Face Recognition and Attendance Tracking Documentation

Table of Contents

Introduction

Prerequisites

Installation

Usage

Code Structure

Features

Sample Input and Output

License

Team members

1. Introduction

The Face Recognition and Attendance Tracking code is a Python script designed to perform face recognition and attendance tracking using the MTCNN (Multi-task Cascaded Convolutional Networks) face detection model and the face\_recognition library. This documentation provides an extensive guide to understand, use, and customize the code for your specific needs.

2. Prerequisites

Before using the code, ensure you have the following prerequisites installed:

Python 3.x

Python is a versatile, high-level programming language used for various applications, including this face recognition and attendance tracking script.

OpenCV (cv2)

OpenCV, or Open Source Computer Vision Library, is a powerful library for computer vision and image processing. In this script, it is used for image processing and face detection.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. This code utilizes Matplotlib to display images with bounding boxes around recognized faces.

face\_recognition

The face\_recognition library is a popular Python library for face recognition and manipulation. It is employed in this script for recognizing and comparing faces with known faces.

MTCNN

MTCNN, or Multi-task Cascaded Convolutional Networks, is a deep learning model used for face detection. In this script, MTCNN is utilized to detect faces in input images.

NumPy

NumPy is a fundamental library for numerical and array operations in Python. It is used for various calculations and data manipulation in this script.

Gradio

Gradio is a Python library for creating user interfaces for machine learning models. While not a mandatory prerequisite, it is an optional dependency if you wish to create a user interface for your face recognition application.

You can install these libraries and dependencies using the provided pip commands in the "Prerequisites" section of the README.

3. Installation

Clone or download the code repository to your local machine.

Organize your known face images in the "Individual images" folder within the code directory.

Create a CSV file named known\_face\_encodings.csv containing known face encodings. The file should be structured as rows, with each row representing a known individual's face encoding.

4. Usage

To use the code, follow the steps outlined in the "Usage" section of the README.

5. Code Structure

The code is organized into several sections, as described in the README. This includes importing libraries, loading known faces and encodings, processing the input image, detecting faces, recognizing faces, and displaying results.

6. Features

Key Features of the Code:

Face Detection: The code uses the MTCNN model for accurate face detection in input images, with support for multiple faces in a single image.

Face Recognition: It leverages the face\_recognition library to compare detected faces with known faces and identify individuals accurately.

Attendance Tracking: The script generates attendance data by recognizing individuals in the input image and saving the results to a CSV file, providing an efficient way to automate attendance recording.

Customizable Threshold: You can adjust the similarity threshold to control the recognition sensitivity, allowing you to fine-tune the recognition process for your specific use case.

Visualization: Matplotlib is used to display images with bounding boxes around recognized faces and their names, making it user-friendly and understandable.

Optional User Interface: The code can be integrated with Gradio to create a user interface for face recognition, enhancing user interaction and ease of use.

Versatile Usage: The code is adaptable for various applications, including attendance tracking, access control, and more.

Open Source: This code is open source and can be customized and extended for your specific requirements.

Efficiency: The code is designed to be efficient and can process images with a large number of faces in real-time.

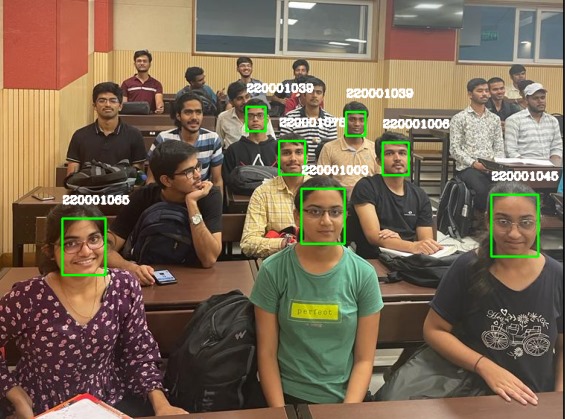
7. Sample Input and Output

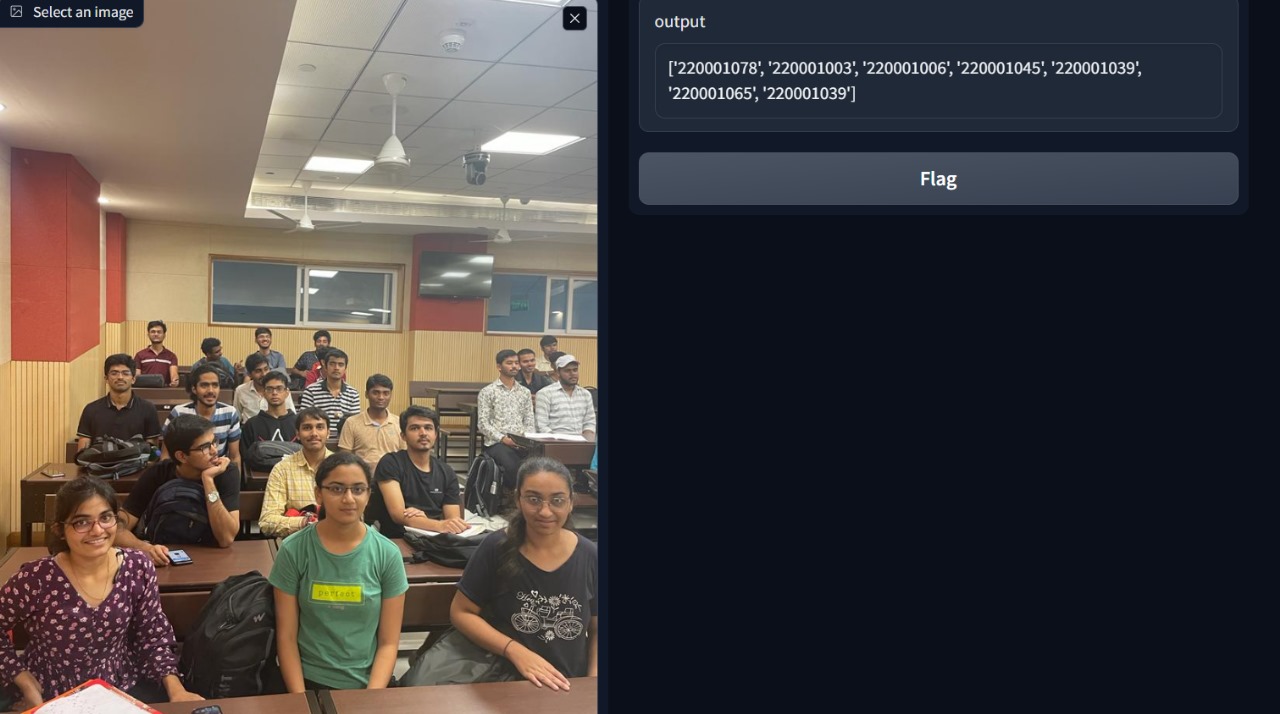
For a better understanding, here's a sample input image and its corresponding output with recognized faces and attendance data, as explained in the README.

Input Image:



Output image:





The code has recognized the faces and generated attendance data based on the known faces.

8. License

This code is provided under the MIT License. See the LICENSE file for details

9. Team members:

Muskan

Nikita Tayade

Kanak Nagar