Face Recognition

I. Objective:

The objective of this project is to develop a face recognition system using a combination of traditional methods and deep learning frameworks. The project encompasses understanding the computational challenges associated with biometric attributes, particularly faces, conducting a literature review, and constructing a customized solution.

A. Project Components:

1. Data Collection:

- Obtained face images from the LFW (Labeled Faces in the Wild) dataset, a publicly available dataset with a diverse set of faces.
- Ensured that the dataset contains more than 20 subjects, each with more than 2 images, meeting the project requirements.

2. Methodology:

- Utilized the FaceNet model, a deep learning framework designed for face recognition tasks.
- Preprocessed images by resizing them to (224, 224) pixels, which is the input size required by the FaceNet model.
- Extracted facial embeddings using the FaceNet model, which represents faces as high-dimensional vectors in a feature space.

3. Implementation:

- Developed Python code to preprocess images, extract embeddings, and perform face recognition.
- Utilized external libraries such as Keras, scikit-learn, and numpy for efficient implementation of machine learning algorithms and data processing.
- Incorporated error handling mechanisms to manage exceptions during image loading and processing.

4. Evaluation Scenarios:

- Selected three images from the dataset, each from a different subject, as "unknown" queries to evaluate recognition performance.
- Choose two subject images not in the dataset for validation, testing the system's ability to handle new faces.

5. Validation and Reporting:

- Evaluated the recognition performance using classification metrics such as precision, recall, and F1-score.
- Generated classification reports for both query images and validation images, saving them as text files for reference.

II. Face Recognition Model

The **face_recognition_model.py** script trains a powerful face recognition model using the LFW (Labeled Faces in the Wild) dataset, employing deep learning techniques like FaceNet for generating compact face embeddings and MTCNN for precise face detection and alignment. TensorFlow underpins these processes, ensuring scalability and efficiency. The script includes preprocessing, training, evaluation, and artifact saving for inference. Leveraging fusion techniques merges FaceNet's embeddings with MTCNN's aligned face regions, enhancing accuracy and robustness in face recognition tasks.

A. Script Overview

1. Data Loading and Preprocessing:

The script loads images from the specified dataset directory (**DATA_DIR**) and preprocesses them using the MTCNN (Multi-Task Cascaded Convolutional Networks) for face detection and alignment. It also encodes the labels using a **LabelEncoder** for model training.

2. Model Training:

The FaceNet model is initialized and used to extract face embeddings from the preprocessed images. These embeddings are then used to train a classification model using a dense neural network architecture.

3. Model Evaluation:

After training, the model is evaluated on a validation set to assess its performance in terms of loss and accuracy. The best-performing model is saved as **face_recognition_model.h5**.

4. Label Encoder:

The script saves the encoded classes (labels) as **label_encoder_classes.npy**, which is crucial for mapping class labels during inference.

B. Usage

To run the script:

- Ensure the dataset path (**DATA_DIR**), image size (**IMAGE_SIZE**), batch size (**BATCH_SIZE**), and number of epochs (**EPOCHS**) are appropriately set.
- Activate the virtual environment (**newenv**) containing the required dependencies. Use source ***newenv/bin/activate*** # For macOS/Linux or ***newenv\Scripts\activate*** # For Windows
- Execute the script using Python: **python face_recognition_model.py**.

C. Outputs

- **face_recognition_model.h5**: Trained face recognition model.
- **label_encoder_classes.npy**: Encoded classes for label mapping.
- Logs and outputs related to model training and evaluation.

III. Face Recognition Query & Validation

The **face_recognition_query_validation.py** script is used for performing face recognition tasks on query and validation images. It utilizes the trained face recognition model (**face_recognition_model.h5**) and label encoder classes to classify and evaluate faces.

A. Script Overview

1. Initialization:

The script loads the trained model and label encoder classes necessary for inference.

2. Image Preprocessing and Embedding Extraction:

It preprocesses query and validation images, extracts face embeddings using the FaceNet model, and prepares them for recognition.

3. Face Recognition:

The script performs face recognition on query images and validation images, generating classification reports (**query_classification_report.txt** and **validation_classification_report.txt**) containing performance metrics such as precision, recall, and F1-score.

B. Usage

To run the script:

- Ensure the virtual environment (**newenv**) is activated. ***source newenv/bin/*****activate** # For macOS/Linux or ***newenv\Scripts\activate*** # For Windows
- Set the appropriate paths for query and validation image directories (**query_dir** and **validation_dir**) and other constants.
- Execute the script using Python: **python face_recognition_query_validation.py**.

C. Outputs

- **query_classification_report.txt**: Classification report for performance evaluation on query images.
- **validation_classification_report.txt**: Classification report for performance evaluation on validation images.
- Logs and outputs related to face recognition tasks.

IV. Sources and References:

A. Sources

1. "LFW (Labeled Faces in the Wild) Dataset" - http://vis-www.cs.umass.edu/lfw/
2. "FaceNet: A Unified Embedding for Face Recognition and Clustering" FaceNet is a deep learning framework developed by Schroff, Adam et al. at Google Research, which is renowned for its face recognition capabilities using triplet loss and Siamese network architecture (Schroff, Adam, & Kalenichenko, 2015). https://arxiv.org/abs/1503.03832
3. Keras Documentation: https://keras.io/
4. scikit-learn Documentation: https://scikit-learn.org/stable/
5. numpy Documentation: https://numpy.org/doc/
6. MTCNN (Multi-task Cascaded Convolutional Networks) is a face detection and alignment framework introduced by Zhang, Zhang, Song, Qi, and Huang in their paper "Joint Face Detection and Alignment Using Multi-Task Cascaded Convolutional Networks" (Zhang et al., 2016).
7. TensorFlow, developed by Google Brain, is an open-source deep learning framework widely used for implementing neural networks and machine learning algorithms (Abadi et al., 2016).

B. References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
2. Schroff, F., Adam, H., & Kalenichenko, D. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).
3. Zhang, K., Zhang, Z., Song, H., Qi, H., & Huang, Y. (2016). Joint face detection and alignment using multi-task cascaded convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3464-3473).

Note:

- Any modifications or enhancements made to the original code or methods are documented within the code files.
- For detailed insights into the implementation, refer to the comments and documentation within the Python scripts.

This readme provides an overview of the face recognition project, including data acquisition, methodology, implementation details, execution instructions, and references. For further information or assistance, contact Mahamadoun Alassane Touré at matoure@iu.edu.