A

Project Report

On

Number Plate Detection without Helmet

Submitted for partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING
In

COMPUTER SCIENCE AND ENGINEERING

By

Ms.A.Naga Akshita (18N81A05K7)
Ms.N.Nandini (18N81A05K5)
Ms.D.Amulya (18N81A05M4)
Ms.V.Navya Sree (18N81A05L1)

Under the guidance of

Mr.Miskeen Ali
Assistant Professor
Department of CSE

B. Tech * MBA

SPHOORTHY ENGINEERING COLLEGE
Department of Computer Science and Engineering
(Affiliated to JNTUH& Recognized by AICTE)
Nadergul, SaroorNagar Mandal, Hyderabad – 501 510

Academic Year: 2021-22

SPHOORTHY ENGINEERING COLLEGE

B.Tech | Visual Arts | BBA

CERTIFICATE

This is to certify that this Project Seminar Report entitled ***"NUMBER PLATE DETECTION WITHOUT HELMET"*** *is a bonafied work carried out by* ***Ms.A.Naga Akshita(18N81A05K7)Ms.N.Nandini (18N81A05K5), Ms.D.Amulya (18N81A05M4), Ms.V.Navya Sree(18N81A05L1****) in partial fulfilment of the Requirements for the award of degree of Bachelor of Engineering College, affiliated to Jawaharlal Nehru Technological University Hyderabad during the Academic year 2021-2022 under our guidance and supervision The results embodies in the this report have not been submitted to any other university or Institute for the award of any degree or diploma*

Internal Guide

Mr.Miskeen Ali
Assistant Professor
Department of CSE
SPHN

Head of the Department

Mr.P.Ram Mohan Rao
Head
Department of CSE
SPHN

DECLARATION

We the undersigned, declare that the Major Project title **"NUMBER PLATE DETECTION WITHOUT HELMET"** carried out at "SPHOORTHY ENGINEERING COLLEGE" is original and is being submitted to the Department of COMPUTER SCIENCE AND ENGINEERING, Sphoorthy Engineering College, Hyderabad towards partial fulfillment for the award of Bachelor of Technology

We declare that, the result embodied in the Major Project work has not been submitted to any other University or Institute for the award of any degree or Diploma

Date:

Place: Hyderabad

Ms. A. Naga Akshita (18N81A05K7)

Ms.N. Nandini (18N81A05K5)

Ms. D. Amulya (18N81A05M4)

Ms. V. Navya Sree (18N81A05L1)

ACKNOWLEDGEMENT

We express my deep sense of gratitude to Our Project Guide Mr.Miskeen Ali, Department of Computer Science &Engineering, Sphoorthy Engineering College Jawaharlal Nehru Technological University Hyderabad (JNTUH) for his Inspiring guidance, consistent, encouragement, constructive criticism and helpful Suggestion during the entire course of my research work.

We express our sincere thanks to **Mr.P.Rammohan Rao**, Associate Professor &Head of the Department, Department of the Computer Science &Engineering Sphoorthy Engineering College, Nadergul (V), Balapur RangaReddy (D) for his Encouragement which helped me to complete my project work.

We deem it great privilege to express our profound gratitude and sincere thanks to **Mr.S.Chalama Reddy**, Chairman, **Mr.S.Jagan Mohan Reddy**, Secretary,**Prof.J.B.V.Subramanyam**, Principal**, Prof.M.V.S Ram Prasad**

,Director,Sphoorthy Engineering College, Nadergul (V), Balapur (M), RangaReddy (D),for their moral support and help in the completion of my research work.

We express our heartfelt thanks to Professors, Associate Professors, and Assistant Professor and other Professional non-teaching staff of department of Computer Science and Engineering, Sphoorthy Engineering College, Nadargul, Balapur, Ranga Reddy for providing the necessary information pertaining to us project work.

BATCH: 2018-2022

ABSTRACT

In current situation, we come across various problems in traffic regulations in India which can be solved with different ideas. Riding motorcycle without wearing helmet is a traffic violation which has resulted in increase in number of accidents and deaths in India. Existing system monitors the traffic violations primarily through CCTV recordings, where the traffic police have to look into the frame where the traffic violation is happening, zoom into the license plate in case rider is not wearing helmet. But this requires lot of manpower and time as the traffic violations frequently and the number of people using motorcycles is increasing day-by-day. What if there is a system, which would automatically look for traffic violation of not wearing helmet while riding motorcycle and if so, would automatically extract the vehicles' license plate number. Recent researches have successfully done this work based on CNN, R-CNN, LBP, HoG, HaaR features,etc. But these works are limited with respect to efficiency, accuracy or the speed with which object detection and classification is done. Non-Helmet Rider detection system is built which attempts to satisfy the automation of detecting the traffic violation of not wearing helmet and extracting the vehicles' license plate number. The main principle involved is Object Detection .The objects detected are person, motorcycle at first level, helmet at second level, License plate at the last level .Then the license plate registration number is extracted using OCR (Optical Character Recognition).

Ms. A. Naga Akshita (18N81A05K7)

Ms.N. Nandini (18N81A05K5)

Ms. D. Amulya (18N81A05M4)

Ms. V. Navya Sree (18N81A05L1)

INDEX

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Nowadays two-wheeler are the most popular modes of transport since all level of people can afford it. When the number of motorcyclist increases, there has been an increasing number of motorbike accidents due to reckless riding. The carelessness of motorcyclists not wearing a helmet is a predominant factor, and it commonly contributes to the biker's head injury. To solve this issue, most countries have laws which mandate the use of helmets for two-wheeler riders. In some countries, the governments have installed a specialized sensor to check the presence of the helmet, but it is economically not reliable to buy sensors for every bike. Without a proper system, the traffic police personnel are deployed to check whether the motorcyclists are wearing the helmet or not. Number Plate Detection without Helmet will help to reduce the burden faced by the traffic police, and it also needs fewer human resources. As a result, the number of motorcyclists not wearing a helmet will get reduced. The main objective of this study is to develop a real-time application for detection for non-helmeted motorcyclist using the single convolutional neural networks.

1.2 OBJECTIVE

Helmet reduces the chances of skull getting decelerated, hence sets the motion of the head to almost zero. Cushion inside the helmet absorbs the impact of collision and as time passes head comes to a halt. It also spreads the impact to a larger area, thus safeguarding the head from severe injuries. More importantly it acts as a mechanical barrier between head and object to which the rider came into contact. Injuries can be minimized if a good quality full helmet is used. Traffic rules are there to bring a sense of discipline, so that the risk of deaths and injuries can be minimized significantly. However strict adherence to these laws is absent in reality. Hence efficient and feasible techniques have to be created to overcome these problems. Manual surveillance of traffic using CCTV is an existing methodology. But here so many iterations have to be performed to attain the

objective and it demands a lot of human resource. Therefore, cites with millions of population having so many vehicles running on the roads cannot afford this inadequate manual method of helmet detection. So here we propose a methodology for full helmet detection and license plate extraction using YOLOv2, YOLOv3 and OCR. Basically helmet detection system involves following steps such as collection of dataset, moving object detection, background subtraction, object classification using neural networks.

1.3 EXISTING SYSTEM

Existing system monitors the traffic violations primarily through CCTV recordings, where the traffic police have to look into the frame where the traffic violation is happening, zoom into the license plate in case rider is not wearing helmet. But this requires lot of manpower and time as the traffic violations frequently and the number of people using motorcycles is increasing day-by-day. What if there is a system, which would automatically look for traffic violation of not wearing helmet while riding motorcycle/moped and if so, would automatically extract the vehicles' license plate number. Recent research have successfully done this work based on CNN, R-CNN, LBP, HoG, HaaR features,etc. But these works are limited with respect to efficiency, accuracy or the speed with which object detection and classification is done

1.4 PROPOSED SYSTEM

In this project we are detecting whether two wheeler rider wearing helmet or not, if he is not wearing helmet then we are extracting number plate of that two wheeler. To extract number plate we have YOLO CNN model with some train and test images and if you want to add some other images then send those images to us so we can include those images in YOLO model with annotation to extract number plate of those new images.
To implement above technique we are following or implemented below modules

1) First image will be upload to the application and the using YOLOV2 we will check whether image

contains person with motor bike or not, if YOLO model detect both person and motor bike then we will proceed to step 2.

2) In this module we will use YOLOV3 model to detect whether object wear helmet or not, if he wear helmet then application will stop hear it. If rider not wears helmet then application proceeds to step 3.

3) In this module we will extract number plate data using python tesseract OCR API. OCR will take input image and then extract vehicle number from it.

1.5 Scope

Riding motorcycle without wearing helmet is a traffic violation which has resulted in increase in number of accidents and deaths. A non-helmet rider detection system is built which attempts to satisfy the automation of detecting the traffic violation of not wearing helmet and extracting the vehicles license plate number.

1.6 Software & Hardware Requirements

The software requirements for the number plate detection are Python, Tensor flow, keras, opencv, and other few Libraries

The Hardware Requirements are easily available as well.

Hardware Requirements

- 8GB RAM
- Processor : Intel core

Software Requirements

➢ Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- system scripting.

The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

Python is an interpreted high-level general-purpose programming languageLibraries such as Numpy, and Matplotlib allow the effective use of Python in scientific computing, SageMath is a computer algebra system with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. OpenCV has Python bindings with a rich set of features for computer vision and image processing.

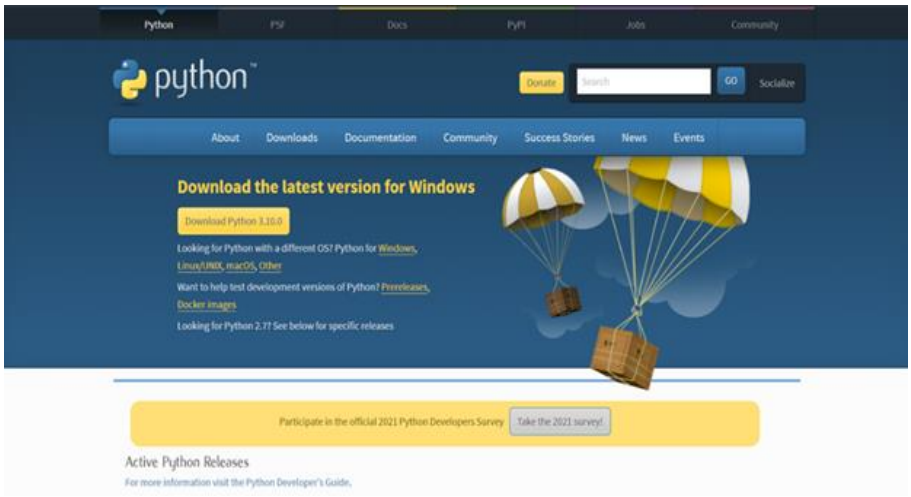


Fig 1.1 Python Home Page

➢ Tensor flow

Tensor Flow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Tensor Flow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use. The data API enables you to build complex input pipelines from simple, reusable pieces.

How to Install Tensor Flow

Installation of Tensor Flow is straightforward if you already have a Python SciPy environment.

Tensor Flow works with Python 3.3+. You can follow the Download and Setup instructions on the Tensor Flow website. Installation is probably simplest via PyPI and specific instructions of the pip command to use for your Linux or Mac OS X platform are on the Download and Setup webpage. In the simplest case, you just need to enter the following in your command line:

pip install tensorflow

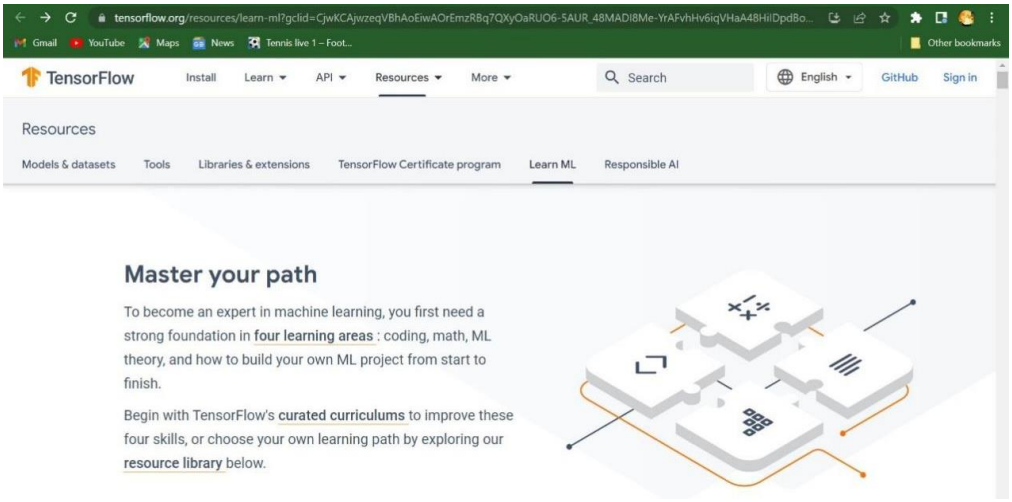


Fig 1.2 Tensor Flow

➢ Keras

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on Tensor Flow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

Keras, you do in-model data preprocessing via preprocessing layers. This includes:

- Vectorizing raw strings of text via the TextVectorization layer
- Feature normalization via the Normalization layer
- Image rescaling, cropping, or image data augmentation

The key advantage of using Keras preprocessing layers is that they can be included directlyinto your model, either during training or after training, which makes your models portable.

Some preprocessing layers have a state:

- TextVectorization holds an index mapping words or tokens to integer indices
- Normalization holds the mean and variance of your features

The state of a preprocessing layer is obtained by calling layer.adapt(data) on a sample of the training data

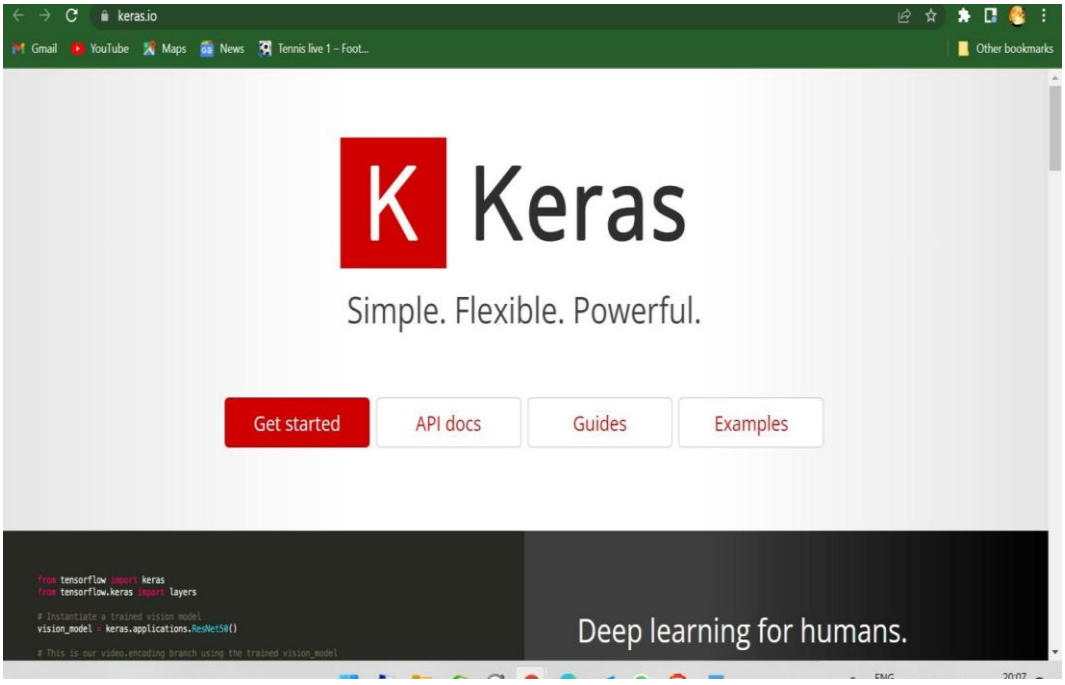


Fig 1.3 Keras Logo

➢ Pandas

Pandas are an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas are fast and it has high performance & productivity for users.

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns.
- What is average value.
- Max value.
- Min value.

Pandas are generally used for data science but have you wondered why? This is because pandas are used in conjunction with other libraries that are used for data science.

Installation of Pandas

If you have Python and PIP already installed on a system, then installation of Pandas is very easy.

Install it using this command:

C:\Users\Your Name\pip install pandas

Fig.1.4 Pandas

- **Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

In Python we have lists that serve the purpose of arrays, but they are slow to process

NumPy aims to provide an array object that is up to 50x faster thantraditional Python lists

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

Installation of NumPy

If you have Python and PIP already installed on a system, then installation of NumPy is very easy.

Install it using this command:

C:\Users\Your Name> pip install numpy

If this command fails, then use a python distribution that already has NumPy installed like, Anaconda, Spyder etc.

Fig 1.6 Numpy

➢ Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.
It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

Installation:

Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages.

Run the following command to install matplotlib package :

python -mpip install -U matplotlib

Fig.1.6 Matplotlib

- **H5py**

The h5py package is a Pythonic interface to the HDF5 binary data format.It lets you store huge amounts of numerical data, and easily manipulate that data from NumPy.

As the name suggests, it stores data in a hierarchical structure within a single file. So if we want to quickly access a particular part of the file rather than the whole file, we can easily do that using HDF5. This functionality is not seen in normal text files hence HDF5 is becoming seemingly popular in fact of being a new concept. To use HDF5, numpy needs to be imported. One important feature is that it can attach metaset to every data in the file thus provides powerful searching and accessing. Let's get started with installing HDF5 to the computer.

To install HDF5, type this in your terminal:

```
pip install h5py
```

H5py uses straightforward NumPy and Python metaphors, like dictionary and NumPy array syntax. For example, you can iterate over datasets in a file, or check out the .shape or .dtype attributes of datasets.
In addition to the easy-to-use high level interface, h5py rests on a object-oriented Cython wrapping of the HDF5 C API. Almost anything you can do from C in HDF5, you can do from h5py.

Fig.1.7 H5py

➢ Pillow

Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors.Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

- ✓ The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

- ✓ The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

This module is not preloaded with Python. So to install it execute the following command in the command-line:

pip install pillow

Fig.1.8 Pillow

➢ Scikit-learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, and clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Important features of scikit-learn:

Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means

- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

Installation:

Scikit-learn requires:

- NumPy
- SciPy as its dependencies.

Before installing scikit-learn, ensure that you have NumPy and SciPy installed. Once you have a working installation of NumPy and SciPy, the easiest way to install scikit-learn is using pip:

pip install -U scikit-learn

Fig1.9 Scikit-Learn

➢ Swarms

PySwarms is an extensible research toolkit for particle swarm optimization (PSO) in Python.

It is intended for swarm intelligence researchers, practitioners, and students who prefer a high-level declarative interface for implementing PSO in their problems. PySwarms enables basic optimization with PSO and interaction with swarm optimizations.

Installation

To install PySwarms, run this command in your terminal:

```
$ pip install pyswarms
```

This is the preferred method to install PySwarms, as it will always install the most recent stable release.
In case you want to install the bleeding-edge version, clone this repo:

```
$ git clone -b development https://github.com/ljvmiranda921/pyswarms.git
```

and then run

```
$ cd pyswarms

$ python setup.py install
```

Fig.1.10 Swarms

CHAPTER 2

LITERATURE SURVEY

Circle arc detection technique supported Hough remodel. They applied it to find the presence of helmet that did not offer correct result. Combination of image process and Optical Character Recognition to find vehicle variety plate underneath totally different background however it's worked on static

2.1 METHODOLOGY

In this study, a Non-Helmet Rider noticing system is built which attempts to satisfy the automation of detecting the traffic violation of not wearing a helmet and extracting the vehicles' license plate number. The main concept involved in Object Detection using Deep Learning at three steps. The objects detected are person, motorcycle at first step using YOLOv2, detecting helmet at a second step using YOLOv3, recognizing license plate at the last step using YOLOv2. Then the license plate registration number is takeout using OCR (Optical Character Recognition). All these techniques are put through to prearrange conditions and constraints, especially the license plate number extraction part. We have used above said procedure to build a holistic system for both helmet and license plate number extraction.

2.2 TECHNIQUES AND ALGORITHMS

➢ Open CV

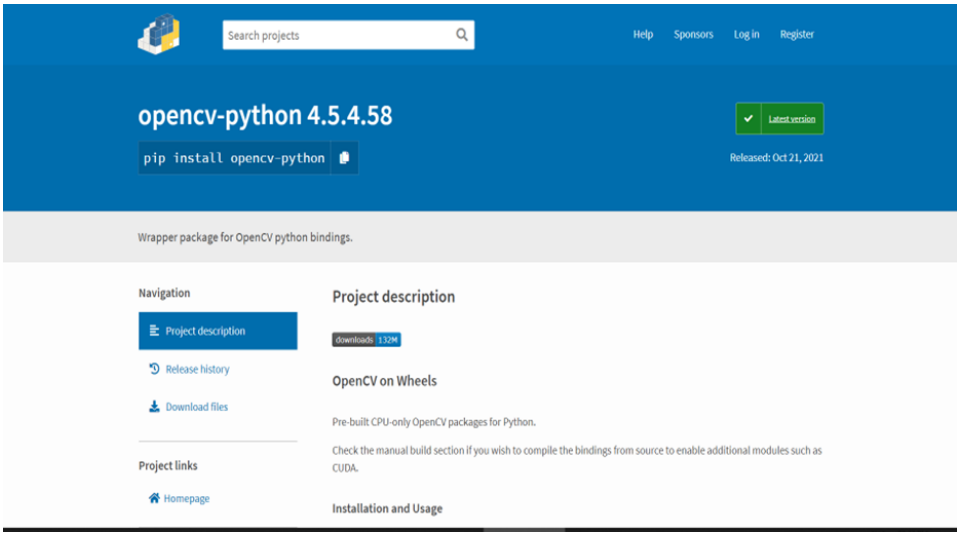


Fig.2.1 OpenCV Home page

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human.OpenCV is used for all sorts of image and video analysis, like facial recognition and detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and a whole lot more.

OpenCV runs on the following desktop operating systems:

Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the following mobile operating systems: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV uses CMake.

Image-Processing:

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then **"Image processing is the analysis and**

manipulation of a digitized image, especially in order to improve its quality".

Digital-Image :

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be. Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps:

1. Importing the image
2. Analysing and manipulating the image
3. Output in which result can be altered image or report that is based on image analysis.

How Does A Computer Read An Image?

We are humans we can easily make it out that is the image of a person. But if we ask computer "is it my photo?". The computer can't say anything because the computer is not figuring out it all on its own.

The computer reads any image as a range of values between 0 and 255. For any color image, there are 3 primary channels -red, green and blue.

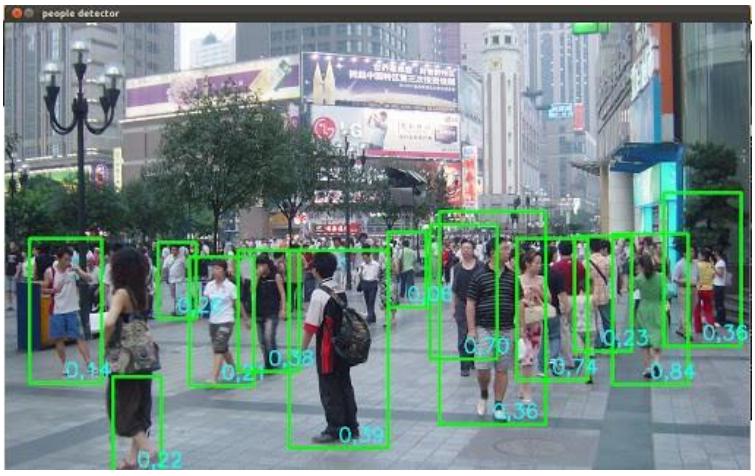


Fig 2.2 Example of Open CV

➢ Tkinter

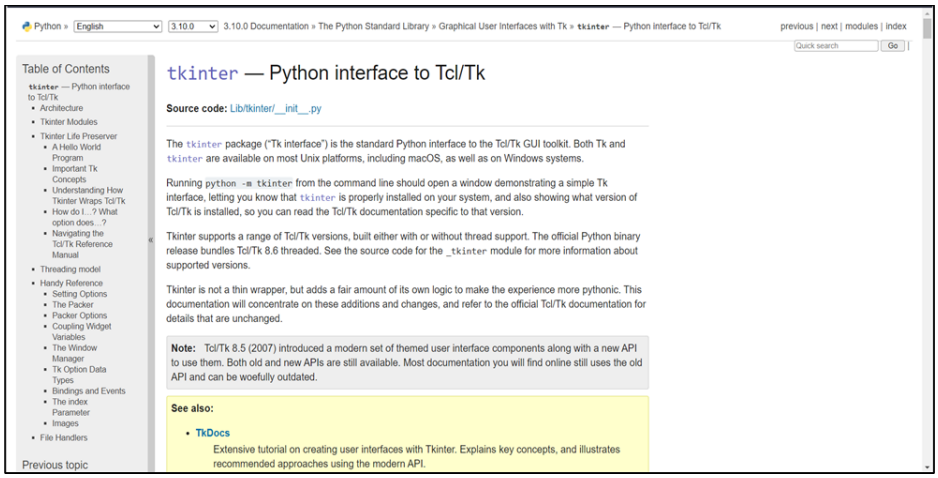


Fig.2.3 Tkinter Home page

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's de facto standard GUI. Tkinter is included with standard GNU/Linux, Microsoft Windows and macOS installs of Python. The name Tkinter comes from Tk interface.

It is used to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact , it's the only framework built into the Python standard library. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

There are two main methods used which the user needs to remember while creating the Python application with GUI-

Tk(screenName=None, baseName=None, className='Tk', useTk=1): To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one.

Tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There is mainly three geometry manager class's

- **pack() method**
- **grid() method**
- **place() method**.

➢ YOLO (Real Time Object Detection)

Fig 2.4 Yolo logo

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results.

When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer.

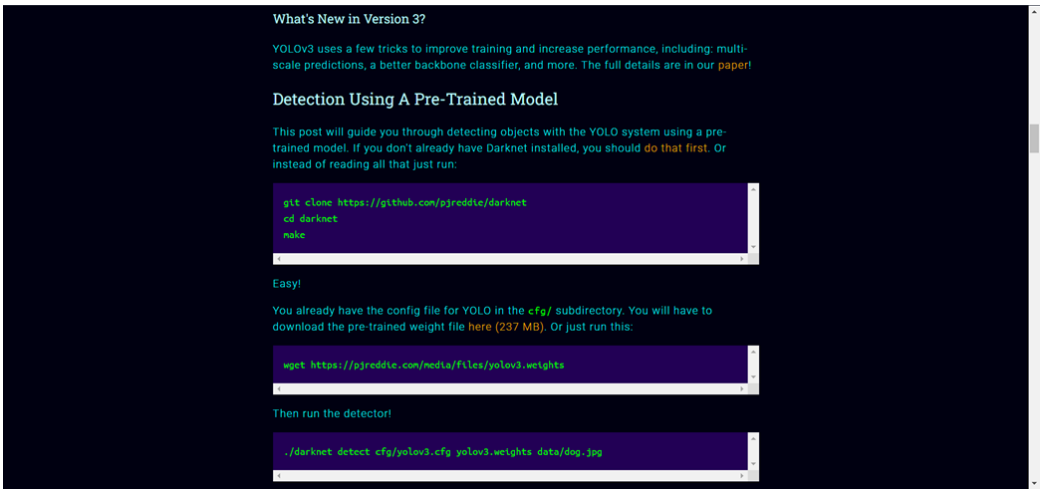


Fig 2.5 Yolo Home Page

YOLO algorithm is important because of the following reasons:

- **Speed:** This algorithm improves the speed of detection because it can predict objects in real-time.
- **High accuracy:** YOLO is a predictive technique that provides accurate results with minimal background

- **Learning capabilities:** The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

YOLO algorithm works using the following three techniques:

- **Residual blocks**

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

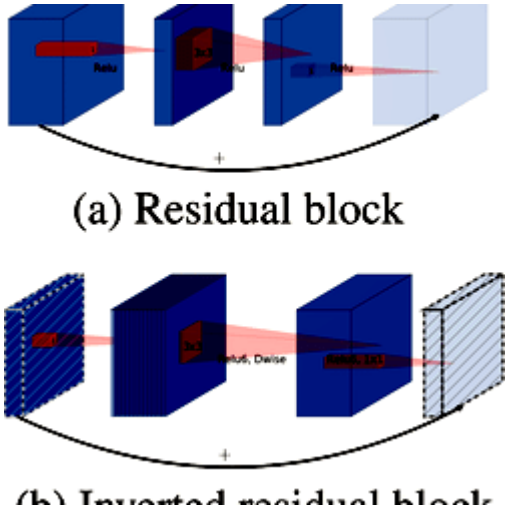


Fig 2.6 Residual Blocks

- **Bounding box regression**

YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box. The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline

YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image below, represents the probability of an object appearing in the bounding box.

Fig 2.7 Bounding box regression

- **Intersection over union (IOU)**

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

YOLO has gone through a number of different iterations, including YOLO9000: Better, Faster, and Stronger capable of detecting over 9,000 object detectors.

Redmon and Farhadi are able to achieve such a large number of object detections by performing joint training for both object detection and classification. Using joint training the authors trained YOLO9000 simultaneously on both the ImageNet classification dataset and COCO detection dataset. The result is a YOLO model, called YOLO9000, that can predict detections for object classes that don't have labeled detection data.

While interesting and novel, YOLOv2's performance was a bit underwhelming given the title and abstract of the paper.

On the 156 class version of COCO, YOLO9000 achieved 16% mean Average Precision (mAP), and yes, while

YOLO can detect 9,000 separate classes, the accuracy is not quite what we would desire.

Redmon and Farhadi recently published a new YOLO paper, YOLOv3: An Incremental Improvement (2018).

YOLOv3 is significantly larger than previous models but is, in my opinion, the best one yet out of the YOLO family of object detectors.

We'll be using YOLOv3 in this blog post, in particular, YOLO trained on the COCO dataset.

The COCO dataset consists of 80 labels, including, but not limited to:

- People
- Bicycles
- Cars and trucks
- Airplanes
- Stop signs and fire hydrants
- Animals, including cats, dogs, birds, horses, cows, and sheep, to name a few
- Kitchen and dining objects, such as wine glasses, cups, forks, knives, spoons, etc.

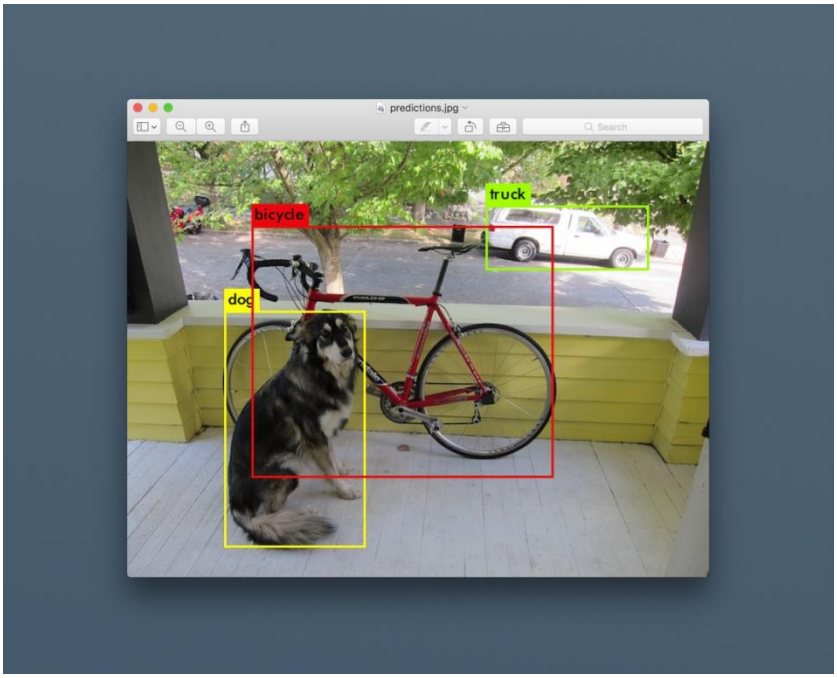


Fig 2.8 YOLO Detection

➢ Convolution Neural Networks (CNNs)

In Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks.

Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or soft ax which converts the output of each class into the probability score of each class.

   The data is then fed into the model and output from each layer is obtained this step is called feed forward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. After that, we back propagate into the model by calculating the derivatives. This step is called back propagation which basically is used to minimize the loss.

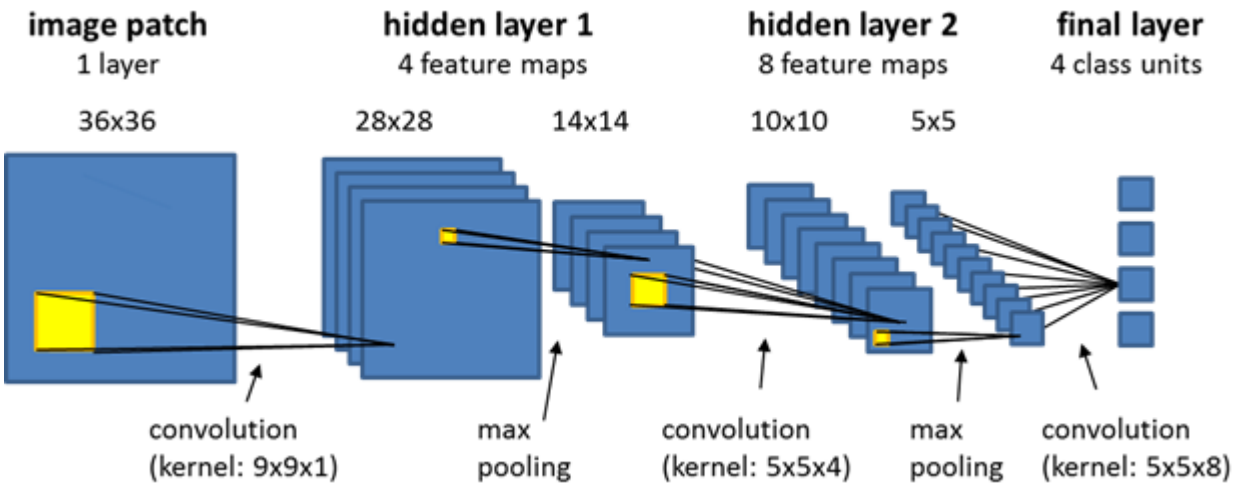


Fig 2.9 CNN layers

What is a Convolution?

A convolution is how the input is modified by a filter. In convolution networks, multiple filters are taken to slice through the image and map them one by one and learn different portions of an input image. Imagine a small filter sliding left to right across the image from top to bottom and that moving filter is looking for, say, a dark edge. Each time a match is found, it is mapped out onto an output image.

Convolution in 2D

Let's start with a (4 x 4) input image with no padding and we use a (3 x 3) convolution filter to get an output image.

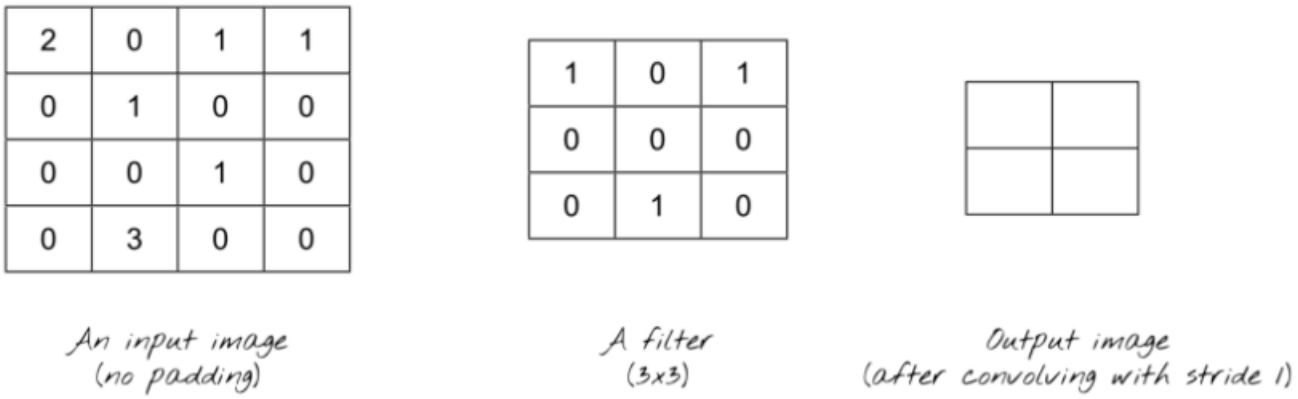
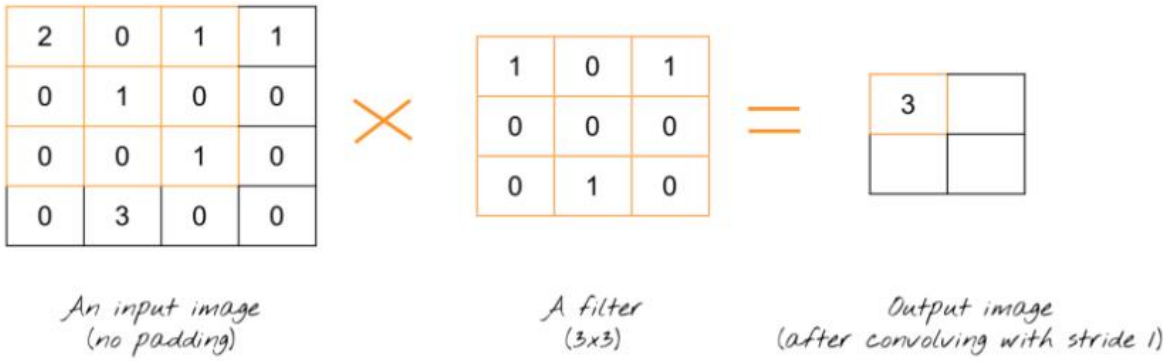


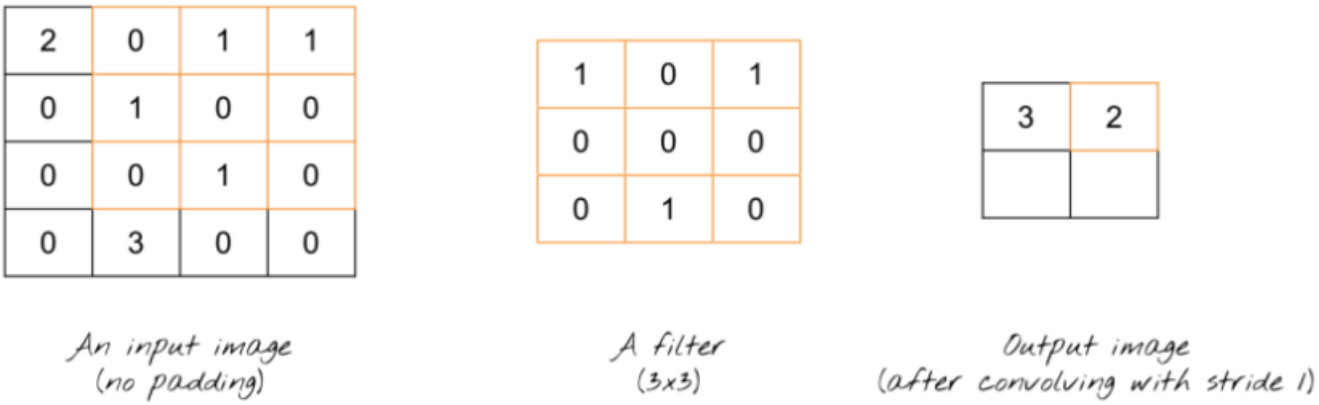
Fig 2.10 Example of Convolution in 2D

The first step is to multiply the yellow region in the input image with a filter. Each element is multiplied with an element in the corresponding location. Then you sum all the results, which is one output value.

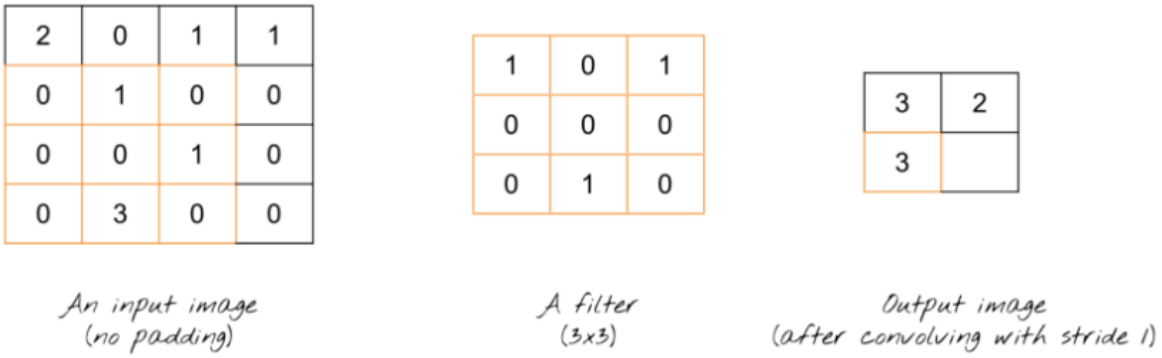


Mathematically, it's (2 * 1) + (0 * 0) + (1 * 1) + (0 * 0) + (1 * 0) + (0 * 0) + (0 * 0) + (0 * 1) + (1 * 0) = 3

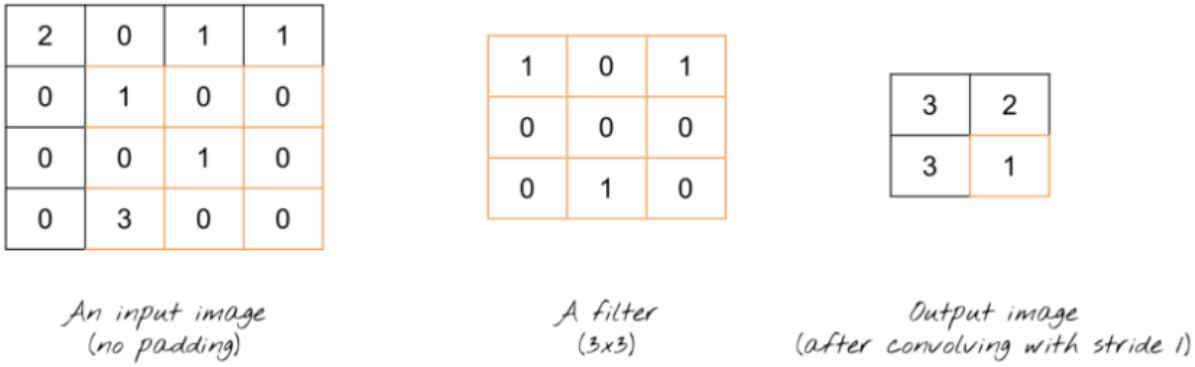
Then, you repeat the same step by moving the filter by one column. And you get the second output.



Notice that you moved the filter by only one column. The step size as the filter slides across the image is called a **stride**. Here, the stride is 1. The same operation is repeated to get the third output. A stride size greater than 1 will always downsize the image. If the size is 1, the size of the image will stay the same.



At last, you are getting the final output.



We see that the size of the output image is smaller than that of the input image. In fact, this is true in most cases.

OCR(Optical character recognition)

Optical Character Recognition refers to a software technology that electronically identifies text inside an image file or physical document, such as a scanned document, and converts it into a machine-readable text form to be used for data processing. It is also known as text recognition.

In short, optical character recognition software helps convert images or physical documents into a searchable form.

Working of optical character recognition

The concept of OCR is straightforward. However, its implementation can be quite challenging due to several factors, such as the variety of fonts or the methods used for letter formation.

The entire process of OCR involves a series of steps that mainly contain three objectives: pre-processing of the image, character recognition, and post-processing the specific output.

Image Pre-Processing

- In the first stage, the technology converts the document's physical shape into a picture, such as a record picture. The purpose of this stage is for the machine's representation to be precise while also removing any undesired aberrations.
- The concept is subsequently transformed to a black and white rendition, evaluated for bright vs. dark regions (characters).
- The image is then segmented into individual pieces, such as spreadsheets, text, or inset graphics, using an OCR system.

AI Character Recognition

AI analyzes the image's dark portions to recognize characters and numerals. Typically, AI uses one of the following approaches to target one letter, phrase, or paragraph at a time:

- **Pattern Recognition**: Technologies use a range of language, text formats, and handwriting to train the AI system. The program compares the letters on the detected letter picture to the notes it has already learned to find matches.
- **Feature Recognition**: The algorithm uses rules based on specific character properties to recognize new characters. The amount of angled, crossing, or curved lines in a letter is one example of a feature. To identify original characters, the algorithm employs rules based on particular character attributes.

Post-Processing

AI corrects flaws in the final file during Post-Processing. One approach is to teach the AI a glossary of terms that will appear in the paper. Then, limit the AI's output to those words/formats to verify that no interpretations are beyond the vocabulary.

In the following, we will show how optical character recognition works and explain the main steps of OCR technologies.

1. Scanning the Document

This is the prime step of OCR which connects to a scanner to scan the document. Scanning the document decreases the number of variables to account for when creating the OCR software since it standardizes the inputs. Also, this step specifically enhances the efficiency of the entire process by ensuring perfect alignment and sizing of the specific document.

2. Refining the Image

In this step, the optical character recognition software improves the elements of the document that need to be captured. Any imperfections such as dust particles are eliminated, and edges, as well as pixels, are smoothed to get a plain and clear text. This step makes it easier for the program to capture and be able to clearly “see” the words being inputted without, for instance, smudges or irregular dark areas.

3. Binarization

The refined image document is then converted into a bi-level document image, containing only black and white colors, where black or dark areas are identified as characters. At the same time, white or light areas are identified as background. This step aims to apply segmentation to the document to easily differentiate the foreground text from the background, which allows for the optimal recognition of characters.

4. Recognizing the Characters

In this step, the black areas are further processed to identify letters or digits. Usually, an OCR focuses on one character or block of text at a time. The recognition of characters is carried out by using one of the following two algorithms:

- **Pattern recognition.** The pattern recognition algorithm involves inserting text in different fonts and formats into the OCR software. The modified software is then used for comparing and recognizing the characters in the scanned document.
- **Feature detection.** Through the feature detection algorithm, OCR software applies rules considering the features of a certain letter or number to identify characters in the scanned document. Examples of features include the number of angled lines, crossed lines, or curves used for comparing and identifying characters.

Simple OCR software compares the pixels of every scanned letter with an existing database to identify the closest match. However, sophisticated forms of OCR divide every character into its components, such as curves and corners, to compare and match physical features with corresponding letters.

5. Verifying the Accuracy

After the successful recognition of characters, the results are cross-referenced by utilizing the internal dictionaries of the OCR software to ensure accuracy. Measuring OCR accuracy is done by taking the output of an analysis conducted by an OCR and comparing it to the contents of the original version.

There are two typical methods for analyzing the accuracy of OCR software:

- **Character-level accuracy**, counting how many characters were detected correctly.
- **Word-level accuracy,** counting how many words were recognized correctly.

In most cases, 98-99% accuracy is the acceptable accuracy rate, measured at the page level. This means that in a page of around 1,000 characters, 980-990 characters should be accurately identified by the OCR software.

Fig 2.11 OCR Detection

Number Plate Recognition with OCR

Automatic number plate recognition (ANPR) uses OCR technology to identify the numbers on license plates.

Today, number-plate recognition is used in a diverse set of commercial applications to find stolen cars, calculate fees for parking, invoice tolls or for access control to safety zones, and more.

Fig 2.12 Number Plate Recognition

Advantages of Optical Character Recognition

Optical Character Recognition offers a wide range of benefits, many of which were reviewed in this article.

However, the most important benefits of OCR are listed below for your reference.

- **Improved accuracy**: Software-based character recognition eliminates human errors, resulting in improved accuracy.
- **Speed-up the processes**: The technology converts unstructured data into searchable information, providing the required data available at faster rates and subsequently speeding up business processes.
- **Cost-effective:** OCR technology does not require a lot of resources which reduces the processing costs and subsequently reduces the overall costs of a business.
- **Enhanced customer satisfaction**: The accessibility of searchable data by the customers ensures a good experience, assuring better customer satisfaction.
- **Improved productivity**: The easy accessibility of searchable data makes a stress-free environment for the employees, allowing them to focus on the main goals, boosting the productivity of a business.

Optical character recognition (OCR) is used to turn scanned images and other visuals into text. This turns paper-based documents into editable and searchable digital mediums and enables the development of automated systems.

OCR Applications :

In Banking

The banking industry is deemed one of the largest consumers of OCR technology as it helps enhance security, improves data management, optimizes risk management, and enhances customer experience.

Before applying OCR technology, most banking documents were physical, including customer records, checks, bank statements, and others. With OCR technology, it became possible to digitize and store even older documents in databases.

OCR technology has also completely revolutionized the banking industry by:

- **Providing easy verification**: OCR allows a real-time verification of money deposit checks and a signature by scanning them using an OCR-based application. An example of this can be seen in mobile

banking apps, where checks can be deposited digitally and processed within days through OCR-based check depositing features.

- **Enhancing security**: The electronic deposition of checks through OCR technology results in fraud

- **In Healthcare**

OCR technology has proved to be beneficial for the healthcare industry. In the healthcare sector, OCR technology allows patient medical histories to be accessed digitally by patients and doctors alike.

In addition, patient records, including their X-rays, treatments, tests, hospital records, and insurance payments, can easily be scanned, searched, and stored using OCR technology.

Thus, optical character recognition helps streamline the workflow and reduce manual work at hospitals while keeping the records up to date.

- **In Travelling**

OCR technology has revolutionized the travel industry and has made traveling much easier. Whether you'rebooking a flight or a hotel, checking in to the airport or your hotel room, or managing your travel expenses,OCR technology is being used at every single place to enhance customer experience.

The majority of airports and mobile travel apps use OCR technology for security and data storage purposes. The applications of OCR range from scanning passports to storing personal data when booking a flight or a hotel.

- **Word Processing**

Word processing is perhaps one of the first and most popular applications of OCR. Print files may be scanned and turned into modifiable and accessible versions—AI assists in ensuring that these papers are transformed as accurately as feasible.

- **Legal Documentation**

Crucial approved legal papers, such as loan documentation, can be scanned and stored in an electronic

database for convenient retrieval. The documents may also be viewed and shared by many people.

Optical character recognition (OCR) is used to turn scanned images and other visuals into text.

2.3 Applications

If the person is not wearing a helmet, convolutional neural networks will be used to recognize the number plate characters of the bike rider and create an accurate and effective output in the next step. Using either an ipcam or a webcam, the bike can be identified in the image.

CHAPTER 3

System Design

3.1 System Architecture

This is a system for detecting whether or not someone is wearing a helmet and reading their license plate based on photos obtained from a traffic monitoring system and fed into convolutional neural networks. If the person is not wearing a helmet, convolutional neural networks will be used to recognize the number plate characters of the bike rider and create anaccurate and effective output in the next step.

Using a webcam, the bike can be identified in the image. In order to categorize whether or not a person is wearing a helmet, this technique first detects the picture of a motorcycle and the driver. In this study, we used CNN models to tackle the challenge of identifying bikers and their helmets from surveillance images.

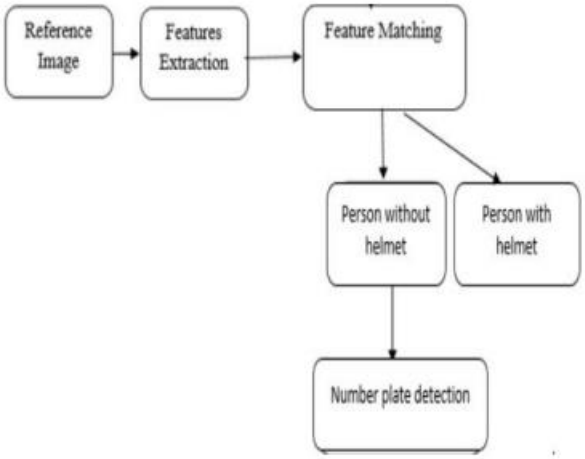


Fig.3.1 System Architecture

When we'd finished collecting the photos that would make up our training dataset, we divided them into two piles: one for training data, and the other for testing data. The results of this experiment are based on the use of CNN models to classify images. To ensure the accuracy of the detection of the biker with and without a helmet in the image, all photographs will be examined and analyzed. Convolutional neural networks, such as CNNs, include several layers (and some other layers). The convolutional process is carried out by a number of filters in a convolutional layer. After comparing the results of earlier phases, we arrive at our final decision. Image

categorization and image detection performance will be evaluated based on experiment accuracy. Performing operations on images at the lowest level of abstraction is referred to as "image pre-processing." If entropy is used as a measure of information content, these procedures actually decrease it. An image's quality can be improved by preprocessing by removing unwanted artifacts or improving certain aspects of the image that are important for subsequent processing and analysis. The license plate number is found using morphologica l procedures on a segmented image. The license plate area will be improved (smoothed) using the dilation and erosion method by deleting any extra pixels from the plate's outer region. We will be able to differentiate the foreground and background after morphological processing. This number plate has been snatched away.

3.2SYSTEM FLOW

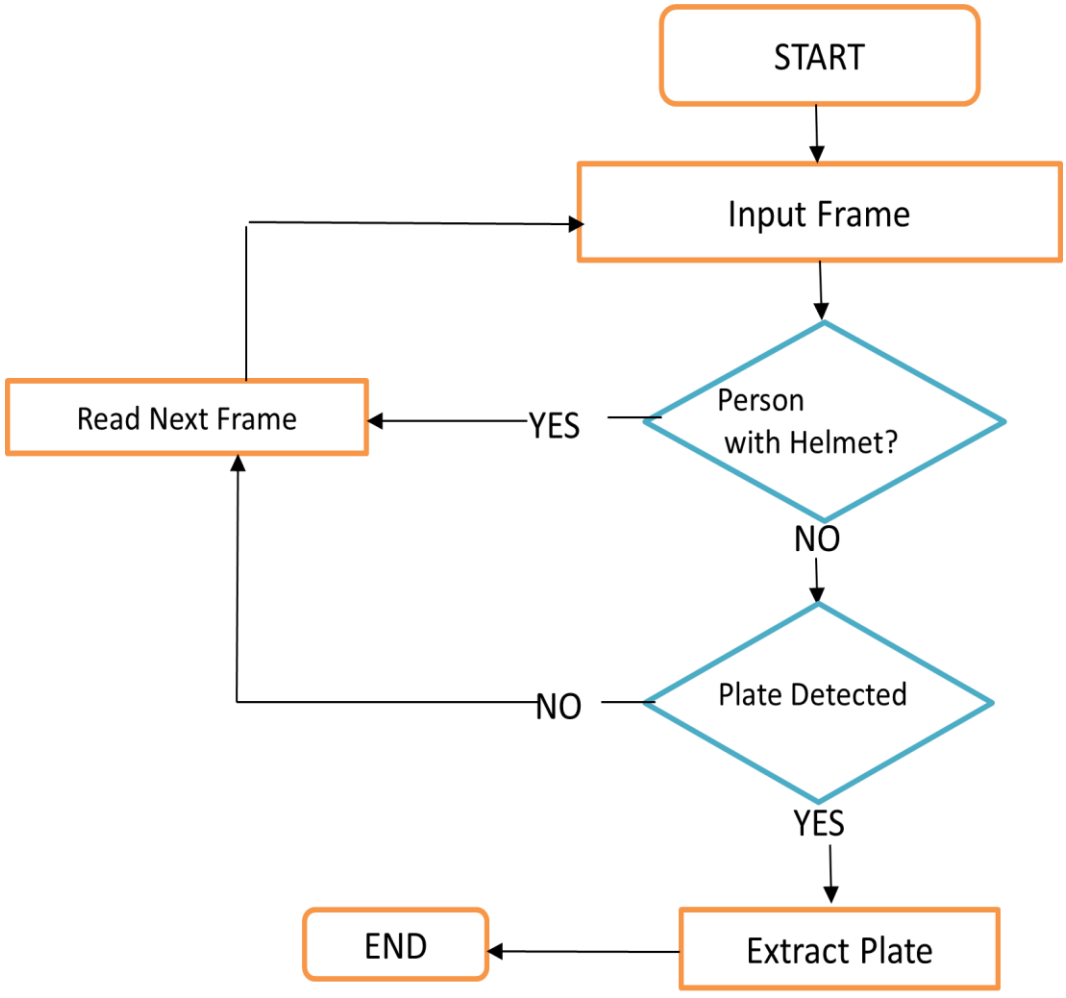


Fig 3.2 System Flow

The Yolov2 was used for real time detection of license plate for a non-helmeted motorcyclist .In our system, the image frame is taken as input, and expected output is the localized license plate for a non helmet motorcyclist The System checks for presence of helmet on motorcyclist. If motorcyclist is without a helmet, the license is extracted

3.3Module Description

1. Upload Image

Upload the image which is trained.

2. Detect Motor Bike & Person

The frame chosen is given as input to YOLOv2 object detection model, where the classes to be detected are „Motorbike", „Person". At the output, image with required class detection along with confidence of detection through bounding box and probability value is obtained.

With the help of functions given by Image AI library, only the detected objects are extracted and stored as separate images and named with class name and image number in order. For example, it will be saved as motorcycle-1, motorcycle-2, etc.… if extracted object is motorcycle or person-1, person-2, etc.… if extracted image is of person. The details of these extracted images which is stored in a dictionary which can be later used for further processing.

3.Detect Helmet

Once the person-motorcycle pair is obtained, the person images are given as input to helmet detection model.

While testing the helmet detection model, some false detection was observed. So, the person image was cropped to get only top one-fourth portion of image. This ensures that false detection cases are eliminated as well as avoid cases leading to wrong results when the rider is holding helmet in hand while riding or keeping it on motorcycle while riding instead of wearing.

4. Exit

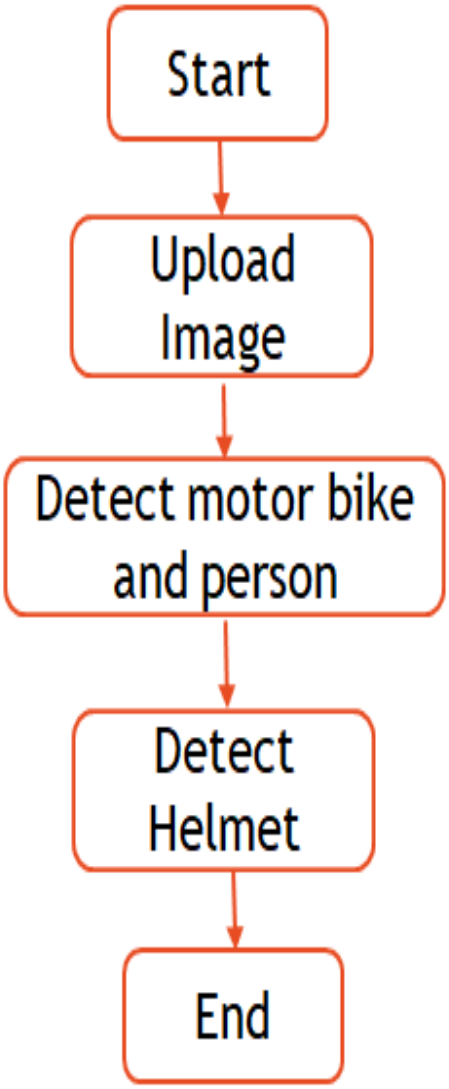


Fig 3.3 Module Description

Chapter 4

Implementation

Thresholding, a segmentation technique, is used to isolate an object from its surrounding environment. The pixel intensity of each pixel is compared to a predetermined threshold in this process. After thresholding, adjacent pixels are joined to form a ternary pattern. A wide range of ternary values can be found in a histogram; therefore the ternary pattern is divided into two binary ones. In order to construct a descriptor twice as large as LBP, histograms are joined together.

The technology used to discover and identify items in an image or image sequence is called object recognition. The color and character elements of a license plate are used to extract it, followed by texture-based segmentation. Low-level image processing techniques such as noise removal may be used first, followed by the extraction of features such as lines, regions, and maybe even areas with specific textures.

It then uses a method known as a feature descriptor to produce feature descriptors/feature vectors from a picture. The convolutional neural network (CNN) is then employed, which includes one or more convolutional layers and is mostly used for image processing, classification, segmentation, and other auto-correlated data processing. ‘Using optical character recognition and thresholding/template matching, a license plate recognition system may then be implemented.

Object Detection

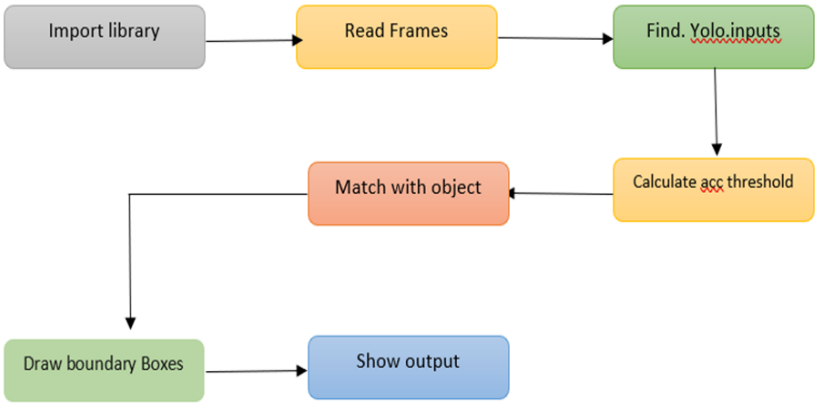


Fig 4.1 Object Detection Block Diagram

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer.

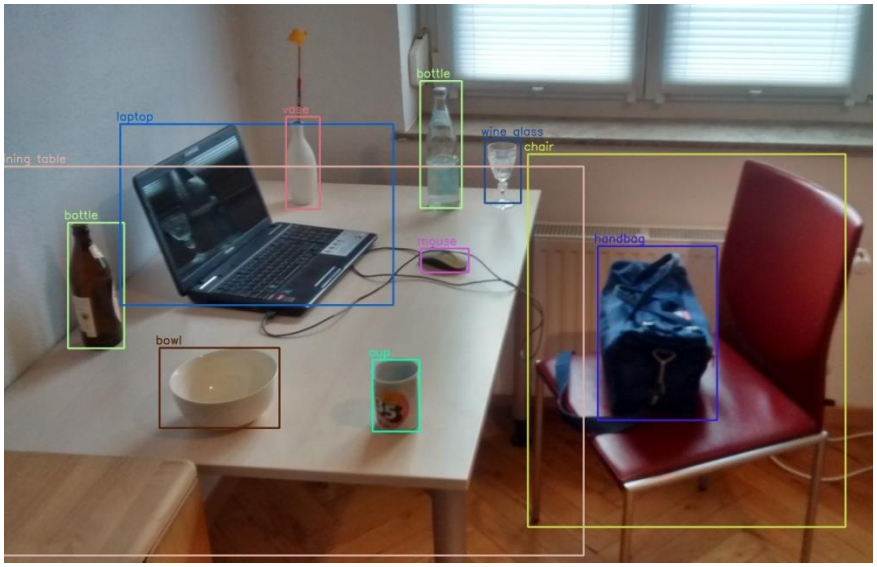


Fig 4.2 Object Detection

4.1 Environmental Setup

Python Installation on Windows

Step 1: Select Version of Python to Install

The installation procedure involves downloading the official Python .exe installer and running it on your system.

Step 2: Download Python Executable Installer

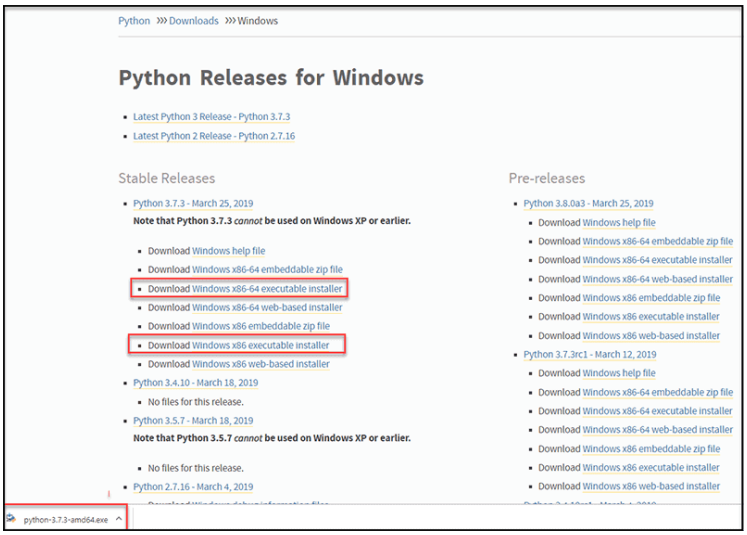


Fig 4.3 Python executable installer

Step 3: Run Executable Installer

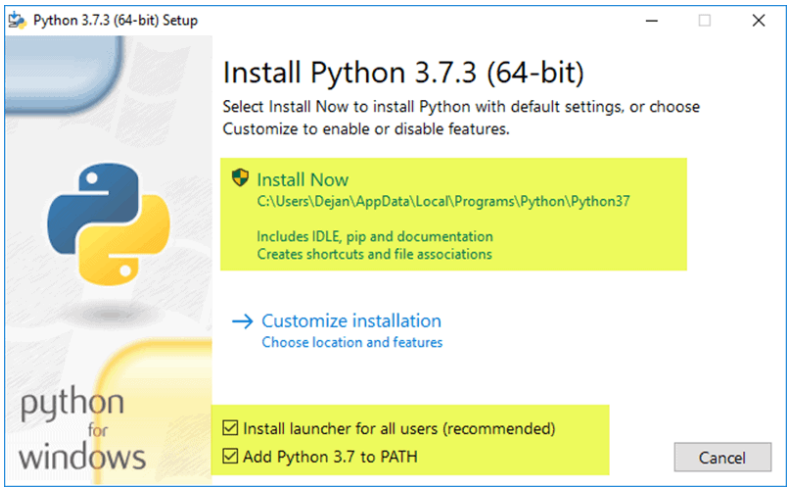


Fig 4.4 Executable Installer

Step 4: Verify Python Was Installed On Windows

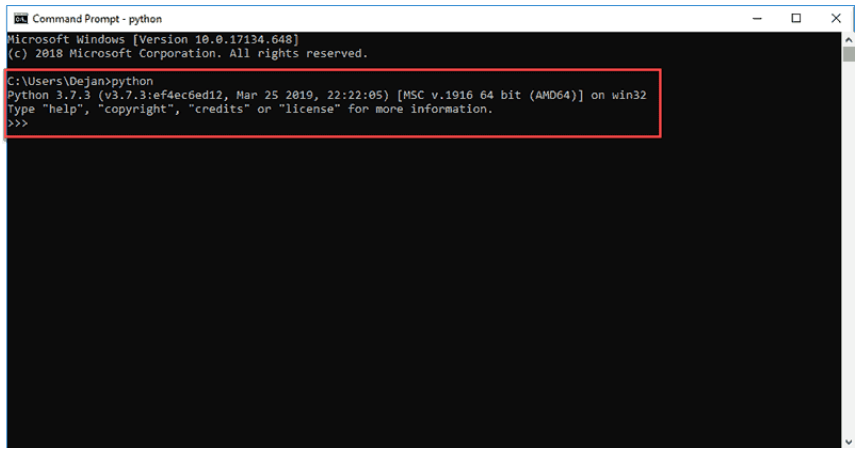


Fig 4.5 Verifying Python Was installed

Step 5: Verify Pip Was Installed

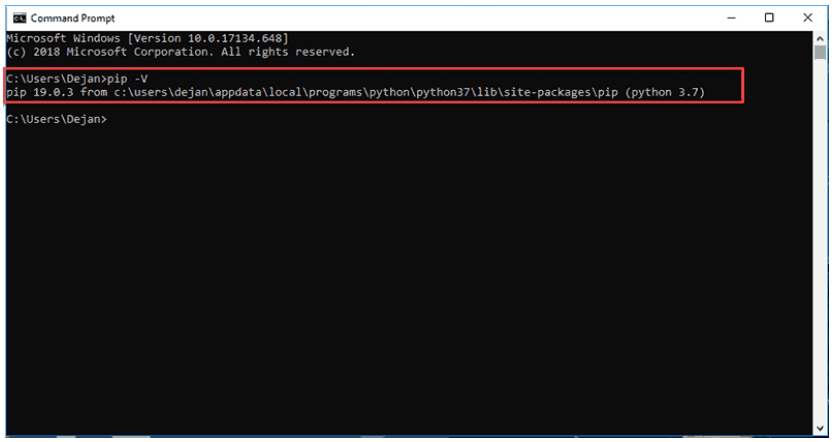


Fig 4.6 Verifying Pip was installed

Step6: Through Pip we have extracted all the libraries

Following libraries are

pip install numpy==1.18.1

pip install matplotlib==3.1.3

pip install pandas==0.25.3

pip install opencv-python==4.2.0.32

pip install keras==2.3.1

pip install tensorflow==1.14.0

pip install h5py==2.10.0

pip install pillow==7.0.0

pip install sklearn-genetic==0.2

pip install SwarmPackagePy

pip install sklearn

pip install scikit-learn==0.22.2.post1

pip install sklearn-extensions==0.0.2

pip install pyswarms==1.1.0

Step7: Click on Run button

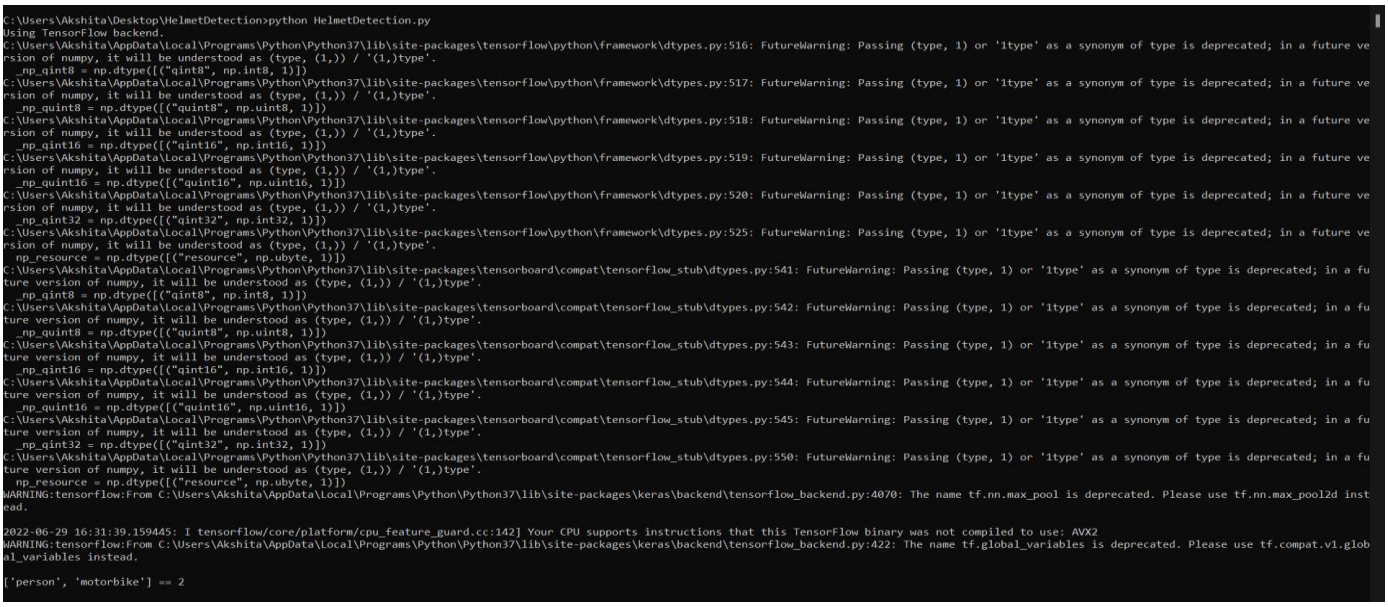


Fig 4.7 Run

Step8: Next we will go to execution page

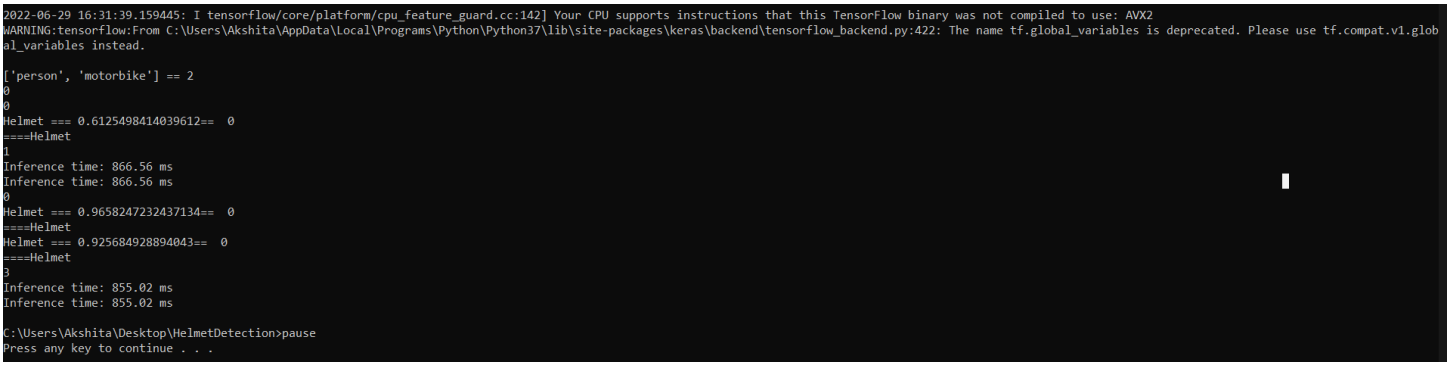


Fig 4.8 Execution page

CHAPTER 5

Evaluation

Datasets

Proper and large dataset is required for all classification research during the training and the testing phase Uploading dataset called ‘dataset.txt’ after uploading dataset.

Here we are taking data set form kaggle

KAGGLE

Fig5.1 Kaggle

Kaggle is an online community platform for **data scientists and machine learning enthusiasts**. Kaggle allows users to collaborate with other users, find and publish datasets, use GPU integrated notebooks, and compete with other data scientists to solve data science challenges

As per the Kaggle website, there are **over 50,000** public datasets and 400,000 public notebooks available. Every day a new dataset is uploaded on Kaggle. Each dataset is a small community where one can discuss data, find relevant public code or create your projects in Kernels.

Let us see few trained data set

FIG 5.2 Person without helmet

Fig 5.3 Person with helmet

5.2 TEST CASES

After setting path double click on 'run.bat' file to run project and to get below screen

1) We will check without helmet image

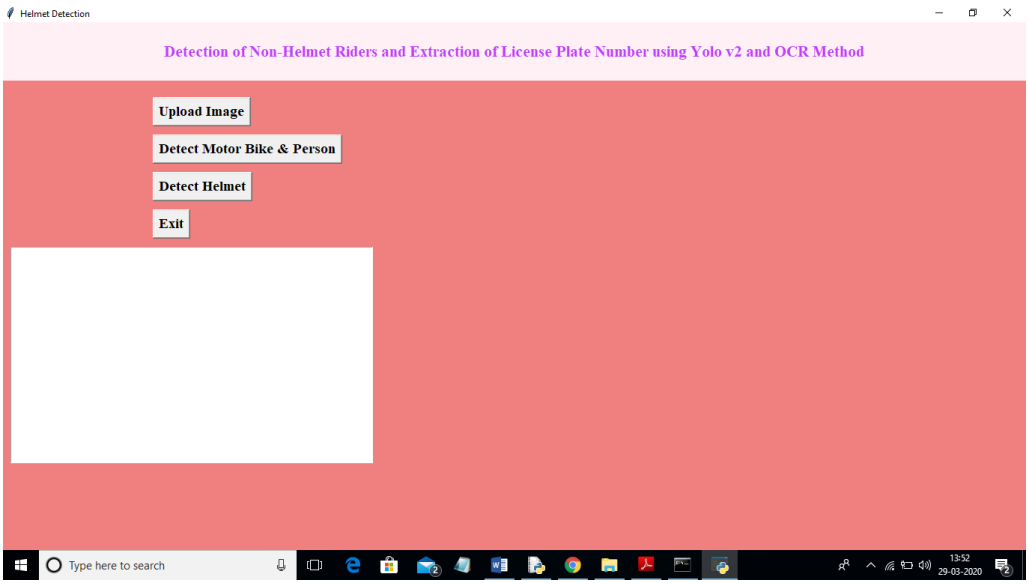


Fig 5.4 home page

In above screen click on 'Upload Image' button and upload image

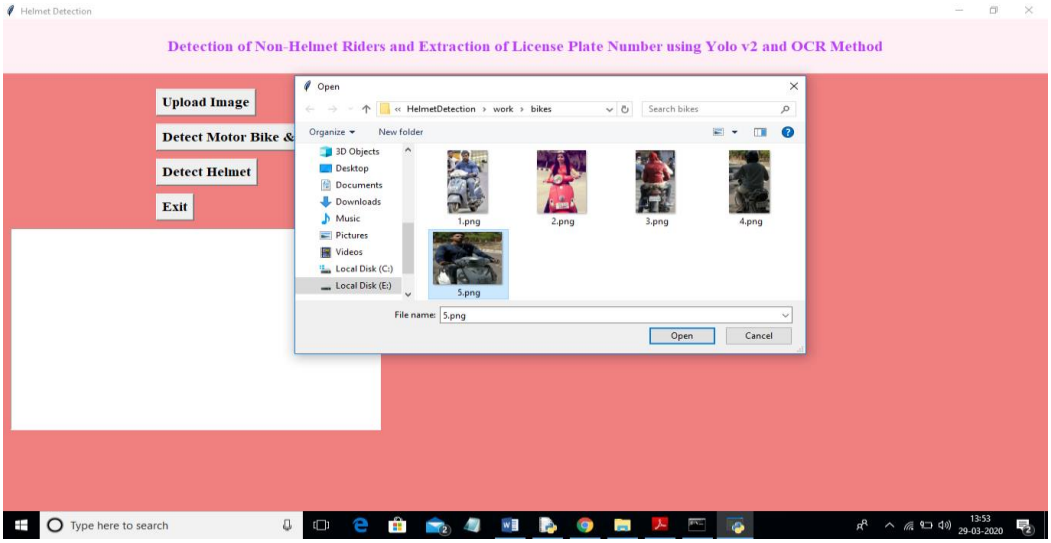


Fig 5.5 Upload image

In above screen I selected one image as '5.png' and click on 'Open' button to load image. Now click on 'Detect Motor Bike & Person' button to detect whether image contains person with motor bike or not

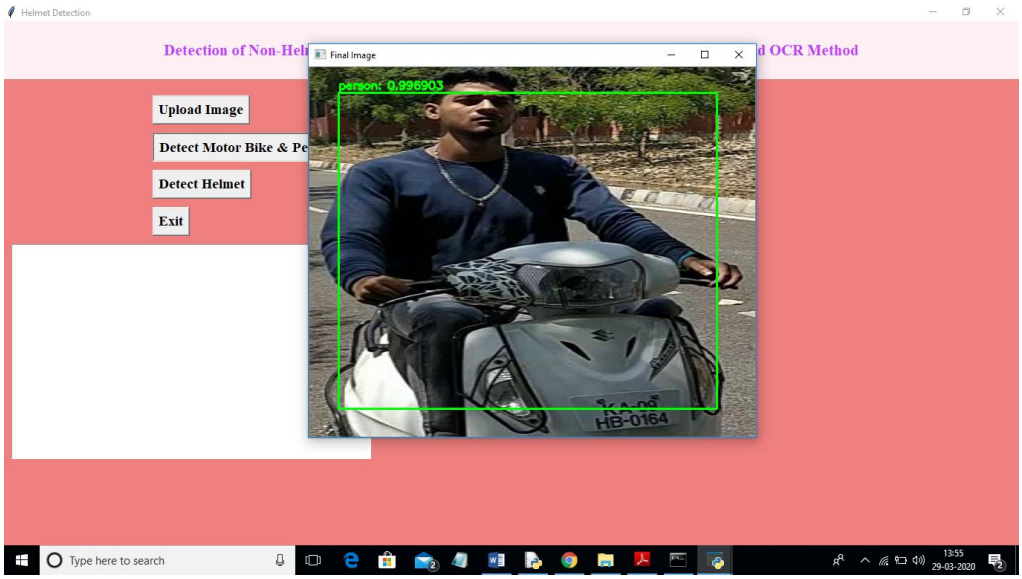


Fig 5.6 detecting helmet or not

In above screen yolo detected image contains person and bike and now click on 'Detect Helmet' button to detect whether he is wearing helmet or not

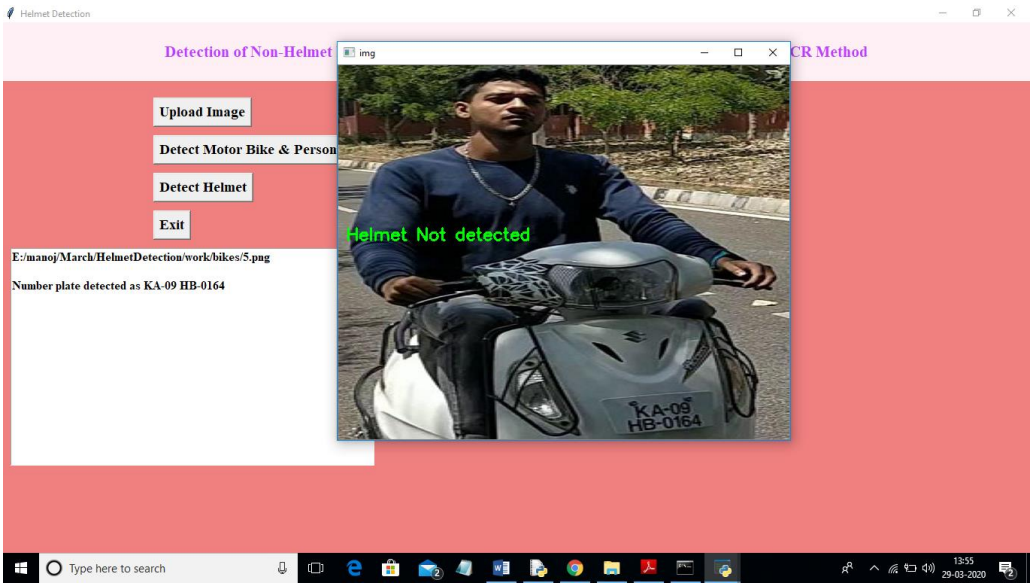
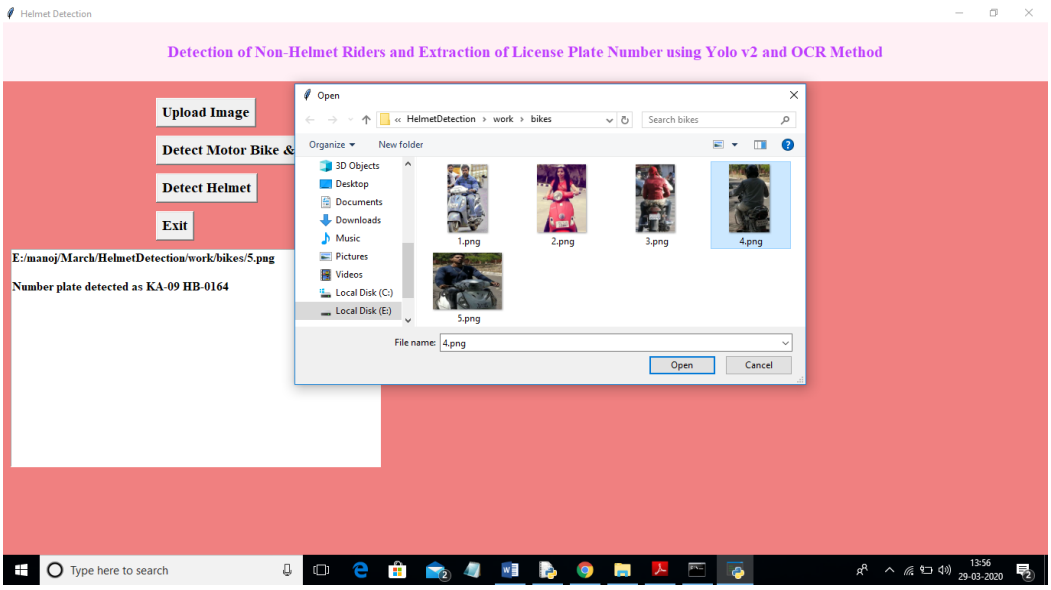


Fig 5.7 helmet not detected

In above screen application detected that person is not wearing helmet and its extracted number from vehicle and display in beside text area.

2) Now we will check with helmet image



In above screen I am uploading 4.png which is wearing helmet and now click on 'Detect Motor Bike & Person' button to get below result,

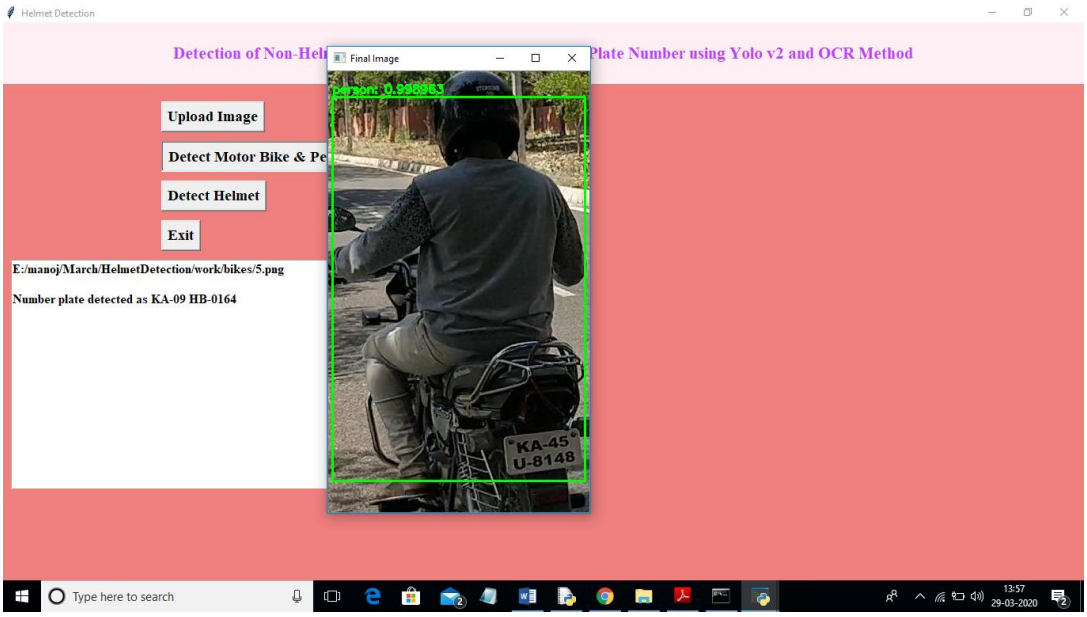


Fig 5.8 Helmet detected

In above screen yolo detected person with motor bike and now click on 'Detect Helmet' button to get below result

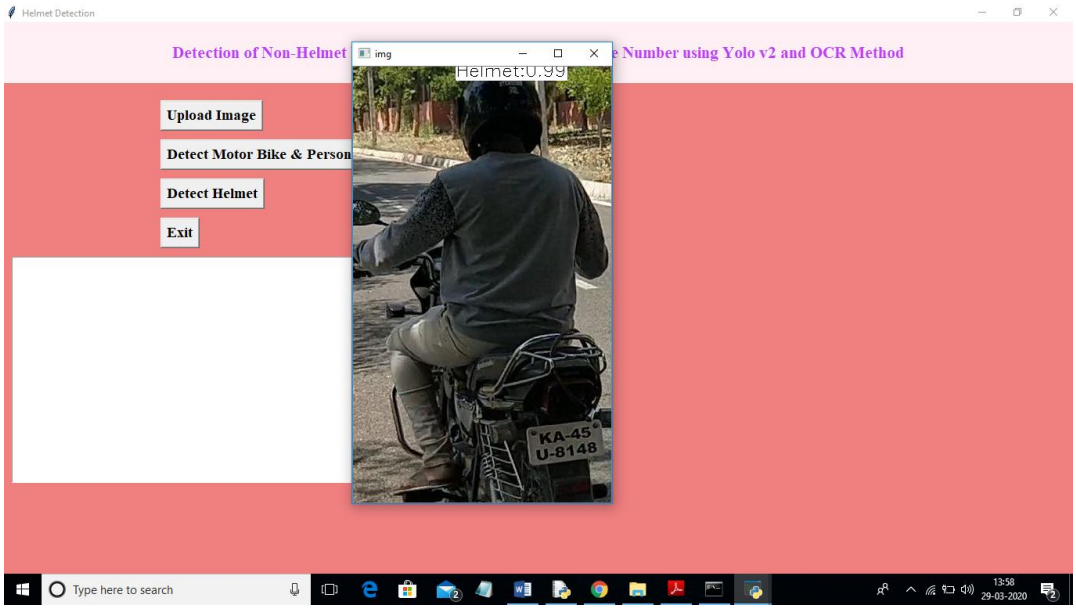
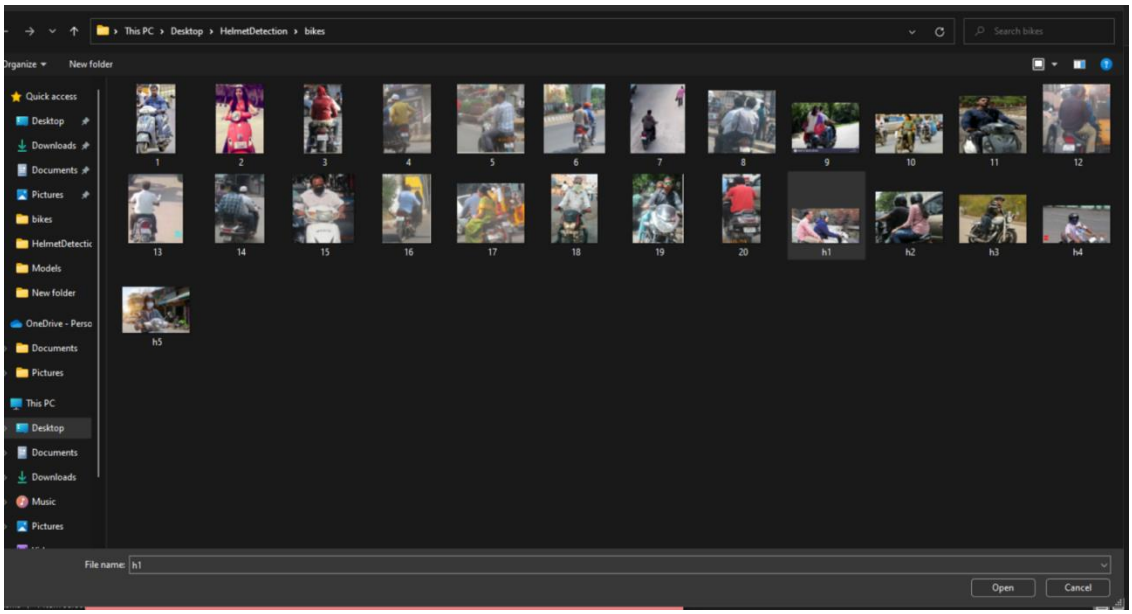


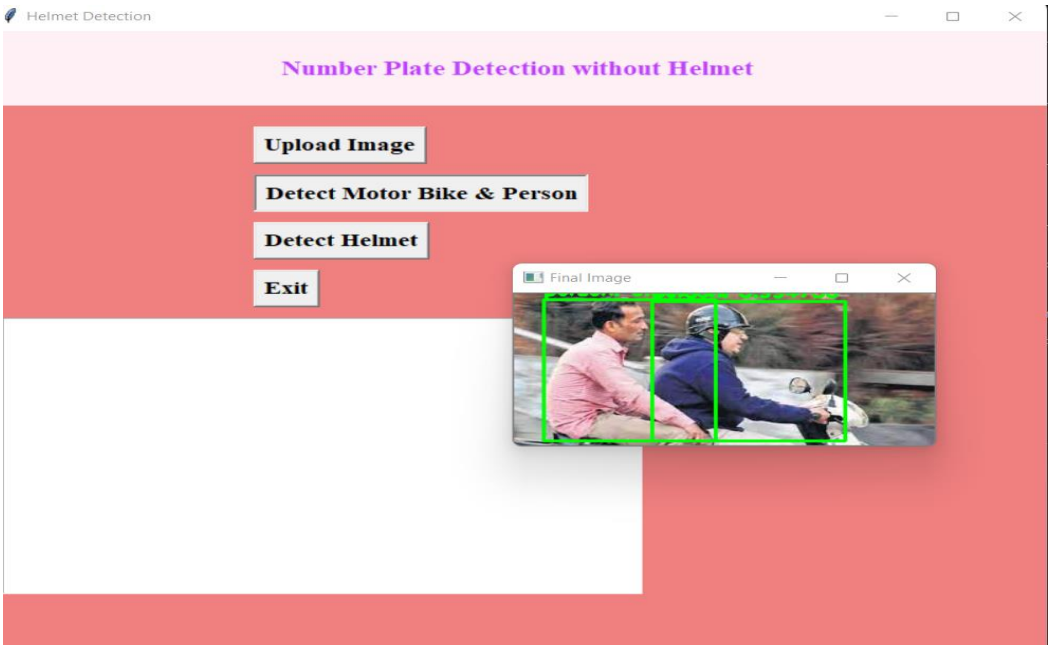
Fig 5.9 person is wearing Helmet

In above screen application detected person is wearing helmet and that label is displaying around his head and application stop there itself and not scanning number plate.

3)Now we will be checking the helmet with a different angle

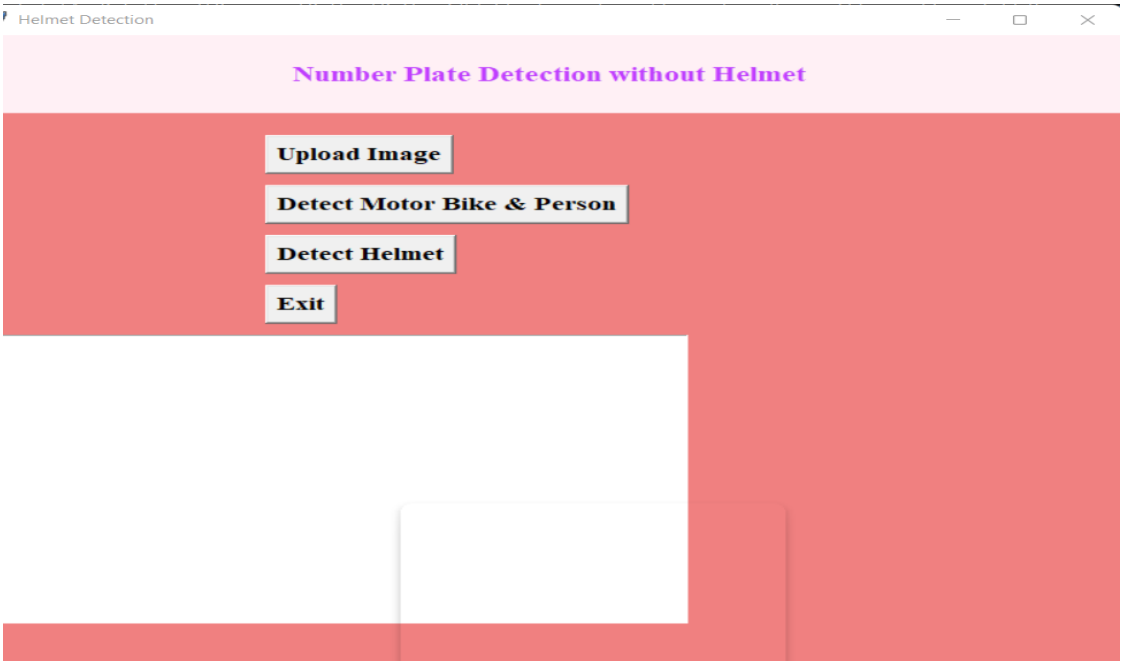


In above screen I selected one image as ‘h1.png’ and click on ‘Open’ button to load image. Now click on ‘Detect Motor Bike & Person’click the button to detect whether image contains person with motor bike or not.



In above screen yolo detected image contains person and bike and now click on 'Detect Helmet' button to detect whether he is wearing helmet or not

In above screen application detected person is wearing helmet and that label is displaying around his head and application stop there itself and not scanning number plate.

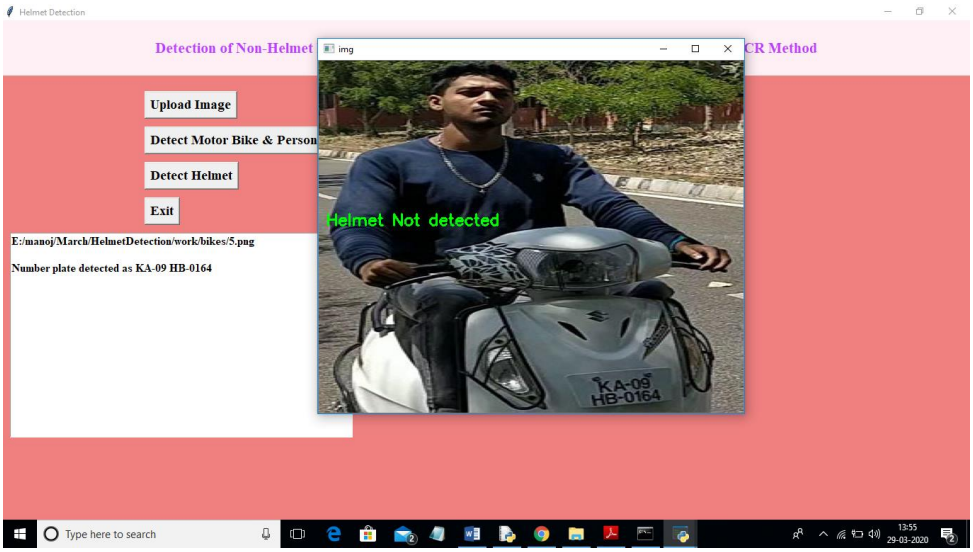


Note: To implement this project and to extract number plate we have trained few images and if u want to extract for new images then send those new images to us, so we include those images in yolo model to extract new images number plate also.

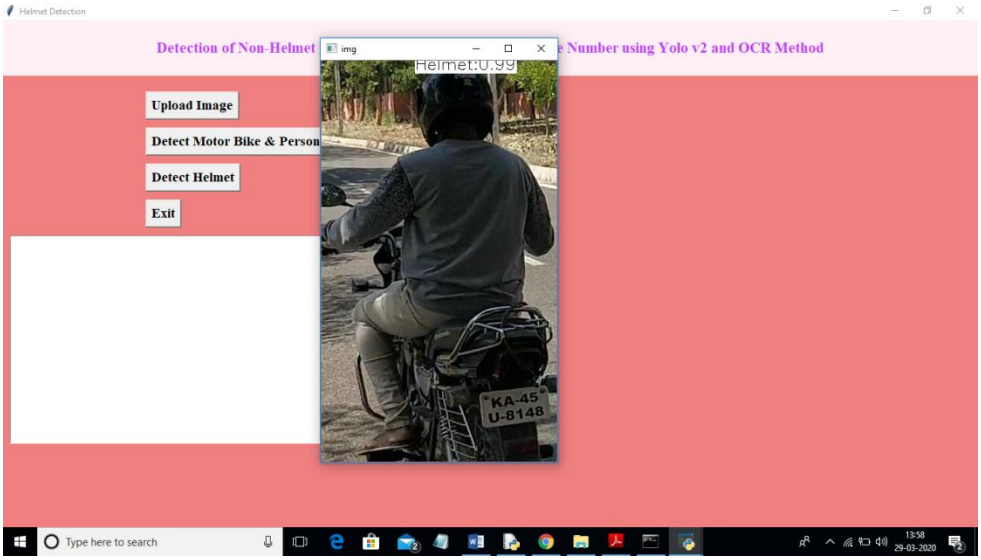
RESULTS

The output image will have two results

1) Detecting number plate without helmet.



2)Detects the helmet.



CHAPTER 6

Conclusion and Future Enhancement

Conclusion

Our goal is to develop a method for identifying cyclists without helmets who are breaking traffic laws. The traffic police would also benefit from the proposed framework, which will help them identify such violators in unusual weather situations, such as the blazing sun. Both the recognition of bike riders and the detection of violators were shown to be accurate in the experiment results With a little tweaking, the proposed architecture can adapt to any new situation.

FUTURE SCOPE

- ✓ We can be linked with traffic cameras and with some modifications it can be used to detect helmets in the real time system, furthermore we can merge the algorithm of automated licence plate detection and make a system which generates challans for those who don't wear helmets.
- ✓ Video frames will be used in the future to extend this notion. We can even recognize the number plate without the helmet quite effectively through films.
- ✓ Automatic Vehicle Recognition System
- ✓ Traffic Rules Violating

6.1 References

[1] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 3, pp . 555–560, March 2008.

[2] B. Duan, W. Liu, P. Fu, C. Yang, X. Wen, and H. Yuan, "Real-time onroad vehicle and motorcycle detection using a single camera," in Procs. of the IEEE Int. Conf. on Industrial Technology (ICIT), pp. 1–6, 10-13 Feb 2009.

[3] C.-C. Chiu, M.-Y. Ku, and H.-T. Chen, "Motorcycle detection and tracking system with occlusion segmentation," in Int. Workshop on Image Analysis for Multimedia Interactive Services, Santorini, pp. 32–32 June 2007.

[4] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 34, no. 3, pp. 334S–352, Aug 2004.

[5] J. Chiverton, "Helmet presence classification with motorcycle detection and tracking," Intelligent Transport ystems (IET), vol. 6, no. 3, pp. 259–269, September 2012.

[6] Z. Chen, T. Ellis, and S. Velastin, "Vehicle detection, tracking and classification in urban traffic," in Procs. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITS), Anchorage, AK pp. 951–956, Sept 2012.

[7] R. Silva, K. Aires, T. Santos, K. Abdala, R. Veras, and A. Soares, "Automatic detection of motorcyclists without helmet," in Computing Conf. (CLEI), XXXIX Latin American, pp. 1–7, Oct 2013.

[8] R. Rodrigues Veloso e Silva, K. Teixeira Aires, and R. De Melo Souza Veras, "Helmet detection on motorcyclists using image descriptors and classifiers," in Procs. of the Graphics, Patterns and Images (SIBGRAPI), pp. 141–148,Aug 2014.

[9] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in Proc. of the Int . Conf. on Pattern Recognition (ICPR), vol. 2, pp. 28–31, Aug.23-26 2004.

6.2 Appendix

yoloDetection.py

```
import numpy as np
import argparse
import cv2 as cv
import subprocess
import time
import os

def detectObject(CNNnet, total layer names, image height, image width, image, name colors,
class_labels,indexno,
            Boundingboxes=None, confidence value=None, class ids=None, ids=None, detect=True):

    if detect:
        blob_object = cv.dnn.blobFromImage(image,1/255.0,(416, 416),swapRB=True,crop=False)
        CNNnet.setInput(blob object)
        cnn_outs_layer = CNNnet.forward(total_layer_names)
        Boundingboxes, confidence_value, class_ids = listBoundingBoxes(cnn_outs_layer,
image_height, image_width, 0.5)
        ids = cv.dnn.NMSBoxes(Boundingboxes, confidence_value, 0.5, 0.3)
        if Boundingboxes is None or confidence_value is None or ids is None or class_ids is
None:
           raise '[ERROR] unable to draw boxes.'
        image,option = labelsBoundingBoxes(image, Boundingboxes, confidence_value, class_ids,
ids, name_colors, class_labels,indexno)

    return image,option


def labelsBoundingBoxes(image, Boundingbox, conf_thr, classID, ids, color_names,
predicted_labels,indexno):
    option = 0
    if len(ids) > 0:
        for i in ids.flatten():
            # draw boxes
            xx, yy = Boundingbox[i][0], Boundingbox[i][1]
            width, height = Boundingbox[i][2], Boundingbox[i][3]

            class_color = (0,255,0)#[int(color) for color in color_names[classID[i]]]

            cv.rectangle(image, (xx, yy), (xx+width, yy+height), class_color, 2)
            print(classID[i])
            if classID[i] <= 1:
                text_label = "{}: {:4f}".format(predicted_labels[classID[i]], conf_thr[i])
                #displayImage(image,indexno)
                cv.putText(image, text_label, (xx, yy-5), cv.FONT_HERSHEY_SIMPLEX, 0.5,
class_color, 2)
                option = 1

    return image,option


def listBoundingBoxes(image, image_height, image_width, threshold_conf):
    box array = []
    confidence_array = []
    class_ids_array = []
```

```
    for img in image:
        for obj_detection in img:
            detection_scores = obj_detection[5:]
            class_id = np.argmax(detection_scores)
            confidence_value = detection_scores[class_id]
            if confidence value > threshold conf and class id <= 1:
                Boundbox   =   obj_detection[0:4]   *   np.array([image_width,   image_height,
image_width, image_height])
                center_X, center_Y, box_width, box_height = Boundbox.astype('int')

                xx = int(center_X - (box_width / 2))
                yy = int(center Y - (box height / 2))

                box_array.append([xx, yy, int(box_width), int(box_height)])
                confidence_array.append(float(confidence_value))
                class_ids_array.append(class_id)

    return box_array, confidence_array, class_ids_array

def displayImage(image,index):
    #cv.imwrite('bikes/'+str(index)+'.jpg',image)
    #index = index + 1
    cv.imshow("Final Image", image)
    cv.waitKey(0)
```

HelmetDetection.py

```
from tkinter import *
import tkinter
from tkinter import filedialog
import numpy as np
from tkinter.filedialog import askopenfilename
import pandas as pd
from tkinter import simpledialog
import numpy as np
import cv2 as cv
import subprocess
import time
import os
from yoloDetection import detectObject, displayImage
import sys
from time import sleep
from tkinter import messagebox
import pytesseract as tess
from keras.models import model_from_json
from keras.utils.np_utils import to_categorical

main = tkinter.Tk()
main.title("Helmet Detection") #designing main screen
main.geometry("800x700")

global filename
global loaded_model

global class_labels
global cnn_model
global cnn_layer_names

frame_count = 0
frame_count_out=0

confThreshold = 0.5
nmsThreshold = 0.4
inpWidth = 416
inpHeight = 416
global option
labels_value = []
with open("Models/labels.txt", "r") as file: #reading MRC dictionary
    for line in file:
        line = line.strip('\n')
        line = line.strip()
        labels_value.append(line)
    file.close()


with open('Models/model.json', "r") as json_file:
    loaded_model_json = json_file.read()
    plate_detecter = model_from_json(loaded_model_json)

plate_detecter.load_weights("Models/model_weights.h5")
plate_detecter._make_predict_function()

classesFile = "Models/obj.names";
classes = None

with open(classesFile, 'rt') as f:
```

```
    classes = f.read().rstrip('\n').split('\n')

modelConfiguration = "Models/yolov3-obj.cfg";
modelWeights = "Models/yolov3-obj_2400.weights";

net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)

def getOutputsNames(net):
    layersNames = net.getLayerNames()
    return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]

def loadLibraries(): #function to load yolov3 model weight and class labels
        global class_labels
        global cnn_model
        global cnn_layer_names
        class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n') #reading
labels from yolov3 model
        print(str(class_labels)+" == "+str(len(class_labels)))
        cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg',
'yolov3model/yolov3.weights') #reading model
        cnn_layer_names = cnn_model.getLayerNames() #getting layers from cnn model
        cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in
cnn_model.getUnconnectedOutLayers()] #assigning all layers


def upload(): #function to upload tweeter profile
    global filename
    filename = filedialog.askopenfilename(initialdir="bikes")
    #messagebox.showinfo("File Information", "image file loaded")



def detectBike():
    global option
    option = 0
    indexno = 0
    label_colors = (0,255,0)
    try:
        image = cv.imread(filename)
        image_height, image_width = image.shape[:2]
    except:
        raise 'Invalid image path'
    finally:
        image, ops = detectObject(cnn_model, cnn_layer_names, image_height, image_width,
image, label_colors, class_labels,indexno)
        if ops == 1:
            displayImage(image,0)#display image with detected objects label
            option = 1
        else:
            displayImage(image,0)




def drawPred(classId, conf, left, top, right, bottom,frame,option):
    global frame_count
    #cv.rectangle(frame, (left, top), (right, bottom), (255, 178, 50), 3)
    label = '%.2f' % conf
    if classes:
        assert(classId < len(classes))
        label = '%s:%s' % (classes[classId], label)
```

```
    labelSize, baseLine = cv.getTextSize(label, cv.FONT_HERSHEY_SIMPLEX, 0.5, 1)
    top = max(top, labelSize[1])
    label_name,label_conf = label.split(':')
    print(label_name+" === "+str(conf)+"==  "+str(option))
    if label_name == 'Helmet' and conf > 0.50:
        if option == 0 and conf > 0.90:
            cv.rectangle(frame, (left, top - round(1.5*labelSize[1])), (left +
round(1.5*labelSize[0]), top + baseLine), (255, 255, 255), cv.FILLED)
            cv.putText(frame, label, (left, top), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)
            frame_count+=1
        if option == 0 and conf < 0.90:
            cv.putText(frame, "Helmet Not detected", (10, top), cv.FONT_HERSHEY_SIMPLEX, 0.75,
(0,255,0), 2)
            frame_count+=1
            img = cv.imread(filename)
            img = cv.resize(img, (64,64))
            im2arr = np.array(img)
            im2arr = im2arr.reshape(1,64,64,3)
            X = np.asarray(im2arr)
            X = X.astype('float32')
            X = X/255
            preds = plate_detecter.predict(X)
            predict = np.argmax(preds)
            #img = cv.imread(filename)
            #img = cv.resize(img,(500,500))
            #text = tess.image_to_string(img, lang='eng')
            #text = text.replace("\n","")
            #messagebox.showinfo("Number Plate Detection Result", "Number plate detected as
"+text)
            textarea.insert(END,filename+"\n\n")
            textarea.insert(END,"Number plate detected as "+str(labels_value[predict]))
        if option == 1:
            cv.rectangle(frame, (left, top - round(1.5*labelSize[1])), (left +
round(1.5*labelSize[0]), top + baseLine), (255, 255, 255), cv.FILLED)
            cv.putText(frame, label, (left, top), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)
            frame_count+=1


    if(frame_count> 0):
        return frame_count

def postprocess(frame, outs, option):
    frameHeight = frame.shape[0]
    frameWidth = frame.shape[1]
    global frame_count_out
    frame_count_out=0
    classIds = []
    confidences = []
    boxes = []
    classIds = []
    confidences = []
    boxes = []
    cc = 0
    for out in outs:
        for detection in out:
            scores = detection[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if confidence > confThreshold:
                center_x = int(detection[0] * frameWidth)
                center_y = int(detection[1] * frameHeight)
                width = int(detection[2] * frameWidth)
                height = int(detection[3] * frameHeight)
```

```
                left = int(center_x - width / 2)
                top = int(center_y - height / 2)
                classIds.append(classId)
                #print(classIds)
                confidences.append(float(confidence))
                boxes.append([left, top, width, height])

    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
    count_person=0 # for counting the classes in this loop.
    for i in indices:
        i = i[0]
        box = boxes[i]
        left = box[0]
        top = box[1]
        width = box[2]
        height = box[3]
        frame_count_out = drawPred(classIds[i], confidences[i], left, top, left + width, top +
height,frame,option)
        my_class='Helmet'
        unknown_class = classes[classId]
        print("===="+str(unknown_class))
        if my_class == unknown_class:
            count_person += 1

    print(str(frame_count_out))
    if count_person == 0 and option == 1:
        cv.putText(frame, "Helmet Not detected", (10, 50), cv.FONT_HERSHEY_SIMPLEX, 0.75,
(0,255,0), 2)
    if count_person >= 1 and option == 0:
        #path = 'test_out/'
        #cv.imwrite(str(path)+str(cc)+".jpg", frame)     # writing to folder.
        #cc = cc + 1
        frame = cv.resize(frame,(500,500))
        cv.imshow('img',frame)
        cv.waitKey(50)


def detectHelmet():
    textarea.delete('1.0', END)
    if option == 1:
        frame = cv.imread(filename)
        frame_count =0
        blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1,
crop=False)
        net.setInput(blob)
        outs = net.forward(getOutputsNames(net))
        postprocess(frame, outs,0)
        t, _ = net.getPerfProfile()
        label = 'Inference time: %.2f ms' % (t * 1000.0 / cv.getTickFrequency())
        print(label)
        cv.putText(frame, label, (0, 15), cv.FONT HERSHEY SIMPLEX, 0.5, (0, 0, 255))
        print(label)
    else:
        messagebox.showinfo("Person & Motor bike not detected in uploaded image", "Person &
Motor bike not detected in uploaded image")


def videoHelmetDetect():
    global filename
    videofile = askopenfilename(initialdir = "videos")
    video = cv.VideoCapture(videofile)
    while(True):
        ret, frame = video.read()
```

```
        if ret == True:
            frame count = 0
            filename = "temp.png"
            cv.imwrite("temp.png",frame)
            blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1,
crop=False)
            net.setInput(blob)
            outs = net.forward(getOutputsNames(net))
            postprocess(frame, outs,1)
            t, _ = net.getPerfProfile()
            #label=''
            #cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))
            cv.imshow("Predicted Result", frame)
            if cv.waitKey(5) & 0xFF == ord('q'):
                break
        else:
            break
    video.release()
    cv.destroyAllWindows()


def exit():
    global main
    main.destroy()

font = ('times', 16, 'bold')
title = Label(main, text='Number Plate Detection without Helmet', justify=LEFT)
title.config(bg='lavender blush', fg='DarkOrchid1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=100,y=5)
title.pack()

font1 = ('times', 14, 'bold')
model = Button(main, text="Upload Image", command=upload)
model.place(x=200,y=100)
model.config(font=font1)

uploadimage = Button(main, text="Detect Motor Bike & Person", command=detectBike)
uploadimage.place(x=200,y=150)
uploadimage.config(font=font1)

classifyimage = Button(main, text="Detect Helmet", command=detectHelmet)
classifyimage.place(x=200,y=200)
classifyimage.config(font=font1)

exitapp = Button(main, text="Exit", command=exit)
exitapp.place(x=200,y=250)
exitapp.config(font=font1)

font1 = ('times', 12, 'bold')
textarea=Text(main,height=15,width=60)
scroll=Scrollbar(textarea)
textarea.configure(yscrollcommand=scroll.set)
textarea.place(x=10,y=300)
textarea.config(font=font1)

loadLibraries()

main.config(bg='light coral')
main.mainloop()
```

A) Sample code

yolo.py

```
import numpy as np
import cv2 as cv
import subprocess
import time
import os
from yoloDetection import detectObject, displayImage
import sys


global class_labels
global cnn_model
global cnn_layer_names

def loadLibraries(): #function to load yolov3 model weight and class labels
        global class labels
        global cnn_model
        global cnn_layer_names
        class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n') #reading
labels from yolov3 model
        print(str(class_labels)+" == "+str(len(class_labels)))
        cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg',
'yolov3model/yolov3.weights') #reading model
        cnn_layer_names = cnn_model.getLayerNames() #getting layers from cnn model
        cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in
cnn_model.getUnconnectedOutLayers()] #assigning all layers

def detectFromImage(imagename): #function to detect object from images
        #random colors to assign unique color to each label
        label_colors =
(0,255,0)#np.random.randint(0,255,size=(len(class_labels),3),dtype='uint8')
        try:
                image = cv.imread(imagename) #image reading
                image_height, image_width = image.shape[:2] #converting image to two
dimensional array
        except:
                raise 'Invalid image path'
        finally:
                image, _, _, _, _ = detectObject(cnn_model, cnn_layer_names, image_height,
image_width, image, label_colors, class_labels,indexno)#calling detection function
                displayImage(image,0)#display image with detected objects label

def detectFromVideo(videoFile): #function to read objects from video

        #random colors to assign unique color to each label
        label_colors =
(0,255,0)#np.random.randint(0,255,size=(len(class_labels),3),dtype='uint8')
        indexno = 0
        try:

                video = cv.VideoCapture(videoFile)
                frame_height, frame_width = None, None  #reading video from given path
                video_writer = None
        except:
                raise 'Unable to load video'
        finally:
                while True:
```

```
 frame_grabbed, frames = video.read() #taking each frame from video
 #print(frame grabbed)
 if not frame_grabbed: #condition to check whether video loaded or not
 break
 if frame_width is None or frame_height is None:
 frame_height, frame_width = frames.shape[:2] #detecting object from frame
 frames,  ,  ,  ,   = detectObject(cnn model, cnn layer names, frame height, frame width,
frames, label_colors, class_labels,indexno)
 #displayImage(frames,index)
 #indexno = indexno + 1
 print(indexno)
if indexno == 5:
 video.release()
 break

        print ("Releasing resources")
        #video_writer.release()
        video.release()


if __name__ == '__main__':
        loadLibraries()
        print("sample commands to run code with image or video")
        print("python yolo.py image input_image_path")
        print("python yolo.py video input_video_path")
        if len(sys.argv) == 3:
                if sys.argv[1] == 'image':
                        detectFromImage(sys.argv[2])
                elif sys.argv[1] == 'video':
                        detectFromVideo(sys.argv[2])
                else:
                        print("invalid input")
        else:
                print("follow sample command to run code")


        #video_path = None
        #video_output_path = "out.avi"
```