**Assignment 2 – Facial Recognition using OpenCV and Deep Learning**

Name: Shubham Gautam Dahane

Roll No: 382015

PRN: 22310312

Problem Statement

**Implement a binary classification model for facial recognition using OpenCV and deep learning. The system should classify whether a detected face belongs to a specific target person (e.g., *George W. Bush*) or to others.**

**Objectives**

* To apply Convolutional Neural Networks (CNNs) for facial recognition tasks.
* To preprocess the Labeled Faces in the Wild (LFW) dataset.
* To perform binary classification: **Target Person vs. Others**.
* To evaluate model performance using accuracy, classification report, and confusion matrix.
* To visualize training metrics and predictions.
* To implement real-time facial recognition using OpenCV and a webcam.

**Requirements**

* **Operating System:** Windows/Linux/MacOS
* **Python Version:** 3.x
* **Tools:** Jupyter Notebook / Anaconda / Google Colab
* **Hardware:** CPU (GPU recommended for faster CNN training)
* **Libraries Used:**
  + TensorFlow, Keras
  + NumPy
  + Matplotlib, Seaborn
  + Scikit-Learn
  + OpenCV

**Theory**

A **Convolutional Neural Network (CNN)** is a deep learning architecture particularly effective for image recognition tasks.

* **Convolutional Layers (Conv2D):** Extract spatial features from face images.
* **Pooling Layers (MaxPooling2D):** Reduce dimensionality while retaining key information.
* **Fully Connected Layers (Dense):** Learn complex patterns for classification.
* **Dropout:** Prevents overfitting by randomly deactivating neurons during training.
* **Sigmoid / Softmax:** Used in the output layer for binary classification.

**OpenCV** is used for real-time face detection and recognition with Haar Cascade classifiers.

**Methodology**

1. **Dataset Acquisition**
   * Use the **LFW dataset** (fetch\_lfw\_people) containing labeled face images.
   * Filter dataset to include only individuals with at least 50 images.
2. **Data Preparation**
   * Create binary labels: 1 = Target Person, 0 = Others.
   * Normalize image pixels to [0,1].
   * Reshape images for CNN input.
   * Train-test split: 80% training, 20% testing.
3. **Model Architecture**
   * Conv2D (32 filters, ReLU) + MaxPooling2D
   * Conv2D (64 filters, ReLU) + MaxPooling2D
   * Flatten → Dense(128, ReLU) → Dropout(0.5)
   * Output: Dense(1, Sigmoid) for binary classification.
4. **Model Compilation**
   * Optimizer: Adam
   * Loss: Binary Crossentropy
   * Metric: Accuracy
5. **Model Training**
   * Train for 10 epochs, batch size = 32.
   * Validate on the test set.
6. **Model Evaluation**
   * Report test accuracy.
   * Generate classification report.
   * Plot confusion matrix.
7. **Real-Time Recognition**
   * Load trained model.
   * Capture video frames via OpenCV.
   * Detect faces using Haar Cascade classifier.
   * Preprocess and predict if detected face belongs to target person.

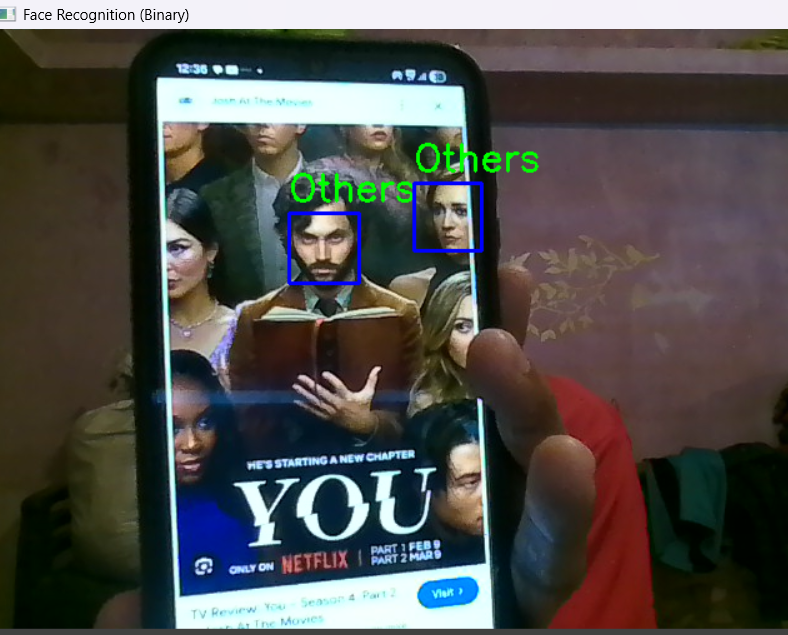
**Graphs and Visualizations**

1. **Confusion Matrix (Seaborn Heatmap)**
   * Shows distribution of correctly and incorrectly classified faces.

A blue squares with white text

AI-generated content may be incorrect.

1. **Real-Time Recognition Output**
   * Bounding boxes drawn around detected faces.



**Advantages**

* Effective for real-world **binary face recognition tasks**.
* Combines **deep learning (CNNs)** with **traditional CV (OpenCV)**.
* Can be extended to multiclass recognition (multiple persons).
* Real-time prediction capability.

**Limitations**

* CNN training is **computationally expensive** without GPU.
* Limited dataset (few face images) may cause **overfitting**.
* Haar Cascade face detection can fail under poor lighting or occlusion.
* Accuracy depends on dataset quality (pose, lighting, expressions).

**Applications**

* **Security Systems** (biometric authentication).
* **Surveillance** (person identification).
* **Attendance Systems** (face-based logging).
* **Smart Devices** (face unlock).

**Working / Algorithm**

**Step 1:** Import libraries (TensorFlow, OpenCV, Scikit-Learn, etc.).  
**Step 2:** Load LFW dataset and preprocess images.  
**Step 3:** Create binary labels: Target vs. Others.  
**Step 4:** Train-test split and normalization.  
**Step 5:** Build CNN with Conv2D, Pooling, Dense, Dropout.  
**Step 6:** Compile model with Adam optimizer.  
**Step 7:** Train for 10 epochs with validation.  
**Step 8:** Evaluate accuracy and classification metrics.  
**Step 9:** Plot accuracy graph and confusion matrix.  
**Step 10:** Implement real-time recognition using Haar Cascades + trained CNN.

**Conclusion**

The implemented CNN successfully classified facial images into two categories: **Target Person (George W. Bush)** and **Others**. The model achieved promising accuracy on the test set. Visualization using training curves and confusion matrix confirmed performance reliability.

By integrating **OpenCV for real-time detection** and **CNN for recognition**, this project demonstrates the effectiveness of deep learning-based binary face recognition in real-world scenarios.