Name: Aziz Sayyad
Roll No: 281069
Batch: A3

Assignment No: - 2

Problem Statement:
Implement a facial recognition system using OpenCV and deep learning to perform binary classification (e.g., recognizing whether a person is "Authorized" or "Unauthorized").

Objective:

- To understand the concepts of facial detection and recognition.
- To preprocess face image data for classification.
- To implement a binary classification model using deep learning.
- To integrate OpenCV for real-time face detection.
- To evaluate recognition accuracy using test data.

S/W Packages and H/W apparatus used:

- **Operating System:** Windows/Linux/MacOS
- **Kernel:** Python 3.x
- **Tools:** Jupyter Notebook, Anaconda, or Google Colab
- **Hardware:** CPU with minimum 4GB RAM; GPU recommended for faster execution
- **Libraries and packages used:** OpenCV, TensorFlow/Keras, NumPy, Pandas, Matplotlib, Scikit-learn

Theory:
Facial recognition involves two major steps: **face detection** and **face classification**.

- **Face Detection:** OpenCV's Haar Cascade or DNN-based detector is used to locate faces in an image or video stream.
- **Feature Extraction:** Detected faces are resized and normalized for input into a deep learning model.
- **Binary Classification:** A neural network (CNN or pre-trained models like VGG16, ResNet, MobileNet) is trained to classify faces into two categories — "Known/Authorized" and "Unknown/Unauthorized."

Structure:

- **Input Layer:** Face image (resized, e.g., 128×128 pixels).
- **Convolutional + Pooling Layers:** Extract spatial features.
- **Fully Connected Layers:** Learn non-linear decision boundaries.
- **Output Layer:** Single neuron with sigmoid activation for binary classification.
- **Activation Functions:** ReLU in hidden layers, Sigmoid in output.
- **Loss Function:** Binary Crossentropy.
- **Backpropagation:** Updates weights to minimize classification error.

Methodology:

1. **Data Acquisition:** Collect dataset of authorized and unauthorized face images.
2. **Data Preparation:**
   - Convert images to grayscale (optional).
   - Resize images to fixed size.
   - Normalize pixel values.
   - Split into training and testing sets.
3. **Model Architecture:**
   - CNN with convolutional and pooling layers.
   - Dense layers with ReLU activations.
   - Sigmoid output layer for binary classification.
4. **Model Compilation:** Adam optimizer, binary crossentropy loss.
5. **Model Training:** Train on training dataset, validate on test set.
6. **Face Detection with OpenCV:** Use Haar cascades or DNN to detect faces in real-time webcam/video feed.
7. **Prediction:** Pass detected faces to trained model for classification.
8. **Evaluation:** Accuracy, precision, recall, F1-score.

Advantages:

- Real-time recognition using OpenCV.
- High accuracy with deep learning models.
- Scalable to multi-class recognition.
- Can be integrated with security systems.
- Works well with pre-trained models (transfer learning).

Limitations:

- Requires large dataset for robust accuracy.
- Sensitive to lighting, pose, and occlusion.
- High computational cost for training CNNs.
- Privacy and ethical concerns.
- May produce false positives/negatives in real-world conditions.

Applications:

- Security and surveillance systems.
- Attendance monitoring in schools and offices.
- Access control (e.g., smart locks, secure areas).
- Personalized user authentication.
- Banking and financial authentication systems.

Working / Algorithm:

1. Import required libraries (OpenCV, NumPy, TensorFlow/Keras).
2. Load dataset (authorized and unauthorized faces).
3. Preprocess images (resize, normalize).
4. Build CNN model (Conv → Pool → Dense → Sigmoid).
5. Compile with Adam optimizer and binary crossentropy loss.

6. Train model and validate on test data.
7. Integrate with OpenCV face detector.
8. Capture real-time video, detect faces.
9. Classify detected faces (Authorized/Unauthorized).
10. Display result with bounding box and label.

Conclusion:
Facial recognition using OpenCV and deep learning for binary classification effectively detects and identifies individuals in real-time. By combining OpenCV for detection and CNNs for recognition, the system achieves high accuracy. Although computationally expensive and sensitive to conditions, with proper dataset preparation and model optimization, it provides a robust solution for security and authentication applications.