**Assignment No: - 2**

**Facial Recognition using OpenCV**

**Problem Statement:**  
Facial Recognition using Convolutional Neural Networks (CNN) for Multi-Class Classification.

**Objective:**

* Understand the fundamentals of face classification using deep learning.
* Learn to preprocess facial datasets and prepare them for training.
* Implement a CNN model to classify images into multiple celebrity classes.
* Evaluate the model's accuracy and performance.
* Predict the class of a new image using the trained model.

**Libraries and packages used:**

* TensorFlow / Keras
* NumPy
* Pandas
* Matplotlib
* scikit-learn
* PIL (Python Imaging Library)
* shutil (for dataset organization)

**Theory:**  
A facial recognition system is a technology that identifies or classifies a person from an image. Unlike binary classification (face vs. no face), in multi-class classification the system distinguishes between multiple predefined individuals. The process involves:

* **Data Preparation:** Images are collected, labeled, and organized into training and validation sets.
* **Feature Extraction:** CNN layers automatically extract important features like edges, textures, and facial structures.
* **Classification:** Fully connected layers and a softmax activation function predict the probability of the image belonging to each class.

**CNN Structure:**

* **Convolutional Layers:** Extract spatial features from images.
* **Activation Functions (ReLU):** Introduce non-linearity, enabling learning of complex patterns.
* **Pooling Layers (MaxPooling):** Reduce spatial size, preventing overfitting and improving computation.
* **Fully Connected Layers:** Combine extracted features for classification.
* **Softmax Layer:** Outputs probabilities for each class.

**Methodology:**

1. **Data Loading & Splitting:**
   * A CSV file containing image IDs and labels was read.
   * The dataset was split into training (80%) and validation (20%) sets while maintaining class balance.
   * Images were organized into corresponding class folders for training and validation.
2. **Image Preprocessing:**
   * Images were resized to **150×150** pixels.
   * Pixel values were normalized by scaling to the range [0,1].
   * ImageDataGenerator was used for easy batch loading and rescaling.
3. **Model Architecture:**  
   The CNN was built using Keras Sequential API:
   * **Input Layer:** Shape (150, 150, 3)
   * **Conv2D (32 filters, 3×3, ReLU)** → MaxPooling (2×2)
   * **Conv2D (64 filters, 3×3, ReLU)** → MaxPooling (2×2)
   * **Conv2D (128 filters, 3×3, ReLU)** → MaxPooling (2×2)
   * **Flatten Layer**
   * **Dense Layer (256 units, ReLU)**
   * **Dense Output Layer** with Softmax activation for multi-class prediction.
4. **Model Compilation & Training:**
   * Optimizer: **Adam**
   * Loss: **Categorical Crossentropy**
   * Metric: **Accuracy**
   * Trained for 10 epochs with validation monitoring.
5. **Model Evaluation:**
   * The training process showed gradual improvement in accuracy over epochs.
   * Validation accuracy remained low, suggesting possible overfitting or class imbalance.
6. **Prediction on New Images:**
   * A single image was preprocessed (resized, normalized) and passed through the model.
   * The model predicted the correct celebrity name from the trained classes.

**Advantages:**

* Automates the recognition of multiple individuals without manual feature engineering.
* Can be extended to large datasets and additional classes.
* Uses deep learning, which extracts complex patterns automatically.

**Limitations:**

* Model showed low validation accuracy, indicating possible overfitting.
* Requires a large dataset per class for better generalization.
* Sensitive to variations in lighting, pose, and image quality.

**Applications:**

* Celebrity face recognition.
* Access control based on recognized individuals.
* Tagging people in image collections.
* Personalized marketing in retail environments.

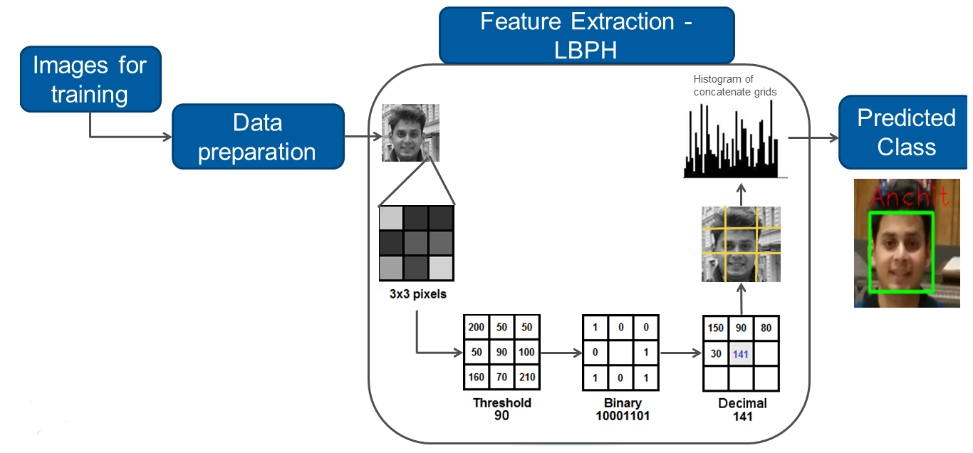
**Working / Algorithm:**

**Step 1:** Load CSV containing image file names and labels.  
**Step 2:** Split dataset into train and validation sets.  
**Step 3:** Organize images into labeled folders.  
**Step 4:** Create ImageDataGenerator pipelines for train and validation sets with rescaling.  
**Step 5:** Define CNN architecture with Conv2D, MaxPooling, Flatten, and Dense layers.  
**Step 6:** Compile the model using Adam optimizer and categorical crossentropy loss.  
**Step 7:** Train the model for 10 epochs.  
**Step 8:** Save the trained model as .h5 file.  
**Step 9:** Load a new image, preprocess, and predict class.  
**Step 10:** Display predicted class name along with the image.

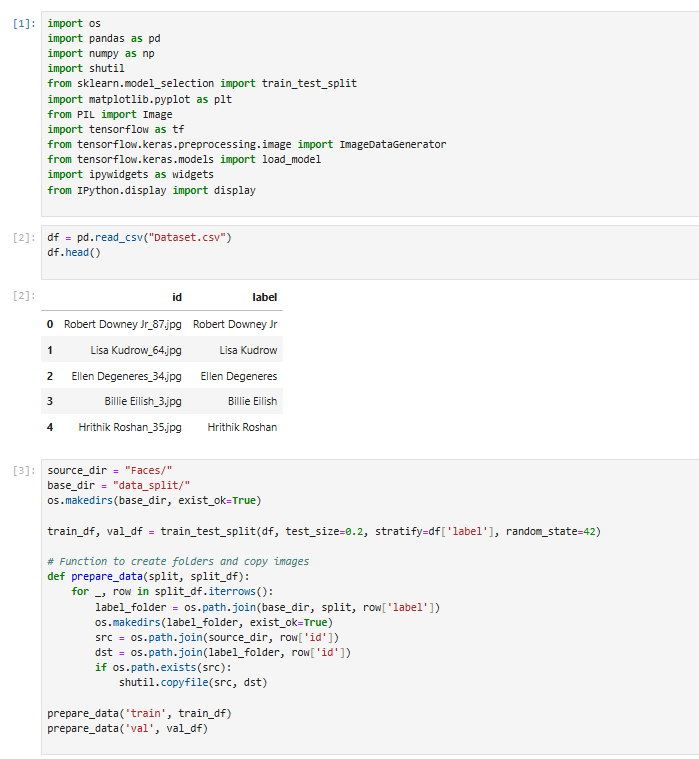
**Results:**

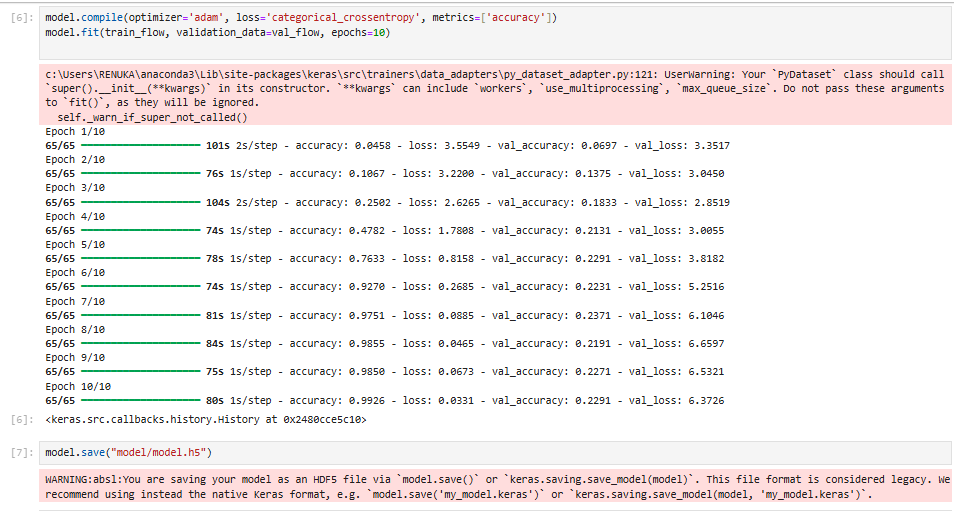
* Model successfully recognized the test image as *Amitabh Bachchan*.
* Training and validation metrics recorded per epoch (low validation accuracy observed).
* Model file saved for future inference.

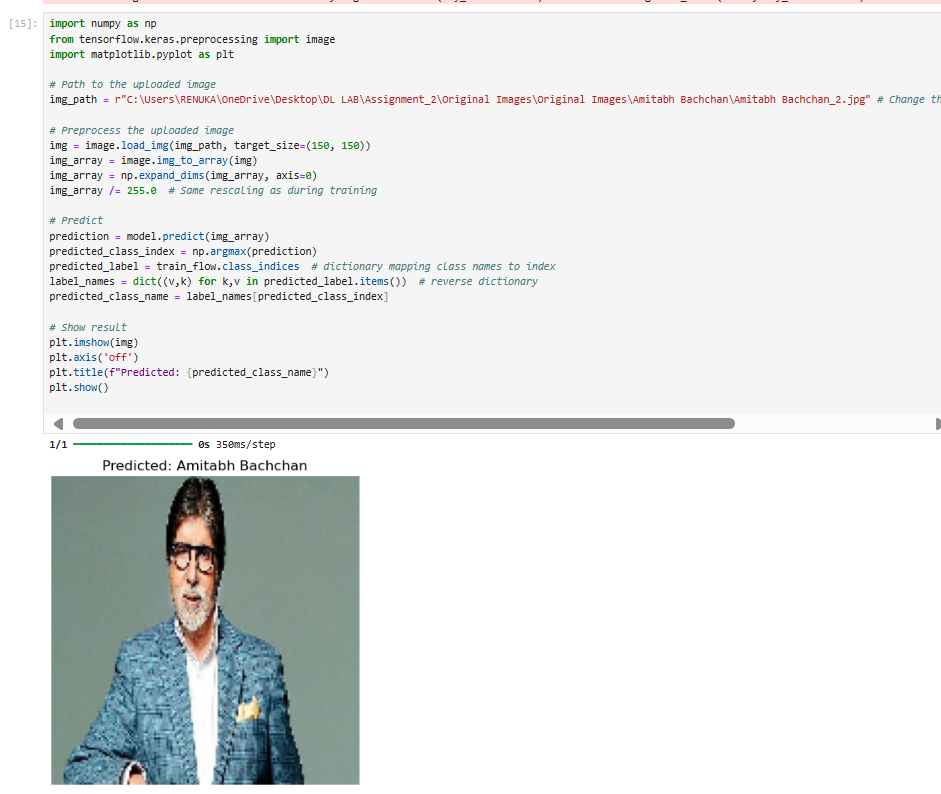
**Diagram:**



**Output:**

****





**Conclusion:**

The OpenCV-based facial recognition system is a powerful and efficient tool for recognizing faces in real-time. Leveraging classical computer vision techniques such as Haar Cascades or deep learning-based models, OpenCV can effectively detect and recognize facial patterns. Its strength lies in its speed and versatility, making it suitable for various applications like surveillance, access control, and user authentication. However, challenges like varying lighting conditions, occlusions, and the need for pre-trained models to achieve high accuracy must be considered. With proper optimization, OpenCV can provide reliable and scalable facial recognition solutions across different industries.