ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DAY – 21

Date: Jul 21, 2025

Workflow: Face Recognition-Based Attendance System

1. Load Known Faces (Initialization Phase)

- Load images of known individuals from the dataset/ folder.
- Extract names from image file names (e.g., John.jpg → John).
- Encode each face using the face_recognition library (128-d feature vector).
- Store these encodings in a list for later comparison.

2. Start Webcam Feed

- Use OpenCV to activate the webcam.
- Continuously capture frames in real time.

3. Preprocess Each Frame

- Resize the captured frame to improve processing speed.
- Convert frame color from BGR to RGB (OpenCV default to face_recognition compatible).

4. Detect and Encode Faces in Frame

- Detect face locations using face_recognition.face_locations().
- Encode detected faces using face_recognition.face_encodings().

5. Compare Detected Faces with Known Encodings

- Use face_recognition.compare_faces() and face_distance():
- Calculate the distance between detected and known face encodings.
- Choose the closest match (smallest distance).

6. Mark Attendance

- If a match is found and not already marked:
- Get current date & time using datetime.now().
- Record the name and timestamp in the CSV file using pandas.
- Display the name and bounding box on the webcam feed.

7. Save & Display Attendance

- Keep updating the CSV file with new recognized faces and timestamps.
- Display attendance list or save it for admin use.

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 = 'dataset'

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

# Create a timestamped CSV filename
# Create a Windows-safe timestamped CSV filename
now = datetime.now()
timestamp_str = now.strftime('%Y-%m-%d_%H-%M-%S')  # <-- CORRECTED (no colons)
csv_filename = f'attendance_{timestamp_str}.csv'




# Initialize the CSV file
df = pd.DataFrame(columns=['Name', 'Time'])
df.to_csv(csv_filename, index=False)

def markAttendance(name):
    global df
    now = datetime.now()
    dtString = now.strftime('%Y-%m-%d %H:%M:%S')

    if name == "Unknown":
        # Skip logging unknown faces
        print("Unknown face detected. Not marking attendance.")
        return

    if name in df['Name'].values:
        print(f'{name} is already marked.')
    else:
        df.loc[len(df)] = {'Name': name, 'Time': dtString}
        df.to_csv(csv_filename, index=False)
        print(f'{name} marked at {dtString}')

# Encode known faces
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)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
```