**Assignment No: - 2**

**Name:** Rushiraj Sunil Suwarnkar  
**PRN:** 22310315  
**Roll No:** 381018  
**Title:** Facial Recognition using OpenCV and Deep Learning for Binary Classification

**Problem Statement**

Implementing facial recognition using OpenCV and deep learning for binary classification (two classes: Class 0 and Class 1).

**Objective**

* To understand the working of facial recognition using computer vision and deep learning.
* To collect and preprocess facial image datasets.
* To design and train a convolutional neural network (CNN) for binary classification.
* To evaluate the trained model using accuracy and loss metrics.
* To deploy the trained model for real-time face recognition using webcam.

**S/W Packages and H/W apparatus used**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook, Anaconda, Google Colab
* **Hardware:** Webcam, CPU/GPU with minimum 4GB RAM
* **Libraries:** OpenCV, TensorFlow, Keras, scikit-learn, NumPy

**Theory**

Facial recognition is a biometric technology that identifies or verifies individuals using facial features. The process involves:

1. **Face Detection:** Using Haar Cascade Classifier in OpenCV to locate faces in real-time from video input.
2. **Data Preprocessing:** Captured images are converted to grayscale, resized to uniform dimensions (100×100), and normalized to the range [0,1].
3. **Deep Learning Model:** A CNN architecture is used for feature extraction and classification:
   * Convolutional layers detect facial features.
   * MaxPooling layers reduce dimensionality.
   * Dense layers perform classification.
   * Sigmoid activation outputs probabilities for binary classes.
4. **Training:** The CNN model is trained with labeled face data for Class 0 and Class 1.
5. **Recognition:** The trained model predicts in real-time whether the detected face belongs to Class 0 or Class 1.

**Methodology**

1. **Data Collection:** Captured face images using webcam with OpenCV (50 samples per class).
2. **Preprocessing:**
   * Converted images to grayscale.
   * Resized to 100×100.
   * Normalized pixel values to [0,1].
3. **Dataset Preparation:** Split into training and testing sets (80:20) using scikit-learn.
4. **Model Architecture (CNN):**
   * Conv2D (32 filters, 3×3, ReLU) → MaxPooling2D (2×2).
   * Conv2D (64 filters, 3×3, ReLU) → MaxPooling2D (2×2).
   * Flatten layer.
   * Dense layer with 64 neurons, ReLU activation.
   * Dense output layer with 1 neuron, Sigmoid activation.
5. **Compilation:** Optimizer → Adam, Loss → Binary Crossentropy, Metric → Accuracy.
6. **Training:** 10 epochs with validation on test data.
7. **Evaluation & Deployment:** Model saved as face\_classifier.h5, then loaded for real-time face recognition using webcam.

**Results**

* **Epoch 1:** Training Accuracy = 48.98%, Validation Accuracy = 50.00%
* **Epoch 2:** Training Accuracy = 51.95%, Validation Accuracy = 100%
* **Epoch 3–10:** Training Accuracy → 100%, Validation Accuracy → 100%
* **Final Test Accuracy:** 100%
* **Final Test Loss:** ~0.000005

**Sample Detection Output:**

Detected: Class 1 (score: 1.00) at [230,134,161,161]

**Advantages**

* Achieves very high accuracy on binary classification.
* Real-time face recognition possible using webcam.
* Uses CNN, which is highly effective for image-based classification.

**Limitations**

* Model trained only on two classes (binary); not scalable to large datasets without retraining.
* Requires sufficient and balanced data samples for each class.
* Sensitive to lighting conditions and camera quality.

**Applications**

* Biometric authentication systems.
* Attendance and surveillance monitoring.
* Security access control systems.
* Human-computer interaction.

**Working / Algorithm**

1. Import required libraries (OpenCV, TensorFlow, scikit-learn, NumPy).
2. Capture and store face images for Class 0 and Class 1.
3. Preprocess data (grayscale conversion, resizing, normalization).
4. Split dataset into training and testing sets.
5. Build CNN architecture for binary classification.
6. Compile and train the model on dataset.
7. Save trained model as face\_classifier.h5.
8. Load model and use webcam for real-time predictions.
9. Display predicted label (Class 0 or Class 1).

**Conclusion**

The CNN-based facial recognition system using OpenCV successfully distinguished between two classes with **100% accuracy** during training and validation. The model effectively captured facial features and generalized well on test data. This approach demonstrates the potential of deep learning for real-time biometric applications. With further data and extension to multiclass classification, the system can be expanded for large-scale face recognition tasks.