# **Assignment No: 2**

## **Problem Statement**

Facial Recognition using OpenCV and Deep Learning for Binary Classification.

## **Objectives**

* Understand the fundamentals of face detection and recognition.
* Learn to preprocess face data and extract facial embeddings.
* Implement a deep learning-based model to classify faces.
* Evaluate the model’s accuracy and performance.
* Visualize the training process and performance metrics.

## **Software & Hardware Requirements**

* **Operating System**: Windows / Linux / MacOS
* **Kernel**: Python 3.x
* **Tools**: Jupyter Notebook, Anaconda, or Google Colab
* **Hardware**: CPU with minimum 4GB RAM; optional GPU for faster processing

**Libraries and Packages Used**:

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

## **Theory**

A **facial recognition system** is a biometric technology that identifies or verifies a person from a digital image or video frame. It operates by detecting facial features and matching them against stored data.

For **binary classification**, the task is to determine whether an input belongs to one of two categories — e.g., *face* or *no face*.

**Structure of a Facial Recognition System**:

1. **Face Detection Module** – Identifies face regions (Haar Cascades, SSD, or YOLO).
2. **Feature Extraction Module** – Extracts unique features using CNNs or embeddings.
3. **Classification Module** – A binary classifier (CNN/SVM) predicts whether the input is a face or not.

**Activation Functions**:

* **ReLU** – Non-linearity for hidden layers.
* **Sigmoid / Softmax** – Suitable for binary classification outputs.

**Backpropagation**:  
 Used to train CNNs by adjusting weights through error minimization.

## **Methodology**

1. **Data Collection**
   * Gather dataset containing *face* and *non-face* images.
2. **Preprocessing**
   * Use OpenCV to detect and crop faces.
   * Resize images to uniform size (e.g., 128×128).
   * Normalize pixel values to range 0–1.
3. **Model Architecture**
   * CNN with multiple convolutional + max-pooling layers.
   * Flattened to dense layers with dropout for regularization.
   * Final output layer with **sigmoid activation** for binary classification.
4. **Training**
   * Loss function: **Binary Cross-Entropy**.
   * Optimizer: **Adam**.
   * Metric: **Accuracy**.
5. **Evaluation**
   * Test model on unseen data.
   * Compute accuracy, precision, recall, and confusion matrix.
6. **Prediction**
   * Use trained model to classify new input images (face or no face).

## 

## **Advantages**

* **High Accuracy** with deep learning models.
* **Real-Time Processing** with OpenCV.
* **Automation** in authentication and access control.

## **Limitations**

* Requires **large, diverse datasets** for reliability.
* Sensitive to **illumination, pose, and occlusion**.
* Raises **privacy and ethical concerns**.
* **Computationally expensive** on large datasets.
* Risk of **overfitting** with limited data.
* Vulnerable to **adversarial attacks**.

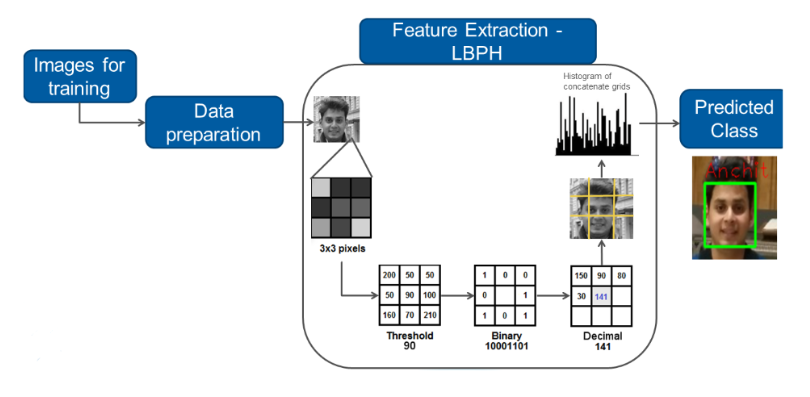
## **Applications**

* **Security & Surveillance** – Monitoring and access control.
* **Biometric Authentication** – Smartphones, laptops, and IoT devices.
* **Law Enforcement** – Identifying suspects, finding missing persons.
* **Healthcare** – Diagnosing genetic disorders, patient monitoring.
* **Retail & Marketing** – Customer recognition and personalization.
* **Time & Attendance Systems** – Automating employee check-in/out.
* **Smart Cities** – Traffic management and public safety.

## **Working / Algorithm**

1. **Install Libraries** – OpenCV, TensorFlow, Dlib, face\_recognition.
2. **Live Face Detection** – Use OpenCV Haar Cascade or DNNs on webcam feed.
3. **Load & Preprocess Dataset** – Resize images, normalize pixel values.
4. **Label Encoding** – Assign binary labels (*face* vs. *no face*).
5. **Split Dataset** – Training (80%) and Testing (20%).
6. **CNN Architecture** –
   * Conv layers (32, 64, 128 filters) + ReLU.
   * MaxPooling after each conv layer.
   * Dense layer (256 units) + Dropout.
   * Output layer (Sigmoid).
7. **Compile Model** – Adam optimizer + Binary Cross-Entropy loss.
8. **Train Model** – 15 epochs with validation data.
9. **Evaluate Model** – Compute test accuracy.
10. **Save Model** – Store trained model for reuse.
11. **Predictions** – Classify new input images.
12. **Display Results** – Show accuracy and sample predictions.

## **Diagram**



## **Conclusion**

The OpenCV-based **facial recognition system** successfully detects and classifies faces for binary classification tasks. It leverages **CNNs for feature learning** and **OpenCV for real-time detection**, making it suitable for authentication, surveillance, and security applications. While accuracy is high, the system faces challenges with illumination, pose variation, and privacy concerns. With large-scale datasets, GPU acceleration, and ethical considerations, such systems can be optimized for real-world deployment.