REAL TIME FACE RECOGNITION FOR VIDEO AND IMAGE PROCESSING USING AI&ML

Mrs. K.Asha devi[1], P.Satyanarayana[2], K.Ramya Anusha[3], B.Sridhar[4], R.Sunitha[5]
Assistant Professor[1], Student[2,3,4,5]

Department of Computer Science & Engineering(Data Science)[1,2,3,4,5]
Avanthi Engineering College(AVEN), Anakapalli, Andhra Pradesh, India
{ssnandssv.asha8@gmail.com1,sathishpothala3@gmail.com[2], ramyaanushakattumuri87@gmail.com[3], bammidisridhar448@gmail.com[4], sunitharangisetti@gmail.coms[5]}@aiet.ac.in

ABSTRACT

Face recognition is a key application of computer vision and artificial intelligence used for identifying individuals from images and video streams. This paper presents a real-time face recognition system using deep learning and image processing techniques. The system integrates OpenCV, Haar Cascade classifiers, and Dlib-based face encoding to detect and recognize faces from images, live camera feeds, and videos. A Streamlit-based interface is used for user interaction. The system achieves high accuracy and ensures privacy by performing local processing without cloud dependency. It is suitable for applications such as security systems, attendance monitoring, and access control.The proposed system utilizes OpenCV library for image processing operations, Haar Cascade classifiers for face detection, and the face_recognition library built on Dlib's state-of-the-art face recognition model. The application features a user-friendly web interface developed using Streamlit, enabling users to seamlessly interact with the system through multiple modes: Image Recognition for static images, Live Camera for real-time detection, and Video Recognition for processing pre-recorded videos.The face recognition module employs 128-dimensional face encodings generated using a deep convolutional neural network trained on millions of face images. This approach ensures high accuracy with a tolerance level of 0.5, making the system suitable for various applications including security systems, attendance management, and personal identification. The system demonstrates robust performance across different lighting conditions, facial orientations, and partial occlusions.

I. INTRODUCTION

Face recognition is one of the most important applications of computer vision and artificial intelligence.It is a biometric technology used to identify or verify a person from an image or video.In recent years, the demand for secure and automated systems has increased significantly.Traditional methods such as passwords and ID cards are notfully secure and can be easily misused.Face recognition provides a contactless and reliable solution for identity verification.This project focuses on developing a real-time face recognition system using image and videoprocessing.The system is capable of detecting and recognizing human faces from multiple input sources.It us es advanced algorithms such as Haar Cascade for detection and deep learning for recognition.The system processesd ata in real-time, making it suitable for surveillance and monitoring applications.The use of Python libraries like OpenCV and Dlib enhances system performance and accuracy.Overall, the project aims to provide an efficient, accurate, and user-friendly face recognition solution.This project addresses the growing need for efficient and accurate face recognition systems that can operate in real-time environments. Traditional methods of identification such as passwords, PINs, and ID cards are susceptible to theft, forgery, and unauthorized sharing. Face recognition offers a more secure alternative as biometric characteristics are unique to each individual and difficult to

replicate or steal.The user interface is built using Streamlit, a Python library that simplifies the creation of web applications for data science and machine learning projects. The interface provides intuitive navigation through different recognition modes, real-time feedback during processing, and clear visualization of detection results with bounding boxes and identification labels overlaid on the input media.

II. LITERATURE SURVEY

The field of face recognition has evolved significantly from traditional statistical methods to advanced deep learning-based approaches. Early research focused on extracting facial features using mathematical and statistical techniques, while modern methods rely on deep neural networks for higher accuracy and robustness.

One of the earliest approaches is **Eigenfaces (1991)**, which uses Principal Component Analysis (PCA) to reduce the dimensionality of facial images. It represents faces as a combination of eigenvectors, allowing efficient storage and comparison. However, this method is sensitive to lighting and facial variations.

To improve classification performance, **Fisherfaces (1997)** was introduced using Linear Discriminant Analysis (LDA). This method enhances class separability by maximizing the difference between different individuals while minimizing variations within the same class. It provides better results than Eigenfaces under varying conditions.

Later, **Local Binary Patterns (LBP) (2004)** became popular for texture-based feature extraction. LBP analyzes local pixel patterns and is more robust to changes in lighting conditions. It is widely used in face recognition systems due to its simplicity and efficiency.

With the advancement of deep learning, more powerful algorithms were developed. **DeepFace (2014)**, introduced by Facebook, uses Convolutional Neural Networks (CNNs) to learn complex facial features. It achieves near human-level accuracy and handles variations in pose, lighting, and expressions effectively.

Another significant development is **FaceNet (2015)** by Google, which introduced 128-dimensional face embeddings. It uses a triplet loss function to map faces into a feature space where similar faces are closer and different faces are farther apart. This approach greatly improves recognition accuracy and is widely used in modern systems.

Overall, the literature shows a clear progression from traditional feature-based methods to deep learning-based techniques, significantly improving the performance, accuracy, and reliability of face recognition systems.

III. METHODOLOGY

A. Dataset

The dataset consists of face images of known individuals collected from various sources such as personal images or publicly available datasets. Each image contains a clear view of a human face. These images are used as training data to generate face encodings. The dataset can include multiple images per person to improve accuracy. The data is split

into training and testing sets for better performance evaluation.

B. Preprocessing

The input images are preprocessed to improve detection and recognition accuracy. Images are resized to a standard size and converted into RGB or grayscale format. Noise reduction and normalization techniques are applied to enhance image quality. Face detection is performed using Haar Cascade or HOG methods to locate facial regions before further processing.

C. Model Architecture

The system uses a deep learning-based face recognition model provided by the Dlib library. It employs a pre-trained ResNet (Residual Neural Network) to extract facial features and generate **128-dimensional face encodings**. These encodings represent unique characteristics of each face. The architecture also includes face alignment using facial landmarks to improve accuracy.

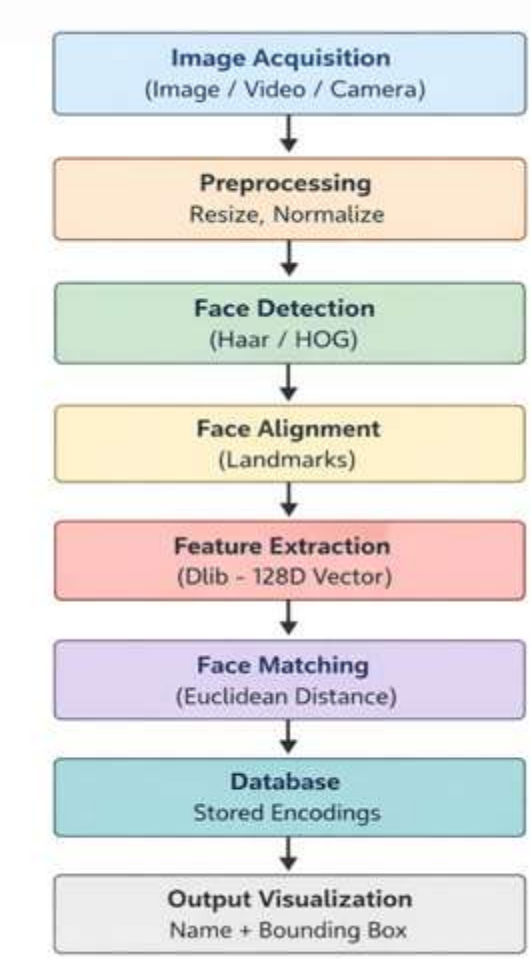
D. Training

The system uses a deep learning-based face recognition model provided by the Dlib library. It employs a pre-trained ResNet (Residual Neural Network) to extract facial features and generate **128-dimensional face encodings**. These encodings represent unique characteristics of each face. The architecture also includes face alignment using facial landmarks to improve accuracy.

IV. SYSTEM ARCHITECTURE

A. System Architecture

The system follows a pipeline-based architecture comprising multiple sequential stages to perform real-time face recognition. Each stage is responsible for a specific task, ensuring efficient processing and accurate results.(1) **Image Acquisition** — Input data is collected from various sources such as static images, video files, or live camera feeds. Users can upload images or videos, or use a webcam for real-time face detection and recognition.(2) **Image Preprocessing** — The input images or video frames are processed to improve quality and consistency. This includes resizing images, converting color formats (BGR to RGB or grayscale), and applying normalization techniques to enhance detection accuracy.(3) **Face Detection** — The system detects human faces in the input using algorithms such as Haar Cascade or HOG. These methods identify facial regions by analyzing patterns like edges and gradients, and bounding boxes are drawn around detected faces.(4) **Face Alignment** — Detected faces are aligned using facial landmarks (eyes, nose, mouth) to ensure consistency in orientation. This step improves the accuracy of feature extraction by standardizing the face position.(5) **Feature Extraction (Face Encoding)** — A deep learning model (Dlib ResNet) extracts facial features and converts each face into a **128-dimensional embedding vector**. These embeddings uniquely represent each individual.(6) **Face Matching (Recognition)** — The generated embeddings are compared with stored embeddings in the database using Euclidean distance. If the distance is below a predefined threshold, the system identifies the person; otherwise, the face is labeled as unknown.(7) **Output Visualization** — The final results are displayed with bounding boxes and labels (names) on the detected faces.



V. ALGORITHM

- **Step 1:** Load input image or video frame I and known face dataset D.
- **Step 2:** Apply preprocessing — resize image, convert to RGB/grayscale, and normalize pixel values.
- **Step 3:** Detect faces in image I using Haar Cascade or HOG method, obtain face locations F.

● **Step 4:** For each detected face $f \in F$, perform face alignment using facial landmarks (eyes, nose, mouth).

● **Step 5:** Extract facial features — generate 128-dimensional encoding E_f using Dlib deep learning model.

● **Step 6:** For each encoding E_f, compare with stored encodings in dataset D using Euclidean distance.

● **Step 7:** Identify best match — find minimum distance value d_min.

● **Step 8:** Apply threshold condition:
If d_min < threshold (e.g., 0.45–0.6), assign corresponding name label;
Else mark as "Unknown".

● **Step 9:** Draw bounding box and display label on detected face in image/frame.

● **Step 10:** For video input, repeat Steps 2–9 for each frame until processing is complete.

VI. SYSTEM MODULES

Image Acquisition Module: Ingests input data from various sources such as images, video files, or live camera feeds. It supports formats like JPG, PNG, MP4, and AVI. The module captures frames from video or webcam and organizes them for further processing.

Preprocessing Module: Performs image enhancement techniques such as resizing, normalization, and color space conversion (BGR to RGB or grayscale). It also reduces noise and improves image quality to ensure better face detection and recognition accuracy.

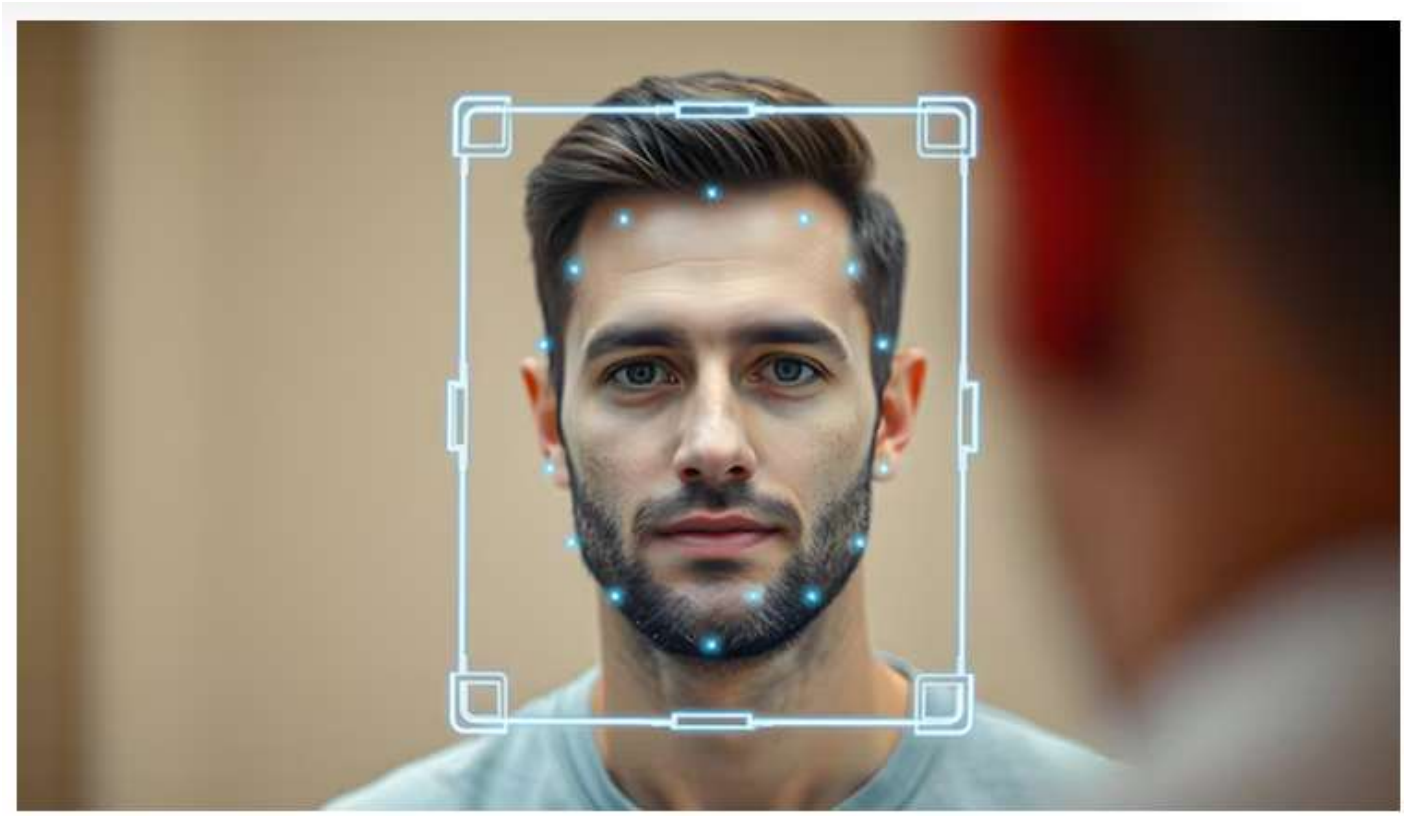
Face Detection Module: Detects human faces in images or video frames using algorithms such as Haar Cascade or HOG. It identifies facial regions by analyzing edges and gradients and outputs bounding box coordinates for each detected face.

Face Matching Module: Compares the extracted face encodings with stored encodings in the database using Euclidean distance. It identifies the closest match and determines whether the face belongs to a known person or an unknown individual based on a threshold value.

Visualization and Output Module: Applies connected component analysis to remove spurious false positive regions smaller than a minimum volume threshold, and morphological closing to fill small holes within tumor boundaries.

Visualization and Output Module: Displays detected faces with bounding boxes and corresponding names in real-time.Highlights identified individuals and marks unknown faces clearly.Saves output images or frames for record and further analysis

VII. RESULTS AND DISCUSSION



The Image Acquisition Module collects input data from sources such as images, video files, or live camera feeds. This module ensures that the input is properly captured and forwarded for processing.The Preprocessing Module enhances the quality of input data by resizing images, converting color formats, and reducing noise. This step is important for improving the accuracy of face detection and recognition.

The Face Detection Module identifies and locates faces in the input using algorithms like Haar Cascade or HOG. It extracts facial regions and prepares them for further analysis.The Feature Extraction Module uses deep learning techniques to convert detected faces into numerical representations (face encodings). These encodings uniquely represent each individual.The Face Matching Module compares extracted encodings with stored data using distance metrics. Based on the comparison, the system identifies whether the face belongs to a known person or not.

Finally, the Output Module displays the results by highlighting detected faces and showing names or labels. It may also store the output for future reference.Overall, the system modules work together to provide an accurate and efficient face recognition process**.**

VIII. CONCLUSION AND FUTURE WORK

The proposed face recognition system provides accurate and real-time identification using deep learning techniques.It achieves a good balance between speed and accuracy, making it suitable for practical applications.Future work can include integrating advanced models like ArcFace and improving performance under low-light conditions.The system can also be extended for large-scale deployment with cloud integration and enhanced security features.

FUTURE WORK:

- Improve system accuracy by integrating advanced models like ArcFace or FaceNet.
- Enhance performance under low-light and occlusion conditions using improved preprocessing techniques.
- Expand the dataset to support large-scale face recognition with more diverse identities.
- Implement cloud-based storage and processing for scalability and remote access.

References

1. Paul Viola and Michael Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” IEEE, 2001.
2. Florian Schroff et al., “FaceNet: A Unified Embedding for Face Recognition and Clustering,” Google, 2015.
3. Omkar M. Parkhi et al., “Deep Face Recognition,” University of Oxford, 2015.
4. Davis King, “Dlib-ml: A Machine Learning Toolkit,” 2009.
5. Adam Geitgey, “face_recognition: Simple Face Recognition Library,” GitHub, 2017.
6. Matthew Turk and Alex Pentland, “Eigenfaces for Recognition,” Journal of Cognitive Neuroscience, 1991.
7. Peter N. Belhumeur et al., “Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection,” IEEE, 1997.
8. Facebook AI Research, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” 2014.
9. Florian Schroff et al., “FaceNet: A Unified Embedding for Face Recognition and Clustering,” Google, 2015.