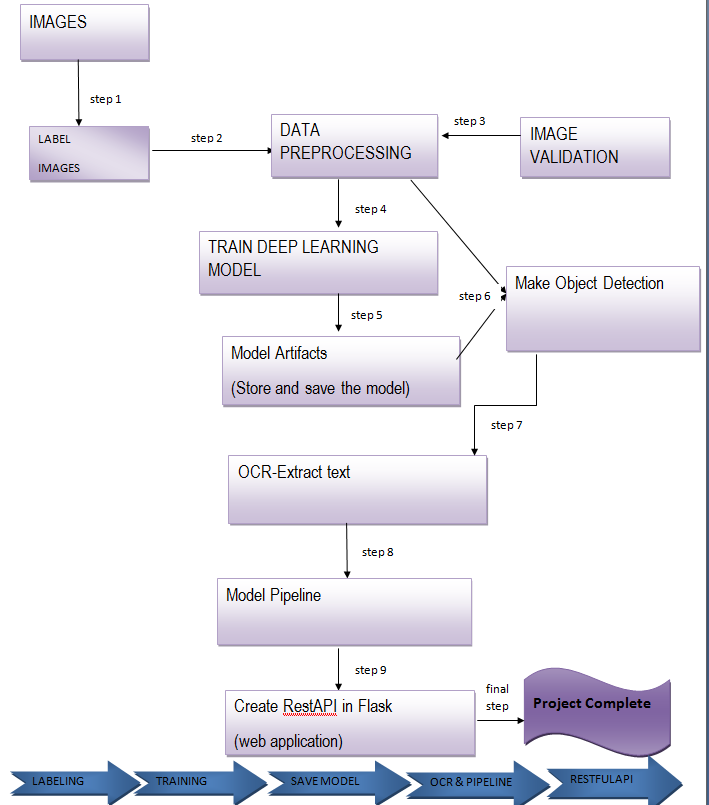
**AUTOMATIC NUMBER PLATE RECOGNITION AND OCR PROJECT**

Introduction: Automatic number-plate recognition is a technology that uses optical character recognition on images to read vehicle registration plates to create vehicle location data.It can be used to store the images captured by the cameras as well as the text from the license plate

Some of the Application:

1. Vehicle Parking (commercial or resedential area)
2. Toll Gates
3. Vehical Survelliance (used by police to detect vehicals)
4. Traffic control Department.

**PROJECT ARCITECTURE**



Pacakages and library use:

1. OpenCV
2. Tenssorflow
3. Glob
4. Panda
5. ElementTree
6. os
7. matplotlib
8. sklearn
9. numpy
10. tensorflow

what is panda?

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks.

what is opencv?

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.

what is glob?

The glob module is a useful part of the Python standard library. glob (short for global) is used to return all file paths that match a specific pattern. These patterns are similar to regular expressions but much simpler. Asterisk (\*): Matches zero or more characters Question Mark (?) Matches exactly one character

what is tenssorflow?

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

what is os?

The OS module in Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system-dependent functionality.

what is matplotlib?

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.

what is numpy?

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It is an open source project and you can use it freely.NumPy stands for Numerical Python.

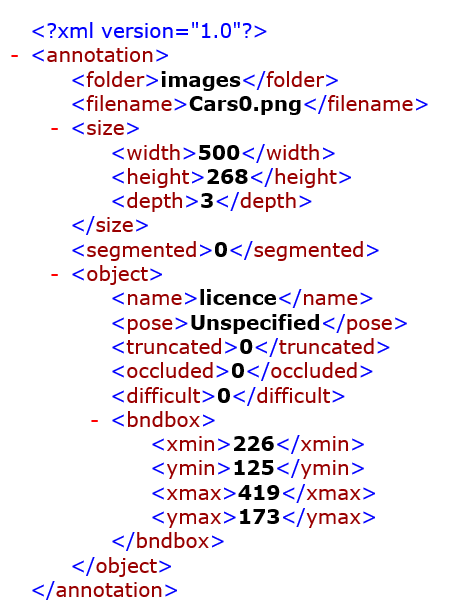
what is sklearn?

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

What is XML and elementree?

XML creates a tree-like structure that is easy to interpret and supports a hierarchy.XML stands for "Extensible Markup Language" The largest, top-level element is called the root, which contains all other elements and the other sub elements are child. The XML tree structure makes navigation, modification, and removal relatively simple programmatically. Python has a built in library, ElementTree, that has functions to read and manipulate XMLs (and other similarly structured files).

**images labels and xml**

****

Root element

**------------------------------------------------------------------------------------------------------------------------------**

**Converting XML to CSV**

.

from glob import glob

path=glob('./images/\*.xml')

path

#All the xml file where loaded and output was list.

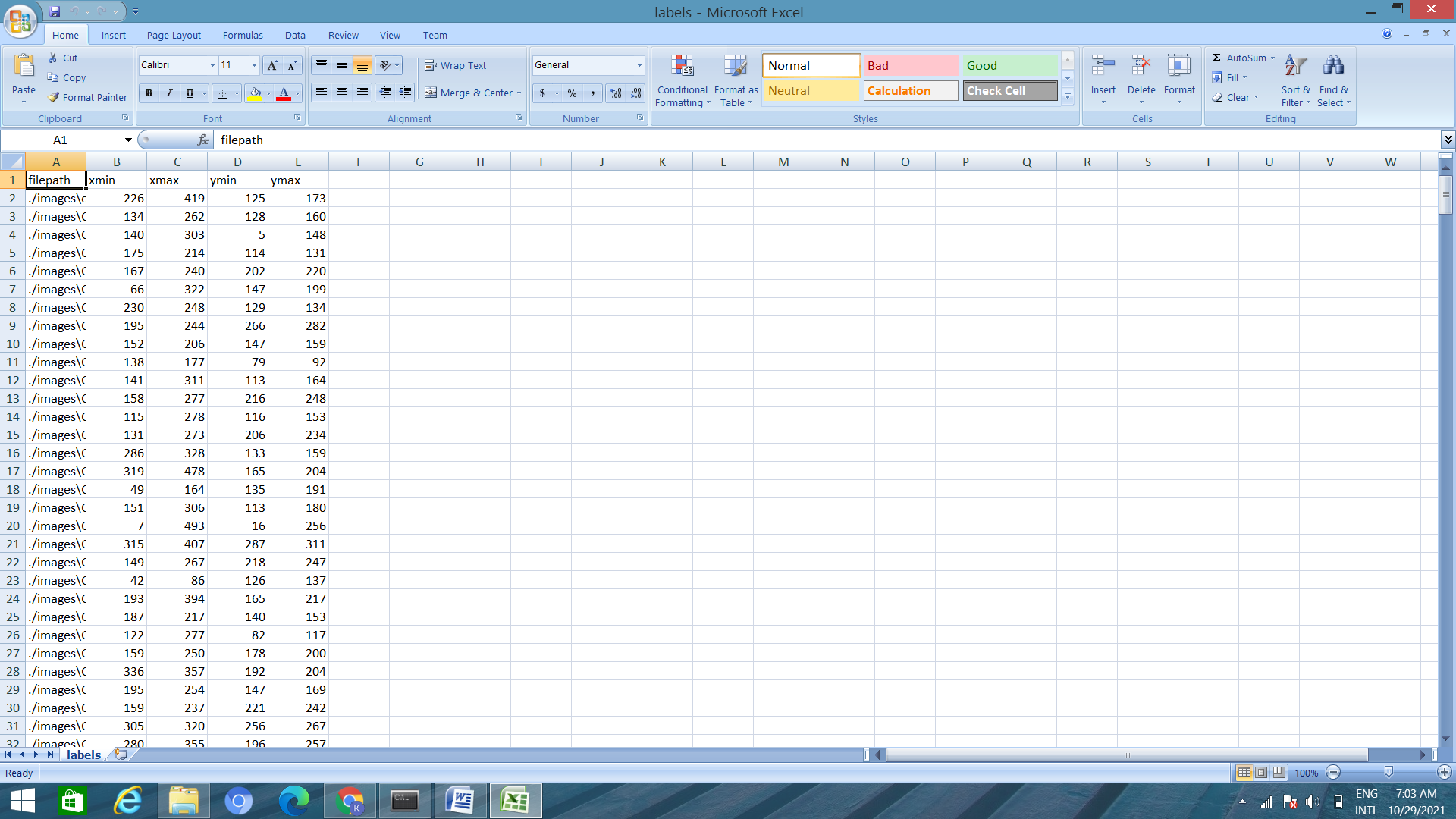
df=pd.DataFrame(labels\_dict)

df

#Then the list was converted into dictionary so that it could be converted into data frame and furthur process can be done.

df.to\_csv('labels.csv',index=False)

#After converting into data frame the xml file was converted into csv file.



**Object detection**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

import cv2

#All the necessary pacakges was installed.

df = pd.read\_csv('labels.csv')

df.head()

#Data frame was read

def getfilename(filename):

filename\_image=xet.parse(filename).getroot().find('filename').text

#**getroot()** This function returns root element of the tree

filepath\_image=os.path.join('./images',filename\_image)

#**os.path.join()** method in Python join one or more path components

return filepath\_image

#Image path was acess by creating a function

image\_path = list(df['filepath'].apply(getFilename))

#data framewas converted into list and all the images was stored by applying the finction tht was created.

image\_path

**reading image**

img=cv2.imread(file\_path)

cv2.namedWindow('example',cv2.WINDOW\_NORMAL)

# **namedWindow() method**is used to create a window with a suitable name and size to display images and videos on the screen.

cv2.imshow('example',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Then the output and image was verified by displaying the image by imread() and imshow().

* What is imread() and imshow()?

1. imread() helps us read an image

2. imshow() displays an image in a window

**Image loaded**



.

**reading image**

cv2.rectangle(img,(226,125),(419,173),(0,255,0),3)

# cv2.rectangle() method is used to draw a rectangle on any image.

*#cv2.rectangle(image, start\_point, end\_point, color, thickness)*

cv2.namedWindow('example',cv2.WINDOW\_NORMAL)

cv2.imshow('example',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**image loaded**

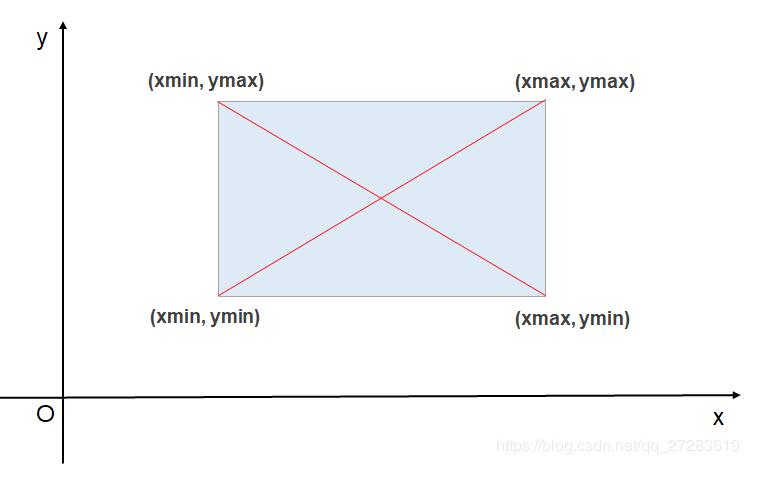


**Parsing image and loaded creating bndbox**

While parsing image data of bounding box is read

**What is bounding box?**

Bounding boxes are rectangles that mark objects on an image. There are multiple formats of bounding boxes annotations The bounding box has the following (x, y) coordinates of its corners: top-left is (x\_min, y\_min) or (98px, 345px), top-right is (x\_max, y\_min) or (420px, 345px), bottom-left is (x\_min, y\_max) or (98px, 462px), bottom-right is (x\_max, y\_max) or (420px, 462px). As you see, coordinates of the bounding box's corners are calculated with respect to the top-left corner of the image which has (x, y) coordinates (0, 0).



**Data processing**

Why we need to process data in Machine Learning?

Data preprocessing is an integral step in Machine Learning **as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn**; therefore, it is extremely important that we preprocess our data before feeding it into our model.

**Step 1:**

from sklearn.model\_selection import train\_test\_split

#train\_test\_split is a function for splitting data array in two subset i.e for training and testing

from tensorflow.keras.preprocessing.image import load\_img,img\_to\_array

#keras provides the load\_img() function for loading an image from file as a PIL image object

**#** Keras provides the img\_to\_array() function for converting a loaded image in PIL format into a NumPy array for use with deep learning models.

labels = df.iloc[:,1:].values

# Python iloc() function enables us to select a particular cell of the dataset

#get all target values in array all rows and one column

data = []

output = []

for ind in range(len(image\_path)):

image = image\_path[ind]

#storing all the image from image\_path by using for loop

img\_arr = cv2.imread(image)

#image was read by the imread function

h,w,d = img\_arr.shape

#for getting height ,width,and depth i.e shape of image

**Preprocessing**

load\_image=load\_img(image,target\_size=(224,224))

# Thetarget\_sizeis the size of your input images, every image will be resized to this size.

load\_image\_arr=img\_to\_array(load\_image)

#function is called and image is loaded to convert it into array norm\_load\_image\_arr=load\_image\_arr/255.0#normalized i/p

#In image processing, normalization is **a process that changes the range of pixel intensity values**

#As the pixel values range from 0 to 256, apart from 0 the range is 255. So dividing all the values by 255 will convert it to range from 0 to 1.

**# normalization to labels**

xmin,xmax,ymin,ymax = labels[ind]

#x is normalized by w,and y is normalized by h

nxmin,nxmax = xmin/w,xmax/w

nymin,nymax = ymin/h,ymax/h

label\_norm = (nxmin,nxmax,nymin,nymax) # normalized output

# -------------- append

data.append(norm\_load\_image\_arr)

output.append(label\_norm)

#data and output are append

## Why do we need to split our data?

To prevent look-ahead bias, overfitting and underfitting.

* Look-ahead bias: Building a model based on data that is not supposed to be known.
* Overfitting: This is the process of designing a model that adapts so closely to historical data that it becomes ineffective in the future.
* Underfitting: This is the process of designing a model that adapts so loosely to historical data that it becomes ineffective in the future.

x\_train,x\_test,y\_train,y\_test=train\_test\_split (x,y,train\_size=0.8,random\_state=0)

#split data into training and testing

# the random\_state parameter is used for initializing the internal random number generator, which will decide the splitting of data into train and test indices in your case.

What is inception resnet model?

Inception-ResNet-v2 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 164 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals

*Dense*: In any neural network, a dense layer is **a layer that is deeply connected with its preceding layer** which means the neurons of the layer are connected to every neuron of its preceding layer.

*Dropout*: A Simple Way to Prevent Neural Networks from Overfitting

*Flatten*: Flattening is merging all visible layers into the background layer **to reduce file size**.

**Deep Learnng Model**

#importing packages

from tensorflow.keras.layers import Input

from tensorflow.keras.applications import mobilenet\_v2,InceptionV3,InceptionResNetV2

from tensorflow.keras.layers import Dense,Dropout,Flatten

from tensorflow.keras.models import Model

import tensorflow as tf

inception\_resnet=InceptionResNetV2(weights="imagenet", include\_top=False,input\_tensor=Input(shape=(224, 224, 3)))

#Supplying weights="imagenet" indicates that we want to use the pre-trained ImageNet weights for therespective model.

inception\_resnet.trainable=False

# A Keras Model is [trainable by default](https://github.com/keras-team/keras/blob/master/keras/engine/base_layer.py#L154) - you have two means of freezing all the weights:

#model.trainable = False *before* compiling the model

#\_-----------------creating layers

headmodel=inception\_resnet.output

headmodel=Flatten()(headmodel)

headmodel=Dense(500,activation='relu')(headmodel)

headmodel=Dense(250,activation='relu')(headmodel)

#  ReLu activation function is to perform a threshold operation to each input element where values less than zero are set to zero

headmodel=Dense(4,activation='sigmoid')(headmodel)

#-------------model

model=Model(inputs=inception\_resnet.input,outputs=headmodel)

**LOSS:**

A loss function in Machine Learning is a measure of how accurately your ML model is able to predict the expected outcome i.e the ground truth.

A high value for the loss means our model performed very poorly. A low value for the loss means our model performed very well.

The MSE is great for ensuring that our trained model has no outlier predictions with huge errors, since the MSE puts larger weight on theses errors due to the squaring part of the function.

OPTIMIZER:

Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models.

Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

MODEL.SUMMARY:

The model summary table reports the strength of the relationship between the model

# complie model

model.compile(loss='mse',optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4))

model.summary()

### model training

from tensorflow.keras.callbacks import TensorBoard

#importing tensorboard from keras

history = model.fit(x=x\_train,y=y\_train,batch\_size=10,epochs=100,

validation\_data=(x\_test,y\_test),callbacks=[tfb])

#training model

model.save('./MODEL/object\_detection.h5')

#saving model in dir MODEL.

**MAKING** **PREDICTION** :

import numpy as np

import cv2

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.preprocessing.image import load\_img,img\_to\_array

#importing labraries

#load model

model=tf.keras.models.load\_model(r'C:\Users\Fehmina\anaconda3\MODEL\object\_detection.h5')

print('model loaded sucessfully')

path=r'C:\Users\Fehmina\anaconda3\test\_images\Cars0.png'

image=load\_img(path)

#PIL obj

image=np.array(image,dtype=np.uint8)

#8 bit array (0,255)

image1=load\_img(path,target\_size=(224,224))

image\_arr\_224=img\_to\_array(image1)/255.0

#convert into array and get normalized o/p

#testing image for prediction

h,w,d=image.shape

print('height of the image=',h)

print('width of the image=',w)

plt.figure(figsize=(10,8))

#size of the original image

plt.imshow(image)

plt.show()

#plotting image

#make prediction

coords=model.predict(test\_arr)

coords

#give normalized o/p

denorm=np.array([w,w,h,h])

coords=coords\*denorm

cords

#denormalized the values

coords=coords.astype(np.int32)

coords

#converting from float to int

#drawing bounding box on top of the image

xmin,xmax,ymin,ymax=coords[0]

pt1=(xmin,ymin)

pt2=(xmax,ymax)

print(pt1,pt2)

cv2.rectangle(image,pt1,pt2,(0,225,0),2)

#plotting image

plt.figure(figsize=(10,8))

plt.imshow(image)

plt.show()

#create pipeline

path=r'C:\Users\Fehmina\anaconda3\test\_images\Cars0.png'

def object\_detection(path):

#read image

image=load\_img(path) #PIL obj

image=np.array(image,dtype=np.uint8) #8 bit array (0,255)

image1=load\_img(path,target\_size=(224,224))

image\_arr\_224=img\_to\_array(image1)/255.0 #convert into array and get normalized o/p

h,w,d=image.shape

test\_arr=image\_arr\_224.reshape(1,224,224,3)

#make prediction

coords=model.predict(test\_arr)

#denormalized the values

denorm=np.array([w,w,h,h])

coords=coords\*denorm

#converting from float to int

coords=coords.astype(np.int32)

#drawing bounding box on top of the image

xmin,xmax,ymin,ymax=coords[0]

pt1=(xmin,ymin)

pt2=(xmax,ymax)

print(pt1,pt2)

cv2.rectangle(image,pt1,pt2,(0,225,0),3)

return image,cords

#plotting image and coordinate on test image

path=r'C:\Users\Fehmina\anaconda3\test\_images\Cars0.png'

image, coords = object\_detection(path)

plt.figure(figsize=(10,8))

plt.imshow(image)

plt.show()

**OCR**

OCR = [Optical Character Recognition](https://nanonets.com/blog/what-is-optical-character-recognition/). In other words, OCR systems transform a two-dimensional image of text, that could contain machine printed or handwritten text from its image representation into machine-readable text. OCR as a process generally consists of several sub-processes to perform as accurately as possible.

**Tesseract — is an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library.OCR uses artificial intelligence for text search and its recognition on images.**

Pytesseract is a wrapper for Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

Installing tesserct and pytesseract

import pytesseract as pt

#importing pytesseract

path=r'C:\Users\Fehmina\anaconda3\test\_images\Cars0.png'

image, coords = object\_detection(path)

#giving path to load image by calling function

plt.figure(figsize=(10,8))

plt.imshow(image)

plt.show()

#plotting image

img

img=np.array(load\_img(path))

#reading image by load\_img function and converting it in array by np.array

xmin,xmax,ymin,ymax=coords[0]

#reading coordinate

roi=img[ymin:ymax,xmin:xmax]

#cropping image i.e number plate

plt.imshow(roi)

plt.show()

#plotting the cropped image

text=pt.image\_to\_string(roi)

print(text)

#extract text from image

#Installing Visual Studio Code

Flask: **Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. [Flask](http://flask.pocoo.org/) is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Bootstrap 4-Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website. It is a front-end framework used for easier and faster web development. It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.

Django- is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

HTML: HTML stands for Hyper Text Markup Language.HTML is the standard markup language for creating Web pages.HTML describes the structure of a Web page.HTML consists of a series of elements.HTML elements tell the browser how to display the content

Html boilerplate- a boilerplate is a unit of writing that can be reused over and over without change. By extension, the idea is sometimes applied to reusable programming, as in “boilerplate code.”

Html snippets-Snippets are stand-alone, re-usable code pieces that can add additional functionality.  An HTML snippet is a small portion of source code in HTML. They can be used to build different elements (like a list view, different styled buttons, text display, customized search bar and so on).

[render\_template](https://github.com/pallets/flask/blob/master/src/flask/templating.py) - is a [Flask](https://www.fullstackpython.com/flask.html) function from the flask.templating package. render\_template is used to generate output from a [template file based on the Jinja2 engine](https://www.fullstackpython.com/template-engines.html) that is found in the application's templates folder.

Note that render\_template is typically imported directly from the flask package instead of from flask.templating

Request - In the client-server architecture, the request object contains all the data that is sent from the client to the server. The data from a client’s web page is sent to the server as a global request object. In order to process the request data, it should be imported from the Flask module.

APP.PY

from flask import Flask, render\_template,request

#importing flask,render tempelate,request

import os

from deeplearning import OCR

#The **Web Server Gateway Interface**  is a simple [calling convention](https://en.wikipedia.org/wiki/Calling_convention) for [web servers](https://en.wikipedia.org/wiki/Web_server) to forward requests to [web applications](https://en.wikipedia.org/wiki/Web_application) or [frameworks](https://en.wikipedia.org/wiki/Web_framework) written in the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language))

# webserver gateway interface

app = Flask(\_\_name\_\_)

#calling our flask app

BASE\_PATH=os.getcwd()

UPLOAD\_PATH=os.path.join(BASE\_PATH,'static/upload/')

# Python method **getcwd()** returns current working directory of a process and **os**.**path**.**join**() method in **Python join** one or more **path** components

@app.route('/',methods=['POST','GET'])

# @app. route("/") is **a Python decorator that Flask provides to assign URLs in our app to functions easily**.

def index():

    if request.method=='POST':

        upload\_file=request.files['image\_name']

        filename=upload\_file.filename

        path=os.path.join(UPLOAD\_PATH,filename)

        upload\_file.save(path)

        text=OCR(path,filename)

        return render\_template('index.html',upload=True,upload\_image=filename,text=text)

    return render\_template('index.html',upload=False)

# Boilerplate code

if \_\_name\_\_ =="\_\_main\_\_":

# So when the interpreter runs a module, the \_\_name\_\_ variable will be set as  \_\_main\_\_ if the module that is being run is the main program.

    app.run(debug=True)

index.html

{% extends 'layout.html' %}

#{**% extends %}**: this declares the template given as an argument as the current template's parent. Usage.

{%block body%}

**#{% block %}{% endblock %}**: This is used to define sections in your templates, so that if another template extends this one, it'll be able to replace whatever html code has been written inside of it.

 <div class="container">

#The <div> tag defines a division or a section in an HTML document. The <div> tag is used **as a container for HTML elements** in which we can do further styling.

    <br><br>

        <form action="#" method="POST" enctype="multipart/form-data">

* #action=# working on current page,
* #method - specifies how to send form data,
* #post- In the post method, after the submission of the form, **the form values will not be visible in the address bar of the new** browser tab,
* #enctype-specifies how the form-data should be encoded when submitting it to the server.
* **#multipart/form-data** - which is used in form element that have a file upload.
* #**multi-part** -means form data divides into **multiple parts** and send to server.

            <div class="input-group">

# input-group class is **a container to enhance an input by adding an icon, text or a button in front or behind the input field**

                 <input type="file" class="form-control" name="image\_name" required>

#Class=form-controls automatically receive some global styling with Bootstrap

#Name =image\_name giving name of image ex car0

                     <input type="submit" value="Upload" class="btn btn-outline-secondary">

#input type=submit-**submit** button which submits all **form** values

#value=upload-upload the selected file

#class="btn btn-outline-secondary-outline border button

             </div>

        </form>

 </div>

    {% if upload %}

        <div class="container">

            <br><br><br>

            <table>

                <tr>

                    <td>

                        <img  class="rounded float-left img-fluid"   src="/static/upload/{{upload\_image}}" alt="" style="margin-right: 500px;">

                    </td>

                    <td>

                        <img class="rounded float-right img-fluid"   src="/static/predict/{{upload\_image}}" alt="" style="margin-right: 500px;">

                    </td>

                </tr>

            </table>

            <br><br>

            <table style="border: solid black;border-width: 10%;width: 100%;">

                <tr style="border: solid black;border-width: 10%">

                    <th style="color: rgb(117, 35, 35);"> Cropped Licence Plate Image </th>

                    <th style="color:rgb(117, 35, 35) ;">Text</th>

                 </tr>

                 <tr style="border: solid black;border-width: 10%">

                     <td style="background-color:rgb(244, 248, 248);">

                         <img class="img-fluid"   src="/static/roi/{{upload\_image}}" alt="">

                     </td>

                     <td style="background-color: rgb(244, 248, 248);">

                         <h1 class="display-8"> {{text}}</h1>

                     </td>

                 </tr>

            </table>

        </div>

  {% endif %}

{% endblock %}

Layout.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Number plate Optical characte recognisition (OCR)</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

</head>

<body>

    <!--navbar-->

    <nav class="navbar navbar-dark bg-dark">

        <div class="container">

            <a class="navbar-brand" href="/">

            <h1>NumberPlate  OCR</h1>

        </a>

        </div>

    </nav>

<hr>

    {% block body %}

    {% endblock  %}

</body>

</html>

Deeplearning.py

import numpy as np

import cv2

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.preprocessing.image import load\_img,img\_to\_array

import pytesseract as pt

model=tf.keras.models.load\_model('./static/MODEL/object\_detection.h5')

def object\_detection(path,filename):

        #read image

        image=load\_img(path) #PIL obj

        image=np.array(image,dtype=np.uint8)  #8 bit array (0,255)

        image1=load\_img(path,target\_size=(224,224))

        image\_arr\_224=img\_to\_array(image1)/255.0   #convert into array and get normalized o/p

        h,w,d=image.shape

        test\_arr=image\_arr\_224.reshape(1,224,224,3)

        #make prediction

        coords=model.predict(test\_arr)

        #denormalized the values

        denorm=np.array([w,w,h,h])

        coords=coords\*denorm

        #converting from float to int

        coords=coords.astype(np.int32)

        #drawing bounding box on top of the image

        xmin,xmax,ymin,ymax=coords[0]

        pt1=(xmin,ymin)

        pt2=(xmax,ymax)

        print(pt1,pt2)

        cv2.rectangle(image,pt1,pt2,(0,225,0),3)

        #convert into bgr

        image\_bgr=cv2.cvtColor(image,cv2.COLOR\_RGB2BGR)

        cv2.imwrite('./static/predict/{}'.format(filename),image\_bgr)

        return coords

def OCR(path,filename):

    img=np.array(load\_img(path))

    coords=object\_detection(path,filename)

    xmin,xmax,ymin,ymax=coords[0]

    roi=img[ymin:ymax,xmin:xmax]

    roi\_bgr=cv2.cvtColor(roi,cv2.COLOR\_RGB2BGR)

    cv2.imwrite('./static/roi/{}'.format(filename),roi\_bgr)

    text=pt.image\_to\_string(roi)

    print(text)

    return text

Software used:

Anaconda

Jupyter notebook

Command prompt

Visual studio code

Limitation and drawback:

* PyTesseract assumed only aligned text it should not be rotated or skewed
* Can’t work for the blur image.
* Not usedfor handwritten and cursive or special effect text.
* Poor resolution image can’t read text

Advantages:

* **Enhanced parking management**
* **Faster Traffic Management.**
* **Better security and prevention of crimes** like car thefts
* Provides **better evidence and lines of inquiry**
* **Automates access control systems**
* Allows **modern and effective law enforcement**

FINAL APP:

