# Face Detection and Recognition –

## Learning Objectives

By the end of this lesson, students will:

* Understand the process of face detection and recognition.
* Implement face detection using **Haar Cascades** and **DNN**.
* Implement face recognition using **LBPH** and **FaceNet**.
* Analyze effects of lighting and pose variations on recognition accuracy.
* Learn enhancements to improve robustness in diverse conditions.

## 1. Face Detection vs Face Recognition

| Task | Description |
| --- | --- |
| **Face Detection** | Locating human faces in an image |
| **Face Recognition** | Identifying the person based on detected face |

## 2. Face Detection Techniques

### A. Haar Cascades (OpenCV)

* Based on **Viola-Jones algorithm**.
* Detects faces by scanning the image with Haar-like features.
* Fast and lightweight, ideal for real-time applications.

**Python Example:**

import cv2  
  
face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')  
img = cv2.imread('test.jpg')  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
faces = face\_cascade.detectMultiScale(gray, 1.1, 4)  
  
for (x, y, w, h) in faces:  
 cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)  
  
cv2.imshow('Detected Faces', img)  
cv2.waitKey()

### B. DNN (Deep Neural Network)

* More robust to scale, orientation, lighting.
* Uses Caffe/ResNet or TensorFlow-based pre-trained models.

**Python Example:**

net = cv2.dnn.readNetFromCaffe('deploy.prototxt', 'res10\_300x300\_ssd\_iter\_140000.caffemodel')  
image = cv2.imread('test.jpg')  
(h, w) = image.shape[:2]  
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300), (104, 177, 123))  
net.setInput(blob)  
detections = net.forward()  
  
for i in range(detections.shape[2]):  
 confidence = detections[0, 0, i, 2]  
 if confidence > 0.5:  
 box = detections[0, 0, i, 3:7] \* [w, h, w, h]  
 (startX, startY, endX, endY) = box.astype("int")  
 cv2.rectangle(image, (startX, startY), (endX, endY), (0, 255, 0), 2)  
  
cv2.imshow("Output", image)  
cv2.waitKey(0)

## 3. Face Recognition Techniques

### 🔹 A. LBPH (Local Binary Pattern Histogram)

* Works well with small datasets.
* Converts face region into a binary pattern, computes histogram, and compares.

**Python Example:**

import cv2  
import numpy as np  
  
recognizer = cv2.face.LBPHFaceRecognizer\_create()  
recognizer.train(faces, np.array(labels)) # 'faces' is list of images, 'labels' is list of IDs  
  
# Recognition  
test\_img = cv2.imread('test.jpg')  
gray = cv2.cvtColor(test\_img, cv2.COLOR\_BGR2GRAY)  
id\_, conf = recognizer.predict(gray)  
print(f"ID: {id\_}, Confidence: {conf}")

### B. FaceNet (Deep Learning)

* Uses CNNs and Triplet Loss to create **128-D embeddings** of faces.
* Compares embeddings with Euclidean distance to recognize faces.

**Key Mathematical Idea**:

If f(x) is the embedding of image x, FaceNet trains using Triplet Loss:

L = max(||f(anchor) - f(positive)||² - ||f(anchor) - f(negative)||² + margin, 0)

* anchor: reference image
* positive: same person
* negative: different person

The goal is to **minimize intra-class distance** and **maximize inter-class distance**.

**Python Tools**:

* Use face\_recognition library (built on FaceNet)

import face\_recognition  
  
image = face\_recognition.load\_image\_file("test.jpg")  
face\_encoding = face\_recognition.face\_encodings(image)[0]  
  
# Compare with known faces  
matches = face\_recognition.compare\_faces(known\_encodings, face\_encoding)

## 4. Impact of Lighting and Pose

| Factor | Effect |
| --- | --- |
| **Lighting** | Can cause shadows or overexposure, reducing detection accuracy |
| **Pose** | Side views or tilted faces may lead to poor matching or detection |

## 5. Enhancements to Improve Accuracy

### A. Face Alignment

* Align faces based on landmarks (eyes, nose, mouth) before recognition.

### B. Histogram Equalization

* Normalize lighting differences using cv2.equalizeHist().

### C. Data Augmentation

* Simulate different lighting and pose during training to improve robustness.

## 6. Summary of Haar & LBPH Use

* **Haar Cascades**: Fast and lightweight face detection for static or frontal faces.
* **LBPH Recognizer**: Easy-to-use face recognition method for small datasets and low computational power.

## 7. Tools and Libraries

| Tool | Use |
| --- | --- |
| **OpenCV** | Detection (Haar, DNN), Preprocessing |
| **face\_recognition** | FaceNet-based face embeddings |
| **dlib** | Landmark detection, face alignment |
| **NumPy** | Matrix operations |