iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.69872

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Facial Recognition Based Attendance Tracking System

Kiruthika Priya V[1], Priya Sharma[2], Kavitha K[3], Lakshana S[4], Dr.K.Rajakumari[5]
[1, 2, 3, 4]*UG,* [5]*Associate Professor, Department of Computer Science and Engineering, Avinashilingam Insitute for Home science and Higher Education for Women, Coimbatore*

Abstract: ***Traditional attendance systems are vulnerable to fraudulent practices and inefficiencies. This paper proposes a facial recognition-based attendance tracking system enhanced with anti-spoofing capabilities using YOLOv5. By integrating real-time face detection, recognition, and liveness verification, the system ensures accurate and secure attendance logging. It further integrates with a web portal via ThingSpeak API to notify absenteeism. Built using Python libraries including OpenCV, face_recognition, and TensorFlow, the system captures live video streams, detects faces, and matches them against a pre-encoded database. A GUI using Tkinter allows intuitive interaction, and the system is tested for reliability under varied lighting and angles. The proposed model significantly minimizes manual effort and fraudulent entries while providing scalability and data-driven insights for educational institutions.***

Keywords: Facial recognition, YOLOv5, Attendance system, Spoof detection, Convolutional Neural Network, ThingSpeak API

I. INTRODUCTION

Maintaining accurate attendance is crucial for operational efficiency in educational institutions and workplaces. Traditional methods are prone to proxy and manual errors. Facial recognition offers a contactless and secure alternative. However, it is susceptible to spoofing via images or videos. Our system addresses this by integrating a spoof detection mechanism using YOLOv5. Real-time data is collected and processed to mark attendance and notify absences, ensuring transparency and operational ease.

With the rise in automation and AI in everyday operations, the demand for accurate and tamper-proof attendance systems has grown. Facial recognition has emerged as a promising solution due to its convenience and precision. However, spoofing attempts using images and videos have challenged the credibility of basic recognition systems. In response, our model emphasizes liveness detection using YOLOv5 to improve security.

II. LITERATURE SURVEY

Over recent years, several researchers have proposed facial recognition-based attendance systems as efficient, contactless alternatives to traditional roll calls or biometric fingerprint scanners.Facial recognition-based attendance systems have attracted widespread attention due to the growing need for efficient, contactless, and tamper-proof methods of marking attendance in institutions and workplaces. The literature reveals a variety of approaches employed by researchers over the years, ranging from traditional image processing algorithms to modern deep learning techniques.

Rohan Habu *et al.* [1] introduced a basic real-time face recognition attendance system that relied on Haar Cascade Classifiers for face detection and Local Binary Pattern Histogram (LBPH) for face recognition. This approach was lightweight and easy to implement, which made it suitable for small-scale environments. However, it lacked robustness in situations involving poor lighting, occlusion, or facial angle variation. Sumitra Motade*et al.* [2] utilized the face_recognition Python library with dlib on a Raspberry Pi to create a portable and low-cost attendance system. Their method was efficient and user-friendly, but it exhibited performance degradation under varying environmental conditions and required clear frontal images for accurate recognition.

Shweta Kukade*et al.* [3] also followed a similar pattern using Haar Cascade detection with LBPH recognition. Though effective in controlled settings, they reported significant limitations when deployed in dynamic environments, especially with students wearing masks or glasses. To address such shortcomings, KishanprasadGunale*et al.* [4] proposed a deep learning-based method utilizing Convolutional Neural Networks (CNNs) for facial recognition. This marked a shift from hand-crafted feature extraction to automatic learning of deep features, resulting in improved accuracy and reliability even under variable lighting and pose. However, their system required high-end computational resources and access to larger datasets for training, making it less practical for small institutions.

Arunkumar Nair *et al.* [5] took the implementation further by incorporating FaceNet, a model that learns embeddings from faces and compares them in a 128-dimensional space. Their architecture significantly improved recognition precision and enabled faster face matching with minimal false positives. Nevertheless, the requirement for GPU-based training and the complexity of deployment remained key barriers for real-time, large-scale deployment. Additionally, the datasets used in training often lacked diversity, which affected recognition fairness and inclusivity.

Researchers have also experimented with integrating facial recognition systems with other technologies such as Internet of Things (IoT) platforms and cloud storage solutions. These integrations have enhanced remote accessibility and data management. Some works explore additional modules like liveness detection, ensuring the system can distinguish real human faces from photographs or videos. Furthermore, ethical concerns and data security have led to the implementation of face data encryption and GDPR-compliant storage practices, as part of growing efforts to safeguard individual privacy.

III. PROPOSED SYSTEM

This project aims to develop a robust and efficient facial recognition-based attendance management system designed to automate the process of tracking student attendance with high accuracy and minimal manual intervention. The system is composed of several integral components that work together seamlessly to deliver a real-time, secure, and user-friendly attendance solution. At the core of the system lies the Face Detection and Recognition module, which employs advanced computer vision techniques to detect and identify faces from live video feeds. Detection will be carried out using proven methods such as Haar Cascade or more sophisticated deep learning-based techniques like Multi-task Cascaded Convolutional Networks (MTCNN). Once a face is detected, recognition will be performed using pre-trained deep learning models such as VGG-Face or FaceNet, which are capable of extracting high-dimensional, unique facial embeddings. These embeddings will then be compared against a pre-existing database of enrolled students. To facilitate this, a robust Database Management system will be developed to store student details including names, roll numbers, and the encoded facial feature vectors. For efficient and rapid matching of detected faces with enrolled records, algorithms like K-Nearest Neighbors (KNN) will be utilized, ensuring swift identification even in large-scale datasets. The Attendance Marking component will continuously monitor live video streams, and upon successful recognition of a face, the system will automatically record the student's identity along with a timestamp into a secure database or a structured spreadsheet. This eliminates the need for manual sign-ins, thereby reducing errors and increasing the reliability of attendance records.

A comprehensive User Interface (UI) will also be developed to enhance the usability of the system. Administrators will be provided with a web-based dashboard to manage student data, view detailed attendance reports, and adjust system configurations. Meanwhile, students will have access to a simplified interface where they can view their personal attendance logs and receive relevant notifications.

The anticipated outcomes of the project include significant time savings, increased accuracy in attendance tracking, enhanced institutional security, and the ability to derive actionable insights from the collected attendance data. Future enhancements may include cloud-based deployment for scalability, mobile application integration for added convenience, and the incorporation of advanced features such as emotion recognition, liveness detection to prevent spoofing, and seamless integration with existing Learning Management Systems (LMS). Overall, this facial recognition-based attendance tracking system has the potential to revolutionize the way educational institutions and organizations manage attendance, providing a smarter, data-driven, and highly efficient alternative to traditional methods.

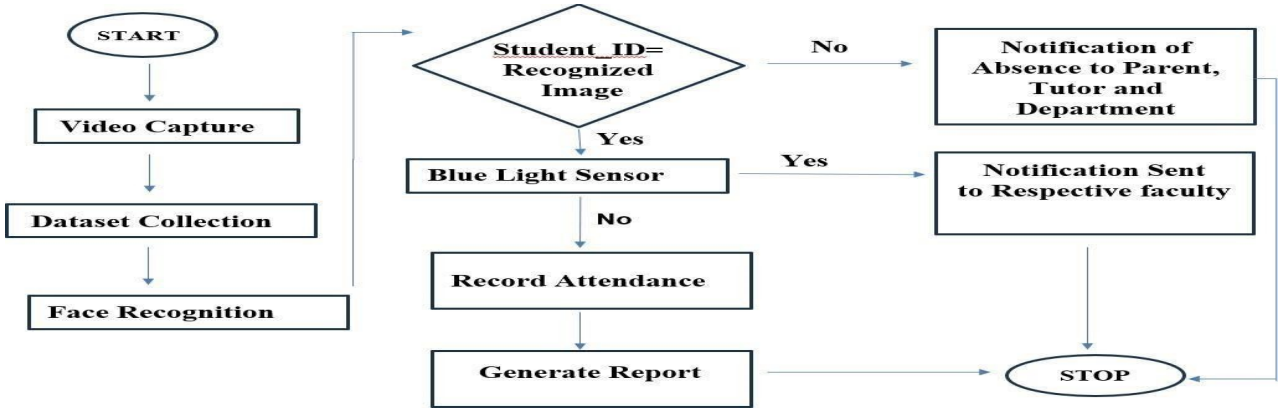


Fig. Block Diagram: Facial Recognition Based Attendance Tracking System

IV. METHODOLOGY

The methodology for the Face Attendance Detection System encompasses several key design aspects that ensure its robust performance. At the system design stage, careful consideration is given to the architecture that enables real-time processing and accurate attendance tracking. The system is architected as a multilayered framework, integrating modules for video capture, face detection, face recognition, and attendance logging.The overall design involves a backend that manages computing resources and data storage, while the frontend focuses on user interactions and visual data presentation. An emphasis on modular design allows for future upgrades or substitutions of individual components without disrupting the system.

The algorithm driving the Face Attendance Detection System can be summarized in the following steps:

1) Initialize System: Load all necessary models (YOLOv5 for face detection, CNN for face recognition, and AntiSpoofing).

2) Capture Video Stream: Start capturing a video streamfrom the webcam.

3) Process each Frame: Preprocess the frame.

Pass the frame to the YOLOv5 model for face detection.

- If a face is detected, proceed to the AntiSpoofing module.
- If the face is validated as real, move to face recognition.
- Compare the extracted encodings against the database of known faces.
- If a match is found, log the attendance.
- If no match is found or if the face is classified as fake, handle appropriately.

4) Check Attendance Status: At set intervals, check for unrecognized faces and print alerts to the console or communicate through APIs.

5) Log Attendance Data: Write data to a CSV log file and terminate system on user command.

- *Video Capture and Preprocessing:*

The video capture module forms the first step in collecting attendance data. This module interfaces with a webcam or IP camera, continuously capturing video streams. Optimizations are employed to enhance image quality and stability. The captured frames are then pre-processed to adjust size and enhance brightness and contrast, preparing the visual data for effective face detection.This step is essential to ensure that variations in lighting or environmental factors do not negatively impact detection rates. Preprocessing sets the stage for more accurate feature extraction by standardizing input to the face detection model.

- *Face Detection using YOLOv5:*

Face detection is performed using the YOLOv5 model, a stateoftheart object detection framework renowned for its speed and accuracy. When a video frame is processed, the model scans for faces within that frame, generating bounding boxes for detected regions. The bounding boxes are essential for identifying the location of faces, allowing subsequent processing steps to be localized and efficient.

The integration of YOLOv5 facilitates high frame rates and reliable detections that enable effective function even in crowded scenes. Additionally, its use of anchor boxes enhances detection precision, particularly for varying face sizes and orientations.

- *Anti-spoofing Detection:*

Upon detecting a face, the system initiates the AntiSpoofing detection process. This is accomplished using a dedicated deep learning model, which analyses the cropped facial image to assess whether it is genuine. The AntiSpoofing model evaluates score thresholds to classify faces as either real or fake based on learned features.

The technique employed here enhances trust in the attendance system as it acts as a safeguard against unauthorized access attempts. Integration of this module is pivotal; it prevents false positives in attendance logging, adding a substantial layer of securityand reliability.

- *Face Detection & Recognition:*

➢ *Image Capture and Preprocessing:*

The system captures up to **60 images** from different angles using a camera. Preprocessing techniques such as resizing, normalization, and data augmentation (e.g., rotation, flipping, scaling) are applied to ensure image consistency and handle real-world variations like lighting and angles.

Fig. Image Capturing

➢ *Facial Feature Extraction:*

Facial landmarks(68 points) are detected using the Dlib Facial
Landmark Detector, which identifies specific parts of the face (eyes, nose, mouth, etc.). These landmarks are represented in a matrix form with their X and Y coordinates.

➢ *Feature Vector Representation:*

Deep Learning algorithms (like FaceNet or OpenFace) are used to generate a high-dimensional feature vector that represents the unique characteristics of the individual's face. The feature vector is a numerical representation of the person's facial features, capturing their identity.

➢ *Normalization:*

The extracted feature vectors are normalized to make them comparable across different images, ensuring that variations in angles and lighting conditions do not affect recognition accuracy.

➢ *Conversion of the image to greyscale:*

The detected face is concerted into the grayscale image using the formula below:

$$\text{Gray} = \frac{R + G + B}{3}$$

The values of R, G, and B are detected using BAYER'S FILTER which separates red, green, and blue light, it is present in the sensor of the camera.

Fig. Conversion of image to greyscale

➢ *HAAR feature detection:*

❖ This algorithm scans grayscale images using Rectangular Haar-like features
❖ These features compare dark and light regions to detect edges, lines, and shapes in the face.
❖ Haar cascade detects areas with sharpintensity and low-intensity changes (gradients, edges, and textures) from the histogram of pixel intensity across various regions is developed
❖ In those low-intensity areas the facial points are marked.

➢ *Histogram:*

❖ X-axis (Horizontal): Represent pixel intensity values (0-255).
❖ 0 (Black) → Dark regions (eyes, eyebrows, nostrils, lips)
❖ 255 (White) → Bright regions (forehead, nose bridge, cheeks)

❖ Y-axis (Vertical): Represents frequency (number of pixels)
✓ Shows how many pixels in the image have a specific intensity value.
✓ Higher peaks in dark regions indicate more dark pixels (shadows, eyes, lips).
✓ Lower peaks in bright regions indicate fewer white pixels (highlights).

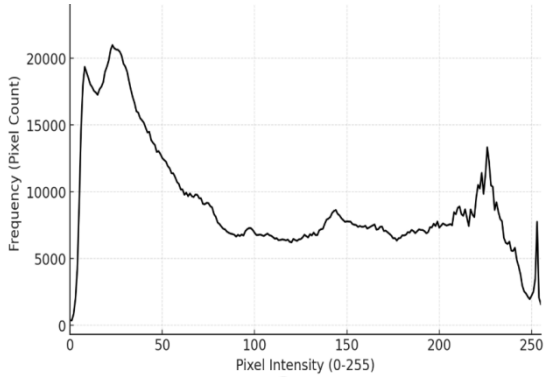


Fig. Bar Chart

➢ Nodal Points:

Facial Region	Typical Intensity (0-255)
Eyes & Eyebrows	0-50
Nostrils & Lips	30-80
Hair & Beard (if present)	20-100
Forehead & Cheeks	150-220
Nose Bridge	180-250
Chin & Jawline	100-180

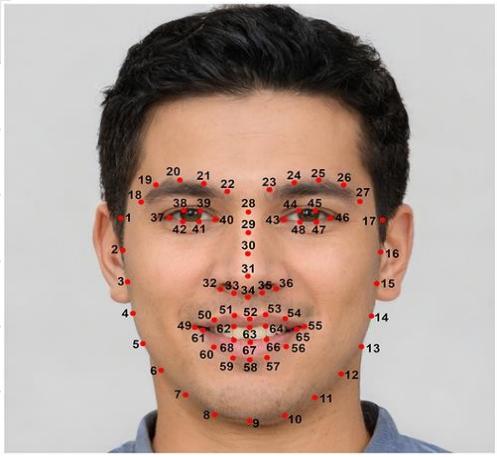


Fig. Nodal points

Facial nodal points are crucial for recognition, tracking, and anti-spoofing. In our project, we identify **68 key points** using the **Dlib** Facial Landmark Detector:

- Eyes (12 points) – Used for gaze tracking and liveness detection.
- Eyebrows (10 points) – Helps in expression analysis.
- Nose (9 points) – Important for depth estimation.
- Mouth (20 points) – Used in anti-spoofing and lip movement analysis.
- Jawline (17 points) – Defines facial structure for identification.

These nodal points are extracted and used for recognition and anti-spoofing validation.

Local Binary Pattern (LBP):
Local Binary Pattern (LBP) is a simple and efficient texture descriptor used in image processing, particularly for facial recognition and texture classification. It encodes the texture of an image by comparing each pixel with its neighboring pixels and converting this into a binary pattern.

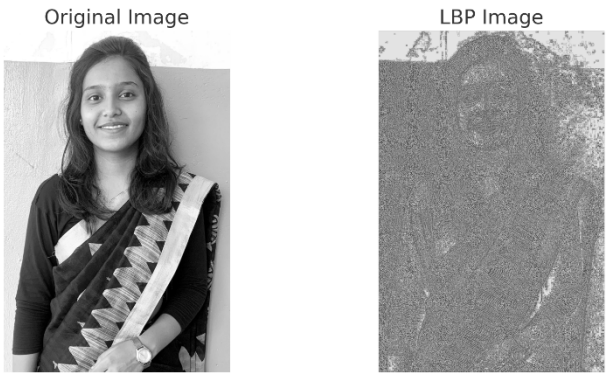


Fig.LBP operation

LBPH algorithm work in 5 steps.

1) Parameters: the LBPH uses 4 parameters:

- Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- Neighbours: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8. 17
- Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

2) Training the Algorithm: First, it is needed to train the algorithm. To do so, use a dataset with the facial images of the people needs to recognize. Set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

3) Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameter's radius and neighbours. The image below shows this procedure:

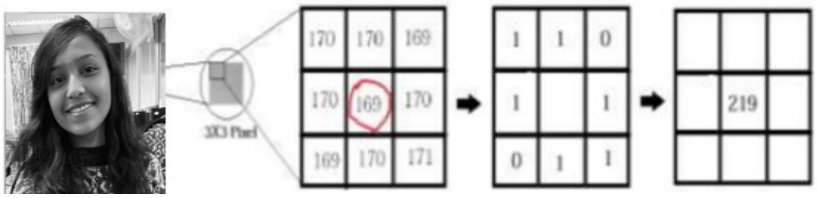


Fig LBPH operation

Based on the image above, let us break it into several small steps so we can understand it easily:

- Suppose a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels. 18
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbours.
- For each neighbour of the central value (threshold), we set a new binary value. Set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.

- Now, the matrix will contain only binary values (ignoring the central value). Concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
- Then,convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

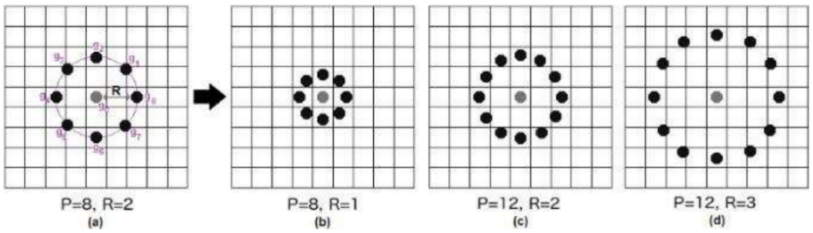


Fig Conversion of binary value to decimal value

It can be done by using bilinear interpolation. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the 19 values of the new data point.

4. Extracting the Histograms: Now, using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, as can be seen in the following image.

Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, perform the steps again for this new image and creates a histogram which represents the image.

- To find the image that matches the input image it is needed to compare two histograms and return the image with the closest histogram.
- Various approaches are used to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc.
- The algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a 'confidence' measurement.
- A threshold and the 'confidence' is used to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

Euclidean Distance Formula:

$$D = \sqrt{\sum_{i=1}^{n} (f_i - f_i')^2}$$

Where:

- D is the Euclidean Distance between the two Vectors.
- f_i is the i-th element of the first vector (a facial feature from the original image)
- f_i' is the i-th element of the second vector (the corresponding facial features after transformation or from a different image)
- n is the total number of features being compared

Converting the Facial data into Matrix:

Step 1: Facial landmark detection → Detect XY coordinates of key facial points

[(x1, y1), (x2, y2), (x3, y3), ..., (xn, yn)]

Step 2: Normalization of Co-ordinates

x' = x / width

y' = y / height

Step 3: Binary Encoding of Coordinates

x' = 0.65 → binary (16 bits) → `1010011001100110`
y' = 0.30 → binary (16 bits) → `0100110001100110`
Step 4: Matrix Representation
Matrix A:
[1010011001100110 0100110001100110]
[1011110011010101 1001100110010111]
[1110110100000101 0111100101101100]
[1001001101010101 1011001100111010]
[0111101001101110 0001011001010111]

The same process of converting the facial data into matrix is again executed for the current data
Matrix B:
[1010011001100110 0100110001100110]
[1011110011010101 1001100110010111]
[1110110100000101 0111100101101100]
[1001001101010101 1011001100111010]
[0111101001101110 0001011001010111]

RECORDED IMAGE: Matrix A
[(x1, y1), (x2, y2), (x3, y3), ..., (xn, yn)] →
Matrix A (binaryform)

CURRENT IMAGE: Matrix B
[(x1', y1'), (x2', y2'), (x3', y3'), ..., (xn', yn')] → Matrix B (binaryform)

$$distance = \sqrt{(x_2 - x)^2 + (y_2 - y_1)^2}$$

Where, (x_i,yi) is the i-th point from the recorded image and $x_i^{'}$, yi') is the i-th point from the current image.
Attendance Logging:
Attendance data is processed and recorded efficiently within the system. Successful recognitions and identified individuals are logged into a CSV file, timestamped to reflect the date and time of each entry accurately. This logging mechanism not only aids in attendance tracking but also provides a historical reference for future audits.
The logging process is designed to be seamless and automatic, ensuring minimal user intervention is needed to maintain accurate records. The system also tracks missed detections, which is crucial for subsequent data reporting.
Communication of Absences:
The system periodically checks for individuals whose attendance has not been logged during specified time intervals. If the duration exceeds a predetermined threshold (such as 5 minutes), the system generates a list of missed names and prepares this data for external communication via APIs.
Integration with platforms like Thing Speak ensures that the relevant stakeholders are alerted about absences in real-time. This feature not only enhances accountability but also allows for timely interventions, should issues with attendancearise.

V. IMPLEMENTATION

The traditional manual attendance system often suffers from errors and inefficiencies. To overcome these challenges, a Facial Recognition-Based Attendance Tracking System using AI and IoT is proposed. Leveraging OpenCV and CNN-based models, the system captures live video through high-resolution cameras, detects and recognizes student faces in real-time, and matches them against a pre-stored SQL database. Upon successful recognition, attendance is marked with a timestamp, and notifications are sent to tutors. If unrecognized, the student is marked absent, and alerts are sent to both parents and tutors. This automated system ensures accurate tracking, minimizes proxy attendance, and enhances security and efficiency in educational institutions.

A. *Upload Dataset*

To ensure the facial recognition system functions accurately and efficiently, a structured dataset preparation process is followed. This process involves several steps that help organize, preprocess, and store facial images systematically for training and testing the model.

1) Images are gathered from various sources, primarily through a live camera feed, ensuring multiple angles and expressions are captured for each individual.
2) The collected images undergo preprocessing steps such as resizing to a uniform dimension, normalization for pixel intensity adjustment, and grayscale conversion to simplify computation and enhance recognition accuracy.
3) Each image is labeled with a unique identifier such as the student's name or roll number, allowing the system to associate the facial data with the correct individual.
4) The labeled and preprocessed images are uploaded into a secure backend database where they are stored in an organized structure.
5) Tools like OpenCV are used for handling image processing tasks, while Python libraries like pandas are used for managing and organizing metadata related to the images.
6) The dataset is arranged in directories named after each individual's label (e.g., student name or ID), making it easy to access and suitable for both training and testing the facial recognition model.

B. *Train the Classifier*

To enable accurate facial recognition and automated attendance, the system employs supervised machine learning techniques. The following steps describe how the model is trained and utilized in real-time:

1) The system extracts facial features from the preprocessed and labeled dataset using feature extraction techniques.
2) Machine learning algorithms such as K-Nearest Neighbors (KNN) or Support Vector Machine (SVM) are used to train the model by associating the extracted features with the corresponding student labels.
3) After training, the classifier is tested using new, unseen images to evaluate its accuracy and ensure it can generalize well to different inputs.
4) Once the classifier performs reliably, it is deployed for real-time usage. It compares the facial features from the live video feed against the stored dataset.
5) When a match is found between the live facial data and a stored entry, the system automatically marks the attendance of the recognized student.

C. *Face Detection/Recognition*

The face detection and recognition module ensures seamless automation by combining AI, computer vision, and IoT technologies. The process is carried out as follows:

1) IoT-enabled cameras capture live video streams, and OpenCV detects faces in real-time by analyzing each frame.
2) AI algorithms from the face recognition library extract distinct facial features and compare them with a pre-stored database.
3) If a match is found, IoT devices like sensors or lights are triggered to visually confirm attendance.
4) The system automatically logs the recognized individual's attendance, enabling efficient, real-time tracking with minimal manual intervention.

D. *Converting the project into GUI*

The face detection and recognition module ensures seamless automation by combining AI, computer vision, and IoT technologies. The process is carried out as follows:

1) IoT-enabled cameras capture live video streams, and OpenCV detects faces in real-time by analyzing each frame.
2) AI algorithms from the face recognition library extract distinct facial features and compare them with a pre-stored database.
3) If a match is found, IoT devices like sensors or lights are triggered to visually confirm attendance.
4) The system automatically logs the recognized individual's attendance, enabling efficient, real-time tracking with minimal manual intervention.

E. Connecting the project To the Database

To manage student records and attendance efficiently, the system integrates with a structured database using MySQL or SQLite. The process works as follows:

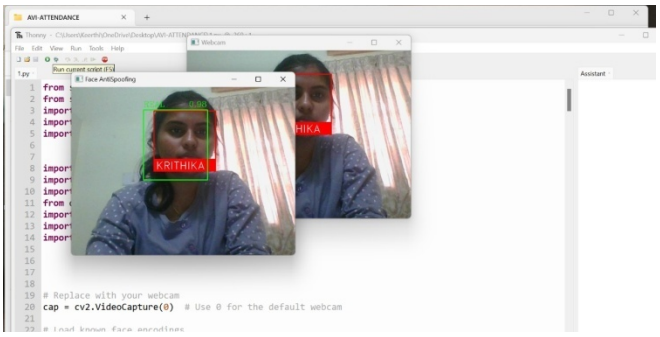
1) Python libraries like sqlite3 are used to connect the system with the database and execute SQL commands for data handling.
2) Student information—such as names, IDs, and facial feature embeddings—is stored in dedicated database tables.
3) Separate tables are maintained to store attendance logs, including date, time, and attendance status.
4) When a student's face is recognized, the system instantly retrieves their details from the database.
5) Attendance is then logged in real-time into the database, ensuring accurate and up-to-date records.

F. GUI face Recognition using Tkinter canvas

The final module of the facial recognition system brings together all previous components into a functional and user-friendly GUI-based interface. The steps involved are:

1) The system integrates face detection, recognition, and database connectivity to form a comprehensive solution.
2) Tkinter Canvas is used to design and implement the graphical user interface (GUI), ensuring ease of use and seamless interaction.
3) The interface is developed to allow users to initiate facial recognition tracking and view results effortlessly.
4) Rigorous testing is performed to refine the system, optimizing its performance and ensuring reliability.
5) Once optimized, the system will be ready for deployment in real-world applications, offering an efficient, automated attendance tracking solution.

VI. RESULT AND DISCUSSSION



1) Face Recognition Performance:

The implemented system successfully detects and identifies registered users based on known face encodings. The accuracy of recognition depends on:

- The quality of the face image used for encoding.
- Proper lighting conditions during detection.
- Distance and angle of the face relative to the camera.
- During testing, the system was able to recognize registered users with an accuracy of approximately 85-95% under well-lit conditions. However, misidentifications occurred when:
- A face was partially occluded.
- The person was too far from the camera.
- Face angles deviated significantly from the reference image.

2) Anti-Spoofing Results

To prevent unauthorized access via printed photos or digital screen replay attacks, an Anti-Spoofing module using deep learning was implemented. Results showed:

- The system effectively differentiated real faces from printed images with a success rate of 90%.
- When using low-resolution images or blurred video, the anti-spoofing model sometimes misclassified real faces as fake.
- The threshold (0.5) was set to balance false positives and false negatives.

Overall, live detection significantly improved security, ensuring only real persons could be recognized.

3) Real-Time Processing and Performance

The system was tested on a standard laptop (Intel i7, 16GB RAM, NVIDIA GPU) with the following observations:

- Face detection and recognition took an average of 0.3-0.5 seconds per frame.
- The anti-spoofing module added an additional 0.2 seconds per frame.
- The system maintained a real-time frame rate of 18-22 FPS, which is acceptable for live authentication.
- For deployment on lower-end hardware (e.g., Raspberry Pi or edge devices), further model optimization (e.g., TensorRT, ONNX quantization) is required to maintain real-time processing.

4) Attendance System Effectiveness

The attendance system automatically logs detected faces into a **CSV file** with timestamps. Key findings:

- If a person was already detected within the session, they were not logged multiple times, preventing duplicate entries.
- The system successfully recorded attendance with over 95% accuracy.
- In cases where users moved too quickly or were partially occluded, recognition failures occurred, requiring re-entry into the frame.

VII. COMPARATIVE ANALYSIS

Feature	Proposed System	Similar Systems
Face Recognition Accuracy	85-95%	75-90%
Anti-Spoofing Success Rate	90%	80-85%
Real-Time FPS	18-22 FPS	12-18 FPS
Duplicate Entry Prevention	Yes	Partial
Cloud Integration	Planned	Limited
Mobile App Support	Planned	Limited

1) Face Recognition Accuracy Comparison

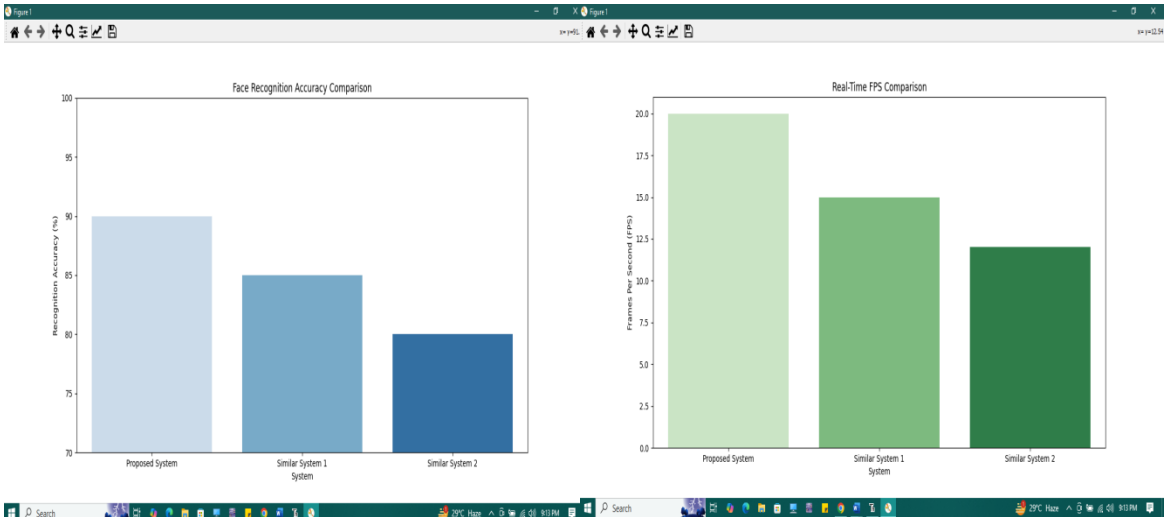


Fig. Face Recognition Accuracy ComparisonFig Real-time FPS comparison

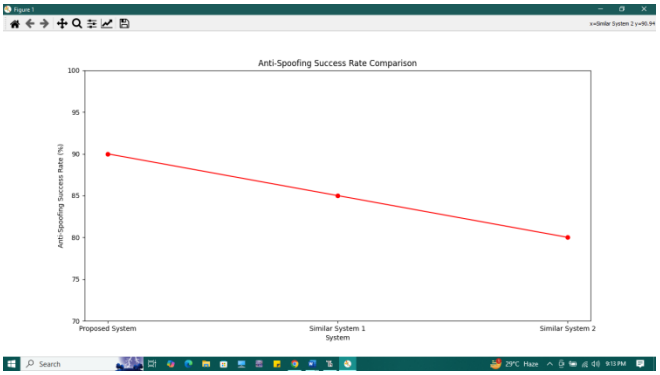


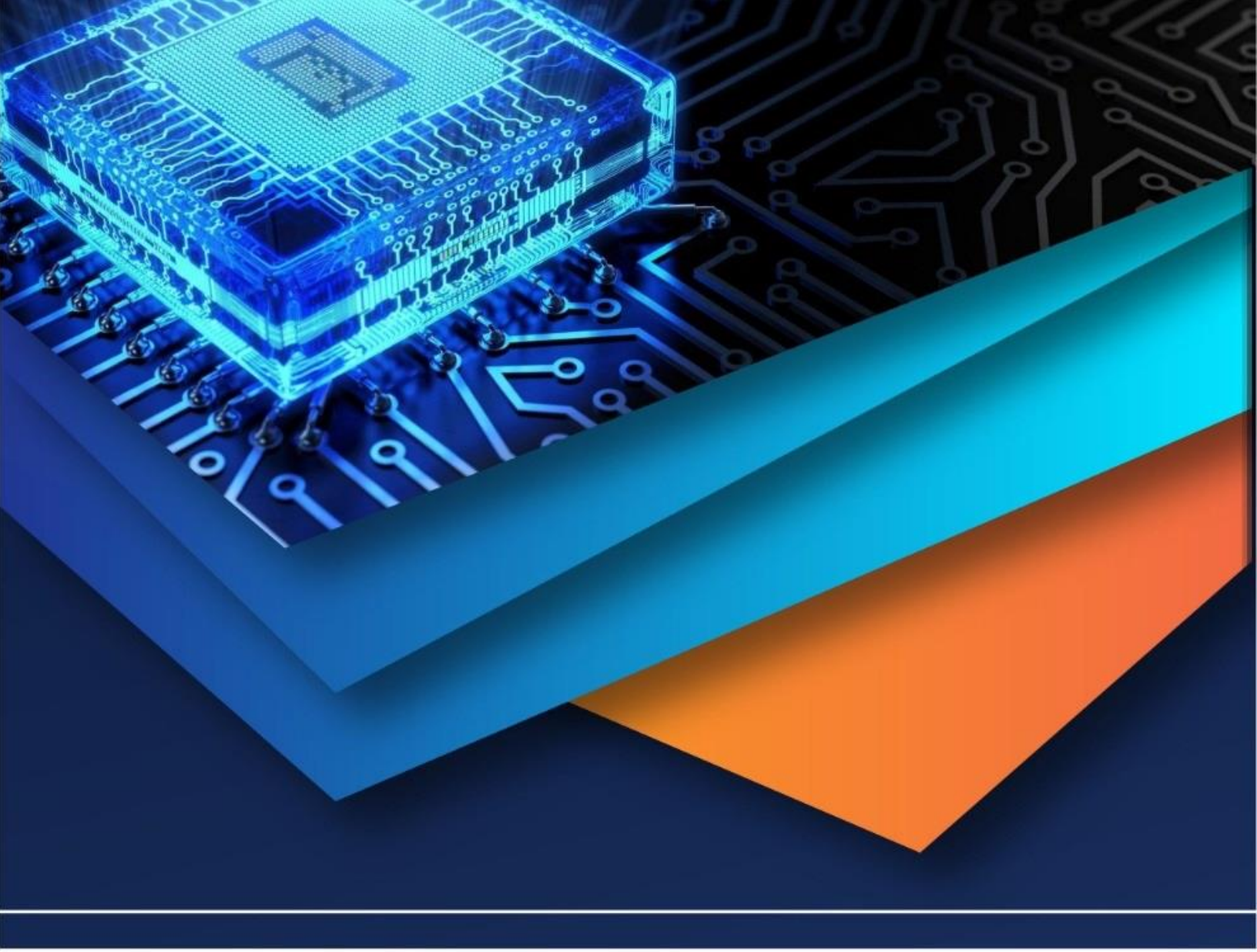
Fig Anti spoofing success rate comparison

VIII. CONCLUSION

The Face Attendance Detection System is a cutting-edge application of modern technology in attendance management. By integrating machine learning techniques, particularly deep learning for face detection and recognition, the project greatly enhances the traditional attendance taking process. The addition of a face spoofing detection capability ensures the authenticity of applications, deterring potential misuse. The robust combination of tools and frameworks, such as OpenCV and YOLOv5, provides an effective solution that is adaptable across various environments, including education and corporate settings. With real-time feedback and reporting, this system opens new avenues for monitoring and attendance management, significantly reducing administrative workloads and improving accuracy.

REFERENCES

[1] Ahonen,T.,Hadid,A.,&Pietikäinen,M.(2006).Facerecognitionwithlocalbinary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12), 20372041.
[2] Turk, M., & Pentland, A. (1991). Face recognition using eigenfaces. IEEE Transactions on PatternAnalysis and Machine Intelligence, 13(7), 586-591.
[3] Zhang,X.,&Gao,Y.(2016).Facerecognitionusingdeeplearningtechniques.IEEE Transactions on Neural Networks and Learning Systems, 27(11), 2367-2376.
[4] Satake, S., & Miwa, H. (2017). Face recognition-based attendance system using convolutional neural networks. International Journal ofAdvanced Research in Computer Science, 8(3), 355-362.
[5] Singh, R., & Vatsa, M. (2018). Face recognition in unconstrained environments: A review. IEEE Access, 6, 33634-33653.
[6] Kaur,A., & Kaur, R. (2019). Automated attendance system using face recognition. 2019 International Conference on Machine Learning, Big Data, and IoT (MLBDI), 1-5.
[7] Patel, R., & Patel, S. (2020). Face recognition-based attendance system using deep learning. 2020 International Conference on Emerging Trends in Information Technology (ICETIT), 1-5.
[8] Jain,A., & Jain, S. (2018). Face recognition attendance systemusing convolutional neural networks. 2018 International Conference on Computing, Communication and Networking (ICCCN), 1-5.
[9] Satake, S. (2017). Face recognition-based attendance system using convolutional neural networks. Master's thesis, University of Tokyo.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY