CS308 report

Weihao Li 12110415 Xuanyu Shi 12111404

Hongming Zhang 12112015 Wei Pan 12211810

June 15, 2024

https://github.com/BaiRiDreamer/SUSTechCVProject

1 Introduction

Facial recognition technology is a method that matches human faces in digital images or video frames with face data in a database to verify personal identity, typically achieved through biometric identification. Nowadays, the definition of facial recognition has become broader, encompassing not only the recognition of faces but also the extraction of information such as emotions and age from facial features. These aspects are considered integral parts of the field of facial recognition. As humans, we can easily complete facial recognition tasks, quickly identifying familiar faces and extracting information such as emotions and age under varying lighting conditions, angles, and facial expressions, often subconsciously. However, for machines, recognizing faces and extracting information is a complex process. Machines require sophisticated algorithms and large amounts of data training to learn and recognize different facial features. For different functionalities, even multiple algorithms and models may be needed to achieve accurate results.

Facial recognition is of great importance to the future development of society, and this research field has garnered significant attention. The importance of facial recognition lies in the rich information it contains compared to other human biometric features, such as identity, emotion, and age. Additionally, facial recognition has high collectability and versatility because, unlike fingerprints and iris features, faces are usually easier to present and large enough to be clearly captured by public security cameras. The significance of facial recognition also lies in its wide range of applications. In the security domain, facial recognition is an important means of personal identity authentication. It is used in everyday devices such as smartphones and vending machines, as well as in critical public locations such as airports for identity verification. On a societal level, facial recognition can be used in the search for missing persons by adding their information to a database and using public cameras for real-time matching. Once identified, the precise location of the missing person can be determined. Additionally, in interrogation scenarios, facial recognition technology can analyze facial expressions and vascular features to determine whether a suspect is lying.

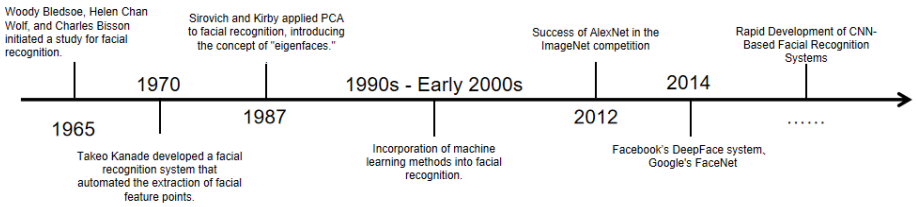


Figure 1: Evolution of Facial Recognition Technology: A Historical Timeline

Facial recognition technology has a long history, dating back to the 1960s. In 1965, Woody Bledsoe, Helen Chan Wolf, and Charles Bisson jointly proposed a study aimed at enabling computers to recognize faces and perform matching. In this project, much of the work was still done manually. For

example, after collecting facial images, researchers would manually mark key feature points on the images, such as the hairline, eyes, and mouth. These feature points were then used to generate feature vectors that describe the geometric structure of the face. The computer would match these feature vectors with those in the database and provide a similarity score. In 1970, Takeo Kanade developed a facial recognition system that required no manual intervention. Although the system still used facial feature points for recognition, the extraction process was more intelligent. In 1987, Sirovich and Kirby first applied Principal Component Analysis (PCA) to facial recognition, introducing the concept of 'eigenfaces'. From the 1990s to the early 2000s, facial recognition gradually incorporated machine learning methods, such as Linear Discriminant Analysis (LDA) for supervised learning and Support Vector Machine (SVM) algorithms. These methods significantly improved the performance of facial recognition systems at the time. During this period, facial recognition primarily developed based on traditional vision methods and machine learning techniques. Until the 2010s, the success of AlexNet in the ImageNet competition demonstrated the immense potential of Convolutional Neural Networks (CNN) in visual image processing, leading the facial recognition field to adopt deep learning technologies and advance in new directions. In 2014, Facebook developed the DeepFace system, achieving a 97.35% accuracy on the Labeled Faces in the Wild (LFW) dataset. In the same year, Google's FaceNet achieved an accuracy of 99.63% on the same dataset. Compared to traditional methods, CNN significantly enhanced the performance of facial recognition systems. With the introduction of CNN, facial recognition systems have also developed new functionalities, such as age recognition, emotion recognition, gender recognition, and ethnicity recognition. Today, the performance, accuracy, and ability to handle complex scenarios of facial recognition systems have greatly improved, leading to many practical new features. The future of facial recognition systems looks promising.

However, many challenges remain in the field of facial recognition:

1. **Illumination:** Illumination variation is a significant challenge for facial recognition systems. Changes in light intensity and direction can significantly alter facial features, causing large discrepancies in features extracted under different lighting conditions.

2. **Pose:** The recognition ability of facial recognition systems significantly decreases when the observation angle or head rotation angle exceeds a certain range. Most facial recognition systems perform poorly in dealing with large pose variations.

3. **Occlusion:** When the face is partially covered by objects, the recognition accuracy of facial recognition systems is greatly affected. Common occlusions such as masks and sunglasses can negatively impact system performance.

4. **Expression:** Facial expression changes can alter the appearance of the face, making it difficult for facial recognition systems to accurately identify. Different expressions, such as smiling or frowning, change facial appearance and affect recognition results.

5. **Low Resolution:** Low-resolution images lead to a loss of facial feature information, reducing the performance of facial recognition systems.

6. **Ageing:** As individuals age, their facial features undergo significant changes. This natural change in appearance poses a challenge to the performance of facial recognition systems.

Figure 2: Challenges in Facial Recognition: Lighting, Age Variations, and Resolution

Despite significant progress in facial recognition technology in recent years, these challenges still require further research and solutions. Addressing these issues will enhance the robustness and accuracy

of facial recognition systems, enabling them to maintain high performance under various complex conditions.

In the future, facial recognition technology will continue to play an important role in multiple fields. Researchers are working on developing more advanced algorithms and models to further improve system performance in complex environments. The integration of multimodal biometric recognition, such as combining face, iris, and fingerprint recognition, will significantly enhance the security and reliability of future facial recognition systems. Additionally, with advancements in computing power, real-time and large-scale facial recognition applications will become more widespread. In practical applications, facial recognition will expand into emerging fields such as personalized services in smart homes and smart city construction.

Despite the numerous challenges, facial recognition technology continues to innovate and develop, contributing to societal progress and providing greater benefits to humanity.

2 Related work

2.1 Traditional work for face recognition

Facial recognition is the task of identifying or verifying individuals from facial images, a long-standing research topic in computer vision. In traditional methods, eigenfaces were a popular technique that used Principal Component Analysis (PCA) to represent faces as a set of feature vectors. These feature vectors, known as eigenfaces, form a subspace where the variance of facial images is maximized. Techniques like Support Vector Machines (SVM) were then used for classification, seeking to find the optimal decision boundary in the reduced feature space.

However, traditional methods have limitations when handling large-scale datasets and complex facial image variations. They often struggle to leverage the vast amounts of data available today and suffer from the curse of dimensionality. Additionally, their performance significantly degrades when facing challenges such as pose variations, occlusions, and changes in lighting conditions.

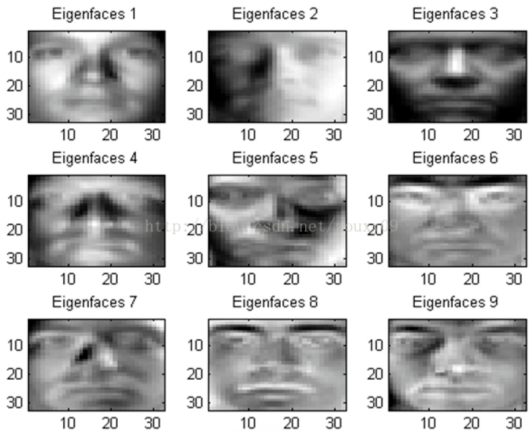


Figure 3: eigen face extracted

2.2 The Rise of Deep Learning in Facial Recognition

With the advent of deep neural networks (DNNs), facial recognition research has witnessed a paradigm shift. DNNs, particularly Convolutional Neural Networks (CNNs), have shown remarkable ability in learning hierarchical representations of data that are robust to variations and can generalize well to unseen data.

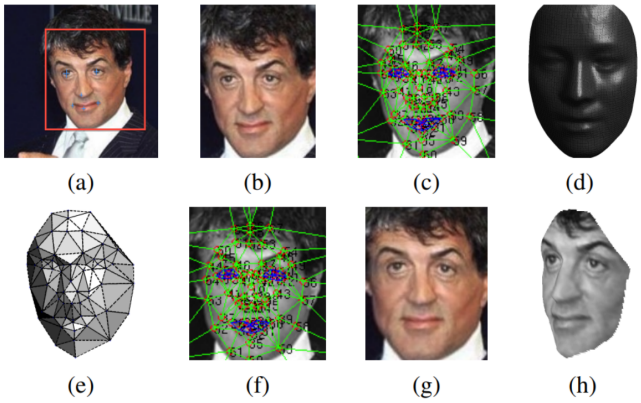


Figure 4: Alignment framework in Deepface using DNN

The development of large-scale facial datasets such as FaceMask CelebA, Flicker-Faces-HQ, and NIST Mugshot Identification has further propelled the advancement of deep learning-based facial recognition techniques. These datasets provide a rich source of labeled facial images that can be used to train deep neural networks in a supervised manner.

3 Deepface

In recent years, facial recognition has made significant progress, with deep learning-based models achieving human-level or even superhuman performance on various benchmarks. Among these models and framework, Deepface, mainly developed by serengil , is a most advanced lightweight face recognition and face attribute analysis framework. We developed our own project based on Deepface.

3.1 models

Deepface includes the most advanced models:, including

- **VGG-Face:** The VGG-Face model is a deep convolutional neural network model developed by the Computer Vision Research Group at the University of Oxford specifically for face recognition tasks. This model is based on VGG architecture and adopts deep convolutional neural network structure. Through training on a huge face data set, it can effectively extract and recognize face features.

- **FaceNet:** FaceNet is a deep learning model developed by Google for face recognition. Different from traditional methods based on feature extraction and classification, FaceNet directly learns a 128-dimensional face feature embedding space, so that similar face feature vectors are closer in the embedding space, while different face feature vectors are farther apart. It uses a training method called triplet loss, which enables the model to effectively distinguish between different individuals by selecting suitable pairs of positive and negative samples.

- **OpenFace:** OpenFace is an open source face recognition model and toolkit based on FaceNet and dlib libraries. Developed by researchers at Carnegie Mellon University and the University of Calgary, OpenFace aims to provide a flexible and efficient face recognition system that is easy to deploy in real-world applications. OpenFace uses deep learning technology to learn the embedding space of face features, so that similar face feature vectors are closer and different face feature vectors are farther apart. Trained using contrastive loss, it performs well on a variety of tasks, such as face verification, recognition, and clustering. As an open source project, OpenFace's code and model parameters are publicly available, with good documentation and community support, making it easy for developers and researchers to apply and customize in their own projects. Due to its open source and ease of use, OpenFace has received a lot of attention and use in both academic research and industrial applications.

- **GhostFaceNet:** GhostFaceNet is a lightweight face recognition model proposed by a research team at Tsinghua University. The model is designed to solve the problem of efficient face recognition in resource-constrained environments such as mobile devices. GhostFaceNet uses the Ghost module, a special lightweight module that simplifies and accelerates the model by reducing the amount of parameters and computation in the model.
- **DeepId:** DeepID is a deep learning model for face recognition developed by a research team at the Chinese University of Hong Kong. Unlike other traditional face recognition methods, DeepID uses deep convolutional neural networks (CNNS) to learn feature representations in face images. The model extracts abstract features from face images through multiple convolution layers and fully connected layers, so that similar face feature vectors are closer in embedding space, while different face feature vectors are farther apart.

With all these models packaged together, Deepface has a face recognition accuracy of 97 percent and has proven to be more successful at face detection than the average face recognition framework. Facebook uses Deepface to prevent impersonation and identity theft on its platform. In our project, we will give the user the freedom to choose the model.

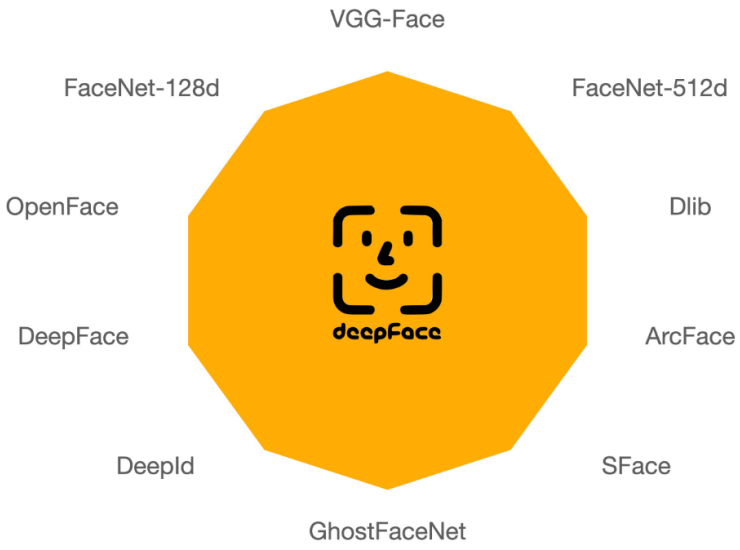


Figure 5: models in deepface

3.2 deepface network architecture

DeepFace utilizes a deep convolutional neural network architecture to learn discriminative representations from facial images. It incorporates techniques such as 3D face alignment to normalize input images and mitigate the effects of pose variations. Additionally, DeepFace employs a multi-task learning approach where the network is jointly trained on both face verification and face recognition tasks, enabling it to learn more robust and generalized representations. Here is the newwork architecture:

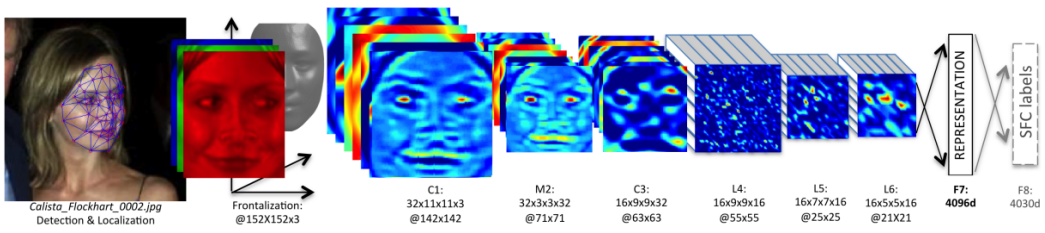


Figure 6: deep face working pipline

- **Input layer:** The input is a three-channel (RGB) aligned face image with dimensions of 152x152 pixels.
- **Convolution layer and pooling layer:**
  - **C1:** The first convolution layer, containing 32 filters of size 11x11x3, generating 32 feature maps.
  - **M2:** The first maximum pooling layer, with 3x3 windows and 2 steps for each feature map.
  - **C3:** The second convolution layer, containing 16 9x9x16 filters, extracts low-level features such as edges and textures.
- **Local connection layer:**
  - **L4, L5, and L6:** These layers are locally connected, and each location learns a different set of filters for aligned face images. Because of the different local statistical properties of different regions, the hypothesis of spatial stationarity of convolution is not applicable. Each output unit is affected by a large region of the input (for example, the output of the L6 is affected by the input region of 74x74x3).
- **Fully connected layer:**
  - **F7 and F8:** These two layers are fully connected layers where each output unit is connected to all input units. They are able to capture correlations between features in different parts of a face image. For example, the relationship between the position and shape of the eyes and mouth.
- **Feature representation:** The output of layer F7 is used as the original face representation feature vector throughout the paper.
- **Output layer:** The output of the final fully connected layer generates a probability distribution of class labels via class K softmax.

The goal of training is to maximize the probability of the correct class (face ID) by minimizing the cross-entropy loss for each training sample, where if k is the index of the true label for a given input, the loss is $L = -logp_r$. This loss is minimized over the parameters by computing the gradient of L with respect to the parameters and updating them using stochastic gradient descent (SGD) , with gradients computed through standard backpropagation of the error. An interesting property of the features produced by this network is that they are very sparse.

3.3 Frameworks

In our project, we utilized several frameworks and libraries to support the implementation and functionality of our facial recognition system. The primary frameworks and libraries we used include:

- **TensorFlow/Keras:** TensorFlow and its high-level API Keras were used for building and training the deep learning models. These frameworks are well-suited for their robustness, ease of use, and extensive community support.
- **OpenCV:** OpenCV was employed for image processing tasks such as reading images, performing face detection, and pre-processing facial images before feeding them into the neural network.
- **DeepFace:** The DeepFace framework developed by Serengil was integrated to leverage its pre-trained models and utilities for face recognition and attribute analysis. DeepFace supports multiple state-of-the-art models like VGG-Face, FaceNet, OpenFace, and Facebook's DeepFace.

The combination of these frameworks provided a robust and flexible foundation for our facial recognition system, enabling us to achieve high accuracy and efficiency.

3.3.1 Structure

Our project mainly conclude 3 functional modules, respectively are deep-face module, LLM interaction module and age transformation module. Here are the brief use flow diagram of our program.

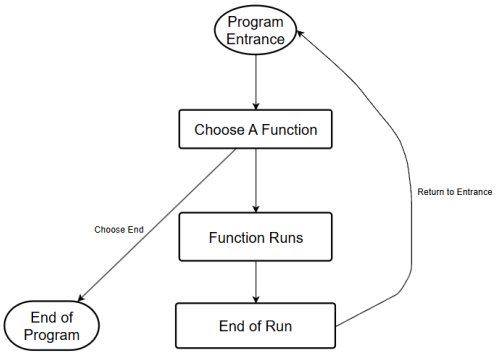


Figure 7: Use Flow Diagram

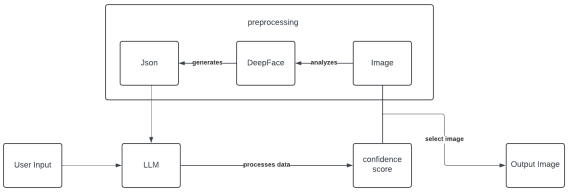


Figure 8: Framework in LLM part

3.4 LLM

The Large Language Model (LLM) component of our project plays a crucial role in providing context-aware interactions and enhancements to the facial recognition system.

Specifically, the LLM helps in creating a natural language interface for users to interact with the system. Users can input queries , which makes the system more user-friendly. Based on LLM, our framework is like this:

For the implementation of the LLM, we leveraged OpenAI's GPT-3.5 model, known for its advanced natural language understanding and generation capabilities. This integration significantly enhanced the overall functionality and user experience of our facial recognition system.

3.5 Age transformation synthesis using GANs

Backpropagation makes discriminative tasks very popular, and RNN uses the discriminative network successfully to predict the next character in the sequence as a generative model, but they cannot generate a new image, so GANs come out.

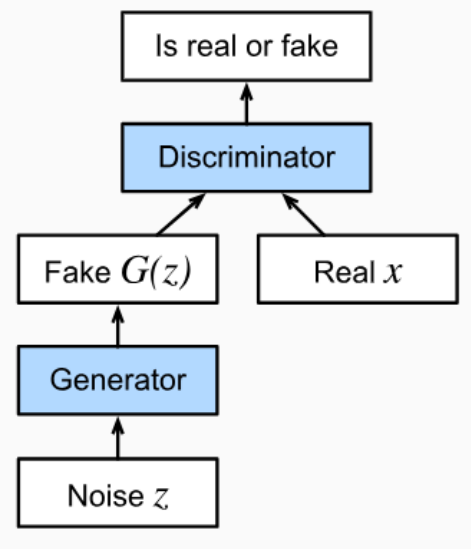


Figure 9: Generative Adversarial Network Architeture

In 2014, a breakthrough paper introduced Generative adversarial networks (GANs) (Goodfellow et al., 2014), a clever new way to leverage the power of discriminative models to get good generative models. At their heart, GANs rely on the idea that a data generator is good if we cannot tell fake data apart from real data. Andrew Ng compared Discriminator and Generator to art forger and art inspector respectively. The art forgers are always trying to paint beautiful pitures, and the art inspectors are always give feedback to the art forgers that whether their artworks are good enough.

Figure 10: Input Image

Figure 11: Output Image

In our project, we implement the lifespan age transform Gans. Given an input image selected by the user, we use the model to generate a lifespan video. In Fig6, there is an example of inputing Jude Bellingham to generate a piture of 40 year old of him.

4 Implementation Details

In this section, we provide detailed descriptions of the implementation steps for our facial recognition system.

4.1 Model Evaluation

The pre-trained models were evaluated on a separate test set using the following metrics:

- **Accuracy:** The overall accuracy of the model in correctly identifying faces.
- **True Positive Rate, False Positive Rate, Area Under the Curve(AUC):** These metrics were calculated to evaluate the model's performance in a more detailed manner, particularly in handling imbalanced classes.

4.2 Integration Challenges with Large Language Models (LLMs)

In the integration of Large Language Models (LLMs) into our system, several challenges were encountered:

4.2.1 Output Usability

The initial problem was that the LLM output was not easily parsable by our program. To address this, we employed a prompt modification technique.

Prompt Modification We capitalized the phrase "**PROVIDE A CONFIDENCE SCORE**" to ensure it was not overlooked by the LLM, a problem observed when this emphasis was absent.

Data Extraction Additionally, regular expressions were utilized to efficiently extract numerical data from the responses.

4.2.2 Processing Latency

Another significant issue was the slow processing of images and responses from the LLM (using ChatGPT-3). To mitigate this, we implemented two optimization strategies:

Preprocessing Data We preprocessed data into JSON files stored in a dedicated directory. For instance, DeepFace's `analyze` method typically requires approximately 3 seconds per image, and this preprocessing reduced redundancy and improved access times.

Multithreading We employed multithreading to manage concurrent access to ChatGPT for all images. This approach significantly reduced the overall processing time by allowing parallel handling of multiple image requests.

These enhancements improved the efficiency and effectiveness of our system, making the integration with LLMs more seamless and responsive.

In summary, the integration of various frameworks and the utilization of a powerful LLM component significantly enhanced our facial recognition system. The detailed implementation process ensured that the system is robust, accurate, and user-friendly.

5 Method

1. **DeepFace.verify()**
   (a) **Functionality:** The verify() function performs facial matching by comparing two images based on feature vectors and distance metrics obtained from the selected face recognition model. It incorporates parameters such as face detector selection, alignment, and normalization to enhance the reliability and accuracy of the verification process.
   (b) **Inputs:**
      i. **img1_path, img2_path:**Accepts image data or paths, which are used to receive the two images to be compared.
      ii. **model_name:** Specifies the deep learning model used for extracting facial features. Options include popular models such as VGG-Face and Facenet.
      iii. **detector_backend:**Chooses the backend for face detection (e.g., OpenCV, RetinaFace).
      iv. **distance_metric:**Selects the metric for measuring the similarity or distance between facial features (e.g., cosine, Euclidean).

v. **enforce_detection:**Controls whether the function should throw an exception if no face is detected in an image.

vi. **threshold:** Sets a custom threshold to determine whether the images are of the same individual.

vii. **align:** Determines whether facial alignment based on eye positions should be performed to enhance feature extraction.

viii. **Additional parameters: expand_percentage, normalization, silent, anti_spoofing**

(c) **Outputs:**

A dictionary containing the following information:

i. **verified:**A boolean value indicating whether the images are of the same individual.

ii. **distance:** The computed distance between the facial embeddings of the two images.

iii. **threshold:** The threshold used to determine the matching outcome.

iv. **model:**The face recognition model used.

v. **distance_metric:**The metric used to measure distance.

vi. **facial_areas:**Coordinates of the detected facial regions in both images.

vii. **time:** The time taken to complete the verification process.

(d) **Implementation Details:**

i. **Face Detection:** The function utilizes the specified detector_backend to detect faces in the images.

ii. **Face Alignment:**If enabled, the function aligns faces based on the positions of the eyes, enhancing the accuracy of the images.

iii. **Feature Extraction:**Features are extracted from aligned or original faces using the selected deep learning model.

iv. **Distance Calculation:** The distance between the two features is calculated using the specified distance metric, indicating the similarity between the images. A lower distance suggests a higher similarity.

v. **Verification:** The calculated distance is compared with the threshold. If the distance is below the threshold, the images are considered a match; otherwise, they are not.

2. **DeepFace.find()**

(a) **Functionality:** The find() function identifies images from a specified database that match the facial features extracted from a source image. It can handle multiple faces in a single image and returns comparison information for each detected face against the database records.

(b) **Inputs:**

i. **img_path:**The path to the image or the image data itself.

ii. **db_path:**Directory path of the facial image database.

iii. **model_name:**Specifies the facial recognition model to be used.

iv. **distance_metric:**Chooses the metric for measuring the similarity or distance between facial features (e.g., cosine, Euclidean).

v. **Other parameters: enforce_detection, detector_backend, align, expand_percentage, normalization. . .**

(c) **Outputs:**

The function returns a series of pandas DataFrames, each including details about the database photos that match the facial features detected in the source image:

i. **identity:** Identity of the individual in the database whose face matches the detected face in the source image.

ii. **target_x, target_y, target_w, target_h:**Bounding box coordinates of the matching face in the database.

iii. **source_x, source_y, source_w, source_h:**ounding box coordinates of the detected face in the source image.

iv. **distance:** The distance between the facial features of the source image and the matching database face, calculated using the specified distance metric.

v. **threshold:** Threshold used to determine whether the two faces belong to the same person.

(d) **Implementation Details:**

i. **Face Detection:** Initially, the function employs the detector_backend to detect faces in the source image.

ii. **Face Alignment:** If align is enabled, the detected faces are aligned based on the position of the eyes. This can enhance the accuracy of the image matching process.

iii. **Feature Extraction:** Features are extracted from each detected and aligned face using the specified model.

iv. **Database Comparison:** The extracted features from the source image are compared with all facial features in the database using the specified distance_metric.

v. **Match Determination:**For each facial feature in the source image, the function calculates the distance to every facial feature in the database. If the distance is below the specified threshold, the two faces are considered a match.

vi. **Results Compilation:** The function compiles the results into DataFrames, detailing the identity and match information for each detected face in the source image.

3. **DeepFace.analyze()**

(a) **Functionality:**The analyze() function is designed to analyze and identify various facial attributes such as age, gender, emotion, and race from an input image. It supports analyzing multiple faces within a single image and provides detailed results for each detected face.

(b) **Inputs:**

i. **img_path:** The path to the image file or the image data itself.

ii. **actions:**Specifies the attributes to be analyzed. The default set includes "emotion", "age", "gender", and "race". Users can customize which attributes to analyze by modifying this parameter.

iii. **Other parameters: detector_backend, enforce_detection, align, expand_percentage, silent, anti_spoofing**

(c) **Outputs:**

The function returns a list of dictionaries, each corresponding to a detected face and including detailed results for the analyzed attributes:

i. **region:** Specifies the bounding box coordinates (x, y, width, height) of the detected face.

ii. **age:** Provides an estimated age.

iii. **face_confidence:**Indicates the confidence level of the face detection.

iv. **dominant_gender and gender:** Provides the most likely gender and the confidence scores for each gender category.

v. **dominant_emotion and emotion:**Identifies the predominant emotion and provides confidence scores for each emotional category.

vi. **dominant_race and race:** Identifies the predominant race and provides confidence scores for each racial category.

(d) **Implementation Details:**

i. **Face Detection:** Initially, the function uses the selected detector_backend to detect faces in the image.

ii. **Face Alignment:**If alignment is enabled, it adjusts the positions of the detected faces based on eye coordinates, enhancing the accuracy of subsequent analyses.

iii. **Attribute Analysis:** The function analyzes the specified attributes (actions) for each detected and aligned face using pre-trained deep learning models customized for each attribute such as age, gender, emotion, and race.

iv. **Results Compilation:**For each face, the function compiles a comprehensive dictionary that includes detailed information such as confidence levels for each attribute.

4. **DeepFace.stream()**

   (a) **Functionality:**The stream() function is designed for real-time face recognition and facial attribute analysis using video streams. It processes frames continuously from a video source. The function employs a predefined database of facial images for individual recognition and optionally performs additional analyses such as detecting emotions, age, gender, or race.

   (b) **Inputs:**

      i. **db_path:** The directory path where the image files for the database are stored. All detected faces in these images are utilized in the recognition process.

      ii. **model_name:**Specifies the face recognition model to be used.

      iii. **enable_face_analysis:**Determines whether to conduct additional facial attribute analyses such as emotion, age, gender, or race.

      iv. **source:** Specifies the video source. The default is 0, which typically represents the primary camera on the device.

      v. **time_threshold:**The time threshold in seconds before a face can be recognized, set by default to 5.

      vi. **frame_threshold:** The number of consecutive frames in which a face must be recognized before a recognition decision is made, with a default value of 5.

      vii. **Other parameters: detector_refter_backend, distance_metric, anti_spoofing**

   (c) **Outputs:** None.

   (d) **Implementation Details:**

      i. **Initialization:**The function initializes the video stream from the specified source and sets up the face recognition model and detector based on the provided parameters.

      ii. **Face Detection:**In each frame of the video, the function employs the detector_backend to detect faces.

      iii. **Recognition and Analysis:**For each detected face, the function compares it against the database using the chosen distance_metric. If a face matches a database entry (considering the time and frame thresholds), the identity of the individual is recognized

      iv. **Facial Attribute Analysis:**If enable_face_analysis is enabled, the function also analyzes the detected faces for specified attributes such as emotion, age, gender, or race.

      v. **Real-time Feedback:**The results of face recognition and any additional analyses are displayed in real-time.

5. **__extract_faces_and_embeddings()**

   (a) **Functionality:**The primary purpose of the __extract_faces_and_embeddings() function is to extract facial regions from a specified image and derive feature vectors from these regions. The extracted features can then be used for face matching or other facial analysis tasks. This function is utilized in the aforementioned face matching functions.

   (b) **Inputs:**

      i. **img_path:**The path to the image or the image itself.

      ii. **model_name:**Specifies the facial recognition model used to generate facial features.

      iii. **detector_backend:** Specifies the backend technology used for facial detection.

      iv. **Other parameters: enforce_detection, align, expand_percentage, normalization, anti_spoofing**

   (c) **Outputs:**

      i. **embeddings:**A list of feature vectors for each detected face.

      ii. **facial_areas:** A list of area information for each detected face.

   (d) **Implementation Details:**

i. **Face Detection:** The **detection.extract_faces()** function is used to detect faces from the specified image. The **extract_faces()** function conducts facial detection based on the backend specified by detector_backend, considering options such as face alignment and expansion, and outputs the regions and related information for each detected face within the image.

ii. **Feature Extraction:**For each face detected by **detection.extract_faces()**, the **representation.represent()** function is employed to obtain the facial feature vectors. This function uses the specified model_name along with other relevant parameters (such as alignment, normalization, etc.) to generate the feature vectors.

6. **find_distance()**

(a) **Functionality:**The find_distance() function is designed to compute the distance between two feature vectors using a specified metric. It measures the similarity or dissimilarity between facial feature vectors. In the context of facial recognition, a smaller distance usually indicates a higher likelihood that the two vectors represent the same individual.

(b) **Inputs:**

i. **alpha_embedding:** A facial feature vector.

ii. **beta_embedding:**Another facial feature vector.

iii. **distance_metric:** The method used to compute the distance between the two feature vectors. Supported metrics include "cosine", "euclidean", and "euclidean_l2"

(c) **Outputs:**

i. **distance:** The computed distance between the two feature vectors, is represented as a floating-point number.

(d) **Implementation Details:** The function calculates the distance based on the metric specified in the distance_metric parameter:

i. **Cosine Distance:**
Cosine distance measures the cosine of the angle between two vectors. It is derived from the cosine similarity formula and is given by:

$$Cosine\,Distance = 1 - \frac{\alpha \cdot \beta}{\|\alpha\|\,\|\beta\|} \tag{1}$$

where α and β are the vectors, and $\|\alpha\|$ and $\|\beta\|$ are the norms (magnitudes) of the vectors. This metric is particularly useful for measuring similarity in normalized data, as it is sensitive to the angle between vectors, not their magnitude.

ii. **Euclidean Distance:**Euclidean distance is the "ordinary" straight-line distance between two points in Euclidean space. The formula is:

$$Euclidean\,Distance = \sqrt{\sum_{i=1}^{n}(\alpha_i - \beta_i)^2} \tag{2}$$

where α_i and β_i are components of the vectors α and β. This distance is effective for measuring actual distances between points.

iii. **Euclidean L2 Distance:**This is the Euclidean distance calculated after applying L2 normalization to both vectors. L2 normalization adjusts the vectors to have a norm of one, but keeps their direction:

$$Euclidean\,L2\,Distance = Euclidean\,Distance(L2(\alpha), L2(\beta)) \tag{3}$$

where L2(v) is the L2 normalization of vector v, calculated as:

$$L2(v) = \frac{v}{\|v\|} \tag{4}$$

This metric is useful when the magnitude of the vectors should not influence the distance measure, focusing solely on the directionality.

6 Experiments

6.1 Dataset

The database used during the experiments is LFW (Labeled Faces in the Wild). The LFW face database was organized and compiled by the Computer Vision Laboratory at the University of Massachusetts Amherst. It contains over 13,000 face images of approximately 5,700 individuals, all collected from the internet, with a resolution of 250x250 pixels. It is currently a commonly used test set for face recognition and is widely utilized in both academia and industry. Due to its diversity and challenge, LFW is considered an important benchmark for testing and comparing the performance of face recognition systems.

6.2 Performance Evaluation

In DeepFace, various models are applied to the LFW (Labeled Faces in the Wild) database for facial matching to obtain evaluation metrics, which determine the performance of multiple models.

The metrics used for evaluation include accuracy, true positive rate (TPR), false positive rate (FPR), and the area under the ROC curve (AUC). Their precise definitions are as follows:

Accuracy is defined as the ratio of correctly identified instances, both positive and negative, to the total instances and is calculated by the formula:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

True Positive Rate (TPR), also known as recall or sensitivity, measures the ratio of correctly predicted positive observations to all observations in the actual class and is calculated by the formula:

$$\text{TPR} = \frac{TP}{TP + FN} \tag{6}$$

False Positive Rate (FPR) represents the ratio of incorrectly predicted positive observations to all observations in the actual negative class and is calculated by the formula:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{7}$$

Area Under the Curve (AUC) measures the overall ability of the model to discriminate between positive and negative classes and is calculated as the area under the ROC curve, which plots TPR against FPR at various threshold settings.

By obtaining these metrics, we can clearly ascertain the real-world performance of each model. The specific experimental results are presented in Section 6.4.

6.3 Experimental design

1. **Download and Process the LFW Dataset:**Download the LFW (Labeled Faces in the Wild) dataset from the internet and save the face image pairs along with their associated labels to a local file.

2. **Create Image Pairs:**Process the downloaded images to generate pairs of images and corresponding labels. These data will be utilized for subsequent face recognition tests.

3. **Test with Various Face Recognition Models:** The purpose of this experiment is to evaluate and compare the performance of various face recognition models on the same dataset. Therefore, the experimentation solely varies in terms of the models used and the distance metrics applied. The study incorporates ten deep learning models: Facenet512, Facenet, VGG-Face, ArcFace, Dlib, GhostFaceNet, SFace, OpenFace, DeepFace, and DeepID. Three distance metrics are employed: euclidean, euclideanL2, and cosine. The Euclidean metric is used to measure the straight-line distance between two points, defined in multi-dimensional space as the square root of the sum of the squares of the dimensional differences. The Euclidean L2 is essentially an L2 normalization applied prior to computing Euclidean distances, whereas the cosine metric assesses

the degree of similarity in direction between two vectors, independent of their magnitudes. Aside from these specified parameters, all other variables are held constant, including the choice of face detector and the settings for image alignment, allowing for a comprehensive evaluation of the models' performance across different distance metrics on the same dataset.

4. **Distance Measurement and Threshold Determination:**After obtaining the feature distances from the models, these distances are compared with a predefined threshold to produce the model outputs. This threshold is determined based on the average distances of positive and negative samples to achieve optimal classification performance.

5. **Result Evaluation and Saving:**Save the results of each test (i.e., whether each image pair has been correctly identified as the same person or different individuals) into a CSV file. These results will later be used to calculate final metrics such as accuracy and recall rates.

6. **Calculate Performance Metrics:**Using the saved result data, calculate the performance metrics for each model configuration, such as accuracy. Finally, generate charts to visually represent these metrics.

6.4 Experimental result

The experiments were conducted using the code in `Perform_Experiments.ipynb`. Due to time constraints, we limited our experiments to a single model, *Facenet512*, and one detector, *YOLOv8*.

The results of these experiments are presented below:

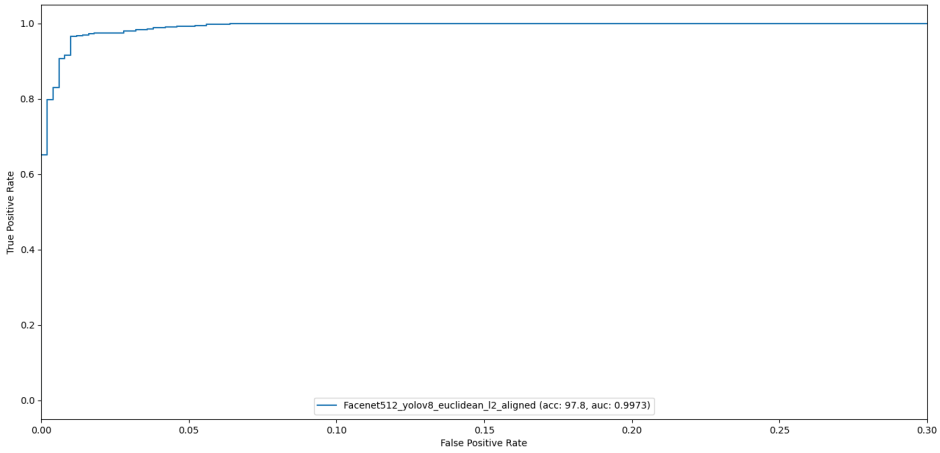


Figure 12: Sample output from the Facenet512 model and YOLOv8 detector.

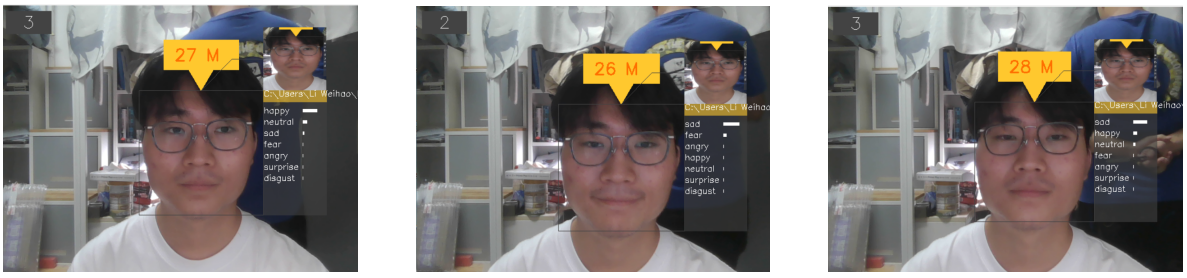
From the official performance metrics announced by DeepFace, we observe that the maximum discrepancy between the official results and our experimental results is approximately 1%, which aligns well with our findings.

6.5 Manual Evaluation of Streaming and DeepFace-LLM Integration

Since there are no established metrics for assessing the streaming capabilities and the integration of DeepFace with LLMs, we conducted manual evaluations.

6.5.1 Manual Evaluation of Streaming

For the streaming component, Weihao Li used his own image dataset for real-time detection. The evaluation confirmed that his facial features and emotional states were accurately detected in real-time.



(a) Weihao Li - Happy (b) Weihao Li - Sad (c) Weihao Li - A Little Sad

Figure 13: Real-time detection results of Weihao Li's facial features and emotional states.

6.5.2 Manual Evaluation of DeepFace-LLM Integration

The experiments were conducted using the code in `llm.ipynb` or directly by running `Main.py`. In the experiments, we utilized images from the `testset` folder. The task assigned to the LLM was to identify an image corresponding to the prompt "He is middle aged." By analyzing the data provided in `analysis_results.json`, the LLM was able to accurately determine and select the most fitting image from the available set, demonstrating its capability to process and understand contextual prompts effectively.

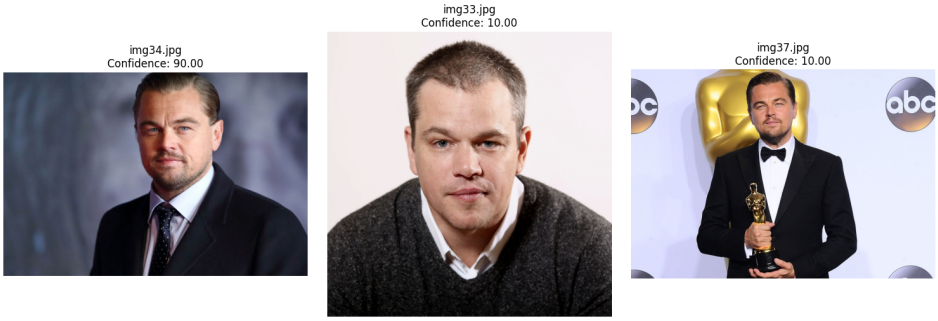


Figure 14: Result of the LLM task based on the prompt "He is middle aged."

7 Conclusion

7.1 Integration between different functions

Lifespan Transformation Synthesis has not been integrated into the PyPI and Anaconda repositories. Therefore, Lifespan transformation Synthesis cannot be installed using conda install or pip install. In addition, the strong coupling between the functions of the Lifespan Transformation Synthesis project brings certain difficulties to the integration of our project. We solve it by some decoupling of functions.

7.2 Advantages

- **Integration Potential**: Successfully integrating DeepFace with LLM demonstrated the potential to combine facial recognition technology with language models. This synergy allows for enriched contextual understanding and provides a foundation for applications requiring both visual and textual data analysis.
- **Enhanced Functionality**: The combined system can theoretically offer more robust and versatile functionality, such as generating descriptions based on facial features and contextual interactions, which could be valuable in domains like security, personalized services, and interactive systems.

- **Innovative Approach**: This project pioneers a novel approach to bridging the gap between vision and language processing, paving the way for future advancements in multi-modal artificial intelligence systems.

7.3 Limitations

- **Accuracy Issues**: The analysis and recognition results were not as accurate as expected. DeepFace's facial recognition performance did not achieve the desired level of precision when augmented by the LLM. The discrepancies in results indicate a need for further refinement and optimization.
- **LLM Limitations**: Despite advancements in language models, the LLM used in this integration still exhibited significant limitations. It struggled with accurate contextual interpretation and often produced erroneous or irrelevant outputs, indicating that current LLMs have not yet reached a level of sophistication necessary for seamless integration with facial recognition systems.
- **Computational Complexity**: Combining DeepFace with an LLM introduced additional computational overhead, complicating the system's deployment and scalability. This increased complexity may limit the practical application in real-time or resource-constrained environments.

8 Contributions

Weihao Li (12110415): 25%
Hongming Zhang (12112015):25%
Wei Pan (12211810):25%
Xuanyu Shi (12111404): 25%

References

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2672–2680.

Or-El, R., Sengupta, S., Fried, O., Shechtman, E., & Kemelmacher-Shlizerman, I. (2020). Lifespan age transformation synthesis. *Proceedings of the European Conference on Computer Vision (ECCV)*.

Qawaqneh, Z., Mallouh, A. A., & Barkana, B. D. (2017). Deep convolutional neural network for age estimation based on vgg-face model. *arXiv preprint arXiv:1709.01664*. https://arxiv.org/abs/1709.01664

Serengil, S. I., & Ozpinar, A. (2020). Lightface: A hybrid deep face recognition framework. *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 23–27. https://doi.org/10.1109/ASYU50717.2020.9259802

Serengil, S. I., & Ozpinar, A. (2021). Hyperextended lightface: A facial attribute analysis framework. *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, 1–4. https://doi.org/10.1109/ICEET53442.2021.9659697

Serengil, S. I., & Ozpinar, A. (2024). A benchmark of facial recognition pipelines and co-usability performances of modules. *Bilisim Teknolojileri Dergisi*, *17*(2), 95–107. https://doi.org/10.17671/gazibtd.1399077