Project Report

AadhaarShield: Aadhaar Card Masking System

-Asmajahan Karkal

Table of Contents

1. Introduction

1.1. Overview

The Aadhaar Card Masking System is a web-based application developed to ensure the privacy and security of sensitive Aadhaar numbers in images. By using Optical Character Recognition (OCR) with EasyOCR for text detection and OpenCV for image processing, the system identifies Aadhaar numbers and masks them automatically, allowing users to safely share Aadhaar card images without revealing sensitive information.

1.2. Objectives

- Develop a user-friendly web application for uploading Aadhaar card images.
- Automatically detect Aadhaar numbers in the uploaded images using EasyOCR.
- Mask the detected Aadhaar numbers using OpenCV.
- Provide users with the option to download the masked image for further use.

2. System Architecture

2.1. Components

- **Frontend:** HTML/CSS and JavaScript are used for the user interface (UI) that allows users to upload images and view/download the results.
- **Backend:** Flask API is utilized for handling image uploads, processing them, and delivering the masked output.
- **Libraries Used:**
  - **EasyOCR:** Extracts text from images to detect Aadhaar numbers.
  - **OpenCV:** Processes the image to apply a mask over the Aadhaar number.
  - **Matplotlib (optional):** Used for displaying images during testing, if needed.

2.2. Workflow

1. **User Interaction:** The user uploads an Aadhaar card image through the web form on the application.
2. **Image Processing:** The backend receives the image, detects the Aadhaar number using EasyOCR, and applies a black mask using OpenCV to hide the first eight digits.
3. **Response:** The processed image is sent back to the user with the Aadhaar number masked, and the user can view and download the image.

3. Implementation Details

3.1. Frontend Development

3.1.1. HTML Form

The HTML form provides the interface for uploading Aadhaar card images and displaying the masked results.

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aadhaar Card Masking</title>
    <style>
        body {
            background-color: #f9f9f9;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .upload-container {
            background-color: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            text-align: center;
            width: 400px;
        }
        h2 {
            color: #333;
            margin-bottom: 20px;
        }
        input[type="file"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 2px solid #4CAF50;
            border-radius: 5px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
        }
        button:hover {
            background-color: #45a049;
        }
        .result-container {
            margin-top: 20px;
            display: none;
        }
        .result-container img {
            max-width: 100%;
            border: 2px solid #ddd;
            border-radius: 5px;
            margin-bottom: 10px;
        }
```

```
        .download-btn {
            padding: 10px 15px;
            background-color: #4CAF50;
            color: white;
            text-decoration: none;
            border-radius: 5px;
            display: inline-block;
            margin-top: 10px;
        }
        .download-btn:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="upload-container">
        <h2>Aadhaar Card Masking</h2>
        <form id="upload-form" enctype="multipart/form-data">
            <input type="file" id="image" name="image" accept="image/*" required>
            <button type="submit">Upload and Mask</button>
        </form>

        <div class="result-container" id="result">
            <h3>Masked Image Preview</h3>
            <img id="masked-image" src="" alt="Masked Aadhaar Image">
            <br>
            <a id="download-link" class="download-btn" href="" download="masked_image.png">Download Masked Image</a>
        </div>
    </div>

    <script>
        const form = document.getElementById('upload-form');
        const resultDiv = document.getElementById('result');
        const maskedImage = document.getElementById('masked-image');
        const downloadLink = document.getElementById('download-link');

        form.addEventListener('submit', async (event) => {
            event.preventDefault();

            const formData = new FormData();
            formData.append('image', document.getElementById('image').files[0]);

            const response = await fetch('/process_image', {
                method: 'POST',
                body: formData
            });

            if (response.ok) {
                const imageBlob = await response.blob();
                const imageUrl = URL.createObjectURL(imageBlob);

                // Display the masked image on the webpage
                maskedImage.src = imageUrl;
                resultDiv.style.display = 'block';

                // Enable the download link with the processed image
                downloadLink.href = imageUrl;
                downloadLink.style.display = 'inline-block';
            } else {
                alert('Failed to process the image.');
            }
        });
    </script>
</body>
</html>
```

3.2. Backend Development

3.2.1. Flask API

The backend, built using Flask, processes image uploads, detects the Aadhaar number in the image, applies a mask, and returns the processed image.

Code:

```python
from flask import Flask, request, jsonify, render_template, send_file
import easyocr
import re
import cv2
import os
import numpy as np
from io import BytesIO

# Initialize Flask app
app = Flask(__name__)

# Initialize EasyOCR reader
reader = easyocr.Reader(['en'], gpu=False)

# Path to save masked images
output_dir = r"C:\Users\asmaj\Downloads\Aadhaar card Sample\Masked_Aadhar"
os.makedirs(output_dir, exist_ok=True)

# Define the route for the homepage
@app.route('/')
def home():
    return render_template('index.html')

# Define the route for processing the uploaded image
@app.route('/process_image', methods=['POST'])
def process_image():
    # Get the image file from the request
    if 'image' not in request.files:
        return jsonify({"error": "No image uploaded"}), 400

    file = request.files['image']

    # Read the image into OpenCV format
    npimg = np.frombuffer(file.read(), np.uint8)
    image = cv2.imdecode(npimg, cv2.IMREAD_COLOR)

    # Read text from the image using EasyOCR
    result = reader.readtext(image)

    # Create a copy of the image for masking
    image_masking = image.copy()

    # Loop through the result to find Aadhaar numbers
    for i in result:
        # Define the pattern for Aadhaar number
        aadhar_pattern = r'\b\d{4}\s?\d{4}\s?\d{4}\b'

        # Find all matches in the text
        found_matches = re.findall(aadhar_pattern, i[1])
        if found_matches:
            for match in found_matches:
                bounding_box = i[0]

                # Define the color and thickness of the mask
                mask_color = (0, 0, 0)  # Black mask
                mask_thickness = -1     # Fill the rectangle

                # Calculate the width for the first 8 digits
                width = bounding_box[1][0] - bounding_box[0][0]
                mask_width = int(width * 0.67)  # Adjust this factor if needed

                # Create a new bounding box to cover the first 8 digits
                x1, y1 = bounding_box[0]
                x2, y2 = x1 + mask_width, bounding_box[2][1]

                # Draw the mask
                cv2.rectangle(image_masking, (x1, y1), (x2, y2), mask_color, mask_thickness)

    # Convert masked image to a format that can be displayed
    _, buffer = cv2.imencode('.png', image_masking)
    img_io = BytesIO(buffer)
    print("Done with the masking")
    # Return the masked image
    return send_file(img_io, mimetype='image/png')

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

3.2.2. Code Explanation

- **Image Upload:** The backend receives the uploaded image file.
- **Text Extraction:** EasyOCR detects and reads text from the image, including Aadhaar numbers.
- **Masking:** OpenCV masks the Aadhaar number by blacking out the first eight digits.
- **Response:** The processed image is returned to the user for viewing and downloading.

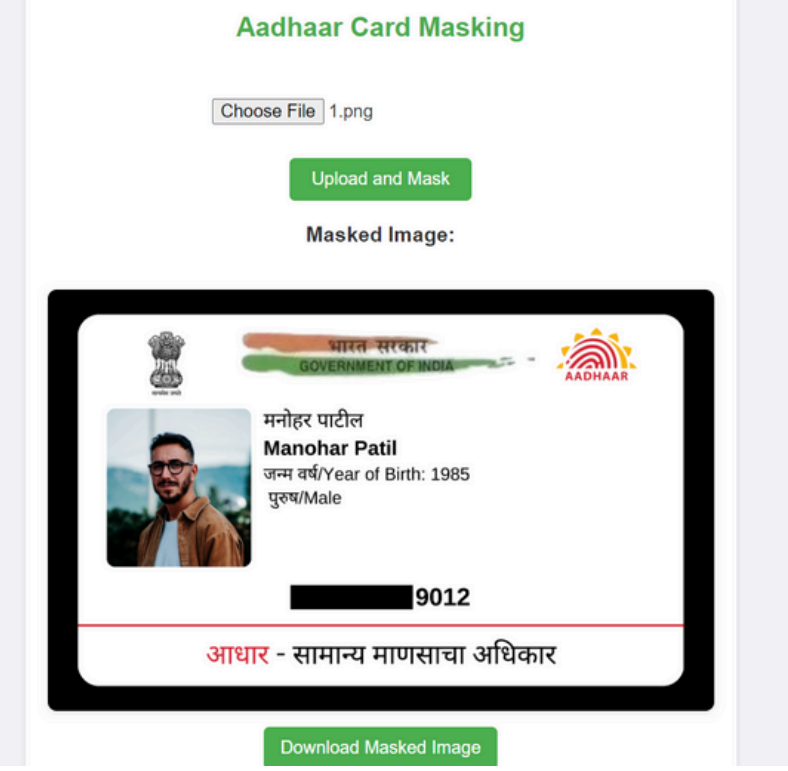
4. Testing and Validation

4.1. Testing

- **Upload Tests:** Tested with multiple Aadhaar card images to ensure accurate number detection and correct masking.
- **Edge Cases:** Handled cases where no Aadhaar number is present or the image is of poor quality.

4.2. Validation

- **Accuracy:** Ensured that Aadhaar numbers are properly detected and masked.
- **Usability:** Verified that the web interface is functional and easy to use on various devices and browsers.



5. Deployment

5.1. Deployment Options

- **Local Deployment:** Flask can be run locally for development and testing purposes.
- **Cloud Deployment:** The system can be deployed on cloud platforms like Heroku, AWS, or Google Cloud for broader access.

5.2. Steps for Deployment

1. Prepare the application for deployment (e.g., using Gunicorn for production).
2. Select a hosting platform and configure the environment.
3. Deploy the application and perform post-deployment testing.

6. Conclusion

6.1. Summary

The Aadhaar Card Masking System provides a secure and effective way to mask Aadhaar numbers in images. By integrating EasyOCR and OpenCV, the project ensures the protection of sensitive information and offers a simple, user-friendly interface.

6.2. Future Enhancements

- Security: Add security features such as encrypted file handling.
- Multilingual Support: Extend OCR support to multiple languages.
- User Experience: Enhance the UI and provide more customization options for users.

6.3. Contact Information

- **Name:** Asmajahan Karkal
- **Email:** asmajahankarkal@gmail.com
- **LinkedIn Profile :** Asmajahan Karkal