Report for Cognitive Computing (UCS-420)

By

GROUP : ASTARR

NAME : Brahmjeet Singh (102317163)
Harsh Badhan (102317140)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,
(A DEEMED TO BE UNIVERSITY), PATIALA

Jan-May, 2025

Face Recognition-Based Attendance System

1. Introduction & Problem Statement

Traditional attendance systems, like roll calls or RFID-based methods, are inefficient, prone to errors, and can be manipulated for proxy attendance. In educational institutions and workplaces, an automated, secure, and efficient method is required to ensure accuracy in attendance tracking. Our project aims to develop an AI-based **Face Recognition Attendance System** that captures, processes, and verifies faces in real time, marking attendance securely and automatically.

2. Dataset Overview

The dataset consists of images captured through a webcam, stored under a structured folder hierarchy for different individuals. These images are later processed to extract facial encodings, which are used for real-time recognition.

- **Data Collection**: The system captures multiple images per person to improve accuracy.
- **Storage Format**: Images are stored in the `dataset/` directory, categorized by person name.
- **Processed Data**: Facial encodings are stored in a `pickle` file for quick retrieval.

3. Technology Stack

Our system is built using the following technologies:

- **Programming Language**: Python
- **Libraries & Frameworks**:
  - `OpenCV` – For image and video processing
  - `face_recognition` – For face detection and feature extraction
  - `pickle` – For storing and retrieving face encodings
  - `pandas` – For managing attendance records
  - `seaborn` & `matplotlib` – For data visualization
  - `sklearn.neighbors.KDTree` – For fast face matching
- **Hardware**: Webcam for real-time face capture
- **Storage**: CSV files for attendance records

4. ML Model Implementation & Evaluation

The system is divided into four main scripts:

1. Data Collection Script

- Captures images of individuals using a webcam.
- Saves images in a structured dataset directory.
- Uses `cv2.VideoCapture()` for real-time video streaming.

2. Face Encoding Script

- Reads images from the dataset and extracts face embeddings.
- Uses `face_recognition.face_encodings()` to generate a unique encoding for each person.
- Saves the encodings and names in a `pickle` file for future use.

3. Real-Time Face Recognition & Attendance Marking

- Loads the stored encodings and starts live video capture.
- Detects faces in real-time and matches them using **KDTree** for fast searching.
- Marks attendance in a CSV file with timestamps.
- If an unrecognized face appears, an alert sound is played.

4. Attendance Visualization

- Reads the attendance CSV file.
- Generates visualizations like bar charts (attendance count per person) and scatter plots (attendance timeline).

5. Results & Insights

- The system successfully identifies registered individuals and marks their attendance.
- Proxy attendance is prevented by using facial recognition instead of RFID or manual methods.
- The **KDTree-based** search speeds up the face matching process significantly.
- Attendance data is well-organized and can be analyzed using visualizations.

6. Challenges & Future Improvements

Challenges Faced

- Low-light conditions affected recognition accuracy.
- Some faces were not detected due to improper angles or obstructions.
- Alert sound playback had compatibility issues on certain systems.

Future Improvements

- Implement **deep learning-based face detection** for better accuracy.
- Use **infrared cameras** to improve performance in low-light conditions.
- Deploy the system as a **web or mobile app** for broader usability.

7. Conclusion & Learnings

This project demonstrated the potential of AI in automating attendance systems. We learned how to integrate computer vision, facial recognition, and machine learning for real-world applications. The system, though effective, has scope for improvement in terms of accuracy, speed, and deployment.

8. References

1. OpenCV Documentation - https://docs.opencv.org/
2. Face Recognition Library - https://github.com/ageitgey/face_recognition
3. Pandas Documentation - https://pandas.pydata.org/docs/
4. KDTree (Scikit-learn) - https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html