Face Recognition System Documentation

1. Project Overview

This project focuses on building a face recognition system using deep learning techniques. The system uses a **Convolutional Neural Network (CNN)** and leverages **transfer learning** with the **ResNet50** model to classify face images from labeled datasets. The model is trained using **TensorFlow** and **Keras**, and its performance is evaluated using metrics such as **accuracy**, **precision**, and **recall**. The system is designed to handle face classification tasks, which can be applied in various use cases such as authentication, surveillance, and access control.

2. System Architecture

The system follows a modular architecture consisting of the following components:

- **Data Collection & Preprocessing**: Face images are collected from various sources, resized, and normalized before training.
- **Model Architecture**:
    - A **Custom CNN** is built to recognize faces from scratch.
    - **Transfer Learning** with **ResNet50** is employed, using pre-trained weights to improve feature extraction and model performance.
- **Data Augmentation**: Techniques such as rotation, width shift, height shift, and horizontal flip are applied to increase the diversity of training data and improve model generalization.
- **Model Training & Evaluation**: The models are trained on the preprocessed images and evaluated using metrics like accuracy, precision, and recall.

3. Dataset

The dataset consists of labeled face images, where each folder in the dataset represents a distinct individual (label). The images are collected from different sources and may come in various formats, such as JPG, PNG, and JPEG. The dataset is divided into training (80%) and testing (20%) sets to evaluate model performance.

4. Libraries Used

The following libraries were used in this project:

- **OpenCV**: Used for reading, processing, and resizing images.
- **TensorFlow & Keras**: Used for building, training, and evaluating the deep learning models.
- **NumPy**: Utilized for numerical operations and array manipulation.
- **Matplotlib**: Used for visualizing training results and images.

5. Model Details

- **Custom CNN**: A convolutional neural network built with multiple convolutional layers, pooling layers, dropout layers, and fully connected layers designed to recognize facial features.

- **ResNet50 Transfer Learning**: The ResNet50 model is used as the base for transfer learning, where pre-trained weights are utilized to extract high-level features from face images. A fully connected layer is added for classification.

6. Training Process

The training process follows these steps:

- **Image Preprocessing**: The images are resized to 128x128 pixels and normalized by scaling pixel values to the range [0, 1].
- **Data Augmentation**: To improve generalization and prevent overfitting, data augmentation techniques like rotation, shift, and horizontal flip are applied.
- **Model Training**: The models are trained using the **Adam optimizer** and **categorical cross-entropy loss**. Both models are evaluated using the validation set to monitor the performance during training.

7. Evaluation Metrics

The system's performance is evaluated using the following metrics:

- **Accuracy**: The percentage of correctly predicted images out of the total number of images.
- **Precision**: The ratio of true positive predictions to the sum of true positives and false positives.
- **Recall**: The ratio of true positive predictions to the sum of true positives and false negatives.

9. Future Enhancements

- **Real-Time Face Recognition**: Implement the system with a webcam or video stream to recognize faces in real-time.
- **Face Detection Integration**: Integrate advanced face detection algorithms (such as **Haar Cascades** or **MTCNN**) to locate faces before classification.
- **Model Optimization**: Use **TensorFlow Lite** or **ONNX** to optimize models for mobile devices or embedded systems, allowing for efficient edge computing.

10. Conclusion

The face recognition system successfully utilizes deep learning to identify and classify faces. By using a custom-built CNN model along with transfer learning, the system demonstrates high accuracy for face classification tasks. This system can be further enhanced for real-time applications and optimized for running on mobile and embedded devices