**Name : Mansi Mahabdi**

**PRN: 22420190**

**Roll No : 382073**

**Batch: B3**

**Assignment No 2**

**Facial recognition using OpenCV and deep learning for binary classification.**

**Problem Statement:**

Implement facial recognition using OpenCV and deep learning to perform binary classification (Human vs Non-Human).

**Objective:**

- To understand the use of Convolutional Neural Networks (CNNs) for image classification.  
- To preprocess and prepare a human vs non-human dataset.  
- To implement facial recognition using OpenCV for detection.  
- To classify detected faces/images using a deep learning model.  
- To evaluate model performance using accuracy and confusion matrix.

**Technical Apparatus used:**

* **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**
* **NumPy**
* **OpenCV**

**Theory:**

Facial recognition involves two main steps: face detection and face classification. OpenCV provides Haar Cascades and DNN-based detectors to identify faces in an image or video stream. Deep learning, specifically Convolutional Neural Networks (CNNs), is then used to classify detected faces. In this assignment, the binary classification task distinguishes between Human and Non-Human categories.

**Methodology:**

**Step 1: Data Acquisition**- Use a human vs non-human dataset containing labeled images.  
 **Step 2: Data Preprocessing**- Resize images to a fixed size (e.g., 128x128).  
- Normalize pixel values to [0,1].  
- Split into training and testing sets.  
 **Step 3: Face Detection (Optional for live recognition)**- Use OpenCV Haar Cascade classifier to detect faces in images/video.  
 **Step 4: Model Building**- Define a CNN model with convolutional, pooling, and fully connected layers.  
- Output layer with sigmoid activation for binary classification.  
 **Step 5: Model Compilation**- Use Binary Crossentropy as the loss function.  
- Adam optimizer and Accuracy as the metric.  
 **Step 6: Model Training**- Train the model on the dataset for multiple epochs.  
- Validate on the test dataset.  
 **Step 7: Evaluation**- Evaluate using accuracy, confusion matrix, and classification report**.  
  
Step 8: Real-time Recognition**- Capture video using OpenCV.  
- Detect faces and classify using the trained model.

**Advantages:**

- Effective for binary image classification tasks.  
- Can be integrated with real-time video processing.  
- Demonstrates practical application of CNNs with OpenCV.

**Limitations:**

- Performance depends on dataset quality.  
- Sensitive to lighting and occlusions in real-time detection.  
- Requires GPU for faster training with large datasets.

**Applications:**

- Security and surveillance  
- Human-computer interaction  
- Biometric authentication  
- Filtering human vs non-human content in images

**Conclusion:**

In this assignment, facial recognition using OpenCV and deep learning was implemented for binary classification. The CNN model was trained on a human vs non-human dataset and integrated with OpenCV for real-time detection. This demonstrates the combined power of computer vision and deep learning in practical image classification applications.