Method

Find：

1. Face detection

In this step, DeepFace uses a pre trained face detection model to locate faces in the image. This is usually achieved by using neural networks to recognize facial features in images. Currently popular face detection models such as MTCNN or SSD can effectively detect faces in complex backgrounds.

2. Feature extraction

After detecting faces, pre trained deep learning models (such as VGG Face, FaceNet, etc.) are used to extract features from each detected face. These features extract key visual information of the face through a network and convert it into high-dimensional data vectors.

Eigenvector formula

\[f (x)=W ^ T x+b \]

Among them, (x) is the input facial image data, and (W) and (b) are the weights and biases of the neural network.

3. Similarity comparison

The extracted feature vectors are used for comparison with the feature vectors stored in the database. There are usually two methods used to calculate similarity here: cosine similarity and Euclidean distance.

Cosine similarity formula

\[\ text {Cosine Similarity}=\ frac {f (x) \ cdot f (y)} {\ | f (x) \ | f (y) \ |} \]

Among them, \ (f (x) \) and \ (f (y) \) are two eigenvectors, dot product represents the dot product between vectors, and the denominator is the product of the moduli of these two vectors.

Euclidean distance formula

\[\ text {Euclidean Distance}=\ sqrt {\ sum\_ {i=1} ^ n (x\_i - y\_i) ^ 2} \]

Among them, \ (x\_i \) and \ (y\_i \) are elements in the two feature vectors.

4. Identification and verification

Based on the calculated similarity, the 'find' function will determine the most matching face. If the similarity exceeds a certain threshold, it is considered the same person.

5. Result output

Finally, the 'find' function returns the recognition result, including similarity score and possible identity information (if there is a match in the database).

**Verify**

1. Facial detection

Firstly, use facial detection algorithms to locate facial regions in the input image. Common facial detection algorithms include Haar feature cascaded classifiers, MTCNN (Multi task Cascaded Convolutional Networks), etc. The output of this step is the bounding box of the face.

2. Facial alignment

In order to eliminate the influence of factors such as shooting angle and lighting on facial feature extraction, facial alignment is necessary. Usually, geometric transformations are performed using the positions of the eyes, nose, and mouth to standardize the face. Assuming we have the coordinates of our eyes (𝑥 1, 𝑦 1) (x1, y1) and (𝑥 2, 𝑦 2) (x2, y2), as well as the coordinates of the standard reference point (𝑥 1 ', 𝑦 1') (x1 ', y1') and (𝑥 2 ', 𝑦 2') (x2 ', y2'), we can align the input image to the standard reference frame through affine transformation.

3. Feature extraction

After facial alignment, use deep neural networks (such as the DeepFace model) to extract facial feature vectors. Assuming that the output of the input image after passing through the network is the feature vectors 𝑓 1f1 and 𝑓 2f2.

4. Eigenvector comparison

Determine whether two feature vectors belong to the same person by calculating the distance between them. Common distance measures include Euclidean Distance and Cosine Similarity. The formula is as follows:

Euclidean distance

𝑑 (𝑓\1, 𝑓\2)=∑τ=1 𝑛 (𝑓\1 \− 𝑓\2 \) 2d (f1, f2)=i=1 ∑ n (f1i − f2i) 2

cosine similarity

Cosine similarity (𝑓 1, 𝑓 2), f2)=∥f1∥∥f2∥f1⋅f2

Among them, 𝑓𝑓𝑓\2f1æ f2 represents the dot product of the eigenvectors, and 𝑓\1} f1 and 𝑓\2} f2 represent the norm of the eigenvectors.

5. Similarity score and threshold determination

Compare the calculated distance or similarity with the preset threshold to determine whether two images belong to the same person. For example, if the Euclidean distance is less than a certain threshold 𝜏τ, it is considered that two images are the same person.

6. Output results

Output verification results based on the comparison results, namely "match" or "mismatch".

Analyze

1. Facial detection

Firstly, use facial detection algorithms to locate facial regions in the input image. This step is the same as the verify function, where methods such as Haar feature cascade classifier and MTCNN can be used to locate facial bounding boxes.

2. Facial alignment

To eliminate the influence of factors such as shooting angle and lighting on facial feature extraction, facial alignment is necessary. By detecting facial key points (such as the positions of eyes, nose, and mouth), geometric transformations are performed on the face to standardize it.

3. Feature extraction

Extract facial feature vectors using pre trained deep neural network models. Unlike the verify function, a specific model is required to extract features related to age, gender, race, and emotion. Assuming that the output of the input image after passing through the network is the feature vector 𝑓 f.

4. Attribute prediction

By inputting the feature vector 𝑓 f into different classifiers or regression models, predict various facial attributes:

Age prediction

Using regression models to predict age:

𝑦 ^ age=𝑓 age (𝑓) y ^ age=fage (f)

Among them, 𝑓 ageage is the regression function used for age prediction, and 𝑦 ^ agey ^ age is the predicted age.

Gender prediction

Using a binary classifier to predict gender:

𝑦 ^ gender=arg max 𝑐 ∈ {male, female} 𝑃 (𝑐𝑓) y ^ gender=argc ∈ {male,female}maxP (c æ f)

Among them, 𝑃 (𝑐𝑓) P (c æ f) is the probability that the feature vector 𝑓 f corresponds to the gender category 𝑐 c.

Race prediction

Using multiple classifiers to predict races:

𝑦 ^ race=arg max 𝑐 ∈ {Asian, Black, White, Indian, Others} 𝑃 (𝑐𝑓) y ^ race=argc ∈ {Asian, Black, White, Indian, Others} maxP (c æ f)

Among them, 𝑃 (𝑐𝑓) P (c æ f) is the probability that the feature vector 𝑓 f corresponds to the racial category 𝑐 c.

Emotional prediction

Using multiple classifiers to predict emotions:

𝑦 ^ emotion=arg max 𝑐 ∈ {happy, sad, angle, sharp, neutral, feel, doubt} 𝑃 (𝑐𝑓) y ^ emotion=argc ∈ {happy, sad, angle, sharp, neutral, front, doubt} maxP (c æ f)

Among them, 𝑃 (𝑐𝑓) P (c æ f) is the probability that the feature vector 𝑓 f corresponds to the emotion category 𝑐 c.

5. Result output

Integrate the output results of various classifiers or regression models to form the final analysis results, including predicted values for age, gender, race, and emotion.

Extract\_faces

1. Facial detection

Use facial detection algorithms to locate facial regions in the input image. This step is the core of the entire process, typically using efficient facial detection algorithms such as MTCNN (Multi task Cascaded Convolutional Networks). The facial detection algorithm will return the bounding box coordinates of each detected face.

Assuming the input image is 𝐼 I, the facial detection algorithm outputs a set of bounding boxes {(𝑥𝑥𝑦𝑦𝑤ℎ\)} 𝑁 {(xi, yi, wi, hi)} i=1N, where 𝑁 N is the number of detected faces, (𝑥π, 𝑦) (xi, yi) is the upper left corner coordinate of the i-th bounding box, 𝑤 ℎ wi and ℎ hi are the width and height of the bounding box, respectively.

2. Facial cropping

Crop each facial region from the input image based on the detected bounding box coordinates. For each bounding box (xi, yi, wi, hi), the corresponding facial image area is 𝐼 [𝑥: 𝑥𝑥𝑦𝑦𝑤𝑦𝑤𝑦𝑤, 𝑦𝑤𝑦𝑦𝑤, ℎℎ) I [xi: xi+wi, yi: yi+hi].

3. Facial alignment

To standardize facial images and eliminate the influence of posture changes, the cropped faces can be aligned. The alignment process is usually carried out by detecting facial key points such as eyes, nose, and mouth. Assuming the coordinates of the keypoints are {(𝑥𝑘, 𝑦𝑘)} 𝑘=1 𝐾 {(xik, yik)} k=1K, affine transformation can be used to align the face to the standard reference frame.

The alignment process includes the following steps:

Detect key points on each face.

Calculate the affine transformation matrix 𝐴 Ai to align the keypoints to the standard reference point.

Apply affine transformation 𝐴𝐴 Ai to each cropped facial region.

4. Facial preservation

Save or return the cropped and aligned facial image as output.