**Name :** Chandrakant Dattatrey Thakare  
**Roll No.:**382014 **PRN:**22310303

**Class :** TY CSE AI **Batch :** B1

**Subject :** DL Lab Assignments

**Assignment 02**

**Facial Recognition using OpenCV and Deep Learning**

**Problem Statement**

In this assignment, we implement a facial recognition system using OpenCV and deep learning techniques for binary classification (face vs. non-face). The task is to detect and recognize faces from images or video streams, then classify them accordingly. Facial recognition is a widely used application in computer vision, with use cases in security, authentication, and human-computer interaction.

**Task Overview**

* Detect faces in raw images using OpenCV’s built-in methods.
* Prepare and preprocess data for deep learning.
* Design and train a CNN-based model to perform binary classification.
* Assess the model with standard evaluation metrics.
* Demonstrate the working model in real-time with a webcam or video feed.

**Objective**

* Explore the integration of OpenCV and deep learning.
* Build a practical system for real-world use (face vs. non-face).
* Gain experience in image preprocessing and augmentation techniques.
* Measure the system’s accuracy and robustness in different scenarios.

**Tools and Resources**

* **Software Environment:** Google Colab / Jupyter Notebook
* **Libraries:** OpenCV, TensorFlow/Keras, Numpy, Matplotlib, Seaborn
* **Dataset:** LFW (Labeled Faces in the Wild) or a custom dataset containing positive (face) and negative (non-face) samples

**Methodology**

**Step 1: Dataset Preparation**

* Images are collected and divided into two categories: face and non-face.
* OpenCV Haar Cascades or DNN modules are applied to detect and crop faces.
* Images are resized to a fixed dimension (e.g., 64×64 pixels) and normalized.

**Step 2: Model Design**

* A CNN architecture is built with convolutional, pooling, and dense layers.
* The final layer uses sigmoid activation for binary classification.

**Step 3: Model Compilation and Training**

* Optimizer: Adam
* Loss Function: Binary Crossentropy
* Metrics: Accuracy
* The model is trained with augmented data (random flips, rotations, zoom) to improve generalization.

**Step 4: Model Evaluation**

* Evaluate the trained network using test data.
* Generate confusion matrix and compute accuracy, precision, recall, and F1 score.

**Step 5: Real-Time Implementation**

* Connect the trained CNN model with OpenCV’s live video feed.
* The system detects faces, passes them through the model, and displays the recognition result in real time.

**Results**

* The trained model was able to achieve high classification accuracy on the test dataset.
* Data augmentation improved the robustness of the classifier to lighting and orientation changes.
* Real-time testing demonstrated successful face recognition in live video streams.

**Conclusion**

This assignment provided hands-on exposure to **face recognition systems**, combining **OpenCV for detection** and **deep learning for classification**. The experiment highlighted the importance of preprocessing, augmentation, and evaluation metrics. Overall, the model performed reliably and laid the groundwork for more advanced tasks such as multi-class face recognition or face verification systems.