**Assignment No. 2**

**Name: Adhija Satish Bachhav**

**PRN: 22311656**

**Batch: TY-A A3**

**Roll No. : 381053**

**Subject: Deep LEarning**

**Problem Statement**

Facial recognition is one of the most widely researched applications of computer vision and deep learning. The task here is to develop a system that can detect a face and classify it into one of two categories (binary classification). The goal is to implement a real-time facial recognition model using OpenCV for face detection and Convolutional Neural Networks (CNNs) for binary classification.

**Objectives**

* Understand the fundamentals of face detection and recognition.
* Preprocess face data and extract features effectively.
* Implement a CNN-based deep learning model for binary classification.
* Evaluate the model’s performance on unseen data.
* Visualize training progress and prediction results.

**S/W Packages and H/W Apparatus Used**

* Operating System: Windows / Linux / MacOS
* Kernel: Python 3.x
* Tools: Jupyter Notebook, Google Colab, Anaconda
* Hardware: CPU with at least 4GB RAM (GPU optional for faster training)

**Libraries and Packages Used:**

* TensorFlow / Keras
* OpenCV
* Dlib
* face\_recognition
* NumPy, Pandas
* Matplotlib / Seaborn
* Scikit-Learn

**Theory**

**Definition:**A facial recognition system identifies or verifies a person from an image or video by analyzing facial features. In binary classification, the system decides between two categories: "face" or "no face" (or "Class A" and "Class B").

**Structure:**

1. Face Detection Module – Uses OpenCV Haar cascades or deep learning-based methods (SSD, YOLO) to locate faces.
2. Feature Extraction Module – CNN layers extract unique features like edges, contours, and patterns from faces.
3. Classification Module – A binary classifier (CNN or SVM) outputs the final prediction.

**Key Concepts:**

* Activation Functions: ReLU, Sigmoid, Softmax.
* Loss Function: Binary Cross-Entropy for binary classification.
* Backpropagation: Updates weights iteratively to minimize error.

**Methodology**

1. **Data Collection**
   * Dataset of face and non-face images collected.
2. **Preprocessing**
   * Convert images to grayscale.
   * Resize to uniform dimensions (e.g., 128×128).
   * Normalize pixel values to [0,1].
   * Perform data augmentation (rotation, flipping).
3. **Model Architecture (CNN)**
   * Input: 128×128×1 (grayscale image).
   * Convolutional layers: filters = [32, 64, 128], ReLU activation.
   * Pooling layers: MaxPooling for downsampling.
   * Fully connected Dense layer with 256 units.
   * Dropout for regularization.
   * Output layer: Sigmoid activation for binary classification.
4. **Training**
   * Optimizer: Adam
   * Loss: Binary Cross-Entropy
   * Metrics: Accuracy
   * Epochs: 15–20
5. **Evaluation**
   * Use accuracy, confusion matrix, precision, recall, F1-score.
6. **Prediction**
   * Input new image → preprocess → pass through CNN → classify as Class 0 / Class 1.

**Advantages**

* High accuracy with deep learning.
* Real-time processing with OpenCV.
* Versatile for multiple applications.

**Limitations**

* Requires large and diverse datasets.
* Sensitive to lighting, pose, and occlusions.
* Computationally expensive (GPU preferable).
* Privacy concerns in real-world use**.**

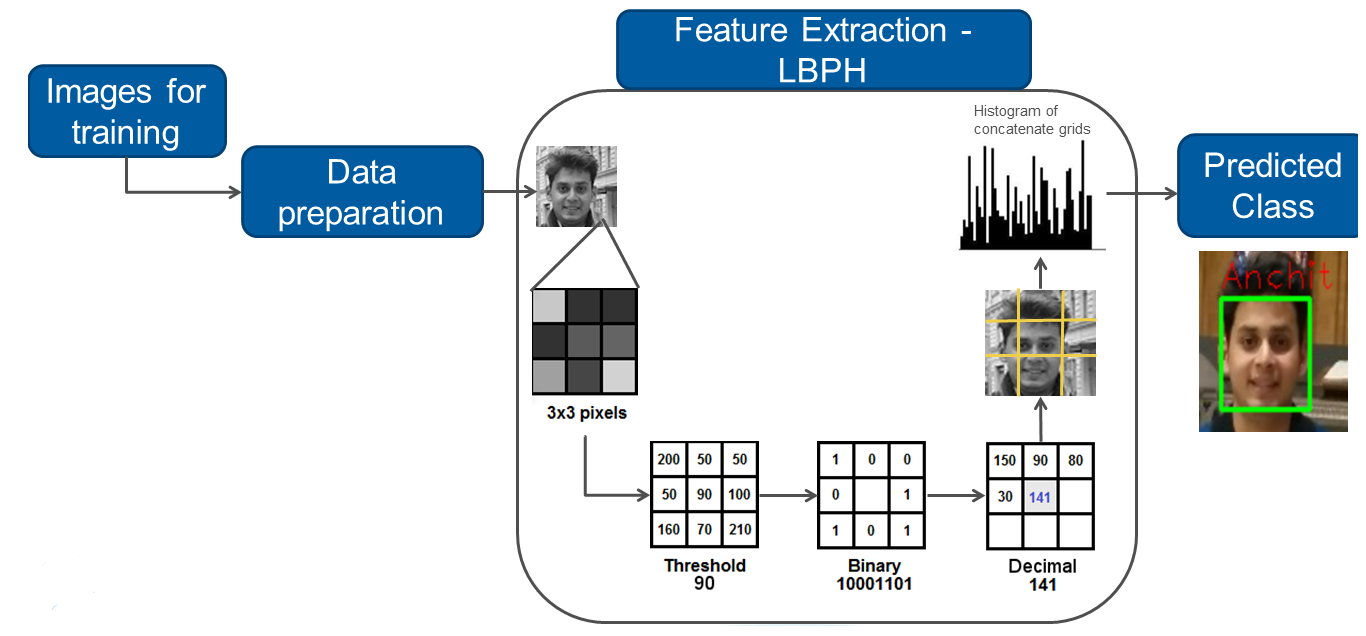
**Applications**

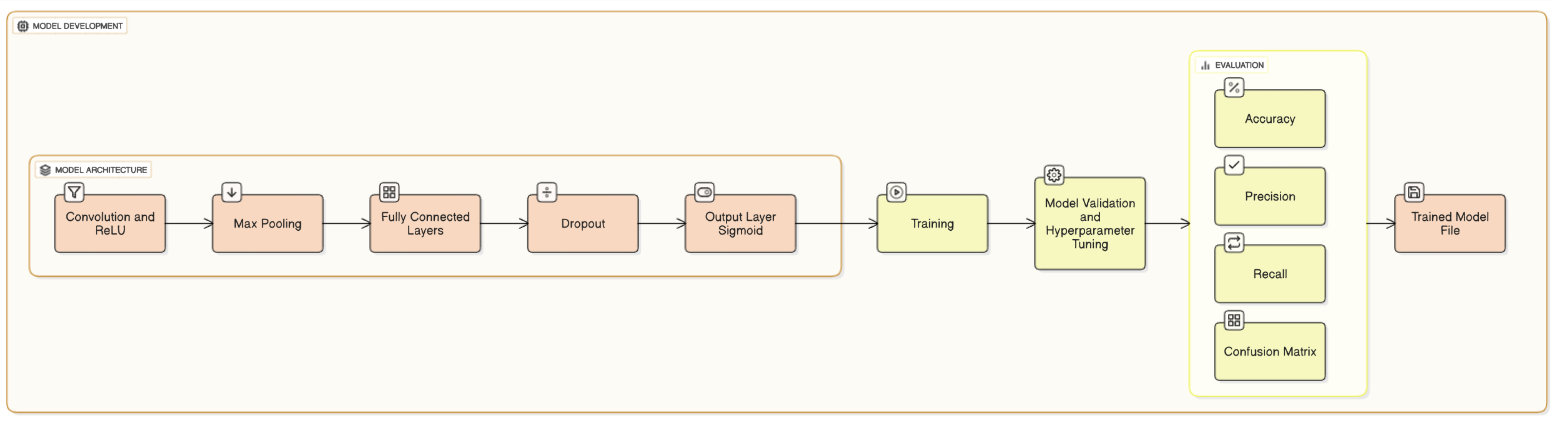
* Security & Surveillance – Identify individuals in CCTV footage.
* Authentication – Face unlock in smartphones/laptops.
* Law Enforcement – Criminal identification.
* Healthcare – Patient monitoring, disorder diagnosis.
* Retail & Marketing – Customer profiling and personalization.
* Attendance Systems – Automated entry/exit logging.
* Smart Cities – Public safety and traffic monitoring.

**Working / Algorithm**

1. Install necessary libraries (OpenCV, TensorFlow).
2. Capture images from webcam or load dataset.
3. Use Haar Cascade / DNN for face detection.
4. Preprocess images (resize, normalize).
5. Encode labels (0 or 1).
6. Split dataset into training and testing sets.
7. Define CNN model architecture.
8. Train the CNN using training data.
9. Evaluate using testing data.
10. Save trained model.
11. Predict on new images.

**Diagram**



****

**Conclusion**

The OpenCV + CNN based facial recognition system successfully detects and classifies faces with high accuracy. The integration of deep learning improves recognition robustness compared to traditional methods. However, challenges like lighting, pose variation, and privacy concerns must be addresses for real-world deployment. With further optimization, such systems can be deployed in authentication, security, healthcare, and smart cities.