UEC 2604 MACHINE LEARNING

FACE RECOGNITION BASED ATTENDANCE SYSTEM

- Balaji G --(3122213002015)

OBJECTIVE

To develop a Face Recognition Based Attendance System that:

- **Automates Attendance**: Eliminates manual input for time-saving and error reduction.
- **Enhances Accuracy**: Reduces proxy and human errors, improving record reliability.
- **Improves Security**: Offers a secure, contactless identification method.

OUTLINE

- The Face Recognition Based Attendance System is designed to streamline attendance tracking in educational settings using advanced machine learning models. The system is structured into two primary components:
- **Monitoring Module:** This component integrates face detection to identify students at key entry points, followed by face recognition powered by a custom-developed machine learning model. The model is trained to distinguish individual students with high accuracy, ensuring reliable logging of attendance.
- **Management Module:** Post-recognition, this module is responsible for managing the collected data. It systematically compiles, sorts, and exports attendance records for specified dates, facilitating straightforward administrative review and oversight.

LITERATURE REVIEW

S.NO.	TITLE	AUTHOR	INFERENCE
1.	A real-time face recognition system based on the improved LBPH algorithm	Xu Zhao, Chen Wei	This study enhances the Local Binary Pattern Histogram (LBPH) algorithm by incorporating median neighborhood values to improve recognition accuracy under varied conditions. The modified LBPH (MLBPH) showed superior performance compared to the standard method in tests on recognized databases.
2.	A Multi-Face Challenging Dataset for Robust Face Recognition	Shiv ram Dubey, Snehasis Muhkerjee	This study created a new set of face images called IIITS_MFace, which has lots of tricky factors like different poses, lighting, and things covering faces. This new test is tougher than the old ones and gives a good challenge to new face recognition methods.

3.	**An approach for Face Detection and Face Recognition using OpenCV and Face Recognition Libraries in Python**	**Vasatha bhavani, Kothamasu santha Priya,Kadavakollu sravani, Srivarshitha**	**The Python programming language, along with OpenCV and face recognition libraries, is chosen for its simplicity and effectiveness in implementing facial recognition systems. The study proposes an algorithm for face detection and recognition, aiming to identify multiple faces in images with high accuracy.**
4.	**Title: Real-time Object Detection using Haar Cascade Classifier for Robot Cars**	**S. Gharge, Aditi Patil, Shrutika Patel, Vaishnavi. K. Shetty, Nidhi Mundhada**	**This study investigates how well the Haar Cascade Classifier can spot objects in a self-driving car's surroundings. It explains how the algorithm is trained and tested, showing its promise in making autonomous vehicles safer and more efficient by accurately spotting and recognizing object**

REQUIREMENTS

Software Requirements:

Programming Language: Python

Libraries: OpenCV, NumPy, Pandas

IDE: PyCharm, Jupyter Notebook, or Visual Studio Code Version Control: Github

Hardware Requirements:

High-resolution camera (infrared capable for low light) - (We used inbuilt cameras in Laptops)

Adequate storage for image and data logs

Ml Algorithms:

Face Detection: Haar Cascades Classifier

Face Recognition: LBPH

Data Preprocessing: Image resizing, noise reduction, histogram equalization

PROCESS OVERVIEW

Data Collection:

Collect multiple images of students.

Apply various filters to simulate different lighting conditions.

Preprocessing:

Resize and perform noise reduction on the images.

Apply adaptive histogram equalization.

Conduct augmentation transformations including rotation and horizontal flipping.

Training :

Utilize In-built functions in OpenCV

(cv2.face.LBPHFaceRecognizer_create()) to create and train the ML model for face recognition with preprocessed images.

PROCESS

Monitoring Module :

Implement the trained model in two distinct scripts:

IN Script: Records entry times of students.

OUT Script: Records exit times of students.

Store timestamps in a CSV file, organized by date for easy access.

Management Module:

Generate and export daily attendance reports from the t imestamp files into a CSV format

DATA COLLECTION

- In educational settings, it's critical to gather a diverse set of images for each student that reflect the various conditions they might face at school. This ensures the face recognition system can reliably identify students under different circumstances.

Implementation:

Initial Image Collection:

During student enrollment, we capture three distinct images per student—frontal and slightly angled shots—to establish a foundational dataset.

Simulated lighting conditions:

To mimic various lighting scenarios (bright daylight, overcast, indoor lighting), we apply filters such as cv2.add and cv2.multiply using OpenCV. This technique creates necessary lighting diversity without needing multiple setups.

PREPROCESSING

- Preprocessing is essential to ensure that the simulated lighting variations do not affect the model's accuracy.

Implementation:

Image Resizing And Noise Reduction:

Images are standardized to 100x100 pixels using cv2.resize. Noise reduction is performed with cv2.fastNlMeansDenoisingColored to enhance clarity.

Augmentation Techniques:

Augmentations including rotation (cv2.getRotationMatrix2D), horizontal flipping (cv2.flip), and scaling are used to broaden the dataset, helping the model recognize faces from different angles and scales akin to real-world scenarios.

HAAR CASCADE CLASSSIFIER

- The Haar Cascade Classifier is a computer vision algorithm for object detection, notably effective for face detection.
- It uses Haar-like features (edges, lines, and rectangles) to identify facial structures in images. The detection process involves several steps:
    1. **Calculating Haar Features**: Recognizes different facial features.
    2. **Creating Integral Images:** Optimizes the computation of Haar features.
    3. **Using Adaboost:** Selects the most relevant features.
    4. **Implementing Cascading Classifiers**: Filters out regions at each stage to focus on likely face areas.

Implementation :

Initialization of the Classifier:

At system start, we load the haarcascade_frontalface_default.xml from OpenCV, containing pre-trained data for detecting frontal faces.

Real-Time Image Capture and Conversion:

Video streams from high-resolution cameras at entry and exit points are captured and converted to grayscale, simplifying the data for faster processing.

Face Detection in Frames:

The detectMultiScale method scans the grayscale images to detect faces, returning bounding boxes that highlight each face's location and size.

LOCAL BINARY PATTERNS HISTOGRAMS

- Local Binary Patterns Histograms (LBPH) is an effective face recognition method that utilizes texture and appearance analysis.
- It works by segmenting an image into small regions and extracting features through comparisons of each pixel with its neighbors, generating a binary pattern.
- These local binary patterns are then transformed into a histogram, which the system uses to recognize faces by comparing against histograms stored in a database.

Implementation:

Model Initialization:

The LBPH recognizer is initialized using OpenCV's cv2.face.LBPHFaceRecognizer_create(), preparing it for training with preprocessed images.

Training The Recognizer:

Images after post-preprocessing and labeling, are used for training. The recognizer learns from the histograms of these images, associating them with specific labels (IDs) of students.

Real-time Recognition Process:

During system operation, detected faces (via Haar Cascade) are processed to create histograms. These are then matched against the trained histograms to identify the student.

Handling recognition results:

The recognizer outputs a label and a confidence score. A high confidence score confirms the recognition, leading to logging the student's attendance. Lower scores might trigger manual checks or further verification.

RESULTS

For testing the process, we assumed a simple case with setup as,

Dataset :

Images of three students with a minimum of three images each.

Applied filters during preprocessing to mimic real-time lighting conditions and angles to simulate real-world variations.

Hardware Setup:

Utilized a laptop to run IN and OUT monitoring scripts.

Standard webcam used, highlighting the need for better hardware in future tests.

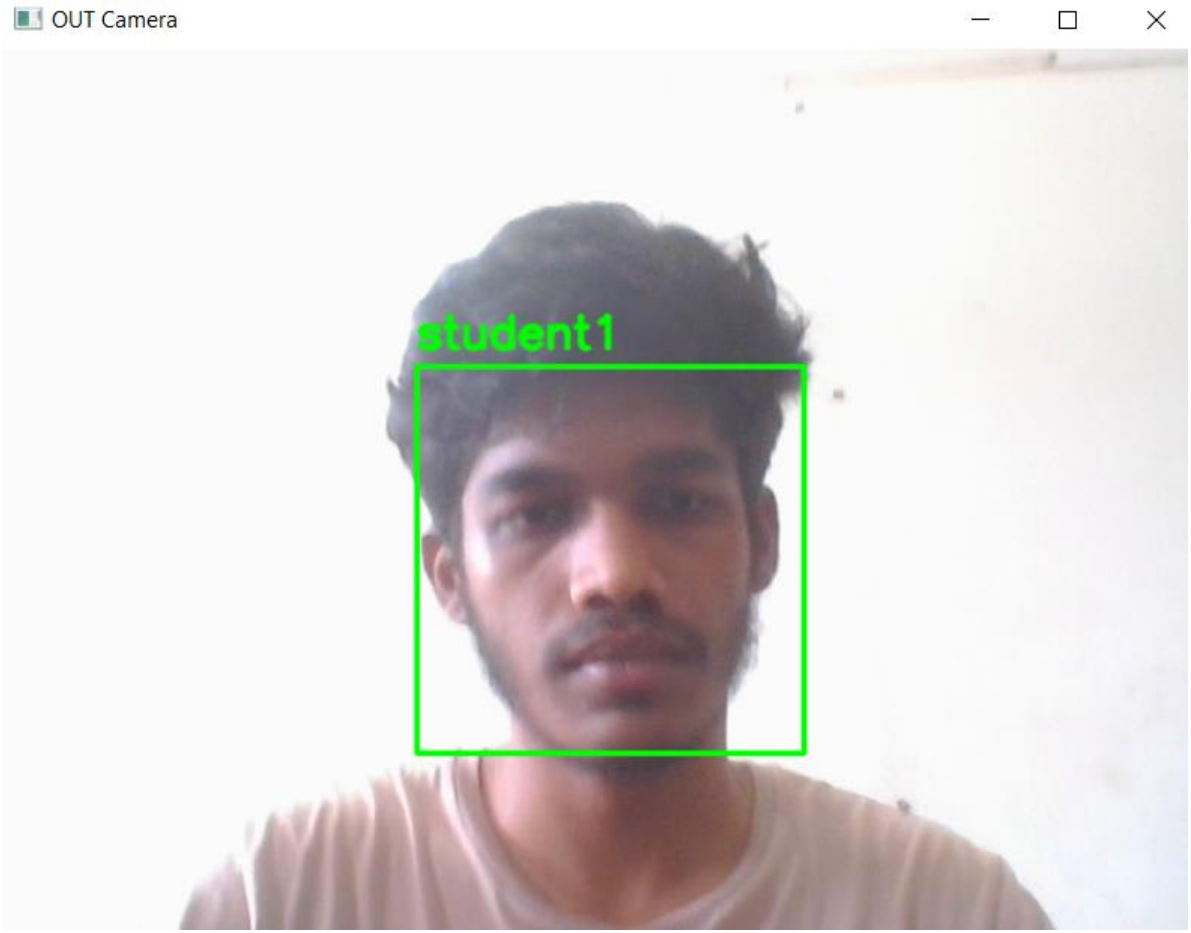
Model Training and Accuracy:

Trained using OpenCV’s LBPHFaceRecognizer_create().Achieved an accuracy of approximately 92%.

Best performance observed under ideal lighting and direct facial orientation.

RESULTS

Output:



OPTIMIZATION

Performance Challenges:

Face Detection Limitations:

The face detection algorithm only recognized frontal faces due to the exclusive use of the haarcascade_frontalface_default.xml classifier. This single classifier failed to detect faces of different profile such as side profile and etc.

Face Recognition Limitations:

Recognition accuracy decreased significantly with increased distance from the camera and even completely nullified after a critical distance of 5 meters.

ENHANCING FACE DETECTION

To address the limitations of single-classifier face detection systems and improve detection performance in real-world scenarios, we employ multiple classifiers, each specialized in detecting faces from various orientations and positions. This multi-classifier approach enhances both robustness and accuracy.

Loading Multiple Classifiers:

Frontal Face Classifier: Detects faces directly facing the camera.(haarcascade_frontalface_default.xml)

Profile Face Classifier: Detects faces in profile view.

(haarcascade_profileface.xml)

Full Body Classifier: Detects the entire body, useful for cases where faces may be small or partially visible.(haarcascade_fullbody.xml)

Upper Body Classifier: Detects the upper part of the body, including the head and shoulders.(haarcascade_upperbody.xml)

Lower Body Classifier: Detects the lower part of the body, providing additional context for partially visible faces.

(haarcascade_lowerbody.xml)

Face Detection Process:

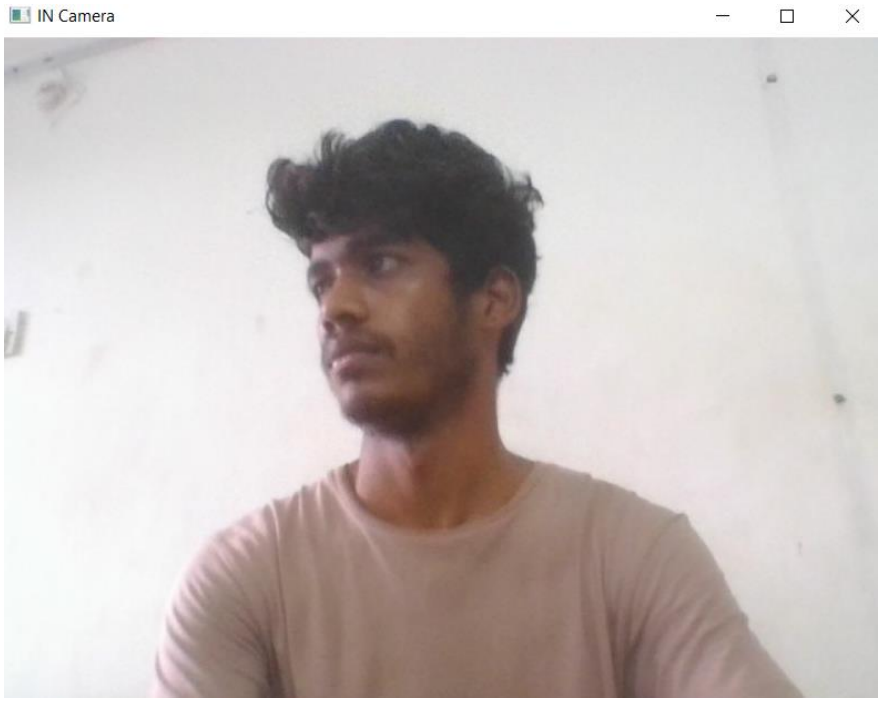
Grayscale Conversion: Convert the input image to grayscale to enhance the performance of the Haar cascade classifiers.

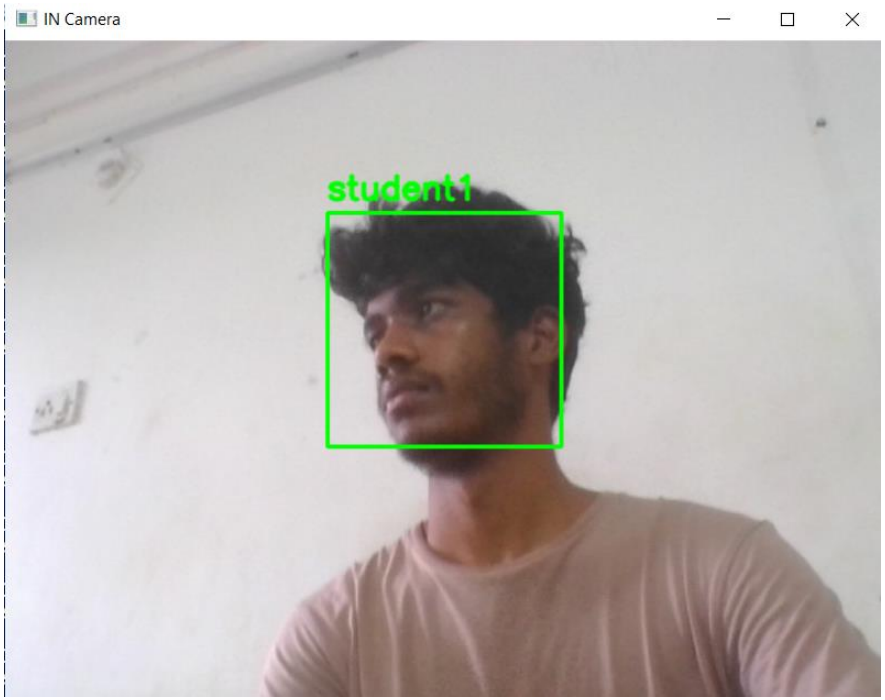
Multi-Classifier Application: Apply each classifier to the grayscale image independently to detect potential faces and body parts.

Combining Results: Aggregate the detected regions from all classifiers to form a comprehensive list of potential face locations.

Non-Maximum Suppression (NMS):

The purpose of Non-Maximum Suppression (NMS) is to handle overlapping detections from multiple classifiers and remove duplicates. NMS selects the bounding boxes with the highest confidence scores and suppresses overlapping boxes that are likely to represent the same face. The implementation of NMS involves calculating the intersection-over-union (IoU) of bounding boxes and retaining only those that meet a specified overlap threshold.





Without multiple classifiers -> side profile is not detected.

With multiple classifiers -> side profile is detected and recognized.

ENHANCING FACE RECOGNITION

Face recognition model can be enhanced by rich diversified dataset, and optimal preprocessing.

Data diversification:

we assumed three standard lighting conditions(Low light, Normal light, Bright Light) and corresponding adjustments are made using inbuilt functions of cv2 module.

Preprocessing:

Preprocessing standardizes and cleans the images for optimal recognition accuracy. For optimizing the preprocessing, we included face detection as the first step. The steps include face detection, resizing, noise reduction, and contrast enhancement. The Diversified images are loaded, Faces are detected and cropped, Cropped faces are resized and padded, Noise is reduced, Contrast is enhanced.

Without Face detection

With Face detection

Despite these improvements, there is still a need to explore alternative approaches in face recognition, such as deep learning models, to further enhance system performance. We can also deploy advanced models such as:

Deep Learning Models: Use deep learning models like Convolutional Neural Networks (CNNs) for face recognition. Models such as VGG-Face, FaceNet, or ResNet provide higher accuracy and better generalization compared to traditional methods.

Transfer Learning: Leverage transfer learning techniques to utilize pre-trained models on large datasets. This approach can significantly enhance recognition performance with a smaller dataset by fine-tuning the model to specific requirements.

CSV log files and exported attendance sheet:

	A	B	C
1	student_id	timestamp	direction
2	student1	38:55.1	IN
3	student1	38:55.1	IN
4	student1	38:55.2	IN
5	student1	38:55.2	IN
6	student1	38:55.3	IN
7	student1	38:55.3	IN
8	student1	38:55.4	IN
9	student1	38:55.4	IN
10	student1	38:55.4	IN

	A	B	C
1	student_id	timestamp	direction
2	student1	23:30.4	OUT
3	student1	23:30.5	OUT
4	student1	23:30.6	OUT
5	student1	23:30.6	OUT
6	student1	23:30.7	OUT
7	student1	23:30.7	OUT
8	student1	23:30.8	OUT
9	student1	23:30.9	OUT
10	student1	23:30.9	OUT

	A	B	C
1	student_id	Subject	Present
2	student1	subject1	TRUE
3	student2	subject1	TRUE
4	student3	subject1	TRUE
5	student2	subject2	TRUE
6	student4	subject2	TRUE
7	student6	subject2	TRUE
8	student5	subject2	TRUE
9	student1	subject3	TRUE
10	student4	subject3	TRUE

CONCLUSION

Our Face Recognition Based Attendance System benefited from preprocessing techniques and multiple classifiers for improved face detection, enhancing accuracy and reducing false negatives. However, challenges remain with recognition accuracy at greater distances and under varied lighting.

Future work will explore advanced approaches like CNNs, ResNets, and transfer learning with models such as VGGFace and FaceNet. These will be integrated and evaluated in real-world scenarios to develop a robust, accurate system for automated attendance tracking in educational settings, with ongoing research to improve reliability, security, and efficiency.

REFERENCES

1. **A Multi-Face Challenging Dataset for Robust Face Recognition (2018)**
   https://ieeexplore.ieee.org/document/8581283

2. **An approach for Face Detection and Face Recognition using OpenCV and Face Recognition Libraries in Python (2023)**
   https://ieeexplore.ieee.org/document/10113066

3. **Real-Time Face Recognition Attendance System Based on Video Processing (2023)**
   https://ieeexplore.ieee.org/document/10393263

4. **A real-time face recognition system based on the improved LBPH algorithm. IEEE 2nd International Conference on Signal and Image Processing (ICSIP), 72-76. (2017)**
   https://doi.org/10.1109/SIPROCESS.2017.8124508.