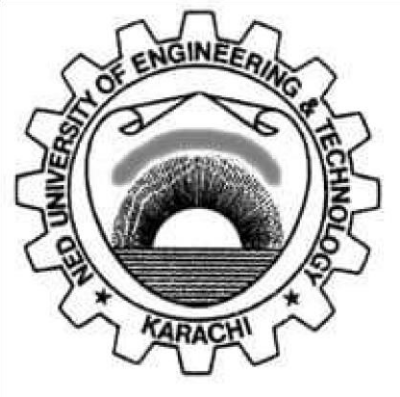

ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEM
(CT-361)

PROJECT REPORT

GROUP MEMBERS:

1. **FAIQ-UZ-ZAMAN (CR-040)**
2. **RAYYAN MIRZA (CR-033)**
3. **ASHHAD NEHAL (CR-014)**
4. **HASHIR BAIG (CR-013)**

Project Title:

AI BASED PERSON IDENTIFIER

1. Problem Statement

Traditionally, security personnel or police officers review hours of CCTV footage to detect theft or identify suspects manually. This process is time-consuming, prone to human error, and inefficient. Our proposed solution automates the detection of known individuals (e.g., thieves) in surveillance footage using Artificial Intelligence (AI), particularly facial recognition. This reduces human effort, accelerates response time, and increases reliability.

---

2. Solution Overview

This project uses AI-driven face recognition to detect the presence of known criminals or suspects in CCTV footage. We load known images of individuals, encode their facial features, and then process each frame of a surveillance video to identify if any of the known faces appear. If a match is found, it is highlighted in the video along with a timestamp.

---

3. Tools and Technologies

Tool/Library	Purpose
Python	Core programming language
OpenCV (cv2)	Video processing, frame capture, and annotation
face_recognition	Facial detection and encoding
Os	File and directory handling for reference images

---

4. AI and Computing Concepts Used

- **Face Detection:** Identifying the location of human faces in video frames.
- **Face Encoding:** Converting facial features into numerical vectors for comparison.
- **Face Matching:** Comparing encodings with a known database using distance metrics.
- **Frame-by-Frame Processing:** Scanning each frame for facial matches and annotating output.

---

5. System Architecture and Flow

1. Load reference images and encode faces.
2. Open the video file.
3. Read frames one by one.
4. Resize and convert color formats.
5. Detect and encode faces in each frame.
6. Compare encodings with known database.
7. If a match is found:
    - Draw a bounding box.
    - Display "Match" label.
    - Store frame number and timestamp.
8. Save the annotated video.

---

6. Code Explanation and Justification

load_reference_faces(reference_folder)

- **Purpose:** Load all images from a folder and encode faces using the face_recognition library.
- **Justification:** This creates a database of known individuals for comparison.

detect_and_annotate_video(video_path, known_encodings, output_path)

- **Purpose:** Process each frame in the video, detect faces, compare them to known faces, and annotate matches.
- **Justification:** This is the core of the solution—using AI to analyze visual data in real-time.

os.listdir()

- **Purpose:** Retrieve filenames from the reference folder.
- **Justification:** Dynamic loading of reference images ensures modularity and flexibility.

cv2.VideoCapture() & cv2.VideoWriter()

- **Purpose:** Handle reading and writing video files.
- **Justification:** Necessary for input/output handling of surveillance footage.

face_recognition.compare_faces()

- **Purpose:** Compare current face encodings with known encodings.
- **Justification:** Implements AI-driven decision making using distance-based comparison.

---

7. Drafted Assumptions

- The reference folder contains clear, front-facing images of known individuals.
- The surveillance footage has sufficient lighting and resolution for facial recognition.
- Only one known person needs to be identified in a video.

---

8. Results

If a known individual is detected, the output includes:

- A bounding box and "Match" label drawn around the face.
- A list of frames and corresponding timestamps where the match was found.
- A video file (`output_with_matches.mp4`) showing the annotated results.

---

9. Conclusion

This project demonstrates the effectiveness of applying Artificial Intelligence, specifically facial recognition, to automate the process of theft detection in surveillance videos. The solution increases security efficiency, reduces human workload, and enhances real-time monitoring capabilities.

---

10. References

- face_recognition GitHub
- OpenCV Documentation
- Python OS Library