracy of the recognition. For example, a spell-checking algorithm may be used to correct any misspelled words, or a voting algorithm may be used to select the most likely character in cases where multiple characters are recognized.
- Output or action: The recognized number plate can be displayed to the user or used to trigger an action, such as opening a gate or generating a parking ticket.

The performance of the number plate identification system can be evaluated using metrics such as accuracy, precision, and recall, which measure the correctness and completeness of the recognition.

CHAPTER 7

IMPLEMENTATION

7.1 APPENDIX - I :

CODING:

```
import cv2
import numpy as np
import os
import imutils
from tensorflow.keras.models import load_model
import time
from numberPlate import *
import smtplib
import base64
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import sys
import pytesseract
gmail_user = "murugancharan2002@outlook.com"
gmail_pwd = "Charan2002@"
FROM = 'murugancharan2002@outlook.com'
TO = ['gokulp1712002@outlook.com'] #must be a list
def mail():
      msg = MIMEMultipart()
    time.sleep(1)
    msg['Subject'] ="SECURITY"

    #BODY with 2 argument
    #body=sys.argv[1]+sys.argv[2]
##    body='no helmet'
    body=text
    msg.attach(MIMEText(body,'plain'))
    time.sleep(1)
    ###IMAGE
    fp = open("1.jpg", 'rb')
    time.sleep(1)
    img = MIMEImage(fp.read())
    time.sleep(1)
    fp.close()
    time.sleep(1)
    msg.attach(img)
    time.sleep(1)
    try:
        server = smtplib.SMTP("smtp.office365.com", 587) #or port 465 doesn't
seem to work!
        print ("smtp.office365")
        server.ehlo()
        print ("ehlo")
        server.starttls()
        print ("starttls")
        server.login(gmail_user, gmail_pwd)
        print ("reading mail & password")
        server.sendmail(FROM, TO, msg.as_string())
        print ("from")
        server.close()
        print ('successfully sent the mail')
    except:
        print ("failed to send mail")

os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'

net = cv2.dnn.readNet("yolov3-custom_7000.weights", "yolov3-custom.cfg")
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
model = load_model('helmet-nonhelmet_cnn.h5')
print('model loaded!!!')
cap = cv2.VideoCapture("video.mp4")    #input video
COLORS = [(0,255,0),(0,0,255)]
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
fourcc = cv2.VideoWriter_fourcc(*"XVID")
writer = cv2.VideoWriter('output.avi', fourcc, 5,(888,500))#output video
def helmet_or_nohelmet(helmet_roi):
        try:
                helmet_roi = cv2.resize(helmet_roi, (224, 224))
                helmet_roi = np.array(helmet_roi,dtype='float32')
                helmet_roi = helmet_roi.reshape(1, 224, 224, 3)
                helmet_roi = helmet_roi/255.0
                return int(model.predict(helmet_roi)[0][0])
        except:
                        pass
```

```
ret = True

while ret:
    ret, img = cap.read()
    img = imutils.resize(img,height=500)
    # img = cv2.imread('test.png')
    height, width = img.shape[:2]
    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)
    confidences = []
    boxes = []
    classIds = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.3:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)

                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                classIds.append(class_id)
                #print(classIds.append(class_id))
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
    for i in range(len(boxes)):
        if i in indexes:
            x,y,w,h = boxes[i]
            color = [int(c) for c in COLORS[classIds[i]]]
            # green --> bike
            # red --> number plate
            if classIds[i]==0: #bike
                helmet_roi = 
img[max(0,y):max(0,y)+max(0,h)//4,max(0,x):max(0,x)+max(0,w)]
            else: #number plate
                x_h = x-60
                y_h = y-350
                w_h = w+100
                h_h = h+100
                cv2.rectangle(img, (x, y), (x + w, y + h), color, 7)
                # h_r = img[max(0,(y-330)):max(0,(y-330 + h+100)) , max(0,(x-
80)):max(0,(x-80 + w+130))]
                if y_h>0 and x_h>0:
                    h_r = img[y_h:y_h+h_h , x_h:x_h +w_h]
                    c = helmet_or_nohelmet(h_r)
                    cv2.putText(img,['helmet','no-helmet'][c],(x,y-
100),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),2)
                    cv2.rectangle(img, (x_h, y_h), (x_h + w_h, y_h + h_h),(255,0,0), 
10)
```

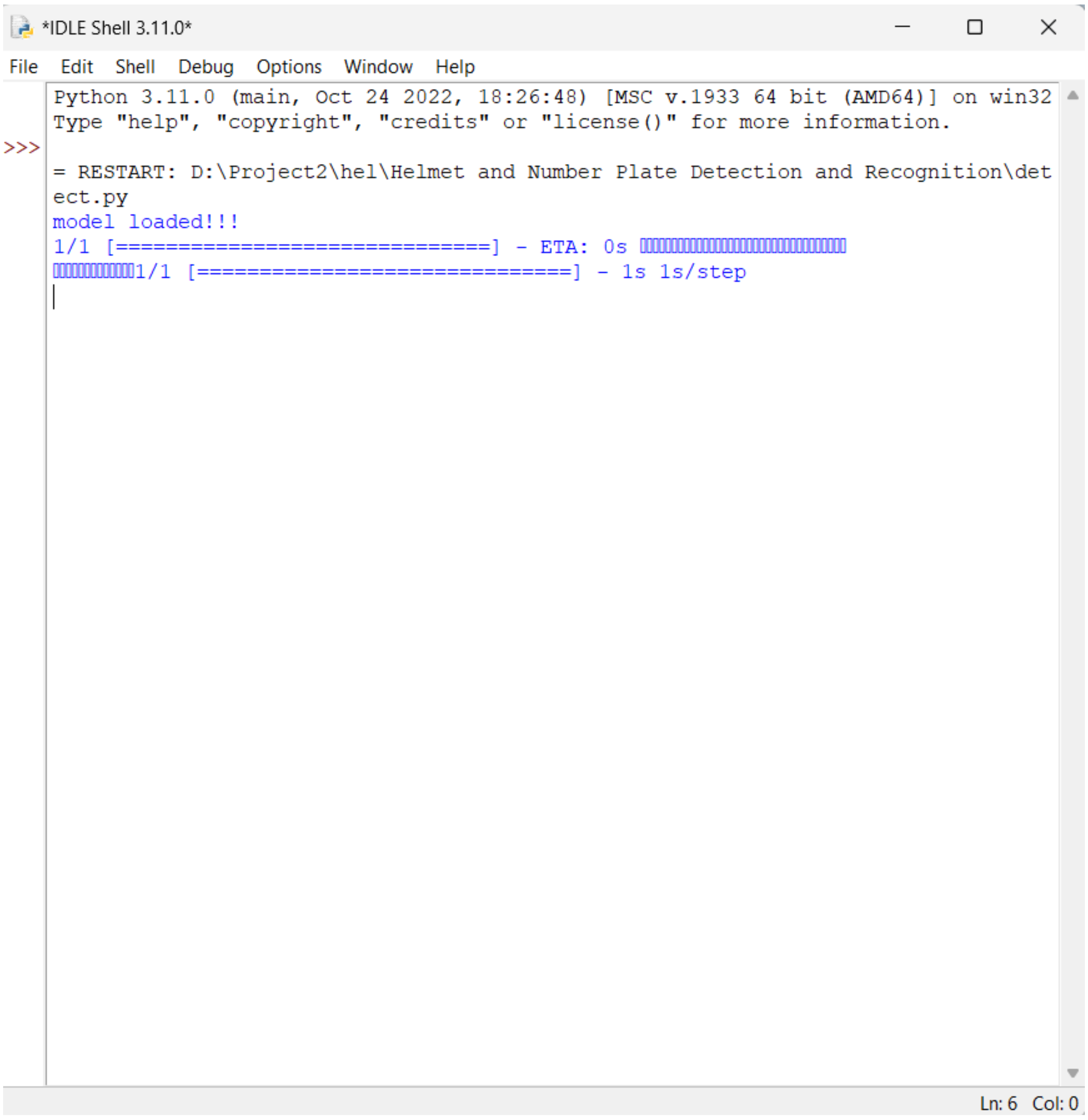
```
                    if classIds[i]==1:
                        plate_img = img[y:y+h, x:x+w]
                        cv2.imwrite('1.jpg', plate_img)
                        text=number_plate();
##                          mail()
                        # Load the saved image
                        img = cv2.imread('1.jpg')

                        # Convert the image to grayscale
                        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                        # Apply thresholding to improve OCR accuracy
                        thresholded = cv2.threshold(gray, 0, 255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
                        # Apply noise reduction to improve OCR accuracy
                        denoised = cv2.fastNlMeansDenoising(thresholded, h=10)
                        pytesseract.pytesseract.tesseract_cmd = r"C:\Program
Files\Tesseract-OCR\tesseract.exe"
                        mail()
    writer.write(img)
    cv2.imshow("Image", img)
    if cv2.waitKey(1) == 27:
        break
writer.release()
cap.release()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

7.2 APPENDIX - II:

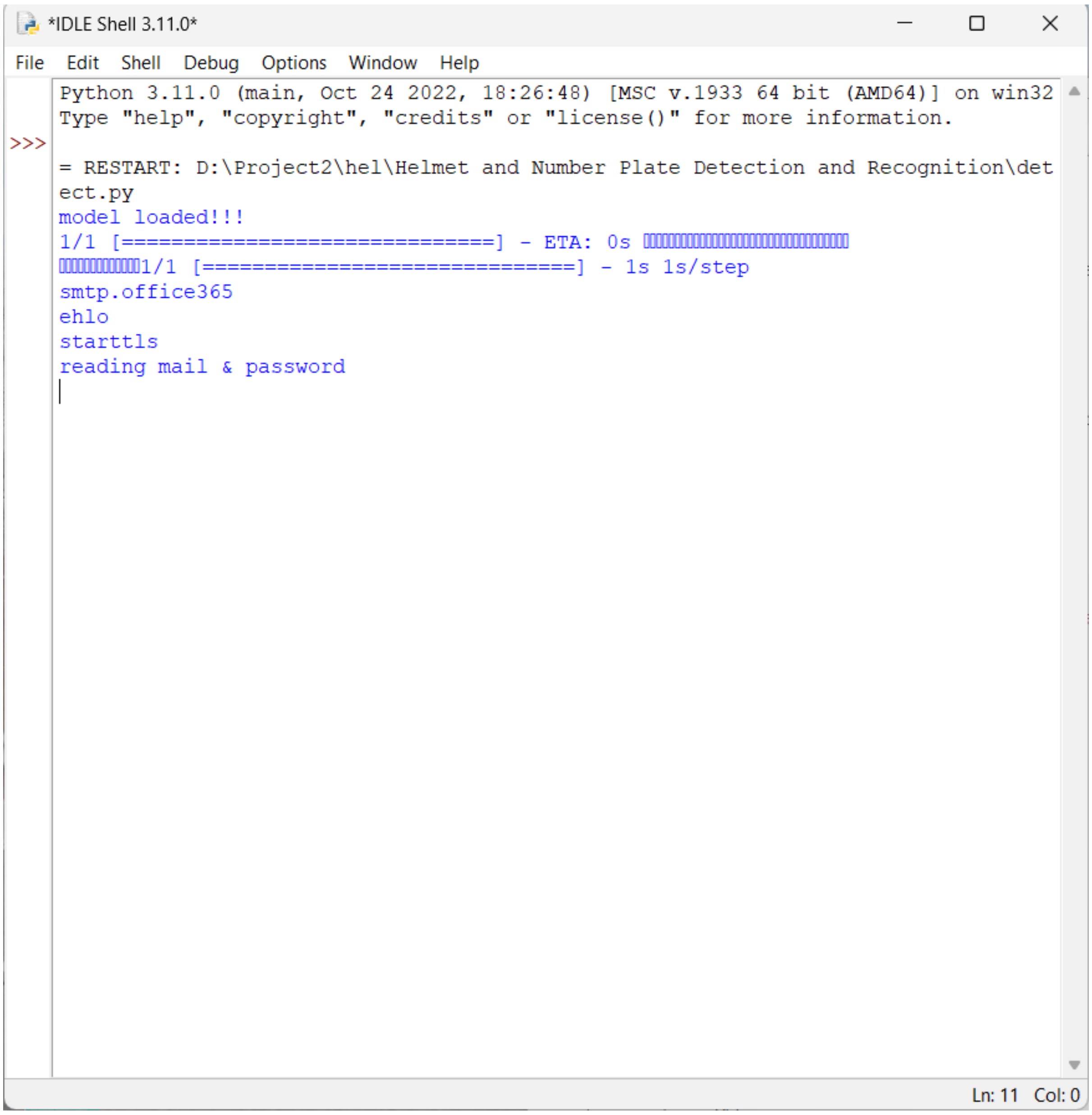
OUTPUT:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Project2\hel\Helmet and Number Plate Detection and Recognition\det
ect.py
model loaded!!!
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 1s 1s/step
```

Fig 7.1: Module loaded

- This is the first step of the project.
- All the models implemented in the project is loaded in the python interface.



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Project2\hel\Helmet and Number Plate Detection and Recognition\det
ect.py
model loaded!!!
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 1s 1s/step
smtp.office365
ehlo
starttls
reading mail & password
```

Fig 7.2: Mail protocols reading

- This is the next step after the loading of the full model of project.
- Here all the mail transfer protocols are checked or verified. If anything misses, it arises an error regarding the missing function.
- It also verifies and read the mail and password for further process of sending the mail.

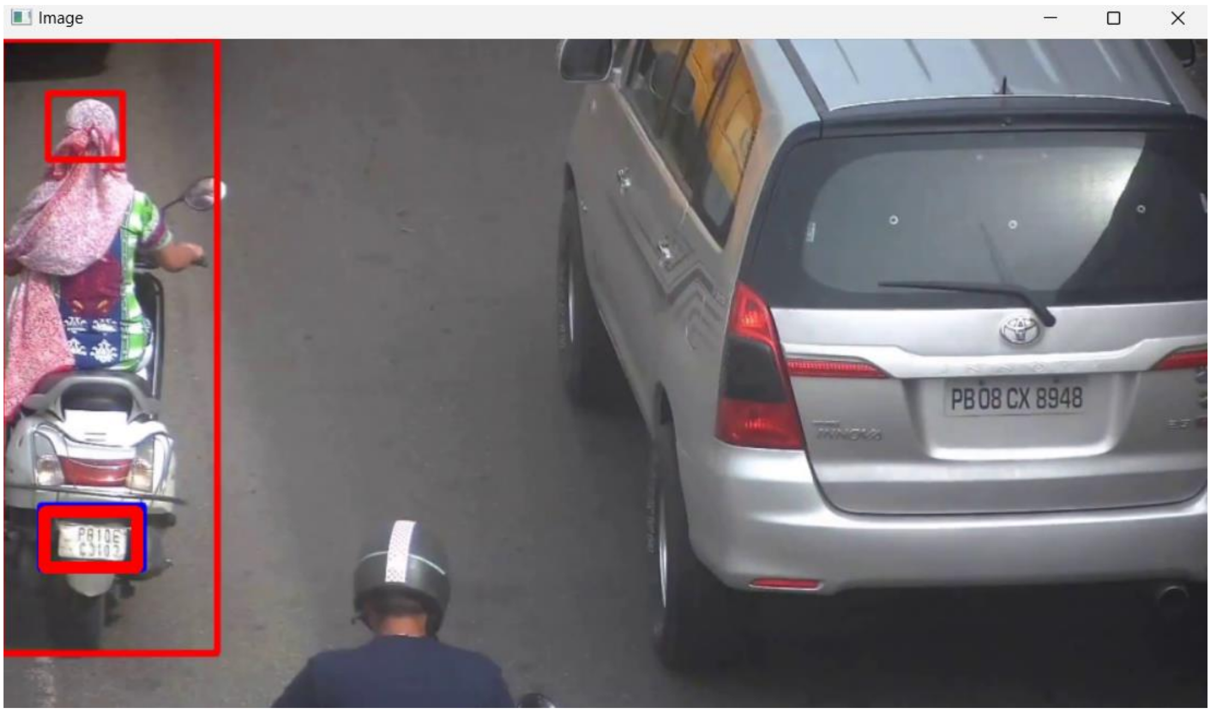
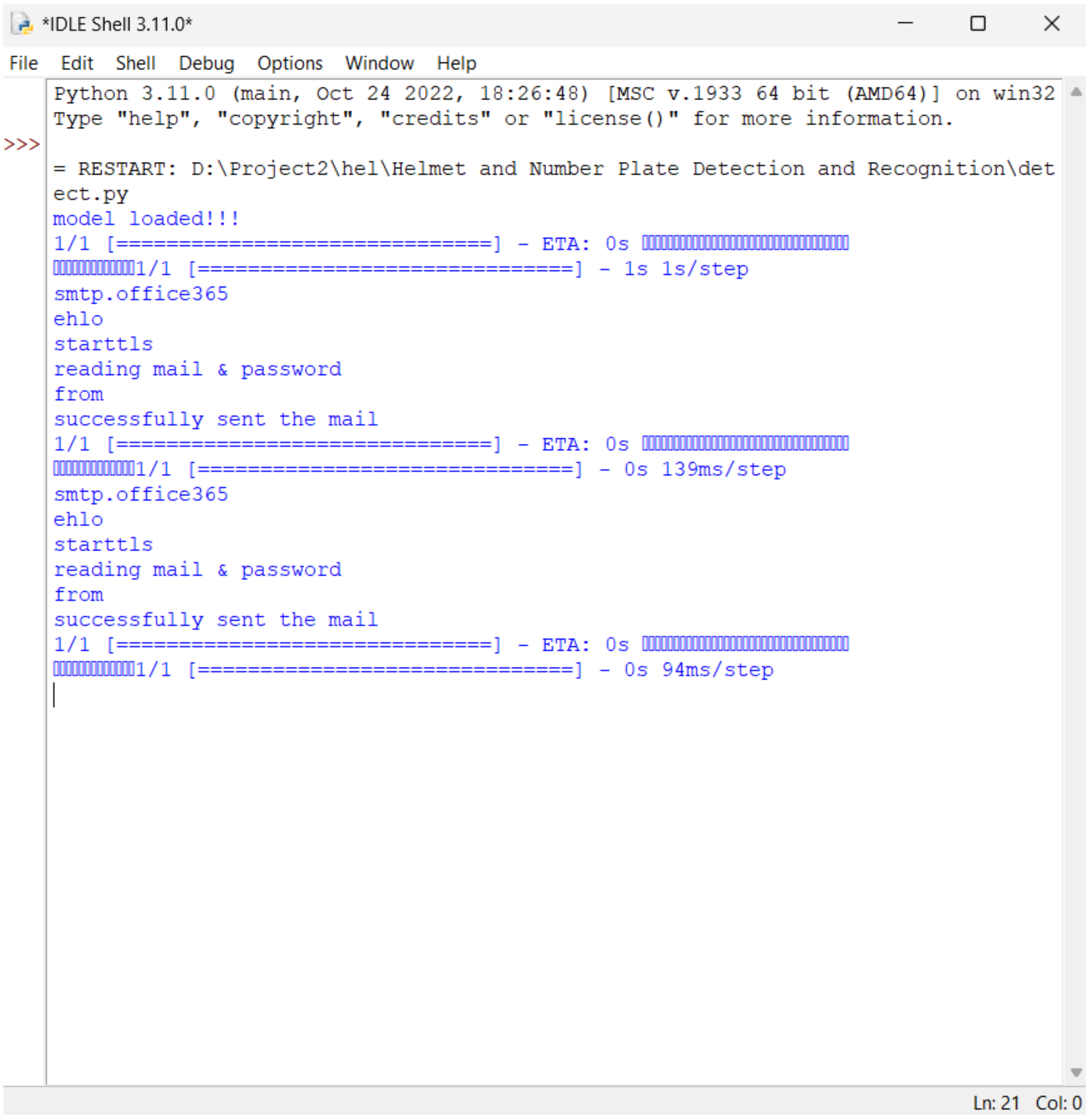


Fig 7.3: Detection of wearing helmet and recognition of number plate

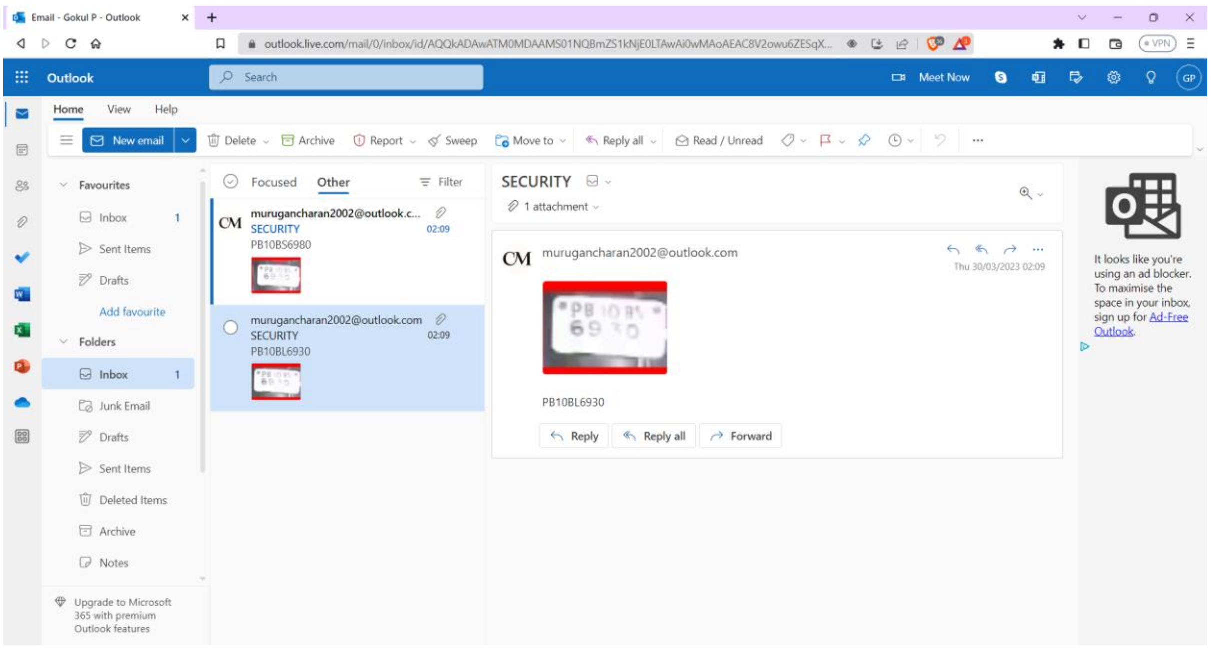
- This is the step after loading all the mail protocols, mail id and password.
- Here the software analyse the video and detects that the lady doesn't wear helmet.
- After the detection, the software then focus on the number plate and takes a screenshot of the number plate.



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Project2\hel\Helmet and Number Plate Detection and Recognition\det
ect.py
model loaded!!!
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 1s 1s/step
smtp.office365
ehlo
starttls
reading mail & password
from
successfully sent the mail
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 139ms/step
smtp.office365
ehlo
starttls
reading mail & password
from
successfully sent the mail
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 94ms/step
```

Fig 7.4: Implementing mail process

- After taking the screenshot of the numberplate in the above process, the software now sends the mail to the control room.
- The control room can process further fines.
- The software provides the screenshot with the character extraction from the number plate using OCR technique.


Email - Gokul P - Outlook
outlook.live.com/mail/0/inbox/id/AQQkADAwATM0MDAAMS01NQBmZS1kNjE0LTAwAi0wMAoAEAC8V2owu6ZESqX...
Outlook
Search
Meet Now
Home
View
Help
New email
Delete
Archive
Report
Sweep
Move to
Reply all
Read / Unread
Favourites
Inbox 1
Sent Items
Drafts
Add favourite
Folders
Inbox 1
Junk Email
Drafts
Sent Items
Deleted Items
Archive
Notes
Upgrade to Microsoft 365 with premium Outlook features
Focused
Other
Filter
1 attachment
Reply
Reply all
Forward
It looks like you're using an ad blocker. To maximise the space in your inbox, sign up for Ad-Free Outlook.

muruganchran2002@outlook.c... SECURITY 02:09
PB10BS6980

muruganchran2002@outlook.com SECURITY 02:09
PB10BL6930

SECURITY

muruganchran2002@outlook.com
Thu 30/03/2023 02:09

PB10BL6930

Fig 7.5: Mail sent

- The person in the control room receives the mail with an attached screenshot in it.
- The extracted text from the image will also be printed in the mail.
- Since the number plate is having some noise, it takes several screenshots and process the extraction of characters in the number plate.
- The number plate from screenshot is not clear, the OCR technique analyse the number and character in it and sends two mails to the control room with possible characters extracted in the screenshot.

CHAPTER 8

TEST PROCEDURE

8.1 SYSTEM TESTING:

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough. Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

8.2 TEST DATA AND OUTPUT:

8.2.1 UNIT TESTING:

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

Test objectives

• All field entries must work properly.

• Pages must be activated from the identified link.

• The entry screen, messages and responses must not be delayed.

Features to be tested

• Verify that the entries are of the correct format

• No duplicate entries should be allowed

• All links should take the user to the correct page

8.2.2 FUNCTIONAL TESTS:

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

8.2.2.1 PERFORMANCE TEST:

Stress Test is those tests designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

8.2.2.3 STRUCTURED TEST:

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.

- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling ends of file condition, I/O errors, buffer problems and textual errors in output information

8.2.3 INTEGRATION TESTING:

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected. The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

8.3 TESTING TECHNIQUES / TESTING STRATERGIES:

8.3.1 TESTING:

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

8.3.1.1 WHITE BOX TESTING:

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing. Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

8.3.1.2 BLACK BOX TESTING:

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

8.3.2 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.

- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

8.3.2.1 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

8.3.2.2 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

8.3.2.3 VALIDATION TESTING:

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

8.3.2.4 USER ACCEPTANCE TESTING:

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

Test Results:

- All the test cases mentioned above passed successfully. No defects encountered.

8.4 OVER ALL TESTING RESULT:

- **Valid Input :** Identified classes of valid input must be accepted.
- **Invalid Input :** Identified classes of invalid input must be rejected.
- **Functions :** Identified functions must be exercised.
- **Output :** Identified classes of application outputs must be exercised.
- **Systems/Procedures :** Interfacing systems or procedures must be invoked.

8.5 TEST CASE SPECIFICATION

TEST CASE ID	**MODULE**	**INPUT**	**EXPECTED OUTPUT**	**ACTUAL OUTPUT**	**STATUS**
TC1	Helmet Detection	Video input of riders wearing helmet in motorcycle	Helmet detected	Detection of Helmet on the rider	Pass
TC2	Number plate detection	Input video of motorcycles with license plate	Number plate recognition	Recognition and extraction of text from license plate	Pass

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION:

The detection of helmets and license plates are two significant applications of computer vision technology. Automatic Helmet detection is used to examine the photos and it gives the result of whether the person is wearing a helmet or not. This technology can be applied in many other contexts, like traffic monitoring, building sites, and other places where safety gear is necessary. Number plate identification is mainly processed for the identification of characters on the license plate of a vehicle. Many uses of this technology exist, such as parking management systems, toll booths, and law enforcement. Both of these applications have a great deal of promise to increase convenience, security, and safety in a variety of settings. It is essential to keep in mind that these systems may have flaws and are not faultless. For instance, when there are several individuals in the frame or when the illumination is bad, automatic helmet identification systems might not function as well. Similarly to this, dirty or obscured license plates may be difficult to identify by number plate recognition software.

9.2 FUTURE WORK:

Despite these drawbacks, automatic helmet detection and number plate recognition are significant technological advancements that could greatly enhance our quality of life.

- Increased accuracy: This system’s accuracy is a major area for development. Advanced algorithms, machine learning models, and more potent technology can all be used to do this. We can improve these systems' overall performance by lowering the number of false positives and false negatives by boosting the accuracy of these systems.

- Cloud-based processing: With the rise of cloud computing, it would be possible to process and analyse images remotely as opposed to locally using hardware. This might result in quicker processing times and more precise outcomes.
- Automatic detection systems could be improved to detect additional types of safety gear in addition to helmets, like seat belts, high-visibility clothes, and safety goggles. This can further increase workplace and road safety.
- Technological restrictions: Environmental aspects like illumination and weather can have an impact on how accurate automated systems can be. Also, some helmets or license plates could be challenging for the algorithms to recognize.
- High initial cost: The expense of such systems implementation may prevent certain businesses or governments from implementing them.
- Maintenance and upkeep: To make sure that automatic helmet detection and number plate identification systems are operating properly, monthly maintenance is required. Maintenance and care expenses should be kept up for the time being.

CHAPTER 10

REFERENCES:

[1] Alarifi, A., & Al-Salman, A. (2022). Automatic license plate recognition using deep learning: A review. IEEE Access, 10, 46331-46353.

[2] Aras, S., & Kiliç, E. (2021). Automatic license plate recognition with deep learning techniques. In 2021 International Artificial Intelligence and Data Processing Symposium (IDAP) (pp. 1-4). IEEE.

[3] Balaji, R., & Kottaimuthu, R. (2022). A deep learning-based number plate recognition system for vehicle identification. Journal of Ambient Intelligence and Humanized Computing, 13(2), 2051-2064.

[4] Dnyaneshwar Kokare, Aaditi Ujwankar, Alisha Mulla, Mrunal Kshirsagar, Apurva Ratnaparkhi "Helmet Detection and Number Plate Recognition using Machine Learning" International Journal of Research in Engineering, Science and Management, June 2022.

[5] Li, Y., Li, J., & Zhang, B. (2021). Deep learning based detection and recognition of vehicle license plates: A review. IET Intelligent Transport Systems, 15(8), 1086-1094.

[6] Lokesh Allamki, Manjunath Panchakshari, Ashish Sateesha, K S Pratheek proposed Helmet Detection using Machine Learning and Automatic License Plate Recognition International Research Journal of Engineering and Technology (IRJET), Dec 2019.

[7] Mr. Thirunavukkarasu.M ; Bugade Amoolya ; Bulusu Vyagari Vaishnavi "Helmet Detection and Licence Plate Recognition" Thirunavukkarasu.M et al, International Journal of Computer Science and Mobile Computing, April- 2021.

[8] Piliouras, K., & Siganos, D. (2020). An end-to-end deep learning approach for license plate recognition. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC) (pp. 3468-3473). IEEE.

[9] Sakhare, N. A., & Bhangale, M. D. (2021). Automatic vehicle number plate recognition system using deep learning. In 2021 International Conference on Sustainable Computing and Intelligent Systems (ICSCIS) (pp. 1-5). IEEE.

[10] S. B. Akintoye, O. E. Famoroti, and B. A. Adejuyigbe, "Helmet detection and number plate recognition system for law enforcement agencies," Journal of Physics: Conference Series, vol. 1529, no. 1, p. 012049, 2020.