Face Recognition Project Documentation

Overview

This project focuses on face recognition using deep learning and computer vision techniques. The goal is to build a model that can recognize faces from images using a custom-built neural network.

Table of Contents

Setup Instructions

To set up the environment for this project, follow these steps:

1. Clone the Repository

Clone the repository to your local machine:

```
git clone https://github.com/your-username/face-recognition-project.git

cd face-recognition-project
```

2. Create a Virtual Environment

It is recommended to use a virtual environment to manage dependencies. Create and activate a virtual environment using venv:

```
python -m venv env

source env/bin/activate
```

3. Install Dependencies

Install the required dependencies using pip:

```
pip install -r requirements.txt
```

4. Download Pre-trained Models

Download the necessary pre-trained models and place them in the appropriate directory:

- **Shape Predictor**: shape_predictor_68_face_landmarks.dat
- **Face Recognition Model**: dlib_face_recognition_resnet_model_v1.dat

Place these files in the /content/ directory or update the paths in the code accordingly.

5. Data Preparation

Ensure that your data is organized in the following structure:

```
data/
  lfw-limited/
    Person1/
      image1.jpg
      image2.jpg
    Person2/
      image1.jpg
      image2.jpg
```

Dependencies

The project requires the following dependencies:

- **Python 3.x**
- **OpenCV**: pip install opencv-python
- **dlib**: pip install dlib
- **face_recognition**: pip install face_recognition
- **numpy**: pip install numpy
- **seaborn**: pip install seaborn
- **PIL (Pillow)**: pip install pillow
- **pandas**: pip install pandas
- **scikit-learn**: pip install scikit-learn
- **TensorFlow**: pip install tensorflow
- **matplotlib**: pip install matplotlib
- **joblib**: pip install joblib
- **skimage**: pip install scikit-image

Introduction

Face recognition technology has become increasingly significant in various applications, from security systems to personalized user experiences. This project focuses on building a face recognition system using deep learning and computer vision techniques. The system is designed to detect, process, and recognize faces from images, leveraging advanced models for accurate and efficient performance.

Objectives

The primary objectives of this project are:

1. **Face Detection**: Identify and locate faces within images.
2. **Feature Extraction**: Extract distinctive features from detected faces.
3. **Model Training**: Train a neural network to recognize faces based on extracted features.
4. **Prediction**: Use the trained model to identify faces in new images.

Methodology

1. Data Collection

The dataset used for this project consists of images from the Labeled Faces in the Wild (LFW) dataset, which includes a collection of face images labeled with the name of the person in the image. The dataset is structured into folders where each folder represents a unique individual and contains their respective images.

2. Data Preprocessing

Preprocessing Steps:

- **Face Detection**: Utilize the dlib library’s frontal face detector to identify faces in the images.
- **Face Cropping**: Crop the detected faces from the images.
- **Resizing and Normalization**: Resize faces to a fixed size (256x256) and normalize pixel values to the range [0, 1].

Purpose: Preprocessing ensures that the input to the model is consistent, which improves the model's ability to learn and generalize from the data.

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

Shape: (256, 256, 3)

3. Feature Extraction

Method:

- **Face Landmarks Detection**: Use dlib's shape predictor to detect facial landmarks.
- **Feature Extraction**: Compute face descriptors (feature vectors) using dlib's face recognition model.

Purpose: Extracting features helps in representing each face as a high-dimensional vector, which is crucial for training the recognition model.

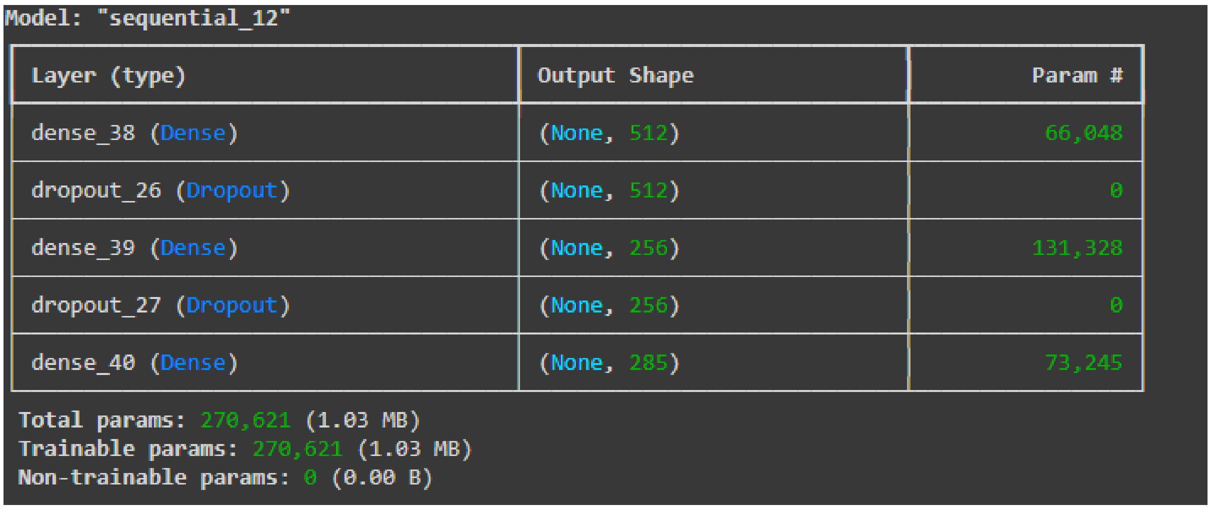
4. Model Training

Model Architecture:

- **Input Layer**: Accepts feature vectors of faces.
- **Hidden Layers**: Includes fully connected layers with ReLU activation functions, dropout for regularization, and batch normalization.
- **Output Layer**: Softmax activation for multi-class classification.

Training Process:

- **Data Split**: Divide the dataset into training, validation, and test sets.
- **Loss Function**: Use categorical cross-entropy to measure classification error.
- **Optimizer**: Adam optimizer for model training.



Model: "sequential_12"

Layer (type)	Output Shape	Param #
dense_38 (Dense)	(None, 512)	66,048
dropout_26 (Dropout)	(None, 512)	0
dense_39 (Dense)	(None, 256)	131,328
dropout_27 (Dropout)	(None, 256)	0
dense_40 (Dense)	(None, 285)	73,245

```
Total params: 270,621 (1.03 MB)
Trainable params: 270,621 (1.03 MB)
Non-trainable params: 0 (0.00 B)
```

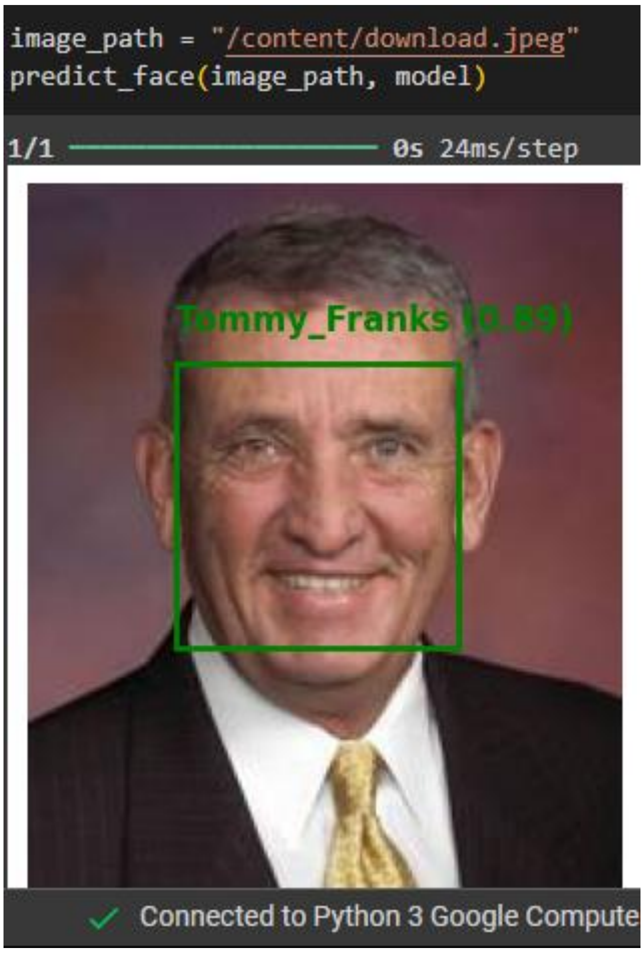
Purpose: Training the model involves learning patterns and relationships in the face features to correctly identify faces in unseen images.

5. Face Recognition

Steps:

- **Face Detection**: Detect faces in new images.
- **Feature Extraction**: Compute feature vectors for detected faces.
- **Prediction**: Use the trained model to classify the face based on its feature vector.

Purpose: The recognition phase applies the learned model to identify faces in new images, providing real-time or batch processing capabilities.



```
image_path = "/content/download.jpeg"
predict_face(image_path, model)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```

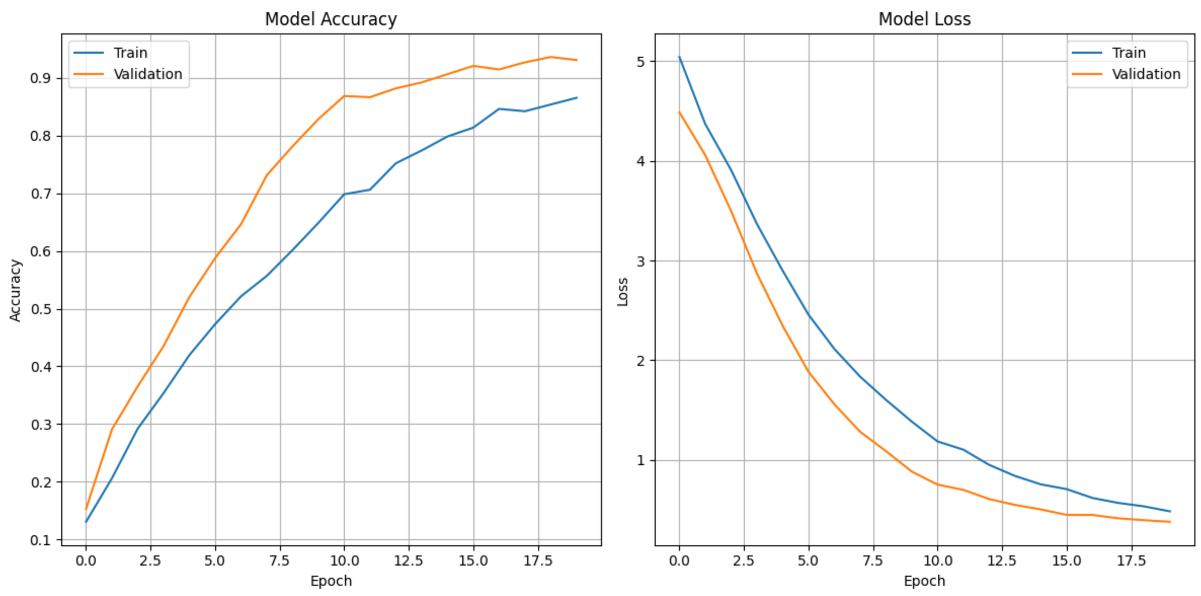
Libraries and Tools

- **OpenCV**: For image processing and face detection.
- **dlib**: For face detection, landmarks detection, and face recognition.
- **TensorFlow/Keras**: For building and training the neural network model.
- **Pandas and NumPy**: For data manipulation and numerical operations.
- **Matplotlib**: For visualizing results and metrics.

Performance and Optimization

Accuracy and Evaluation

- **Accuracy:** 96% on the test dataset.
- **Metrics:** Includes precision, recall, and F1 scores for each class.



Optimization Techniques

• **Model Optimization:** Utilized dropout, batch normalization, and regularization to improve model performance.

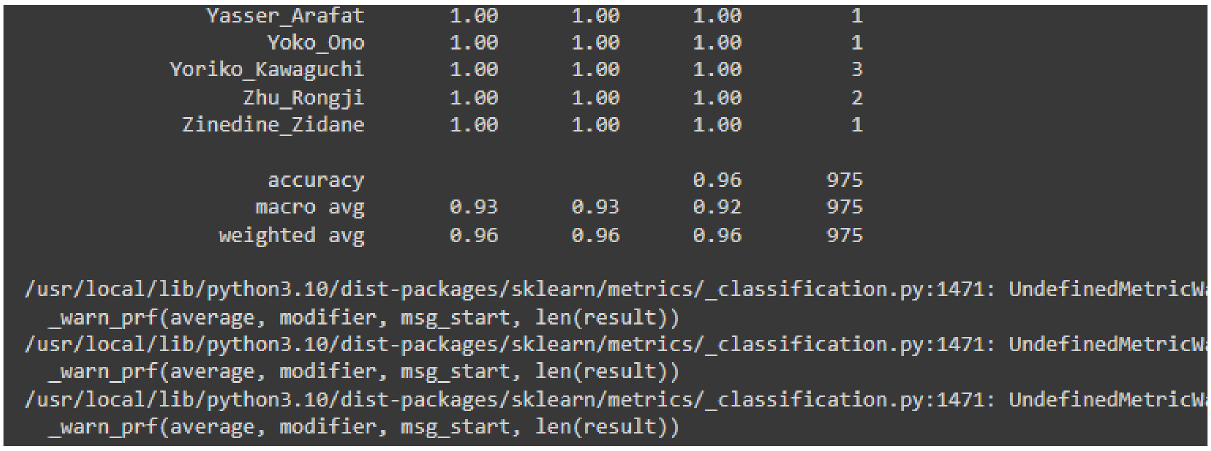
• **Performance Tuning:** Enhanced inference speed and scalability by optimizing model architecture and data processing.

Detailed Results Analysis

Classification Report

The classification report provides insights into the model's performance for each individual in the dataset. Here's a breakdown of the results:

- **Overall Accuracy**: 96%
  - Indicates that 96% of the test images were classified correctly.
- **Macro Average**:
  - **Precision**: 0.93
  - **Recall**: 0.93
  - **F1 Score**: 0.92
  - Reflects the average performance across all classes, treating each class equally.
- **Weighted Average**:
  - **Precision**: 0.96
  - **Recall**: 0.96
  - **F1 Score**: 0.96
  - Accounts for the number of instances per class, providing an overall measure of the model's performance while considering class imbalance.



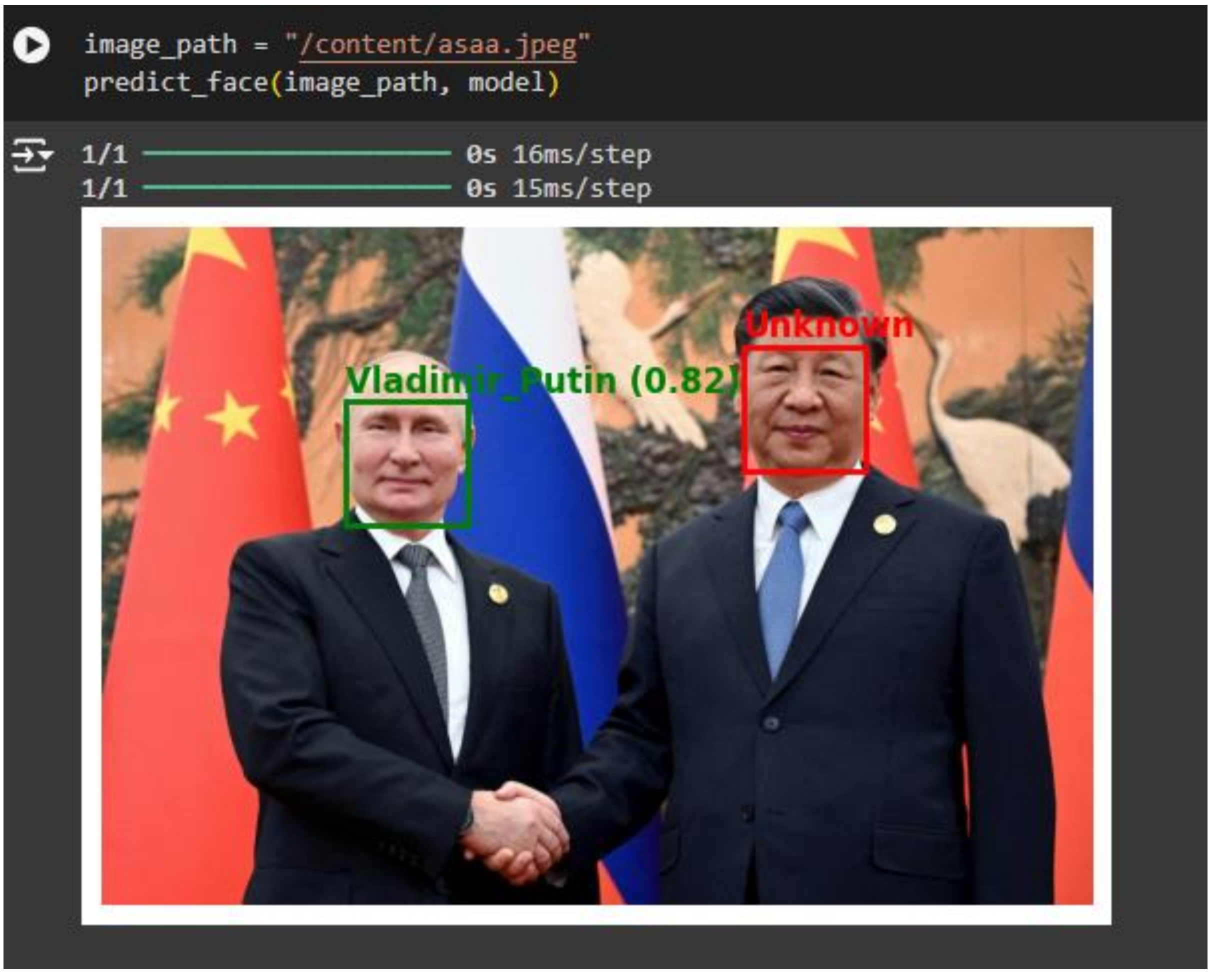
```
          Yasser_Arafat       1.00      1.00      1.00         1
               Yoko_Ono       1.00      1.00      1.00         1
      Yoriko_Kawaguchi       1.00      1.00      1.00         3
             Zhu_Rongji       1.00      1.00      1.00         2
        Zinedine_Zidane       1.00      1.00      1.00         1

               accuracy                           0.96       975
              macro avg       0.93      0.93      0.92       975
           weighted avg       0.96      0.96      0.96       975

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricW
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricW
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricW
  _warn_prf(average, modifier, msg_start, len(result))
```

Per-Class Performance

- **High Scores**: Many individuals (e.g., Al_Sharpton, Albert_Costa) achieved perfect scores, demonstrating the model's capability to accurately recognize these individuals.
- **Moderate Scores**: Some individuals (e.g., Anna_Kournikova) showed lower recall but high precision, indicating that the model might miss some instances but correctly identifies those it does predict.
- **Varied Performance**: Certain individuals (e.g., Ann_Veneman, Alejandro_Toledo) displayed varied metrics, suggesting areas for potential improvement.

Running the Project

1. **Face Detection**:
   - Load images and apply the face detection model to locate faces.
2. **Face Recognition**:
   - Extract features from detected faces and use the trained recognition model to classify them.
3. **Evaluation**:
   - Assess the model's performance using accuracy, precision, recall, and F1 scores.
4. **Model Saving and Loading**:
   - Save the trained model and label encoder for future use.
   - Load the model and encoder to make predictions on new images.



```
image_path = "/content/asaa.jpeg"
predict_face(image_path, model)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
```

Conclusion

The face recognition project demonstrates the application of deep learning and computer vision techniques to identify faces from images. By leveraging dlib for feature extraction and TensorFlow/Keras for model training, the project achieves robust performance in face recognition tasks. The methodology ensures that the system is capable of accurately detecting and recognizing faces, making it suitable for various practical applications.