AI-Based License Plate Recognition

Project Overview

This project focuses on developing an AI-based license plate recognition system utilizing technologies such as YOLOv8 for object detection, OpenCV for image processing, Tesseract for optical character recognition (OCR), Flask as the backend framework and Nginx for reverse proxy and load balance. The system is designed to accurately detect and recognize license plate numbers from images or video feeds.

Technologies Used

- **YOLOv8**: A real-time object detection model used to locate and identify license plates in images.
- **OpenCV**: A library for image processing tasks such as pre-processing images before feeding them to the detection model.
- **Ultralytics**: Provides tools for training and evaluating the YOLOv8 model.
- **Tesseract**: An OCR engine that converts detected license plate images into readable text.
- **Flask**: A web framework that serves as the backend of the application, handling requests and responses.
- **Nginx**: Used for separating the frontend and backend, acting as a reverse proxy to improve performance and security.

Project Architecture

The architecture of the project is divided into two main components: the frontend and the backend.

1. **Frontend**:
    - Built with HTML and CSS, the frontend includes an `index.html` file that provides the user interface.
    - The frontend communicates with the backend via REST APIs.
2. **Backend**:
    - Implemented using Flask, it includes the necessary API endpoints for processing images and returning recognized license plate numbers.
    - Nginx is configured to manage incoming requests and route them to the appropriate services.

Model Training

The YOLOv8 model was trained using labeled images generated with LabelImg. The training process involved:

- Collecting a dataset of license plates.

- Annotating the dataset using LabelImg to create bounding boxes around the license plates.

API Endpoints

The backend Flask application exposes the following API endpoints:

- **POST /api/upload**: Accepts an image file and returns the detected license plate number.
- **GET /api/status**: Returns the status of the service.

Nginx Configuration

The Nginx server is configured to serve static files and to act as a reverse proxy for the Flask application.

```
server {

    listen 80;
    server_name localhost;

    location /static/ {
        root C:\nginx-1.26.2\static;
    }

    location / {
        root C:\nginx-1.26.2\html;
        index index.html;
        try_files $uri $uri/ =404;
    }

    location /api/ {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Code Snippets

Flask Application (app.py)

```
from flask import Flask, render_template, request, redirect
import cv2
import pandas as pd
import numpy as np
import pytesseract
from ultralytics import YOLO
from datetime import datetime
import os

app = Flask(__name__)
UPLOAD_FOLDER =
r"C:\Users\Dell\Downloads\license_plate_recognition1-master\license_
plate_recognition1-master\uploads"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

pytesseract.pytesseract.tesseract_cmd = r'C:\Program
Files\Tesseract-OCR\tesseract.exe'
model =
YOLO(r"C:\Users\Dell\Downloads\license_plate_recognition1-master\lic
ense_plate_recognition1-master\best.pt")

processed_numbers = set()

area = np.array([(10, 400), (10, 480), (1030, 480), (1030, 400)],
np.int32)

def preprocess_image(crop):
    gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
    gray = cv2.bilateralFilter(gray, 10, 20, 20)
    return gray

def extract_plate(frame):
    results = model.predict(frame)
    a = results[0].boxes.data
    if len(a) == 0:
        return [], pd.DataFrame()

    px = pd.DataFrame(a).astype("float")
    extracted_texts = []

    for index, row in px.iterrows():
        x1 = int(row[0])
        y1 = int(row[1])
        x2 = int(row[2])
        y2 = int(row[3])
        crop = frame[y1:y2, x1:x2]
        processed_crop = preprocess_image(crop)
        text = pytesseract.image_to_string(processed_crop).strip()
        cleaned_text = text.replace('(', '').replace(')',
'').replace(',', '').replace(']', '').strip()
        if cleaned_text and cleaned_text not in processed_numbers:
            processed_numbers.add(cleaned_text)
            extracted_texts.append(cleaned_text)
            current_datetime = datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
            with open("car_plate_data.txt", "a") as file:
                file.write(f"{cleaned_text}\t{current_datetime}\n")

    return extracted_texts, px

def process_video(video_path):
    cap = cv2.VideoCapture(video_path)
    extracted_texts = []
    count = 0

    while True:
        ret, frame = cap.read()
        count += 1
        if not ret:
            break

        if count % 3 != 0:
            continue

        frame = cv2.resize(frame, (1020, 500))
        cv2.polylines(frame, [area], True, (255, 0, 0), 2)

        results = model.predict(frame)
        detections = results[0].boxes.data
        px = pd.DataFrame(detections).astype("float")

        if px.empty:
            print("No detections in this frame.")
            continue

        for index, row in px.iterrows():
            x1, y1, x2, y2, conf, cls = row
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            print(f"Detected box: {(x1, y1, x2, y2)} with
confidence: {conf}")


            if (x1 >= area[0][0] and x2 <= area[2][0] and
                y1 >= area[0][1] and y2 <= area[2][1]):


                crop = frame[y1:y2, x1:x2]
                processed_crop = preprocess_image(crop)

                text =
pytesseract.image_to_string(processed_crop).strip()
                cleaned_text = text.replace('(', '').replace(')',
'').replace(',', '').replace(']', '').strip()

                if cleaned_text and cleaned_text not in
processed_numbers:
                    processed_numbers.add(cleaned_text)
                    extracted_texts.append(cleaned_text)
                    current_datetime =
datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                    with open("car_plate_data.txt", "a") as file:

file.write(f"{cleaned_text}\t{current_datetime}\n")
                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0,
255, 0), 2)
                else:
                    print(f"Detected but already processed:
{cleaned_text}")

        cv2.imshow("Video Frame", frame)
        if cv2.waitKey(1) & 0xFF == 27:
            break

    cap.release()
    cv2.destroyAllWindows()

    return "\n".join(set(extracted_texts))

def process_image(image_path):
    img = cv2.imread(image_path)
    frame = cv2.resize(img, (1020, 500))
    extracted_texts, _ = extract_plate(frame)
    return extracted_texts

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return redirect(request.url)

    file = request.files['file']
    if file.filename == '':
        return redirect(request.url)

    file_path = os.path.join(app.config['UPLOAD_FOLDER'],
file.filename)
    file.save(file_path)

    if file.filename.lower().endswith(('.mp4', '.avi', '.mov')):
        extracted_texts = process_video(file_path)
        extracted_text_output = extracted_texts.split("\n") if
extracted_texts else "No plates detected."
        return render_template('index.html',
extracted_text=extracted_text_output)
    else:
        extracted_texts = process_image(file_path)
        extracted_text_output = extracted_texts if extracted_texts
else "No plates detected."
        return render_template('index.html',
extracted_text=extracted_text_output)

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

Frontend (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0">
    <title>License Plate Detection</title>
    <link rel="stylesheet" href="static/style.css">
</head>
<body>
    <h1>License Plate Detection</h1>
        <form action="{{ url_for('upload_file') }}" method="post"
enctype="multipart/form-data">
        <div class="button">
            <input type="file" name="file" accept="image/*,video/*">
            <input type="submit" value="Upload File">
        </div>
    </form>
    {% if extracted_text %}
        <h2>Extracted License Plate: {{ extracted_text }}</h2>
    {% endif %}
</body>
</html>
```

Steps To Run

1. Make sure the following libraries are installed
    - Opencv
    - Yolo from Ultralytics
    - Tesseract
    - Numpy
    - Pandas
    - OS
2. Also ensure Nginx is installed.
3. Make the changes in nginx.conf and add the codes for html in the templates folder.
4. Include app.py in nginx folder.
5. Open terminal from nginx folder and do the following :
    - start nginx
    - py app.py
    - Open the link from the terminal.
6. Upload image or video and select upload.
7. Reload if necessary(nginx –s reload)

Conclusion

This project successfully demonstrates an AI-based license plate recognition system, integrating several advanced technologies. The use of YOLOv8 for detection, combined with Tesseract for text recognition, allows for efficient and accurate processing of license plates. The separation of the frontend and backend using Nginx enhances performance and security.