VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SMART INTERNZ – ARTIFICIAL INTELLIGENCE

AUTOMATIC NUMBER PLATE RECOGNITION

TEAM MEMBERS:

Velaga Lalith Srinivas	20MIS0010
A.Rithika	20MIS0018
B.Lithikaa	20MIS0061
R.Adharsha	20MIS0354

1. INTRODUCTION

1.1 Overview

Automatic Number Plate Recognition (ANPR) is a technology that enables the automatic detection and reading of vehicle license plates. It utilizes optical character recognition (OCR) techniques to extract alphanumeric characters from images or video frames captured by cameras. ANPR systems are commonly used for various applications such as traffic management, law enforcement, parking management, toll collection, and vehicle tracking. The application will allow users to upload an image containing a vehicle license plate, and it will extract and display the characters present on the license plate.

To develop the ANPR OCR web app, we will need the following components:

Flask: Flask is a lightweight web framework for Python. It allows us to quickly build web applications with minimal boilerplate code.

OpenCV: OpenCV (Open-Source Computer Vision Library) is a powerful open-source library for image and video processing. We will use it for image manipulation and preprocessing tasks.

1.2 Purpose

The purpose of developing an Automatic Number Plate Recognition (ANPR) OCR web app project can be manifold. Here are some potential purposes and benefits of such a project:

Enhanced Security: ANPR technology can be used to enhance security at various locations such as airports, government buildings, and private premises. The web app can aid in identifying unauthorized vehicles, creating access logs, and implementing security protocols.

Toll Collection: ANPR OCR systems can be utilized for automated toll collection on highways and toll roads. By integrating the web app with a payment gateway, users can easily pay toll fees online, minimizing traffic congestion and improving the overall toll collection process.

Law Enforcement: ANPR technology can aid law enforcement agencies in identifying stolen vehicles, tracking suspects, and enforcing traffic regulations. The web app can facilitate the quick and automated identification of license plates, allowing law enforcement officers to focus their efforts on potential threats or suspicious activities.

Traffic Management: ANPR systems can be used to monitor and manage traffic flow by capturing and analyzing license plate data. The web app can assist in collecting data for traffic analysis, identifying traffic patterns, detecting traffic violations, and optimizing traffic management strategies.

2. LITERATURE SURVEY

2.1 Existing problem

Variability in license plate designs: License plate designs vary across different countries and regions, and even within a single country. This variability poses a challenge for ANPR systems, as they need to adapt to different plate sizes, fonts, colors, and layouts. A system trained on one set of license plate designs may struggle to accurately detect plates from another region.

Poor image quality: ANPR systems rely on clear and high-resolution images to accurately detect license plates. However, in real-world scenarios, image quality can be compromised due to factors such as low lighting conditions, adverse weather, motion blur, or camera angles. Poor image quality can make it difficult for ANPR systems to extract the necessary information from license plates.

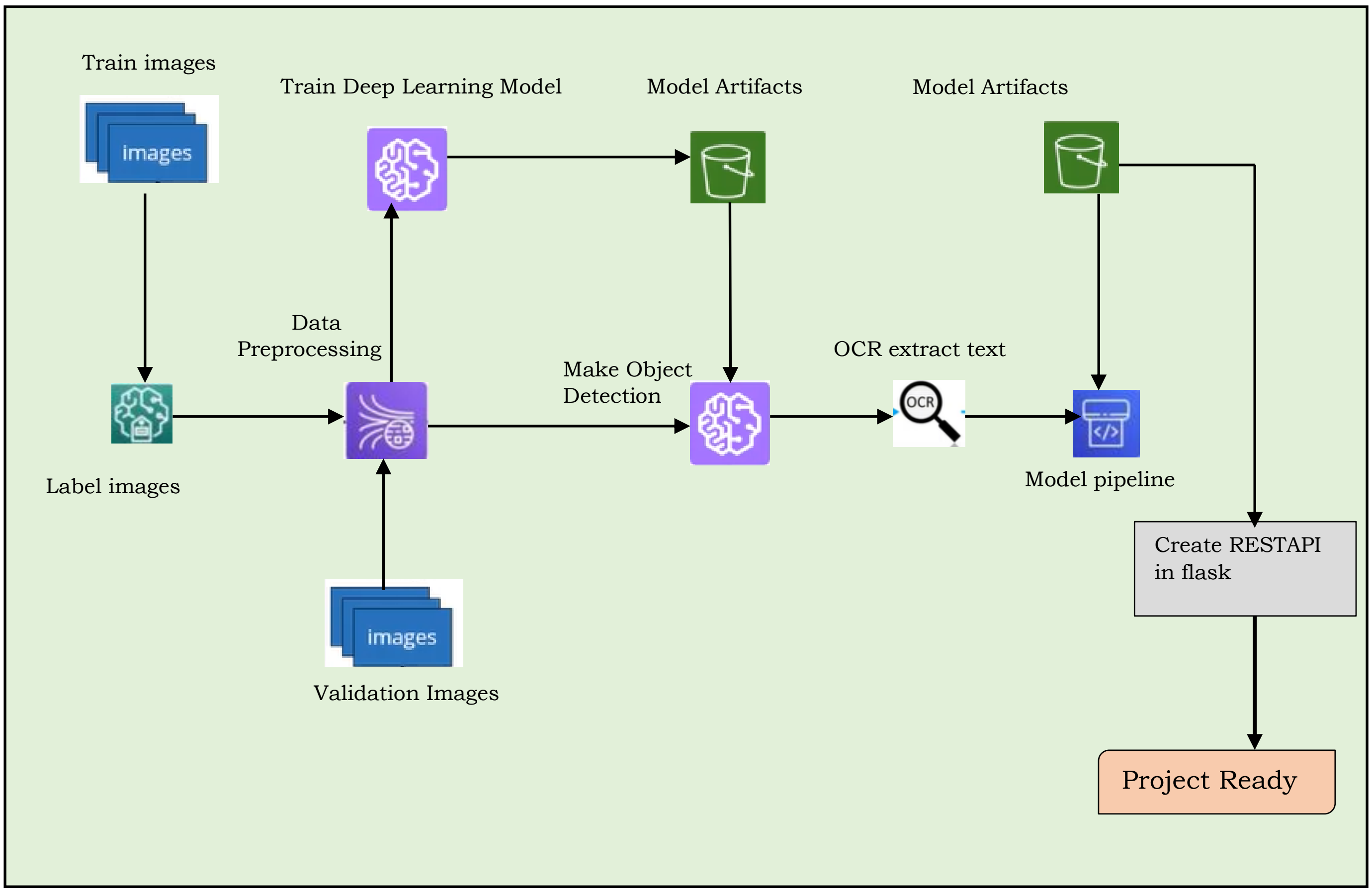
Privacy Concerns: The fact that images and records are kept and stored raises some privacy concerns. People are usually afraid that the records of someone's whereabouts in all these footages might be misused. It can become a subject of data thefts or people with all kinds of nefarious intentions.
Speed and motion blur: ANPR systems are often used in applications that require the detection of license plates from moving vehicles. However, the speed of the vehicle can introduce motion blur, which can affect the clarity of the license plate image. The blurring effect can make it difficult for ANPR systems to extract the necessary information accurately.

2.2 Proposed solution

Set up the flask web app by installing flask and other necessary dependencies. Create the necessary routes and templates for handling user requests and displaying results. Design the user interface where we could upload image with number plate. Open CV is o be used to process the uploaded image. Resize image using OCR. By applying techniques like counters or machine learning algorithm locate the license plate region in uploaded image. Segment the characters within the license plate region to isolate individual characters for OCR using Character segmentation. Extract the segmented characters using library like pytesseract. Clean up and validate the extracted OCR results. Present the recognized characters to the user on the web interface. Show the extracted license plate number or any additional information associated with it. Improve the user interface with CSS styles and layouts to enhance the visual appeal of the web app.

3. THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware / Software designing

Hardware : Laptop

Software : Jupyter notebook,VS code, Flask

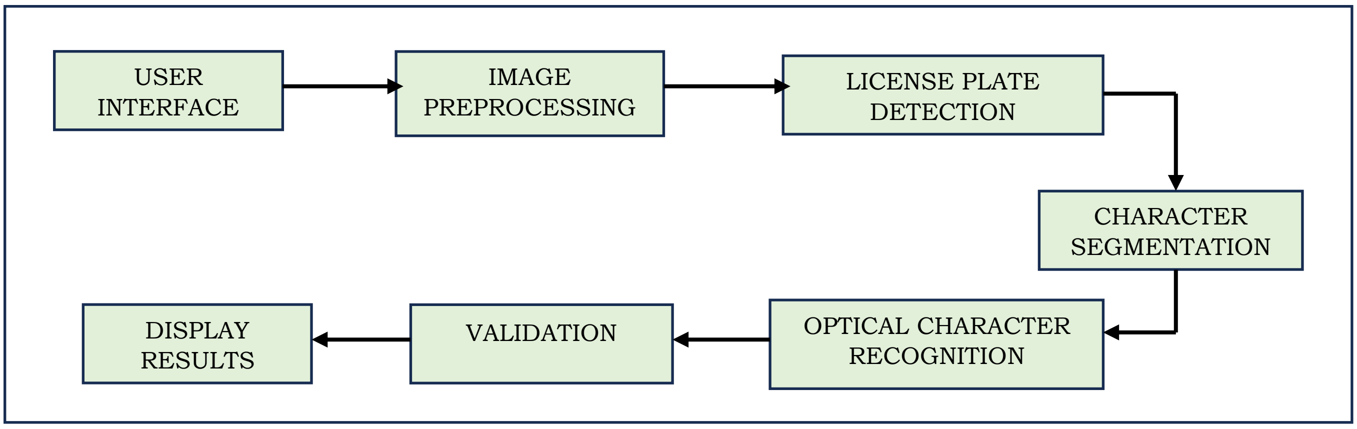
4. EXPERIMENTAL INVESTIGATIONS

Tesseract OCR: Tesseract is a popular OCR engine developed by Google. It supports various languages and can recognize text from images. We will utilize the pytesseract library, which provides a Python interface for Tesseract.

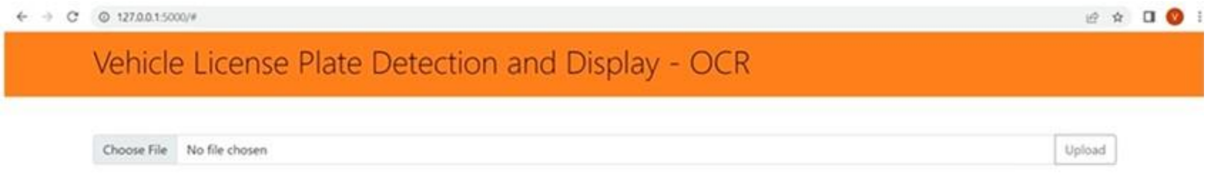
Image Preprocessing: This step involves enhancing the input image to improve the quality and make it suitable for further processing. Techniques like resizing, grayscale conversion, noise removal, and contrast adjustment are commonly used.

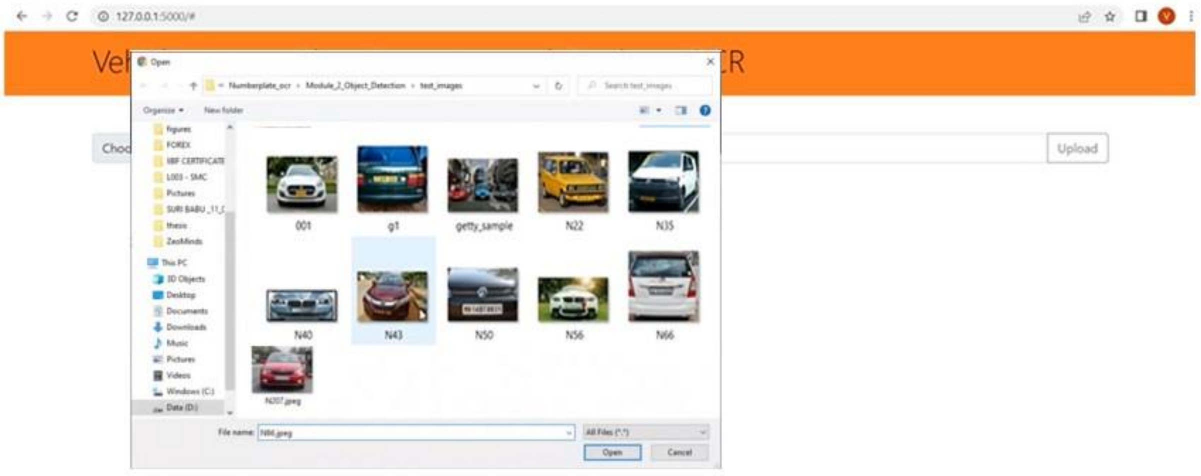
Character Recognition: After segmentation, the segmented characters are recognized by employing optical character recognition (OCR) techniques. OCR algorithms analyze the visual patterns of the characters and match them against a pre-trained character set or a machine learning model trained on a dataset of characters.

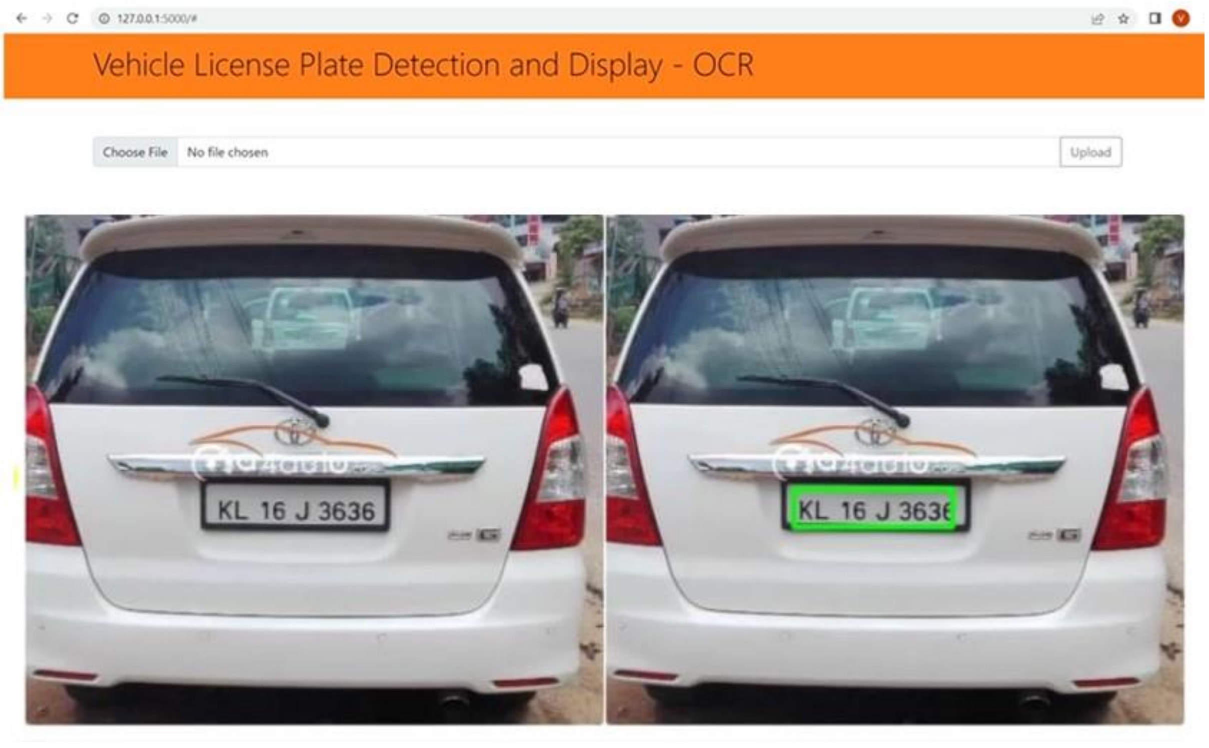
5. FLOWCHART

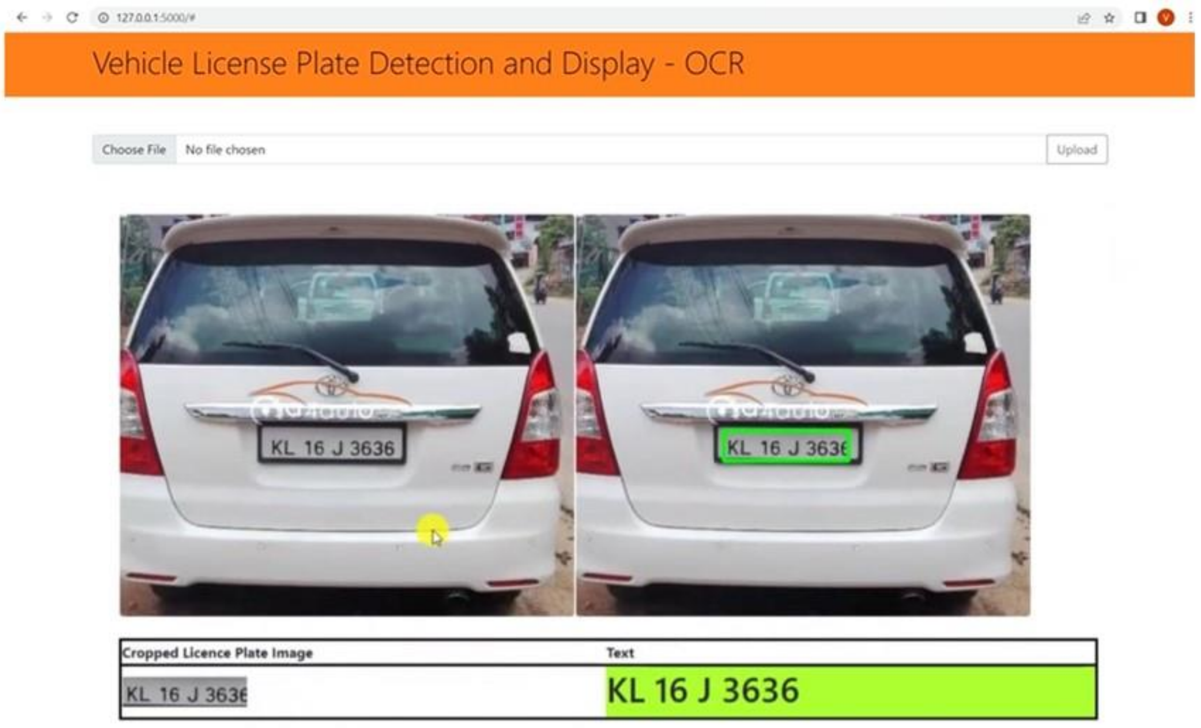


6. RESULT









Cropped Licence Plate Image	Text
KL 16 J 3636	KL 16 J 3636

7. ADVANTAGES :

Automation: ANPR OCR web apps automate the process of license plate recognition, eliminating the need for manual intervention. This saves time and reduces human error, leading to increased efficiency and productivity.

Improved Security: ANPR systems enhance security by quickly identifying vehicles of interest, such as stolen vehicles or those associated with criminal activities. The web app can provide real-time alerts to law enforcement or security personnel, enabling prompt action.

Parking Efficiency: ANPR OCR systems streamline parking management by automating entry and exit processes, tracking parking durations, and enforcing parking regulations. The web app facilitates a seamless parking experience for users and reduces congestion in parking areas.

Data Collection and Analysis: ANPR OCR web apps collect valuable license plate data that can be used for various purposes, including traffic analysis, crime prevention, and marketing research. The data can provide insights into traffic patterns, customer behavior, and other relevant information for decision-making.

DISADVANTAGES :

Privacy Concerns: ANPR OCR systems raise privacy concerns as they capture and process license plate information. There is a need to handle and store this data responsibly, ensuring compliance with privacy regulations and protecting individuals' personal information.

Accuracy Limitations: The accuracy of ANPR OCR systems can be influenced by factors such as image quality, lighting conditions, and variations in license plate formats. It may not be perfect, and errors in license plate recognition can occur, leading to potential false positives or false negatives.

Cost and Infrastructure: Setting up an ANPR OCR system, including cameras, hardware, and server infrastructure, can involve significant costs. Additionally, ongoing maintenance, software updates, and data storage requirements contribute to the overall expenses.

Environmental Factors: ANPR OCR systems can be affected by environmental conditions such as bad weather, poor lighting, or obscured license plates. These factors can impact the accuracy and reliability of the system, requiring additional measures or fallback options.

8. APPLICATIONS

Parking Management: ANPR OCR systems can automate entry and exit processes in parking lots, garages, and facilities. The web app can integrate with parking management systems to accurately track parking durations, enforce parking regulations, and streamline payment processes.

Vehicle Access Control: ANPR OCR technology can enhance security by providing automated vehicle access control at secured premises, gated communities, and parking areas. The web app can verify authorized vehicles, maintain access logs, and trigger alerts for any unauthorized or suspicious vehicle activity.

Fleet Management: ANPR OCR systems can aid fleet management companies in vehicle tracking, route optimization, and driver management. The web app can integrate with fleet management platforms to track vehicles, monitor driver behaviour, and improve logistics and delivery operations.

Border Control and Security: ANPR OCR can assist in border control and security operations by capturing and analyzing license plate data at border checkpoints. The web app can integrate with security systems to identify vehicles of interest, track border crossings, and enhance border security measures.

Safety and Surveillance: ANPR OCR technology can be used for safety and surveillance purposes, such as monitoring and identifying vehicles involved in accidents, hit-and-runs, or other criminal activities. The web app can help authorities quickly retrieve license plate information for investigation purposes.

Marketing and Analytics: ANPR OCR systems can provide valuable data for marketing and analytics purposes. The web app can analyze license plate data to gain insights into customer behavior, measure traffic volumes, and support targeted advertising campaigns.

Smart City Initiatives: ANPR OCR technology is a key component of smart city initiatives, contributing to efficient traffic management, improved security, and enhanced urban planning. The web app can be part of a larger ecosystem integrating various smart city solutions.

9. CONCLUSION

It is quite clear that number plate-recognition is difficult system because of different number of phases and presently it is not possible to achieve 100% overall accuracy as each phase is dependent on previous phase. Certain factors like different illumination conditions, vehicle shadow and non-uniform size of license plate characters, different font and background colour affect the performance of ANPR. Some systems work in these restricted conditions only and might not produce good amount of accuracy in adverse conditions. We are currently working on an optimization to the detection algorithm implementation; our goal is to adapt the algorithm to better fit to the architectural characteristics of our platform. In the future we will also have a look at special region segmentation methods.

10. FUTURE SCOPE

Future research in ANPR still faces several challenges; For instance, there is a need to concentrate on more robust algorithms for non-standardized formats, irrespective of regions. Also, all proposed/designed algorithms need to be tested for real time scenarios rather pre-acquired images. In addition, high resolution cameras need to be integrated, allowing robust algorithms to reduce processing times and increase recognition capabilities. Yet another avenue is Obscure character recognition, since there are a lot similarities in characters like the pairs- (O,0), (P,B), (Z,2), (S,5), (3,8), (B,8), (P,R), (D,O), (1,I), (Q,O), (C,G), (A,4), (K,X), (F,E), (b,6), (q,9), (p,b), (V,W), (X,Y), (V,U), (6,8), (5,3), (5,8), (0,8), (3,9), (4,9) etc. The similarities, together with impairments, can easily deceive the optical character recognition mechanism if there is small tilt, fonts change, broken, snow or dirt on characters or if the image is acquired at different angles. Lastly, it is recommended that moving vehicles, fast speeds, low contrast, insufficient or over exposed lights and real time scenarios must be tested to check robustness of the proposed algorithms.

11. BIBILOGRAPHY

N. P. Ap, T. Vigneshwaran, M. S. Arappradhan and R. Madhanraj, "Automatic Number Plate Detection in Vehicles using Faster R-CNN," 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 2020, pp. 1-6, doi: 10.1109/ICSCAN49426.2020.9262400.

M.R.C.Niluckshini and Dr.M.F.M.Firdhous, "Automatic Number Plate Detection using Haar-Cascade Algorithm Proposed for Srilankan Context," 2022 2nd International Conference on Advanced Research in Computing (ICARC), Belihuloya, Sri Lanka, 2022, pp. 248-253, doi: 10.1109/ICARC54489.2022.9753915.

G. N et al., "Automatic Number Plate Detection using Deep Learning," 2022 Smart Technologies, Communication and Robotics (STCR), Sathyamangalam, India, 2022, pp. 1-5, doi: 10.1109/STCR55312.2022.10009582.

S. Babbar, S. Kesarwani, N. Dewan, K. Shangle and S. Patel, "A New Approach for Vehicle Number Plate Detection," 2018 Eleventh International Conference on Contemporary Computing (IC3), Noida, India, 2018, pp. 1-6, doi: 10.1109/IC3.2018.8530600.

C. K. Sahu, S. B. Pattnayak, S. Behera and M. R. Mohanty, "A Comparative Analysis of Deep Learning Approach for Automatic Number Plate Recognition," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2020, pp. 932-937, doi: 10.1109/I-SMAC49090.2020.9243424.

APPENDIX

FLASK APP:

App.py:

```python
from flask import Flask, render_template, request
import os
from deeplearning import OCR
#webserver
app = Flask(__name__)
BASE_PATH = os.getcwd()
UPLOAD_PATH = os.path.join(BASE_PATH,'static/upload/')
@app.route('/',methods=['POST','GET'])
def index():
    if request.method == 'POST':
        upload_file = request.files['image_name']
        filename = upload_file.filename
        path_save = os.path.join(UPLOAD_PATH,filename)
        upload_file.save(path_save)
        text =OCR(path_save,filename)
        return render_template('index.html',upload=True,upload_image=filename,text=text)
    return render_template('index.html',upload=False)
```

```
if __name__ == "__main__":
    app.run(debug=True)
```

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>Numberplate OCR</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
</head>
<body>
    <nav class="navbar navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="/">
                <h1 class="display-8">Numberplate OCR</h1>
            </a>
        </div>
    </nav>
    {% block body %}
    {% endblock  %}
    <footer>
        <hr>
        <a href="http:\\AIanywhere.com">AI anywhere</a>
    </footer>
</body>
</html>
```

index.html

```
{% extends 'layout.html' %}
{% block body %}
    <div class="container">
        <br><br>
        <from action="#" method="POST" enctype="multipart/form-data">
            <div class="input-gropu">
                <input type="file" class="form-control" name="'image_name" required>
                <input type="submit" value="Upload" class="btn btn-outline-secondary">
            </div>
        </from>
    </div>
    {% if upload %}
        <div class="container">
            <br><br>
            <table>
                <tr>
                    <td>
                        <img class="rounded float-left img-fluid" src="/static/upload{{
upload_image }}" alt="">
                    </td>
                    <td>
                        <img class="rounded float-right img-fluid" src="/static/predict{{
upload_image }}" alt="">
                    </td>
                </tr>
            </table>
            <br>
            <table style="border: solid black; width: 100%; ">
                <tr style="border: solid black;">
                    <th>Cropped License Plate Image</th>
                    <th>Text</th>
                </tr>
                <tr style="border: solid black;">
                    <td>
                        <img class="img-fluid" src="/static/roi/{{ upload_image }}"alt="">
                    </td>
                    <td style="background-color: cadetblue;">
                        <h1 class="display-8"> {{ text }} </h1>
                    </td>
                </tr>
            </table>
        </div>
    {% endif %}
{% endblock body %}
```

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/models/object_detection.h5')
#path = './test_images/69b56ab9-accd-4c67-aa7a-a884c018e5ab___speedex-number-plates-
nandanam-chennai-car-number-plate-dealers-3zh7n2s.jpg.jpeg'
def object_detection(path,filename):
    image = load_img(path)
    image = np.array(image,dtype=np.uint8)
    image1 = load_img(path,target_size=(224,224))
    image_arr_224= img_to_array(image1)/255.0
    h,w,d = image.shape
    test_arr = image_arr_224.reshape(1,224,224,3)
    coords = model.predict(test_arr)
    denorm=np.array([w,w,h,h])
    coords = coords*denorm
    coords=coords.astype(np.int32)
    xmin,xmax,ymin,ymax= coords[0]
    pt1=(xmin,ymin)
    pt2=(xmax,ymax)
    print(pt1,pt2)
    cv2.rectangle(image,pt1,pt2,(0,225,0),3)
    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))
    cods = object_detection(path), filename
    xmin,xmax,ymin,ymax = cods[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
```

INITIALIZE WEIGHTS:

Object_detection

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
import xml.etree.ElementTree as xet
df = pd.read_csv('label.csv')
df.head()
filename = df['filepath'][0]
filename
def getFilename(filename):
    filename_image = xet.parse(filename).getroot().find('filename').text
    filepath_image = os.path.join('./google_images',filename_image)
    return filepath_image
image_path = list(df['filepath'].apply(getFilename))
image_path
file_path = image_path[0]
file_path
import cv2
img = cv2.imread(file_path)
cv2.rectangle(img,(159,561),(316,653),(0,255,0),3)
cv2.namedWindow('example',cv2.WINDOW_NORMAL)
cv2.imshow('example',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img, img_to_array
label = df.iloc[:,1:].values
label[0]
data=[]
output=[]
for ind in range(len(image_path)):
    image = image_path[ind]
    img_arr = cv2.imread(image)
    h,w,d = img_arr.shape
    #preprocessing
    load_image = load_img(image, target_size=(224,224))
    load_image_arr = img_to_array(load_image)
    norm_load_img_arr = load_image_arr/255.0
    #normalization
    xmin,xmax,ymin,ymax = label[ind]
    nxmin,nxmax= xmin/w , xmax/w
    nymin,nymax = ymin/h , ymax/h
    label_norm = (nxmin,nxmax,nymin,nymax)
    data.append(norm_load_img_arr)
    output.append(label_norm)
x = np.array(data,dtype=np.float32)
y = np.array(output,dtype=np.float32)
x.shape,y.shape
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
from tensorflow.keras.applications import MobileNetV2, InceptionV3, InceptionResNetV2
from tensorflow.keras.layers import Dense, Dropout, Flatten, Input
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)))
inception_resnet.trainable = False
headmodel = inception_resnet.output
headmodel = Flatten()(headmodel)
headmodel = Dense(500, activation = "relu")(headmodel)
headmodel = Dense(250, activation = "relu")(headmodel)
headmodel = Dense(4,activation='sigmoid')(headmodel)
model = Model(inputs= inception_resnet.input , outputs=headmodel)
model.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4))
model.summary()
from tensorflow.keras.callbacks import TensorBoard
tfb = TensorBoard('object_detection')
history = model.fit(x=x_train, y=y_train , batch_size = 10, epochs =200, validation_data =(x_test,
y_test),callbacks=[tfb])
model.save('./model/object_detection.h5')
```

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
model = tf.keras.models.load_model('./model/object_detection.h5')
print('model loaded successfully')
path = './test_images/69b56ab9-accd-4c67-aa7a-a884c018e5ab___speedex-number-plates-
nandanam-chennai-car-number-plate-dealers-3zh7n2s.jpg.jpeg'
image = load_img(path)
image = np.array(image,dtype=np.uint8)
image1 = load_img(path,target_size=(224,224))
image_arr_224= img_to_array(image1)
h,w,d = image.shape
print('height= ',h)
print('width= ',w)
plt.figure(figsize=(4, 4))
plt.imshow(image)
plt.show()
test_arr = image_arr_224.reshape(1,224,224,3)
test_arr.shape
coords = model.predict(test_arr)
denorm=np.array([w,w,h,h])
coords = coords*denorm
coords=coords.astype(np.int32)
xmin,xmax,ymin,ymax= coords[0]
pt1=(xmin,ymin)
pt2=(xmax,ymax)
print(pt1,pt2)
cv2.rectangle(image,pt1,pt2,(0,225,0),3)
#plt.figure(figsize=(4, 4))
plt.imshow(image)
plt.show()
path = './test_images/69b56ab9-accd-4c67-aa7a-a884c018e5ab___speedex-number-plates-
nandanam-chennai-car-number-plate-dealers-3zh7n2s.jpg.jpeg'
def object_detection(path):
    image = load_img(path)
    image = np.array(image,dtype=np.uint8)
    image1 = load_img(path,target_size=(224,224))
    #preprocessing
    image_arr_224= img_to_array(image1)/225.0
    h,w,d = image.shape
    test_arr = image_arr_224.reshape(1,224,224,3)
    coords = model.predict(test_arr)
    denorm=np.array([w,w,h,h])
    coords = coords*denorm
    coords=coords.astype(np.int32)
    #draw boundary
    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
path = './test_images/69b56ab9-accd-4c67-aa7a-a884c018e5ab___speedex-number-plates-
nandanam-chennai-car-number-plate-dealers-3zh7n2s.jpg.jpeg'
image ,cods= object_detection(path)
plt.imshow(image)
plt.show()
img=np.array(load_img(path))
xmin,xmax,ymin,ymax = cods[0]
roi= img[ymin:ymax,xmin:xmax]
plt.imshow(roi)
plt.show()
pt.image_to_string(roi)
```