ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DAY – 22 22 July 2025

PROJECT : Face Recognition-Based Attendance System

Workflow

The development of the Face Recognition-Based Attendance System was carried out using a structured and step-by-step methodology. This ensured clear planning, efficient implementation, and successful testing of the project. The approach followed can be summarized as follows:

- Problem Understanding and Planning
    - Identified the drawbacks of traditional attendance systems (manual roll calls, fingerprint scanners, proxy attendance, etc.).
    - Defined project scope, objectives, tools, and requirements.
- Dataset Preparation
    - Collected facial images of known individuals and stored them in a directory named dataset/.
    - Each image was named using the corresponding person's name (e.g., john.jpg), which helped in labeling during recognition.
- Face Encoding
    - Using Python and a face recognition library, we converted each face image into numbers called encodings.
    - These encodings represent the unique features of each person's face.
    - The system saved these encodings and the names for later matching.
- Real-Time Face Detection and Recognition
    - Captured live video stream using OpenCV's webcam interface.

- Extracted encodings from detected faces and compared them to known encodings.
- Calculated the similarity between encodings using Euclidean distance.

➢ Marking Attendance

- When a person was recognized, their name and the current time were saved in a file called attendance.csv.
- The system only marked attendance once per person per session, to avoid duplicates.
- We used the date and time to know when each person was present.

➢ User Interface and Feedback

- Displayed real-time video with bounding boxes around detected faces:
  - Green boxes for recognized individuals (with names).
  - Red boxes for unknown individuals.
- Allowed users to terminate the attendance session using a keyboard input.
- Displayed system messages and attendance status via on-screen overlays.

➢ Testing and Validation

- Conducted tests under different lighting conditions and camera angles.
- Evaluated the system's ability to detect and recognize faces in real-time.
- Verified that the attendance file was updated correctly without duplicates.
- Tested the system with both known and unknown faces to validate accuracy and robustness.

Code Implementation

```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
import pandas as pd

# Load known faces from 'dataset' folder
path = 'Data'
images = []
classNames = []
myList = os.listdir(path)

print('Encoding known faces...')

for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    if curImg is not None:
        images.append(curImg)
        classNames.append(os.path.splitext(cl)[0])

def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encodings = face_recognition.face_encodings(img)
        if encodings:
            encodeList.append(encodings[0])
    return encodeList
```

```
csv_filename = 'attendance.csv'
# Initialize the CSV file if it doesn't exist
if not os.path.exists(csv_filename):
    df = pd.DataFrame(columns=['Name', 'Time'])
    df.to_csv(csv_filename, index=False)
marked_names = set()

def markAttendance(name):
    if name == "Unknown":
        print("Alert: Detected face not found in the database. Not logging attendance.")
        return
    if name in marked_names:
        print(f"{name} already marked present.")
        return
    now = datetime.now()
    dtString = now.strftime('%Y-%m-%d %H:%M:%S')
    df = pd.DataFrame([[name, dtString]], columns=['Name', 'Time'])
    df.to_csv(csv_filename, mode='a', header=False, index=False)
    marked_names.add(name)
    print(f'{name} marked at {dtString}')
encodeListKnown = findEncodings(images)
print('Encoding Complete.')

cap = cv2.VideoCapture(0)

while True:
    success, img = cap.read()
    if not success:
        break
```

```
    imgS = cv2.resize(img, (0, 0), fx=0.25, fy=0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
        name = "Unknown"
        if encodeListKnown:
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
            if len(matches) > 0:
                matchIndex = np.argmin(faceDis)
                if matches[matchIndex] and faceDis[matchIndex] < 0.5:
                    name = classNames[matchIndex].upper()
        y1, x2, y2, x1 = faceLoc
        y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
        color = (0, 255, 0) if name != "Unknown" else (0, 0, 255)
        cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
        cv2.rectangle(img, (x1, y2 - 35), (x2, y2), color, cv2.FILLED)
        cv2.putText(img, name, (x1 + 6, y2 - 6),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
        markAttendance(name)
    cv2.imshow('Webcam', img)
# Press 'q' to exit the video stream
    if cv2.waitKey(1) & 0xFF == ord('q'):
        print("Exit command received. Closing webcam...")
        break
cap.release()
cv2.destroyAllWindows()
```