`

Code Report: Real-time Face Recognition for Attendance Tracking using AI

1. Introduction:

The Python script provides a robust real-time face recognition system for attendance tracking. Leveraging OpenCV, face_recognition, and deepface libraries, the code ensures efficient processing and flexibility in model selection. The system caters to various scenarios, allowing users to choose between 'face_recognition' and 'deepface' models.

2. Import Statements:

The script imports a diverse set of libraries, demonstrating a comprehensive approach to computer vision, deep learning, concurrency, and data manipulation. The use of concurrent. futures.ThreadPoolExecutor indicates a strategic focus on parallelizing face recognition tasks.

3. Class Definition: MultiFacerec:

Initialization Method (__init__):

- **Encapsulation of parameters and initialization logic.**
- **Dynamic choice of face recognition model.**
- **Efficient storage and management of attendance-related data structures.**

Methods:

- **load_or_download_encodings: Implementation of data persistence and retrieval for known face encodings.**
- **download_and_store_encodings: Dynamic processing of image files, handling duplicate entries intelligently.**
- **mark_attendance: Dynamic timestamp handling for attendance marking.**
- **mark_absentees: Robust computation of absentees using set operations.**
- **sort_and_write_csv: Flexible CSV writing mechanism considering options for timestamps and removal of unrecognized entries.**

`

- **detect_known_faces: Implementation of real-time face detection and recognition using concurrent processing.**
- **process_face: Advanced handling of face recognition results.**
- **draw_face_rectangle: Advanced visual feedback for recognized faces.**
- **get_face_embedding: Modular method for obtaining face embeddings, allowing for model flexibility.**

4. Models Used:

- **Face_recognition Model:**
  - **In-depth utilization of the face_recognition library for face-recognition tasks.**
  - **Application of face_recognition.face_locations and face_recognition.face_encodings functions.**
- **Deepface Model:**
  - **Advanced integration of deepface library for face recognition.**
  - **Utilization of DeepFace.detectFace and DeepFace.represent functions.**
- **Model Flexibility:**
  - **A key feature allowing users to seamlessly switch between 'face_recognition' and 'deepface' models.**
  - **Future extensibility for incorporating additional models.**

5. Main Code Execution:

- **Dynamic Model Selection:**
  - **Instantiation of MultiFacerec class with the default model 'face_recognition'.**
  - **Dynamic switching between recognition models enhances code adaptability.**
- **Real-time Video Processing:**
  - **OpenCV integration for real-time video stream capture.**
  - **Optimized frame skipping for computational efficiency.**
- **Attendance Management:**

`

- **Seamless integration of attendance marking, absentee computation, and CSV writing.**
- **User-friendly termination key handling for a smooth program exit.**

6. Overall Analysis:

- **Robustness:**
  - **The code demonstrates robustness in handling real-time face recognition with efficiency.**
  - **Smart handling of attendance-related data structures ensures accuracy and reliability.**
- **Flexibility:**
  - **Model flexibility allows users to experiment with different face recognition approaches.**
  - **Class-based structure promotes code modularity and reuse.**
- **Readability and Organization:**
  - **The code is well-organized with meaningful method names and comments.**
  - **Logical structuring enhances readability, aiding developers in understanding and extending the codebase.**
- **Performance Optimization:**
  - **The implementation showcases performance optimization strategies such as parallel processing and frame skipping.**
  - **Efficient use of sets and dictionaries minimizes redundancy and improves computation speed.**
- **Advanced Visual Feedback:**
  - **Visual feedback for recognized faces is advanced, providing a rich user experience.**
  - **Rectangle drawing and text placement enhance the interpretability of recognition results.**
- **Data Persistence and Retrieval:**
  - **The code efficiently manages face encodings through serialization and deserialization.**
  - **Intelligent processing of image files ensures data integrity.**

`

7. Advantages:

- **Modularity and Organization:**
  - **The class-based organization enhances modularity, making the code maintainable and extensible.**
  - **Encapsulation of functionality within methods promotes a clean and structured design.**
- **Dynamic Model Selection:**
  - **Users can choose between 'face_recognition' and 'deepface' models, providing flexibility based on requirements.**
  - **This adaptability allows experimentation with different recognition approaches.**
- **Efficient Data Handling:**
  - **The code efficiently computes absentees by smartly subtracting presentees from all known entries.**
  - **Data structures like sets and dictionaries are used intelligently to minimize redundancy.**
- **Real-time Video Processing:**
  - **Frame skipping enhances computational efficiency, ensuring smooth real-time video processing.**
  - **Optimized video stream capture contributes to a responsive and user-friendly experience.**
- **Advanced Visual Feedback:**
  - **Visual feedback for recognized faces, including rectangle drawing and text placement, enhances the interpretability of recognition results.**
  - **The code provides a rich user experience by conveying information about attendance in real time.**
- **Performance Optimization:**
  - **Strategies like parallel processing with ThreadPoolExecutor enhance performance during face recognition.**
  - **Frame skipping and dynamic attendance handling contribute to efficient real-time processing.**

8. Conclusion:

`

The provided code is an advanced solution for real-time face recognition in attendance tracking scenarios. Leveraging the capabilities of face_recognition and deepface libraries, the implementation achieves a balance between performance, flexibility, and readability. The modular class-based structure, along with dynamic model selection, positions the code for future enhancements and extensions. Overall, this script serves as a powerful foundation for the integration of AI-based attendance tracking systems in various contexts.

Manual

Introduction:

The Real-time Face Recognition for Attendance Tracking script is designed for facial recognition in a live video stream, facilitating attendance tracking. This manual provides an in-depth explanation of the code's structure, functionality, and usage. It aims to enhance readability and assist users in understanding and customizing the script for their specific requirements.

Dependencies:

Before running the code, ensure the installation of the required libraries:

bash

pip install opencv-python numpy face_recognition deepface

Class: MultiFacerec

Initialization Method (__init__):

The __init__ method encapsulates parameters for initialising the MultiFacerec class.

Key parameters include:

- **model (default: 'face_recognition'): Specifies the face recognition model ('face_recognition' or 'deepface').**
- **frame_resizing (default: 0.5): Sets the resizing factor for frames.**
- **csv_file_path: Sets the file path for the main attendance CSV.**
- **absentees_csv_file_path: Sets the file path for the absentees CSV.**

`

- **presentees, absentees, unique_entries, unique_names: Sets for tracking attendance and unique entries.**
- **known_face_encodings, known_face_names: Lists for storing known face encodings and names.**

Methods:

- **load_or_download_encodings Method:**
  - **Checks if the face encodings file exists and loads it; otherwise, downloads and stores the encodings.**
- **download_and_store_encodings Method:**
  - **Iterates through face images extracts names and USNs, processes images, and updates known face encodings and names.**
  - **Saves the obtained face encodings to a pickle file.**
- **mark_attendance Method:**
  - **Marks attendance for a recognized person if conditions are met, including dynamic timestamp handling.**
- **mark_absentees Method:**
  - **Determines absentees by finding the set difference between all known entries and the set of presentees.**
- **sort_and_write_csv Method:**
  - **Writes a CSV file with sorted and unique entries, considering options to include timestamps and remove unrecognized entries.**
- **detect_known_faces Method:**
  - **Uses face_recognition library to detect face locations and encodings in a given frame.**
  - **Uses ThreadPoolExecutor for parallel processing of face recognition.**
- **process_face Method:**
  - **Compares face encodings with known face encodings and marks attendance if a match is found.**
  - **Draws rectangles around faces and prints attendance information.**
- **draw_face_rectangle Method:**
  - **Draws a rectangle and adds text on the given frame around the detected face.**

`

- **get_face_embedding Method:**
    - **Retrieves face embeddings using either the face_recognition or deepface library based on the selected model.**

Main Code

- **Initialization:**
    - **Creates an instance of MultiFacerec with the default model 'face_recognition'.**
- **Video Capture:**
    - **Opens a video stream from an iPhone camera using OpenCV.**
- **Processing Loop:**
    - **Loops through frames, skipping some frames for efficiency.**
    - **Calls detect_known_faces to perform face recognition.**
    - **Displays the processed frame and handles the termination key (Esc) to mark absentees, write attendance CSV, and exit the program.**

Conclusion:

The provided script is an advanced and flexible solution for real-time face recognition in attendance tracking scenarios. The modular structure, dynamic model selection, and performance optimization strategies make it a powerful foundation for AI-based attendance tracking systems. Users are encouraged to explore and adapt the code to meet specific requirements.