Analysis of face detection and exploration

2021/2022

CMT400

MSc Advanced Computer Science

Tiecheng Wang

ID: C2010921

Supervisor: Xianfang Sun

Moderated by: Padraig Corcoran

18/09/2022

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Xianfang Sun, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to thank my parents for their wise counsel and sympathetic ear. You have been always there for me.

In addition, I would like to thank the beautiful city of Cardiff for the peace and quiet that keeps me safe. The beautiful surroundings and the warmth of the citizens made us international students feel warm.

Finally, I would like to express my grateful feelings to the lady who supported me all over the way to get my Master degree and I owe her as much as I owe to my parents. Thanks to Ms.Ke Liu for all her great contributions and support during my Master year

Declaration of Originality

I certify that this thesis was written completely by me and that it has not previously been submitted, in whole or in part, in any prior application for a degree, except when stated otherwise by reference or acknowledgment.

Proforma

· Candidate Number: C2010921
· Title of the project: Analysis of face detection and exploration
· Word Counts: 15656 Words
· Project Supervisor: Xianfang Sun
· Project Moderator: Padraig Corcoran

· Abstract: In this work, FaceNet and InsightFace's face search accuracy will be compared The face feature extraction technique will be chosen in accordance with the outcomes. In order to create a more effective face feature retrieval technique, this article will also compare the performance of using the Python loop and the vector database Milvus.
Based on the aforementioned findings, a face retrieval system employing MTCNN+InsightFace+Milvus will be finished. This article will also go through the benchmark tests that confirm its suitability for being implemented in a sizable production setting. To gauge how similar two faces are, the idea of face similarity has been put forth.

Table of Contents

1 Introduction

1.1 Introduction

Face recognition is one of the technologies in biometrics that has garnered a lot of interest. Face recognition is increasingly widely used as a form of identification in people's schooling, jobs, and daily lives. Face recognition is one of the most user-friendly biometric technologies, despite the fact that its development is not as far as that of iris recognition and fingerprint recognition at this point. This is because it is non-contact, non-compulsory, and concurrent (Chellappa, Wilson & Sirohey, 1995). Face acquisition in face recognition technology can be done remotely, removing the need for passive collaboration. The identity verification procedure becomes more logical and practical by making full use of the face database resources already available for comparison and identification. The use of artificial intelligence compensation to lessen the impact of face alterations and changes in the surrounding lighting environment to generate a more desired application effect are two obstacles that face recognition-based time and attendance systems must overcome.

1.2 Project Aims and Objectives

Typically, a whole face retrieval system includes the following three components:

Face Feature Extraction, Face Feature Retrieval, and Face Detection.

Among them, face detection with the MTCNN model is currently more common. The extraction and retrieval of face features cannot be handled using a single method, though.

FaceNet and InsightFace will be used in the project to extract face characteristics and improve the accuracy of face search in an effort to find a better face feature extraction technique. To identify a more effective face feature retrieval technique, the performance variations between the vector database Milvus and the Python loop will also be examined.

Experiments are done to compare the search accuracy of employing InsightFace and FaceNet during the facial feature extraction phase. Experiments are carried out to compare the search performance between the Python loop and the vector database Milvus in the face feature retrieval phase.

Based on the findings of the aforementioned investigation, a full face retrieval system will be developed utilizing MTCNN+InsightFace+Milvus. This study also presents the idea of face similarity percentage to gauge the level of face similarity and benchmark tests the Milvus database to confirm its viability for use in a large-scale production setting.

2 Background

2.1 Related Work

In the journal Nature, Francis Galton first proposed face recognition methods in the late nineteenth century. He classified faces by finding the norms of their temple curves and used the differences between the actual face profile and the norms for identification. At the end of the last century, computers developed rapidly and their image processing capabilities were significantly improved, and people have since really moved into automatic machine recognition. This was followed by the emergence of a large number of new technologies and methods. A search for "face recognition" on the EI (Engineering Index) Compendex (http://www.ei.org) reveals that the number of face recognition-related literature is growing at an EI (Engineering Index) Compendex The growth in the amount of face recognition-related literature on EI (Engineering Index) Compendex is phenomenal. The change in the volume of EI Compendex literature from 1980 to 2014 is shown in Figure 2.1.

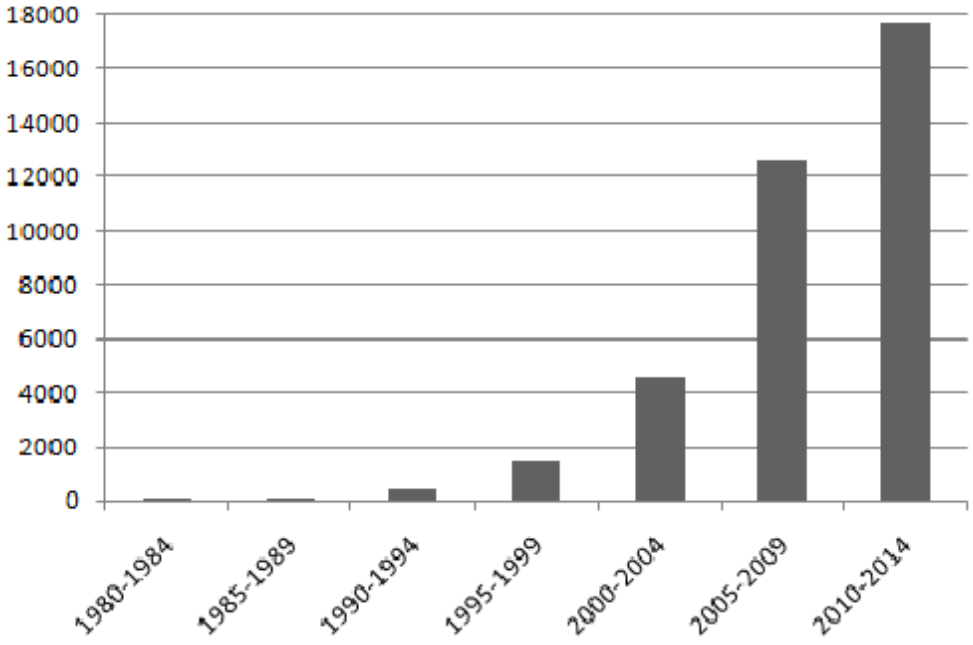


Figure2.1

(The data used for the statistics are from (http://www.ei.org))

Nowadays, professionals are working on face recognition technology in countries all over the world.

For example, Stone Z of Harvard University has proposed a large-scale face recognition method based on social contexts(Stone, Zickler &Darrell,2010)

Professor Mike of the University of Glasgow and Bruce of the University of Stirling The research group led by Professors Mike at the University of Glasgow and Bruce at the University of Stirling focuses on the role of the brain in face recognition and face perception. (Schweinberger & Burton, 2003)

Institute of Automation, Chinese Academy of Sciences. A research group led by researcher Tieniu Tan proposed a method to fuse face features with other biometric features, which substantially improved the effectiveness of single biometric recognition (Huang, Ren & Tan，2014). The Biometric Recognition and Security Technology Research Centre A research group led by Professor Li

Ziqing has obtained remarkable results in the research of near-infrared face recognition and medium-and long-range face recognition, effectively reducing the impact of light on face recognition performance (Zhao et al., 2002) (Lin et al., 2015).

A group led by Professor Jing-Yu Yang from the School of Computer Science, Nanjing University of Technology, has proposed a new perspective on face recognition based on algebraic features. He introduced the Singular Value Decomposition (SVD) method for face recognition(Wang, Chen & Yang, 2005). He investigated the problem of Fisher's linear discriminant analysis and the sensitivity of features to perturbations, proposed that the discriminant vectors may not be orthogonal to each other, gave the concept of a balanced distribution matrix, and constructed a new linear discriminant criterion.

Professor Xu Guangyou from the Department of Computer Science and Technology of Tsinghua University and Professor Zhang Changshui from the Department of Automation, whose respective groups proposed a nonlinear multi-pose face recognition method(Liang et al., 2001) (Yuan & Zhang, 2000).

Professor Zhihua Zhou of the Department of Computer Science and Technology at Nanjing University(Zhou et al., 2001), whose group led by him proposed single-person, single-training-sample face recognition.

In addition, some of the major foreign units specialising in face recognition research include A4Vison, Human Scan, Identix, Neuro Technology, and others.

Some face recognition research results have evolved from laboratory prototype systems to professional-grade commercial face recognition systems that can be produced in series.

Since August 2014, Japan has restarted experiments with face recognition systems at the immigration control offices at Haneda and Narita airports, targeting Japanese people travelling on flights at both airports. The Japanese government had previously installed automatic border gates at airports that allowed people to pass through security checks with fingerprint recognition

alone, but the utilisation rate was not high because of the need to register fingerprints in advance, whereas face recognition does not require prior registration.

The Hannover IT Fair (CeBIT) opened in Germany on March 15, 2015. Jack Ma, the founder of Alibaba Group, gave a keynote speech at the opening of the conference as the only invited entrepreneurial representative. After the speech, Jack Ma also demonstrated Ant Financial's Smile to Pay face scanning technology on the spot for German Chancellor Angela Merkel and Chinese Vice Premier Ma Kai, and personally swiped his face to buy gifts for the guests.

2.2 Existing Tools

Three different approaches to face recognition research

-Methods based on a feature description of the whole.

-Processes based on the study of local features.

-Combining local aspects with overall features.

(1) A comprehensive, feature-based approach

In addition to retaining the topological relationships of the face components that characterize the global information of the face, the holistic feature description approach also maintains some of the features of the face components. This method considers the overall qualities of the pattern. The primary techniques include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), (Ahmed, Osman & Nandi, 2019) and 2-D Principal Component Analysis (2DPCA), among others.

-PCA

The most representative method of principal component analysis (PCA) is the Eigenface method, proposed by Turk M and Pentland A in 1991 (Turk & Pentland, 1991). The principle of PCA is to extract the main components of a face using the K-L transform. The covariance matrix of the face images in the

training set is used to decompose the features to obtain the corresponding eigenvectors, which are used to represent the feature faces, thus forming the feature face subspace. Each face can be represented as a linear combination of these "feature faces". A set of projection coefficients is obtained by projecting the face image to be recognised into this space, and the final result is obtained by comparing it with the correlation coefficients of existing faces.

This is a theoretical drawback of the feature-face approach, as it takes into account the differences in the face image as a whole, but it is not known whether the differences are external (e.g., background, lighting, hairstyle, etc.) or internal to the face itself. In addition, the literature (Grudin, 1997) points out that the distribution of face space approximates a Gaussian distribution, and the distribution of ordinary faces generally lies around the mean, with individual special faces lying at the edge of the distribution, making the method unsuitable for practical recognition. To address the two major drawbacks mentioned above, researchers have also proposed some improved methods(Belhumeur, Hespanha & Hespanha, 1997) (Liu & Wechsler, 1998) (Moghaddam & Pentland, 1997)

-LDA

In order to aggregate samples from the same class and maintain as much separation as possible between samples from different classes, the Linear Discriminant Analysis (LDA) method seeks to extract the most discriminative low-dimensional characteristics from a high-dimensional feature space. The method is based on Fisher's criterion, so the problem of finding enough training samples to guarantee the invertibility of the intra-class scattering matrix, i.e., the small sample problem, arises due to the high dimensionality of the image vector (Park, 2008). Immediately afterwards, some solutions have also emerged, such as Fisherfaces (Belhumeur, Hespanha & Hespanha, 1997), Direct LDA, DLDA (Yu & Yang, 2001), Regularized LDA, RLDA(Guo, Hastie & Tibshirani, 2007) (Sharma & Paliwal, 2010), etc.

(2) Analysis based on local characteristics

By removing local contour information and grayscale information from faces, local feature analysis, which consists of a feature vector with the relative ratio of face data points and other shape or category parameters characterizing face features, is used to create particular recognition algorithms. The three primary techniques are Gabor wavelets, Hidden Markov Models, and Local Binary Patterns (LBP) (HMM).

-LBP

Local Binary Patterns (LBP), which extract local texture features of a face as a basis for recognition judgment, are a type of pattern recognition algorithm. The LBP algorithm, proposed in 1996 by T. Ojala et al. of Oulu University, Finland (Ojala, Pietikainen & Harwood, 1996), is a non-parametric operator that describes the local spatial structure of an image and is used to analyze image texture features, with strong discriminatory power in texture classification. The significant advantage of this method is that it is less sensitive to illumination but still does not address the issue of pose and expression. However, compared to the feature-face method, the recognition rate of the method has improved considerably. According to the literature (Ahonen, Hadid & Pietikainen, 1996), LBP has achieved recognition rates of over 90% for several face libraries.

-Gabor wavelet-based.

This method has been one of the research hotspots in the field of face recognition in recent years, and there are three major advantages.

- Gabor wavelets (Lee,1996) (Cao et al., 2006)have excellent properties in extracting local spatial information and frequency domain information of the target. They are remarkably close to the visual stimulus–response of simple cells in the human visual system.

- Gabor wavelets can offer good orientation selection properties and scale selection properties since they are sensitive to an image's edges.

- The sensitivity and adaptability of Gabor wavelets to changes in light are both good.

-HMM

The Hidden Markov Model (HMM) is a probabilistic model with a parametric representation used to describe the statistical properties of a stochastic process, which is a dual stochastic process consisting of a Markov chain and a general stochastic process. (Samaria & Young, 1994) Samaria F and Young S developed a Markov model for the human face. The hair, forehead, eyes, nose, and mouth represent five different states in a certain order and are described by a set of numerical features.

The HMM-based face recognition method allows for partial expression changes and more pronounced head rotation and has the advantage of good scalability and high recognition rates. However, the method is computationally complex and very difficult to implement.

(3) Integration of the global and the local

The main disadvantage of the method based on overall feature description is that it cannot solve the problem of the high dimensionality of image feature data and is easily affected by the environment. The main disadvantage of the method based on local feature analysis is that local features are more sensitive to lighting, expression, pose, and occlusion, so the recognition stability is not high. Therefore, methods that combine the whole with the local (Lanitis, Taylor & Cootes, 1995) (Penev & Atick, 1996)have been proposed and are gradually gaining attention. One of the typical methods is the Elastic Bunch Graph Matching (EBGM) method (Wiskott et al., 1997).

The Elastic Bunch Graph Matching (EBGM) technique uses the Gabor wavelet transform to separate facial features from a face's overall features and local features. With the lattice 2D architecture shown in Figure 2.2, it depicts the face. The Gabor wavelet decomposition of the picture location yields a feature vector that labels each vertex in the topology and contains details about the vertex and

its surroundings. Figure 2.3 depicts the feature vector expressing the facial traits, and each edge on the topological map is annotated with the distance between the vertices (Luo,2004). When matching face photos, the most similar topological graph to the target image is first identified, and then each vertex location in the graph is best matched. As a result, a deformed graph is created whose vertices roughly correspond to the topological graph's corresponding vertices.

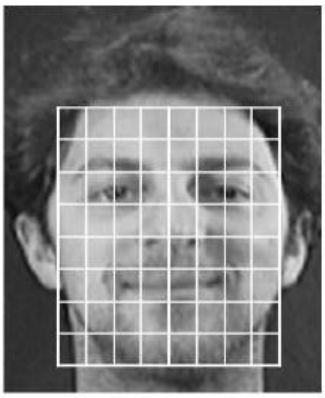


Figure 2.2 Lattice 2D topology defined on a human face（Luo,2004）

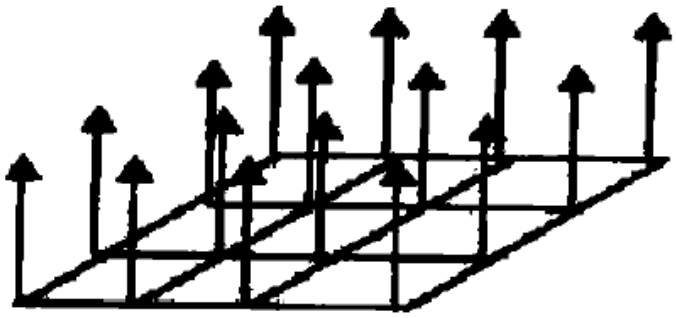


Figure 2.3 Face feature vector（Luo,2004）

This method exceeds the feature face method because it uses Gabor wavelets, which are superior to the feature face method since they are invariant to light, angle, size, and other factors. However, each training sample's 2D topology must be determined throughout the feature extraction process. Additionally, new topology data must be updated whenever a new face is added to the face

library. As a result, the calculation and storage of the facial data grow significantly.

2.3 Existing Applications.

The most prevalent use of facial recognition in daily life is the attendance system. To suit the demands of some organizations or units to determine the attendance status of users, a variety of attendance systems have been developed. The traditional paper sign-in and punch clock attendance techniques have been replaced with RF card attendance systems, fingerprint attendance systems, facial recognition attendance systems, mobile APP (Application) location attendance systems, etc. as the primary ways of attendance.

The biometric fingerprint and face recognition attendance systems have no consumables, no lost or missing cards, and no equipment requirements for users compared to the other two attendance methods. With fingerprint recognition, there are disadvantages such as 5% of the population being born without fingerprint recognition; finger wear and tear having a large impact on recognition; and direct contact being unhygienic. The mobile phone APP location attendance system, which requires a certain network coverage, has higher requirements for the intelligence of the attendance user's mobile phone and also has the loophole of being able to replace it with others. Therefore, at this stage, for small and medium-sized institutions or units, the face recognition attendance method still has the advantages of convenience and high-cost performance and is also the main direction of attendance system development.

2.4 Improvements and difficulties

The challenges of face recognition arise mainly from image acquisition and image recognition techniques.

Face recognition systems generally require high-quality image acquisition during the day and night or under various natural light conditions. Therefore, environmental changes are the biggest challenge affecting the quality of image capture and, therefore, the performance of the whole face recognition

attendance system. Some researchers have tried to solve this problem with 3D face recognition technology and thermal imaging technology, but these two technologies are still very immature and there are mainly the following problems to be solved.

1) The required image acquisition equipment is relatively expensive and it is difficult to achieve the desired usage rate.

2) The acquired data is not compatible with visible image data This limits the development of the two technologies.

Near-infrared (NIR) image capture technology was proposed and rapidly developed. When a face is facing the camera at a distance of 0.5m to 1m, the NIR active light source located around the camera illuminates the face in the frontal direction and prevents light from the visible band from passing through the filter, so that the captured image can be used as the NIR image for the next step. The technique thus improves the effect of external light on the face image.

Another problem with image acquisition is the effect of image noise. This means removing image blur and bokeh noise, or removing the effect of noise at the image acquisition stage. This is still a difficult problem that cannot be completely overcome.

The system is influenced by several human factors that relate to face recognition technology, in addition to the impact of noise in the image itself as a result of image capture.

1) Face images that contain complex backgrounds can easily lead to inaccurate face localization or failure to detect faces.

2) Substantial changes in facial pose can affect the accuracy of face detection and localization.

3) Changes in facial make-up, such as injuries, female make-up, male beard growth, etc., can increase recognition difficulties.

4) Hairstyles, glasses, scarves, and other accessories can obscure one's face.

Face recognition is considered to be a challenging class of topics involving multiple disciplinary areas, mainly because the face itself is a non-rigid object with highly variable structural details. While existing 2D face recognition methods have achieved some success, more face recognition methods with strong robustness are needed to cope with the many factors. (Li & Hu, 2005) The research in this paper will focus on face recognition techniques.

3 Approach

3.1 Overview

This section covers the project methods and the methods used to create the tools. Methods for face detection, face feature value extraction, and face feature value retrieval will be listed.

3.2 Face Detection

3.2.1 MTCNN

-Introduction

MTCNN is a multi-task convolutional neural network.

It implements both face detection and keypoint recognition, which is also called face alignment.

Detection and alignment are the basis for many other facial applications, such as face recognition and expression recognition.

Features of the network

1. cascade network
2. online hard sample selection online hard sample dining
3. Fast, real-time detection

The main architecture of MTCNN is a cascade of 3 progressive networks, called P-Net, R-Net, and O-Net, where P-Net quickly generates rough candidate

frames, R-Net performs filtering to obtain high-precision candidate frames, and O-Net generates bounding frames and key point coordinates.

-Principle

Looking at Figure 3.8, it is clear that mtcnn does four steps to process the incoming images.

The first major step is resizing, in which the data is only preprocessed and not yet passed into the neural network. It will resize the original image to a smaller and smaller size, with a minimum edge length of 12.

The second step is Stage 1, P-Net. The various sized images from the first resizing step are passed into the convolutional neural network of the P-Net. The neural network outputs two parameters. One is the confidence level of the white box and the other is the regression coefficient information. The overall network looks like this (Figure 3.9).

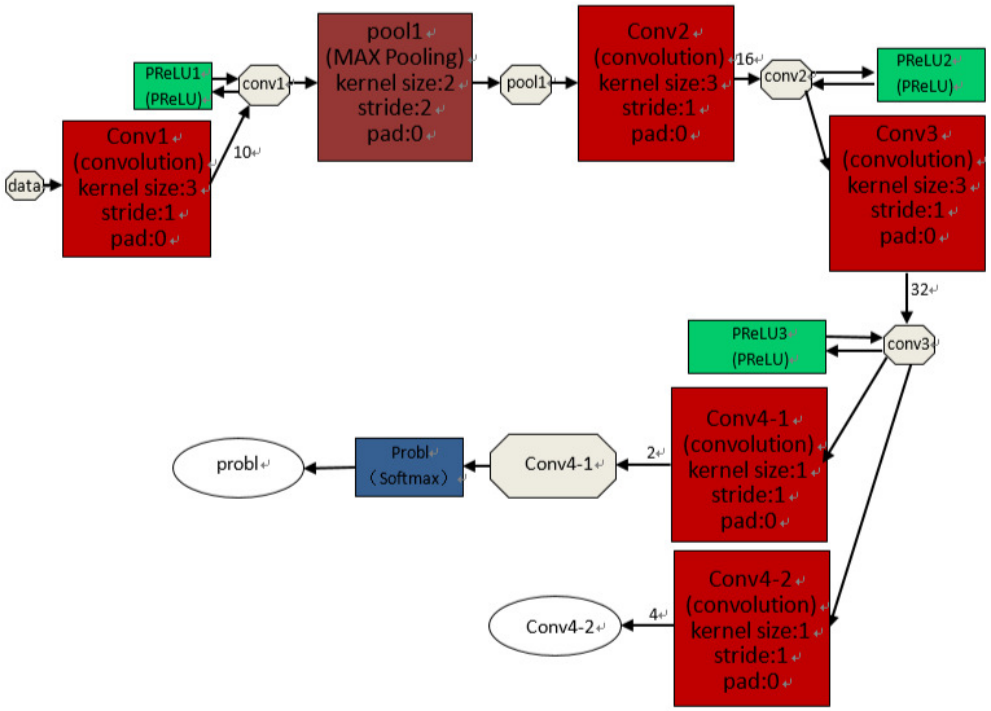


Figure 3.9

The output of conv4-2 is the regression coefficients and prob1 is the confidence information and confidence level (probability of whether it is a face or not) of the box. This means that the image after each reshape is passed into this network, and conv4-2 and prob1 are outputted afterwards.

For the reason that the minimum edge length is 12. If the edge length is 12, then the input network is 3x3x16 in dimension by the conv2 layer. This will then have another 3x3 convolution kernel of 16 channels by conv3, which is the minimum edge length to convolve at this layer. If the edge length is any smaller, the network will not work.

The P-Net is a fully convolutional layer and does not involve a fully connected layer, so the size of our input image can be fluid. For P-Net, the input image is an image pyramid, i.e., a single image. The image is reduced according to different factors. The variable size here means that different images from the image pyramid can be used as input.

The P-Net network is a 12*12 sized box to detect if a face is present. Assuming an input image size of 100*120, the smallest acceptable face image size is 20*20 and the largest is 100*100.

If the smallest 20*20 can be scaled down to be recognised by P-Net, Set the first scaling factor to 12/20 = 0.6. After that, set the scaling factor of the adjacent layers of the image pyramid to the same 0.7.

The classification outcome, which determines whether a face is included or not, is the first one. The regression outcome, which yields four offsets, is another. These two findings are created concurrently during the inference step. Processing of the classification result begins with values larger than a threshold (this value is artificially specified and a face is considered to be included when it is greater than this value).

The next step is to turn the rectangular frame into a square, an operation called "rec2square. Imagine a picture where the face frame is a long, thin shape and needs to be turned into a square with a larger width than the width of the picture

would otherwise be. This is where an operation similar to padding is needed. So to summarise, after the box is corrected, rec2square is performed, and then padding is performed. Finally, the boxes are resized to 24x24x3 and are ready to be sent to the next R-Net network. At this point, the P-Net is complete.

In the third step, the images of the boxes filtered by P-Net are fed to R-Net. The candidate boxes obtained from P-Net are resized, and each candidate box is classified and regressed. Then the convolutional kernel is fully concatenated. The final network will have two outputs, one with the confidence information from prob1 and the other with the regression coefficient information from conv5-2. An NMS (i.e., non-maximal suppression) is performed on the candidate frames. This is used in the detection algorithm to eliminate a large number of overlapped boxes while retaining the highest quality boxes. The regression results are then combined with the regression adjustment. Finally, the boxes were resized to 48x48x3 and prepared for the next layer of O-Net. The network structure of R-Net is posted below (Figure 3.10).

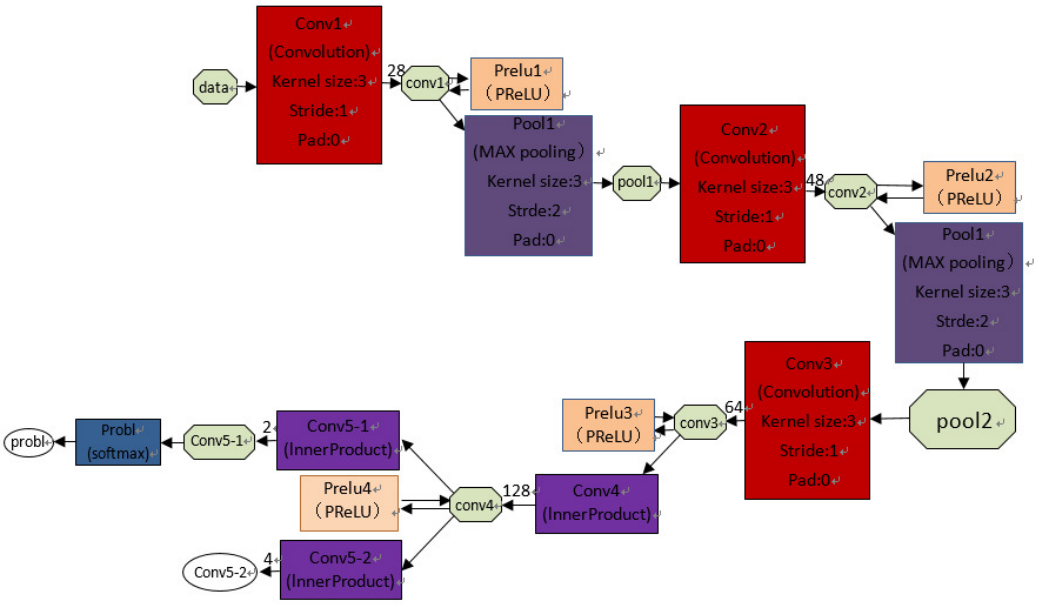


Figure 3.10

Step 4. The candidate boxes of R-Net are resized. o-Net will have three outputs: key point positions, regression results, and classification results. That is, confidence information of the prob1 layer, key point information of the conv6-3 layer, and regression coefficient information of the conv6-2 layer. The extra keypoint information, which is what MTCNN finally implements, is the detection of the key points of the face. The O-Net network structure is posted below.(Figure 3.11)

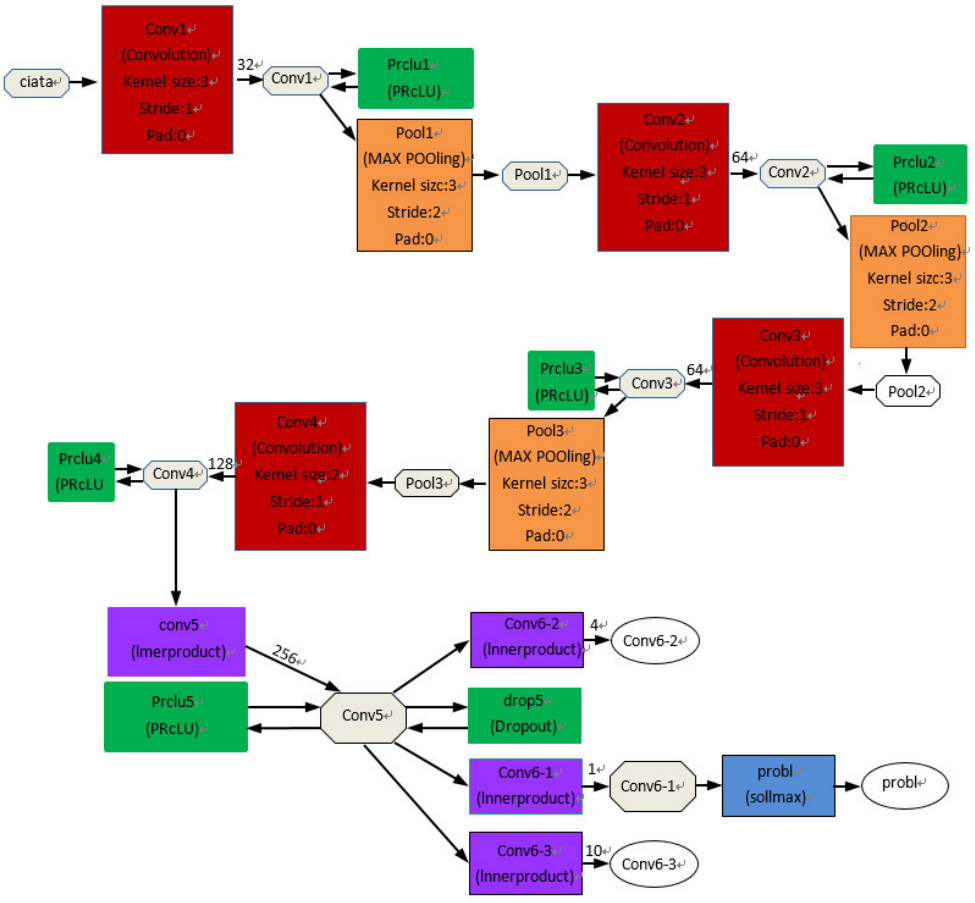


Figure3.11

-Summary

The MTCNN process is divided into the following steps:

-In order to obtain the image pyramid, the test image is first continually scaled.
-To obtain a large number of candidates, the image pyramid is entered into Pnet (candidate).
-The candidate images are filtered by Pnet and refined by Rnet.
-Many candidate images filtered by Rnets are fed into Onet, which outputs the exact box and landmark coordinates.

The process of face detection is complete. With the output of O-Net, face boxes and face keypoint information can be drawn using tools such as OpenCV.

3.3 Face Feature Extraction

3.3.1 InsightFace (ArcFace)

InsightFace is a toolkit related to face detection and face recognition. Its upper layer interface can directly do face detection, gender determination, etc. The underlying implementation is trained models of various kinds. The main algorithm of InsightFace is described below. It starts by using a ResNet100 convolutional network, followed by a fully connected layer. It is trained using a modified loss function based on Softmax called ArcFace.

-Softmax and SphereFace

ArcFace is optimised based on other algorithms. So to understand ArcFace, Softmax and SphereFace need to be understood first.

The most widely used loss function for classification is the softmax loss(1).

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} \qquad (1)$$

$x_i \in R_d$ denotes the depth feature of the i-th2 sample, which belongs to a class y_i.

$W_j \in R_d$ indicates the jth column of weights, W $\in R_d$×n and $b_j \in R_n$ is the bias term. The batch size and number of classes are N and n. Softmax loss has traditionally been widely used for deep face recognition (Cao et al., 2018) (Parkhi, Vedaldi & Andrew, 2015). However, the softmax loss function does not explicitly optimise feature embeddings to enhance the similarity of within-class samples and the diversity of between-class samples. This leads to performance gaps in deep face recognition for large intra-class appearance variations (e.g. pose variations (Sengupta et al., 2016) (Zheng & Deng,2018) and age gaps (Zheng, Deng & Hu, 2017) (Moschoglou et al., 2017) and large-scale test scenarios (e.g. millions (Maze et al., 2018) (Whitelam et al., 2017) or trillions of pairs (Shlizerman et al.2016).

The bias $b_j = 0$ in (Zhang et al., 2017) was corrected for ease of representation. logit (Zhang et al., 2018) was then converted to $W_j^T x_i = \|W_j\|\|x_i\|\cos\theta_j$, where θ_j is the angle between weight W_j and feature The individual weights $\|W_j\| = 1$ were fixed by realisation according to (Zhang et al., 2017) (Wang et al., 2018) (Wang et al., 2017). According to (Wang et al., 2018) (Wang et al., 2017) (Wang et al., 2018) (Ranjan, Castillo & Chellappa,2017), the embedded features are fixed by l_2 nonormalizing them $\|x_i\|$ and rescaling them. As a result of the characteristics and weights being normalized, the prediction now just depends on the angle between the two. The learned embedding features are thereby dispersed across a hypersphere of radius. This is the current loss expression.

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}} \quad (2)$$

To improve intra-class compactness and inter-class variance, an additional angular margin is placed between them since the embedded features are dispersed around each hypersphere feature. The geodesic distance margin penalty in the normalized hypersphere is the same as the proposed additive angular margin penalty. ArcFace is the name of the method, and the current loss expression is as follows.

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}} \quad (3)$$

To train distinct 2D feature embedding networks with softmax and ArcFace losses, face photos with appropriate samples (about 1500 images/class) were chosen from 8 different identity face images. Figure 3.1 illustrates how the softmax loss produces roughly separable feature embeddings but leaves the decision boundary with a glaring uncertainty. On the other hand, it is evident that the ArcFace loss results in a more pronounced discrepancy between the closest classes (class spacing is significantly greater).

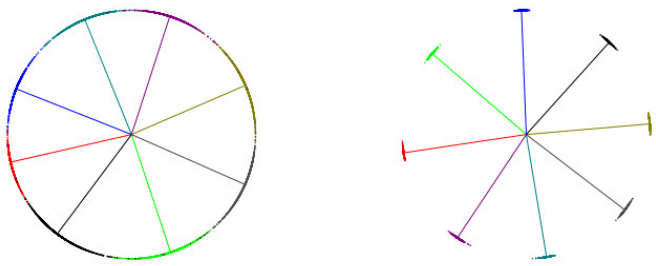


Figure3.1 Softmax and Arcface

(Deng, et al. 2018)

-SphereFace and Coface

Numerical similarity in SphereFace (Zhang et al., 2017) (Liu et al., 2016), ArcFace, and CosFace (Wang et al., 2018). Three different margin penalties are proposed, for example, multiplicative angular margin m_1, additive angular margin m_2, and additive cosine margin m_3. From the perspective of numerical analysis, the different margin penalties all need to enhance intra-class compactness and inter-class diversity by penalising the target logit (Zhang et al., 2018). The plotted target Logit curves for the SphereFace, ArcFace, and CosFace at the ideal margin values are displayed in Figure 3.3. As shown in Figure 3.2, these target logit curves are only shown in the range [20°, 100°], as the angle between W_{y_j} and x_i during ArcFace training starts at approximately 90° (random initialisation) and ends at approximately 30°. The starting point, the ending point, and the slope are, intuitively, the three variables in the target logit curve that influence performance. SphereFace, ArcFace, and CosFace are implemented in a unified framework with .m_1, m_2 and m_3 as parameters by integrating all margin penalties

$$L_4 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(m_1\theta_{y_i}+m_2)-m_3)}}{e^{s(\cos(m_1\theta_{y_i}+m_2)-m_3)}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}} \quad (4)$$

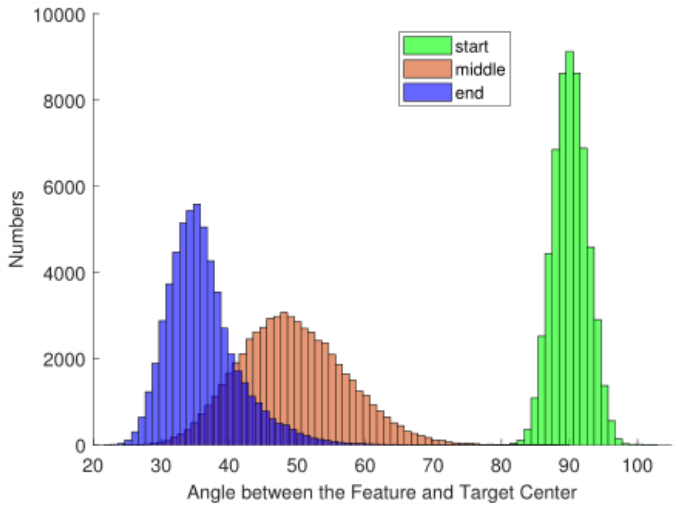


Figure 3.2 θ_j Distributions

(Deng, et al. 2018)

3.3.2 FaceNet

Deep convolutional networks are used in FaceNet. Two different core architectures will be discussed: a Zeiler & Fergus (Zeiler & Fergus, 2013) style network and an Inception type network. The details of these networks are described in Section 3.3. (Szegedy et al., 2014)

Assume that the model is a black box (Figure 3.3). The use of triadic loss is made possible by the fact that it provides direct access to the outcomes of face verification, recognition, and clustering. Specifically, the number of squared distances between pairs of faces from face images with different identities is high whereas the number of squared distances between all faces with the same identity is modest (regardless of the imaging settings) when f(x) from the image x is embedded into the feature space Rd.

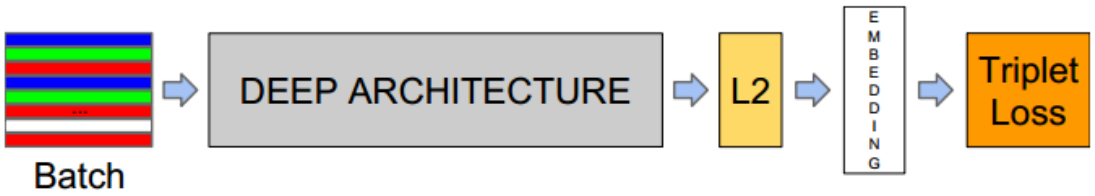


Figure 3.3

(Schroff, Kalenichenko & Philbin, 2015)

Model the structure and composition of the network. Deep CNN with batch input layer—Perform L2 normalization—Implement face embedding (face embedding)—Triplet loss function during training.

Ternary loss attempts to strengthen the margin between each pair of faces from one person to all other faces. This allows different faces of the same identity to rely on the same copy. It also strengthens the distance between faces and thus differentiates between other identities. (Ahmed, Osman & Nandi, 2019)

The next section describes this triadic loss and how it can be learned effectively in large-scale situations.

-Triplet loss

Triplet loss means that three images need to be viewed at the same time. An anchor, positive or negative,

(Anchor and Positive refer to the same person, while Anchor and Negative refer to a picture of a different person) (Figure 3.4)

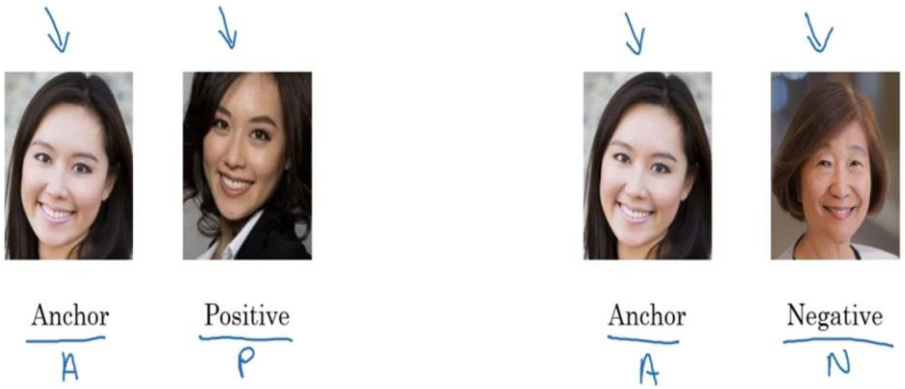


Figure 3.4

When comparing anchor images with positive images, the smaller the output distance, the better.

When comparing Anchor with Negative, the greater the output distance, the better.

Embedding is represented by $f(x) \in \mathbb{R}^d$. It embeds the image x into a d-dimensional Euclidean space. Furthermore, this embedding is restricted to the d-dimensional hypersphere, i.e. $\|f(x)\|_2 = 1$. This loss is induced in the context of nearest neighbor classification in (Weinberger, Blitzer & Saul,2006). Here, it is important to ensure that a particular person's image x_i^a (anchor) is closer to all other images of the same person x_i^p (positive) than any other person's image x_i^n (negative) (Figure 3.5).

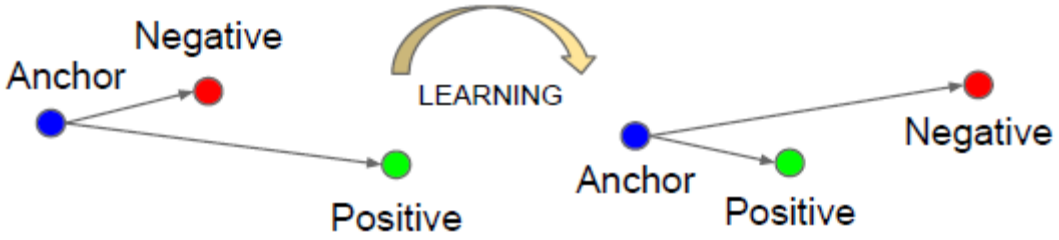


Figure 3.5

(Schroff, Kalenichenko & Philbin, 2015)

-Advantages

Triplet loss based neural network models can distinguish details very well, especially in image classification tasks. When two inputs are similar, triplet loss can learn a better representation of the two less dissimilar input vectors and thus perform well in the classification task.

Triplet loss typically learns superior fine-grained feature features during training compared to other classification loss functions, and more precisely, triplet loss is able to define specific thresholds based on the demands of the model training.

In order to manage the distance between positive and negative samples, the designer might alter the threshold margin used to train the network structure using triplet loss.

-Disadvantages

Triplet loss is efficient, but it also has drawbacks, including late convergence and a higher propensity for overfitting than classification loss. This is because choosing triples produces data that isn't always uniformly distributed, which makes the model's performance during training more unstable.

This method will therefore typically be applied during the model's pre-training phase. Alternatively, it is combined with the softmax function (classification loss).

—Classification

In image tasks, triplet loss is often used as a distance vector representation of sample points in Euclidean space, which can greatly improve the discriminative power of deep features. In order to make training more rapid and efficient, care needs to be taken to design a training strategy that selects only the appropriate

triplet for training. Since there are so many triplet combinations in a dataset, training all of them will greatly reduce the efficiency.

Based on our definition of triplet loss, in general, triplets will be classified into three categorie.

1. Easy triplets: refers to the triplet in the case of untrained, triplet loss value has been 0, when the network does not need training to learn to meet the requirements of the loss function.

2. Semi-hard triplets: means that the distance between negative samples and the base sample is greater than the distance between positive samples and the base sample. However, the Triplet loss value has not yet reached 0. At this point, the network can continuously reduce the loss value through proper learning.

3. Hard triplets: refers to the distance between negative samples and benchmark samples being smaller than the distance between positive samples and benchmark samples, which is the most difficult sample group for the network to learn when the loss value will have a large oscillation.

Since the loss value of a simple triplet is 0, if the training network can only learn simpletriplets, it will lead to a limitation of the network's generalisation ability and the network will not learn any features that are not helpful for the training task. Therefore, when a training batch contains a large number of easy triplets, it will lead to a much lower convergence rate of the training network.

The general triplet is ideal for pre-training the network, helping it to converge and providing a large amount of valid statistical information at the same time.

Difficult triples are good for learning in the later stages of training and can help improve the performance of the network. By allowing the network to learn from difficult sample features, it can greatly improve the classification ability of the network, especially for difficult samples.

-Triplet Selection

How to effectively perform difficult sample mining becomes the key. Difficult samples should include samples with large distances in the same category (matching samples) and samples with small distances in the non-category (non-matching samples).

And it is not easy to use triplet loss with a large number of images. Accordingly, it is necessary to optimize the current strategy for large-scale circumstances. The use of triplets can lead to a surge in the amount of data and the number of combinable triples, which makes it impractical or extremely inefficient to traverse all combinations.

The first is to convert triplet loss to softmax loss (the triplet net article combines the output of triplet loss with softmax and mse); the other is batch OHNM; but the second approach to mining samples is to directly consider a batch in all sample spaces. If there is a batch that does not satisfy margin, it is considered a difficult sample. However, this does not guarantee that a lot of very similar samples are selected at the time of sampling, and the (a,p,n) image set is fixed once it is selected and no other combinations will be made, so it is less efficient. So finding similar individuals (categories) is the core key to improving the triplet net.

Experiments and comparisons of the above two algorithms are presented in the next section.

3.4 Face Feature Retrieval

3.4.1 Overview

Face retrieval refers to the comparison of an input face with a face in a database. It is an identity verification technique used to determine whether an input face is attributed to a specific person.

After obtaining the face feature vectors, the Euclidean distances between the input vectors and the other feature vectors in the face database are calculated. The calculated distances are then threshold filtered and sorted, and the filtered features are likely to be images of the same face.

This section will introduce Milvus, which is a method of face-feature invocation. The next section will perform an experimental comparison with a Python loop to verify the detection speed. Choose a more efficient face retrieval method.

3.4.2 Vector similarity

Similarity retrieval is the process of comparing a target object with data in a database and recalling the most similar result. Similarly, vector similarity retrieval returns the most similar vector data.

The Approximate Nearest Neighbor Search (ANN) algorithm can calculate the distance between vectors, thus increasing the speed of vector similarity retrieval. If two vectors are very similar, it means that the source data they represent is also very similar.

3.4.3 Milvus

Milvus is a cloud-native vector database. It is highly available, high-performance, and easily scalable for real-time recall of massive vector data.

Milvus is built on vector search libraries such as FAISS, Annoy, HNSW, etc. The core of Milvus is to solve the problem of dense vector similarity retrieval. Based on the vector search library, Milvus supports data partitioning and slicing, data persistence, incremental data ingestion, scalar-vector hybrid queries, time travel, and other functions. The performance of vector retrieval is also significantly optimised to meet the needs of any vector retrieval scenario.

Milvus uses a shared storage architecture with complete separation of storage computation and horizontal scaling of compute nodes. Milvus follows the separation of data flow and control flow, and is divided into four layers. These are the access layer, the coordinator service, the worker node, and the storage layer. Each layer is independent of the other.

-Access Layer: consists of a set of stateless proxies. It provides the endpoint for external user connections, validates client requests and consolidates the returned results.

-Co-ordinator Service: The brain of the system. It is responsible for assigning tasks to execution nodes. The coordinator service has four roles: root coord, data coord, query coord, and index coord.

-Worker Node: The limbs of the system. They are responsible for completing the commands issued by the coordination service and the Data Manipulation Language (DML) commands initiated by the proxy. There are three roles for worker nodes: data node, query node, and index node.

-Storage service: the skeleton of the system. It is responsible for the persistence of Milvus data and is divided into three parts: meta store, log broker, and object storage.

4 Implementation

4.1 Overview

The following script will be used to calculate the face search accuracy for Insightface and FaceNet in the first part of this section.

code/face_detec_reco_insightface/face_search_accuracy.py

code/face_detec_reco_facenet/face_search_accuracy.py

After the results are obtained, they are analysed by comparison. The aim is to select a face feature extraction method with a high accuracy rate to be used in future systems.

Later in this section, the efficiency of the two feature-based retrieval methods will be experimented with.

4.2 Preparation for the experiment

4.2.1 Datasets Download

The dataset was selected from the following links.

http://www.cs.columbia.edu/CAVE/databases/pubfig/download/

PubFig: A database of the faces of public figures

The PubFig collection contains 58,797 photos of 200 real-world faces that were downloaded from the Internet. These pictures were collected under totally uncontrolled conditions, and the individuals weren't very cooperative, unlike the majority of other face datasets that are already available.

In the development dataset, there are over 16,000 images of 60 people. The eval dataset, on the other hand, contains 140 individuals and approximately 42,000 photographs. The eval dataset has a much larger number of people and a much richer amount of data. Therefore, this study uses the 42,461 photo records in the eval_url.txt in this dataset as the base data. Figures 4.1 and 4.2 show screenshots of the dataset.

4.2.2 Data cleaning

There will be a great deal of irrelevant data without effective data cleansing. There will be challenges in the way of future data development, application, management, etc. Data cleaning is the last stage in identifying and resolving observable faults in data files, including dealing with incorrect numbers, missing information, etc.

A data warehouse is a place where subject-specific data, including historical data, is collected from various systems. As a result, some of the data is unavoidably inaccurate. Some of the data are incompatible with one another, and it is evident that inaccurate or inconsistent data should not be used.-Store face data in PostgreSQL

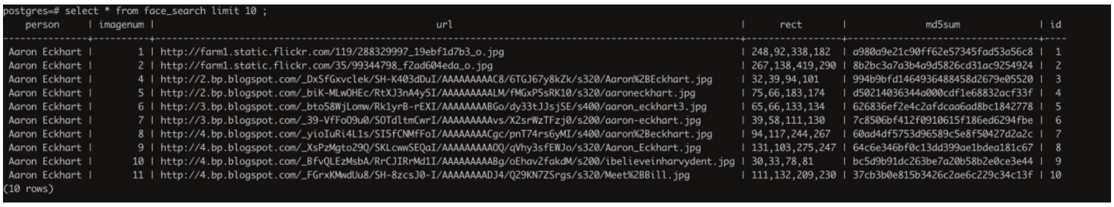
This is why PostgreSQL was chosen. Since PostgreSQL responds faster than the often used MySQL, facial recognition is made easier. Facial recognition systems also rely on its dependability and data integrity. It also allows the usage of Python and has improved SQL programming capabilities.

eval_url.txt contains 5 fields: person, imagenum, url, rect, and md5sum. Ideal for storage management using a relational database.

Firstly, remove the first two lines from eval_url.txt. A final data file for importing into PostgreSQL is generated and named face_data.txt.

The next step is to start PostgreSQL, build the table, and import the data.

After storage in PostgreSQL, the data file is shown in Figure 4.3.



```
postgres=# select * from face_search limit 10 ;
    person     | imagenum |                                              url                                               |      rect       |              md5sum              | id
---------------+----------+------------------------------------------------------------------------------------------------+-----------------+----------------------------------+----
 Aaron Eckhart |        1 | http://farm1.static.flickr.com/119/288329997_19ebf1d7b3_o.jpg                                  | 248,92,338,182  | a980a9e21c90ff62e57345fad53a56c8 |  1
 Aaron Eckhart |        2 | http://farm1.static.flickr.com/35/99344798_f2ad604eda_o.jpg                                    | 267,138,419,290 | 8b2bc3a7a3b4a9d5826cd31ac9254924 |  2
 Aaron Eckhart |        4 | http://2.bp.blogspot.com/_DxSfGxvclek/SH-K403dDuI/AAAAAAAAAC8/6TGJ67y8kZk/s320/Aaron%2BEckhart.jpg | 32,39,94,101    | 994b9bfd1464936488458d2679e05520 |  3
 Aaron Eckhart |        5 | http://2.bp.blogspot.com/_biK-MLwOHEc/RtXJ3nA4y5I/AAAAAAAAALM/fMGxP5sRK10/s320/aaroneckhart.jpg | 75,66,183,174   | d50214036344a000cdf1e68832acf33f |  4
 Aaron Eckhart |        6 | http://3.bp.blogspot.com/_bto58WjLomw/Rk1yrB-rEXI/AAAAAAAABGo/dy33tJJsjSE/s400/aaron_eckhart3.jpg | 65,66,133,134   | 626836ef2e4c2afdcaa6ad8bc1842778 |  5
 Aaron Eckhart |        7 | http://3.bp.blogspot.com/_39-VfFo09u0/SOTdltmCwrI/AAAAAAAAAvs/X2srWzTFzj0/s200/aaron-eckhart.jpg | 39,58,111,130   | 7c8506bf412f0910615f186ed6294fbe |  6
 Aaron Eckhart |        8 | http://4.bp.blogspot.com/_yioIuRi4L1s/SI5fCNMfFoI/AAAAAAAACgc/pnT74rs6yMI/s400/aaron%2Beckhart.jpg | 94,117,244,267  | 60ad4df5753d96589c5e8f50427d2a2c |  7
 Aaron Eckhart |        9 | http://4.bp.blogspot.com/_XsPzMgto29Q/SKLcwwSEQaI/AAAAAAAAAOQ/qVhy3sfEWJo/s320/Aaron_Eckhart.jpg | 131,103,275,247 | 64c6e346bf0c13dd399ae1bdea181c67 |  8
 Aaron Eckhart |       10 | http://4.bp.blogspot.com/_BfvQLEzMsbA/RrCJIRrMd1I/AAAAAAAAABg/oEhav2fakdM/s200/ibelieveinharvydent.jpg | 30,33,78,81     | bc5d9b91dc263be7a20b58b2e0ce3e44 |  9
 Aaron Eckhart |       11 | http://4.bp.blogspot.com/_FGrxKMwdUu8/SH-8zcsJ0-I/AAAAAAAADJ4/Q29KN7ZSrgs/s320/Meet%2BBill.jpg | 111,132,209,230 | 37cb3b0e815b3426c2ae6c229c34c13f | 10
(10 rows)
```

Figure4.3

-Detection of invalid images

Script file used: code/delete_useless_url. ipynb

In this step, the url of the invalid image is detected and the record is deleted from the postgres database. Invalid images are usually missing, duplicated, or incorrect data types. Figures 4.4 and 4.5 show examples of invalid URLs.

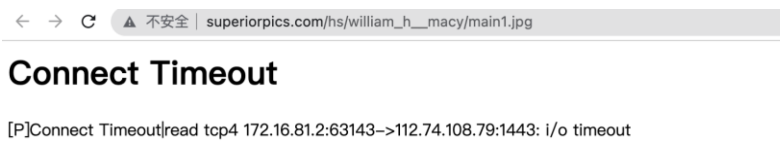


Figure 4.4

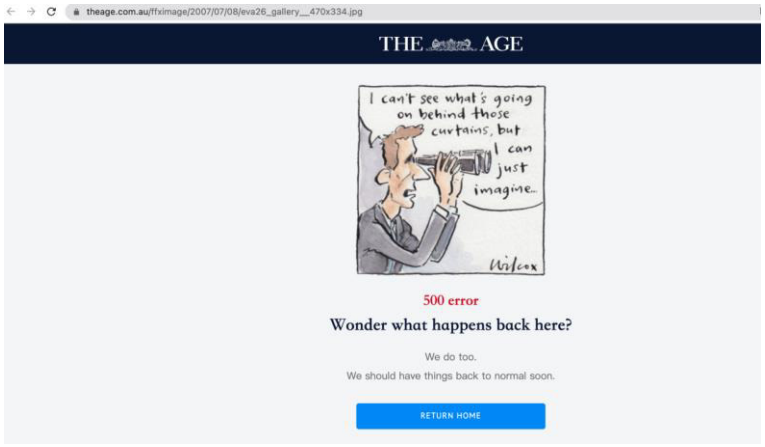


Figure 4.5

-Image download and deletion

First, use the script file: code/save_image.ipynb. Download all images with valid URLs.

After that, use the script file: code/clean_broken_img.ipynb

Delete the invalid (broken) images (invalid images, Figure 4.6).

After the above steps, the face image dataset for this experiment can be obtained and placed in this directory (Figure 4.7). code/image

Figure 4.6

Figure 4.7

-Summary

This section the preparation done before the experiment. The main content is data cleaning. Adequate data cleaning was carried out on the image data downloaded from the internet. Corrupted images, blurred images, and similar images were removed. The aim was to improve the efficiency and accuracy of the experiment.

4.3 Accuracy testing experiments

-Overview

This section will be oriented towards the analysis of images with feature values extracted by both algorithms, insightface and facenet, and the accuracy will be calculated. A total of six comparison experiments will be performed; each experiment will perform ten rounds of image detection. Each experiment will examine 100 images.

-Part of Insightface

To begin, run the following script:

code/face_detec_reco_insightface/gen_embed.py

Convert the face image detected by MTCNN into a vector using InsightFace.

A res folder will be created with the vectors of the faces extracted from each image, named faceEmbedding.npy

Create a file name.txt containing the name of the face image corresponding to the vector (the image name is the same as the image md5 in the previous postgreSQL) (Figure 4.8).

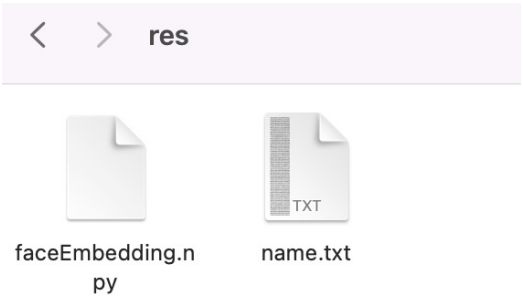


Figure4.8

Next, use the script: **code/face_detec_reco_insightface/cp_image.py** to copy the images from name.txt to a new folder: insight_reco_image

To verify the accuracy of the two face feature retrieval models. Only data containing a single face image will be selected for this study.

The directory of the set of images of single faces:

code/face_detec_reco_insightface/insight_reco_image (Figure 4.9).

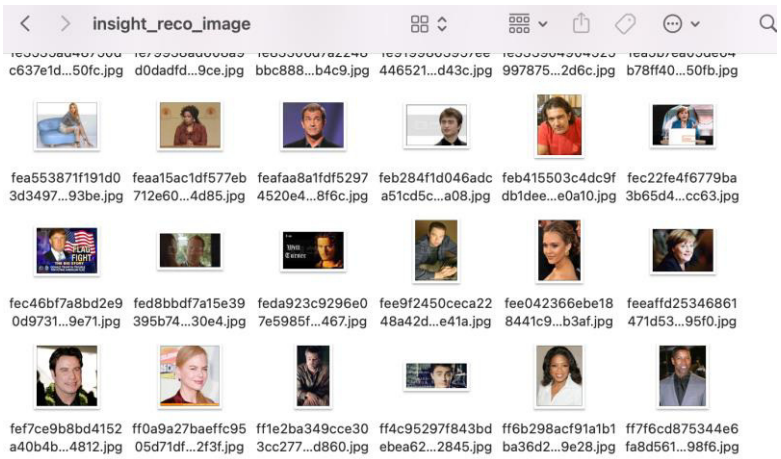


Figure 4.9

The third step will be to calculate, using the script:

To calculate the face search accuracy using InsightFace, use **code/face_detec_reco_insightface/face_search_accuracy.py.**

Experiment 1

Calculating the hit rate of InsightFacetop5 results

One image is randomly chosen from the insight_reco_image directory to calculate this. The base library's top 5 faces that were most similar to the target face were then located, and the hit ratio was determined as the proportion of those top 5 faces that belonged to the same individual as the target face.

Ten rounds of trials were carried out, with 10 faces chosen for each round and the top five most comparable faces being sought after in each. 100 faces will be searched cumulatively after 10 rounds of trials, which is a big enough sample size to satisfy the demand.The results of the programme

Rounds	0	1	2	3	4	5	6	7	8	9
Hit_ratio	0.98	0.98	0.98	1.0	1.0	0.98	1.0	1.0	1.0	1.0
Mean_hit_ratio	99.2%									

------Round 0

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.98.

------ Round 1

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.98.

------ Round 2

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.98.

------ Round 3

A random search of ten people for five similar photos was conducted. The mean accuracy is 1.0

------ Round 4

A random search of ten people for five similar photos was conducted. The mean accuracy is 1.0

------ Round 5

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.98.

------ Round 6

The average accuracy of a random search of 10 people for 5 similar photos is 1.0.

------ Round 7

The average accuracy of a random search of 10 people for 5 similar photos is 1.0.

------ Round 8

The average accuracy of a random search of 10 people for 5 similar photos is 1.0.

------ Round 9

The average accuracy of a random search of 10 people for 5 similar photos is 1.0.

Experiment 2

Calculating the hit rate of top10 using InsightFace

There will be ten rounds of experiments. The same 10 faces will be chosen for each round as before, and each round will look for the top 10 faces that are most similar. A total of 100 faces will be searched throughout the course of 10 rounds of trials, and the sample size will be adequate to match the demand.

The calculation's methodology. One target image will be chosen at random for this experiment from the insight_reco_image directory. Then, in the bottom collection, identify the top 10 images that are most similar to the target image. The hit ratio is calculated last. The hit ratio is the proportion of the target face that is shared by the top 10 comparable faces.The results of the programme

Round	0	1	2	3	4	5	6	7	8	9

Hit_ratio	0.96	0.87	1.0	0.99	0.97	0.99	0.97	1.0	1.0	0.89
Mean_hit_ratio	96.4%									

------Round 0

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.96

------ Round 1

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.87

------ Round 2

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 3

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.99

------ Round 4

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.97

------ Round 5

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.99

------ Round 6

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.97

------ Round 7

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 8

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 9

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.89

Experiment 3

In this experiment, the hit rate of the top20 using the InsightFace method will be calculated. To calculate the hit_ratio, the percentage of the top 20 similar faces that are the same person as the target face will be calculated. The images will be selected randomly. One target image will be selected from the insight_reco_image directory. Then find the top 20 most similar images to the target face from this directory in the base library.

The guild conducts ten rounds of experiments, each round picking 10 images to search for the top 20 most similar images for each target image. A cumulative total of 100 faces will be searched, and the sample size will be sufficient to meet the demand.

The results of the programme

Round	0	1	2	3	4	5	6	7	8	9
Hit_ratio	0.99	0.97	0.75	0.99	0.87	0.99	0.98	0.88	0.98	0.96
Mean_hit_ratio	93.6%									

------Round 0

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.99

------ Round 1

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.97

------ Round 2

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.75

------ Round 3

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.99

------ Round 4

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.87

------ Round 5

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.98

------ Round 6

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.88

------ Round 7

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.98

------ Round 8

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 1.0

------ Round 9

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.96

-Part of FaceNet

This section will calculate the accuracy of the photos used for FaceNet's face feature recognition. To compare with the InsightFace approach, this is employed.

The exact same collection of facial photos as those used by inightface is utilized to compare the accuracy: insight_reco_image The first step is to use FaceNet to turn the faces that are detected by mtcnn into vectors.

Using the script provided: **code/face detec reco facenet/gen embed.py**

The folder res is where each extracted vector picture is saved. This necessitates running gen embed.py. All of the face vectors that were extracted from the photos are contained in a file named faceEmbedding.npy that is located in this folder. The faces in this file, name.txt, have names that match the vectors. (The image names are the same as the image md5 in postgreSQL)(Figure 4.10)

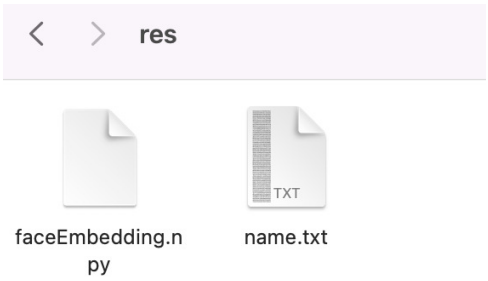


Figure4.10

The next step is to carry out the experiments. When conducting the experiment. In order to avoid any coincidental situations. So the experiment is divided into three groups in order to make the results more convincing and scientific. For each group of experiments the pictures of faces will be randomly selected for analysis. The top 5, top 10 and top 20 similar images will be selected for comparison in each of the three sets of experiments and the ratio will be used as the hit rate. A total of 300 images will be searched.

Experiment 4

The first set of experiments will calculate the hit rate of the first 5 similar faces using the FaceNet method.

Method of calculation. The top 5 faces in the basic library that are most similar to the target face are discovered from a randomly chosen face image in the insight_reco_image directory. The percentage of the top 5 comparable faces that belong to the same person as the target face is used to compute the hit ratio.

The experiments were run in 10 rounds, with 10 faces chosen for each round, and the top 5 most similar faces were looked up for each face.

The results of the programme

Round	1	2	3	4	5	6	7	8	9	10
Hit_ratio	0.88	0.84	1.0	0.98	0.94	0.9	0.98	1.0	0.92	1.0
Mean_hit_ratio	94.4%									

------Round 0

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.88

------ Round 1

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.84

------ Round 2

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 1.0

------ Round 3

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.98

------ Round 4

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.94

------ Round 5

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.9

------ Round 6

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.98

------ Round 7

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 1.0

------ Round 8

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.92

------ Round 9

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 1.0

Experiment 5

This round of experiments will select the results of calculating the first 10 similar images.

100 photos will be searched in total for this experiment. Ten rounds of trials will be conducted according to the experiment's unique design. The top 10 most similar images will be looked up for each of the 10 photographs in each round of search. The sample size can supply what is needed.

The information needed for this experiment is determined by the hit radio calculation logic. The percentage of faces that are comparable to the target face among the first 10 faces is known as the hit_ratio. This is accomplished by choosing one face picture from the insight_reco_image directory, and then locating the top 10 faces in the base library that are most similar to the target face. This script is used for the calculation:

code/face_detec_reco_facenet/face_search_accuracy.py.

Round	0	1	2	3	4	5	6	7	8	9
Hit_ratio	0.86	0.89	1.0	0.96	0.84	0.87	0.91	0.95	0.99	0.95
Mean_hit_ratio	92.2%									

The results of the programme

------Round 0：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.86

------ Round 1：：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.89

------ Round 2：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 3：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.96

------ Round 4：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.84

------ Round 5：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.87

------ Round 6：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.91

------ Round 7：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.95

------ Round 8：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.99

------ Round 9：

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.95

Experiment 6

The calculation process involves selecting a single face image at random from the insight_reco_image directory, locating the top 20 faces in the base library that are most similar to the target face, and calculating the hit ratio—the proportion of the top 20 similar faces that match the target face.

There are a total of 10 rounds of tests, with 10 faces and the top 20 most similar faces chosen for each round. 100 faces will be searched in total after 10 rounds of tests, which is a big enough sample size to satisfy the demand.The results of the programme

Round	0	1	2	3	4	5	6	7	8	9
Hit_ratio	0.87	0.93	0.92	0.84	0.84	0.92	0.87	0.83	0.89	0.90
Mean_hit_ratio	88.0%									

------Round 0：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.87

------ Round 1：：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.89

------ Round 2：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 1.0

------ Round 3：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.96

------ Round 4：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.84

------ Round 5：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.87

------ Round 6：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.91

------ Round 7：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.95

------ Round 8：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.99

------ Round 9：

A random search of ten people for twenty similar photos was conducted. The mean accuracy is 0.95

4.4 Gaussian noise

-Overview

The two face-feature recognition techniques will continue to be tested in this part. The two approaches' search precisions are contrasted. Gaussian noise addition is the method for doing this.

due to image capturing technology, environmental conditions, and numerous other parameters in common applications This will undoubtedly result in a difference between the processed image and the original image, with noise playing a factor in that difference.

The basic image utilized, however, is a pure image when the process is simulated via simulation. Noise must be introduced to the pure image in order to create an effect that resembles a genuine image in order to test the algorithm's performance, and the findings produced by the algorithm are only convincing when noise is taken into account.

- Choice of Gaussian noise

Let's start with the claim that Gaussian noise is justifiable compared to alternative synthetic noise distributions. When the noise source is extremely complicated, Gaussian noise is arguably the best portrayal of real noise. It is challenging to effectively mimic unknown true noise since in reality, noise is typically not produced by a single source but rather a composite of several distinct sources. If the actual noise is thought of as the sum of many random variables with various probability distributions, its normalized sum converges to a Gaussian distribution as the number of noise sources rises. Based on this assumption, handling this tough situation where the noise distribution is unknown can be handled simply and respectably by employing synthetic Gaussian noise.

- Compared to true noise

Let's start with the logical conclusion that a method that performs well with Gaussian noise does not necessarily perform well with real noise. This relies on the characteristics of the actual noise and if the algorithm was built to be resistant to noise dispersion. There is no assurance that the technique created to

handle real noise will perform equally well because Gaussian noise is merely an approximation and simulation of real noise. The Gaussian noise test does, however, have some validity, therefore in the case of actual noise, these algorithms will have some noise reduction. Overall, the simulation is reasonable when Gaussian noise is used.

-Gaussian noise experiment

Accuracy is assessed in this section after adding Gaussian noise to the FaceNet and InsightFace facial feature detection algorithms. The following are the key steps:

1. To begin with, the original image's pixel values are normalized by being divided by 255, bringing them within the range of 0 and 1.

2. Produce a Gaussian-distributed image matrix with a mean and variance sigma.

3. To create the noise-added image, apply the noise to the original image.

4. To recover the pixel values of the noisy image, denormalize and multiply by 255.

5. Add noise.

Four rounds of this experiment will be completed, including two rounds for FaceNet and two rounds for InsightFace. The first five and the first ten similar photographs are used to determine accuracy. The Gaussian function is discussed in the section that follows.

```
#gauss_noise function
def gauss_noise(img, mean, sigma):
image = np.array(img / 255, dtype=float)
noise = np.random.normal(mean, sigma/255.0, image.shape)
out = image + noise
```

resultImg = np.clip(out, 0.0, 1.0)

resultImg = np.uint8(resultImg * 255.0)

return resultImg

img = gauss_noise(img,0,25)

- The results of the programme

1. Comparison of search accuracy of top5 results after adding Gaussian noise.

Algorithm\ Round	0	1	2	3	4	5	6	7	8	9	avg
InsightFace	1.0	0.98	1.0	0.98	0.91	0.92	1.0	1.0	1.0	0.98	**0.98**
FaceNet	0.9	0.9	0.8	0.84	0.86	0.74	0.84	0.96	0.8	0.88	**0.85**

InsightFace:

------ Round 0:

A random search of ten people for five similar photos was conducted. The mean accuracy is 1.0

------ Round 1:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.9778

------ Round 2:

A random search of ten people for five similar photos was conducted. The mean accuracy is 1.0

------ Round 3:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.9778

------ Round 4:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.9111

------ Round 5:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.92

------ Round 6:

A random search of ten people for five similar photos was conducted. The mean accuracy is 1.0

------ Round 7:

A random search of ten people for five similar photos was conducted. The mean accuracy is 1.0

------ Round 8:

A random search of ten people for five similar photos was conducted. The mean accuracy is 1.0

------ Round 9:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.975

FaceNet:

------ Round 0:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.9

------ Round 1:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.9

------ Round 2:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.8

------ Round 3:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.84

------ Round 4:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.86

------ Round 5:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.74

------ Round 6:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.84

------ Round 7:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.96

------ Round 8:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.8

------ Round 9:

A random search of ten people for five similar photos was conducted. The mean accuracy is 0.88

2. Comparison of the search accuracy of top10 after adding Gaussian noise.

Algorithm\ Round	0	1	2	3	4	5	6	7	8	9	avg
InsightFace	1.0	1.0	0.97	0.96	0.99	0.96	1.0	0.99	1.0	0.96	**0.98**
FaceNet	0.85	0.83	0.77	0.81	0.72	0.93	0.89	0.91	0.97	0.87	**0.85**

InsightFace:

------Round 0:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 1:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 2:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.97

------ Round 3:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.9625

------ Round 4:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.9889

------ Round 5:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.96

------ Round 6:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 7:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.9889

------ Round 8:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 1.0

------ Round 9:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.96

FaceNet:

------ Round 0:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.85

------ Round 1:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.83

------ Round 2:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.77

------ Round 3:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.81

------ Round 4:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.72

------ Round 5:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.93

------ Round 6:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.89

------ Round 7:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.91

------ Round 8:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.97

------ Round 9:

A random search of ten people for ten similar photos was conducted. The mean accuracy is 0.87

4.5 Retrieval Efficiency Experiments

-Overview

This section compares the use of a general Python loop for face feature retrieval with the vector database Milvus for face feature retrieval, with the aim of finding the more efficient method.

Using the code: **code/face_detec_reco_insightface/face_search_accuracy.py**

By changing the way the features are calculated in the code, then a different method can be chosen to face the retrieval of feature values (Figure 4.11).

```
    if img is None:
        print('------------------No face detected')
        # print(name)
    else:
        print('-----------------feature')
        target_emb = model.get_feature(img)

        # python loop for vector search
        t1 = time.time()
        for i in compare_emb:
            dist = cacu_l2_dist(target_emb, i)
            l2_dist.append(dist)
        index_l2_dist = list(map(list, zip(range(len(l2_dist)), l2_dist)))
        small_l2_dist = sorted(index_l2_dist, key=lambda x: x[1], reverse=False)
        t2=time.time()

        # # milvus for vector search
        # t1 = time.time()
        # ids = search_vec_by_milvus(target_emb, TOP_K)
        # t2 = time.time()

        print("one vector search, return {} similar results, total cost: {}!".format(TOP_K, (t2-t1)))

        hit_num = 0
        for i in range(topK):
            pic_name_index = small_l2_dist[i][0] #python loop
            # pic_name_index = ids[i] #milvus
            pic_name = name_list[pic_name_index]
            search_person = get_person_by_md5(pic_name)
            if search_person == person:
                hit_num = hit_num + 1
        accuracy = round(hit_num/topK, 4)
        accuracy_list.append(accuracy)
mean_accuracy = round(sum(accuracy_list)/len(accuracy_list), 4)
return mean_accuracy
```

Figure 4.11

-Python Loop Experiment

A loop is a collection traversal, and a Python loop is a loop that computes the distance between vectors from a list of vectors by moving through them violently. For calculating the elapsed time in a loop. The function **time.time()**, which calculates the elapsed time, needs to be added so that the retrieval time can be derived.

Using Python Loop, a face feature vector was selected for similarity calculation (Euclidean distance) with 7200 vectors in the base library and the top 5 most

similar vectors were selected. A total of 10 rounds of experiments were run. The average elapsed time was calculated.

Program results (Figure 4.12):

Round	0	1	2	3	4	5	6	7	8	9
Time(ms)	537	537	522	618	525	519	513	543	545	609
Mean time (ms)	547									

```
one vector search, return 5 similar results, total cost: 0.5369718074798584!
----------------feature
one vector search, return 5 similar results, total cost: 0.5366508960723877!
----------------feature
one vector search, return 5 similar results, total cost: 0.5223171710968018!
----------------feature
one vector search, return 5 similar results, total cost: 0.6182479858398438!
----------------feature
one vector search, return 5 similar results, total cost: 0.5253000259399414!
----------------feature
one vector search, return 5 similar results, total cost: 0.5189049243927002!
----------------feature
one vector search, return 5 similar results, total cost: 0.5137717723846436!
----------------feature
one vector search, return 5 similar results, total cost: 0.543433427810669!
----------------feature
one vector search, return 5 similar results, total cost: 0.5457899570465088!
----------------feature
one vector search, return 5 similar results, total cost: 0.6085751056671143!
```

Figure 4.12

Milvus Experiment.

By first choosing a face feature, determining its similarity to the 7200 vectors in the base library (i.e., calculating the Euclidean distance between them), and then choosing the top 5 most similar vectors, this round of experiments will determine the time required to retrieve data using the Milvus database. Ten

rounds of trials will be conducted. It is computed how long each of the 10 rounds of experiments took on average.

Program results (Figure 4.13):

Round	1	2	3	4	5	6	7	8	9	10
Time(ms)	48	30	28	33	29	28	29	30	39	26
Mean Time(ms)	32									

```
one vector search, return 5 similar results, total cost: 0.047647953033447266!
----------------feature
one vector search, return 5 similar results, total cost: 0.030046939849853516!
----------------feature
one vector search, return 5 similar results, total cost: 0.02876591682434082!
----------------feature
one vector search, return 5 similar results, total cost: 0.03343701362609863!
----------------feature
one vector search, return 5 similar results, total cost: 0.02915215492248535!
----------------feature
one vector search, return 5 similar results, total cost: 0.028673887252807617!
----------------feature
one vector search, return 5 similar results, total cost: 0.029078960418701172!
----------------feature
one vector search, return 5 similar results, total cost: 0.030118942260742188!
----------------feature
one vector search, return 5 similar results, total cost: 0.03873896598815918!
----------------feature
one vector search, return 5 similar results, total cost: 0.025981903076171875!
```

Figure 4.14

-Additional data

The dataset of 7200 items and 512 dimensions was utilized for the tests in the preceding section. This section will conduct three extra sets of tests because the

dataset size would be very high in real-world scenarios. The examination of more datasets resulted in more robust and believable experimental outcomes.

This step creates three datasets at random, each with 512 dimensions containing 1w items, 5w things, and 10w items (512 dimensions). There is a comparison test run.

1. 1w /512 dimensions

Generate a random 512-dimensional vector, then search for the 5 closest vectors in the 1W 512-dimensional vector base library.

Calculation method \ Round	0	1	2	33	4	5	6	7	8	9	avg
Python loop (ms)	751	746	753	759	749	741	732	756	737	743	747
Milvus (ms)	28	27	34	37	36	28	30	41	39	32	33

2. 5w/ 512 dimensions

Generate a random 512-dimensional vector, then search the 5w 512-dimensional vector base for the 5 closest vectors.

Calculation method \ Round	0	1	2	3	4	5	6	7	8	9	avg
Python loop (ms)	3423	3458	3502	3456	3478	3568	3519	3471	3672	3488	3503
Milvus (ms)	43	45	38	39	37	45	43	41	44	33	41

3. 10w /512 dimensions

Generate a random 512-dimensional vector, then search the 5w 512-dimensional vector base for the 5 closest vectors.

Calculation method \ Round	0	1	2	3	4	5	6	7	8	9	avg
Python loop (ms)	7103	7067	7081	7051	7033	6989	7032	7022	7013	7021	7041
Milvus (ms)	58	56	55	56	57	53	52	49	48	55	54

4.6 Benchmark Testing

-Overview

Running a computer program, a collection of programs, or other actions to compare the performance of an object is known as a benchmark. Run a number of common tests and experiments to accomplish this.

To establish a baseline of performance at a specific point in time, benchmarking is the practice of monitoring and evaluating software performance parameters. When the system goes through software and hardware updates, it keeps performance from being impacted.

Face images are captured by cameras in practical applications. There are a lot of cameras in use right now. There are a ton of cameras everywhere in public. The number of facial photos that would result would also be astounding. The number of facial photos that are analyzed daily will so easily approach the tens of millions or billions.

On the basis of this backdrop, a crucial problem will be how to quickly discover comparable faces among the tens of millions or even billions of photos. The open source vector dataset SIFT1B will be used in this experiment to explore the performance limits of milvus.

Three groups of one million, ten million, and one hundred million vectors are chosen to evaluate the system's functionality as it is right now and to simulate a heavier load in order to find any potential scalability bottlenecks as the load increases. The experimentally obtained data can validate some of the system-based assumptions and validate whether these assumptions are reasonable. Planning future directions for improvement is another option.

For more information on this dataset, please refer to

http://corpus-texmex.irisa.fr/.

Version used for testing: milvus 1.0

The script used for testing is: **code/bechmark_script**

Some of the symbols and special terms used are described below.

Terminology	Definition
nq	Number of target vector entries queried
topk	The number of results that are most similar to the target vector of the query.
time	Total query time for batch queries

—Experiment 1

The first million vectors in the sift1B dataset will be chosen as the dataset for the initial series of tests.

Milvus will import the data in batches of 100,000. The import time for 100,000 vectors with 128 dimensions is 0.7 seconds.

The recall was tested three times, with the 500 vectors deleted each time being different. The recall was averaged across 500 randomly chosen vectors from the set of vectors to be queried.

Accuracy:

Round / Accuracy / topk	Round 1	Round 2	Round 3
topk=1	98.2%	98.3%	98.2%
topk=10	98.3%	98.2%	98.4%
topk=100	98.4%	98.4%	98.5%
topk=500	98.0%	87.9%	98.0%

Performance:

topk / time(s) / nq	topk=1	topk=10	topk=100	topk=500
nq=1	0.0026	0.0024	0.0025	0.0023
nq=10	0.0099	0.0101	0.0106	0.0122
nq=100	0.0699	0.0708	0.0706	0.0763
nq=500	0.3356	0.3351	0.3567	0.4390
nq=1000	0.907	0.9300	0.9226	0.9735

-Experiment 2

The top 10 million vectors from the sift1B dataset will be used to create the dataset for the second series of tests.

500 non-identical vectors will be randomly eliminated from the set of vectors to be questioned after each of the three testing sessions. The average of the 500 outcomes is used to determine the final recall.

Accuracy:

Round / Accuracy / topk	Round 1	Round 2	Round 3
topk=1	97.8%	97.6%	98.4%
topk=10	98.1%	98.3%	98.0%
topk=100	98.2%	98.2%	98.1%

topk=500	97.6%	97.3%	97.7%

Performance:

topk / time(s) / nq	topk=1	topk=10	topk=100	topk=500
nq=1	0.0051	0.0047	0.0048	0.0060
nq=10	0.0499	0.0500	0.0461	0.4812
nq=100	0.3306	0.3263	0.3252	0.3379
nq=500	1.4227	1.4162	1.5058	1.5132
nq=1000	2.6893	2.7907	2.9995	3.0083

-Experiment 3

The top 100 million vectors in the sift1B data set were used to create the data set for the third series of trials.

In this instance, 500 vectors were chosen at random from the vector set to be queried. The average of these 500 findings is used to calculate the recall. Three experiments in total were run, and each time, not all 500 extracted vectors were the same.

Accuracy:

Round / Accuracy / topk	Round 1	Round 2	Round 3
topk=1	96.8%	96.0%	96.4%

topk=10	96.3%	95.8%	96.2%
topk=100	95.5%	94.9%	95.3%
topk=500	93.5%	92.4%	92.9%

Performance:

topk / time(s) / nq	topk=1	topk=10	topk=100	topk=500
nq=1	0.0193	0.0195	0.0204	0.0240
nq=10	0.0856	0.0857	0.0860	0.0920
nq=100	0.5063	0.5042	0.5212	0.6316
nq=500	2.3709	2.3793	2.4304	2.6234
nq=1000	4.3712	4.3877	4.4647	4.8881

5 Results And Evaluation

5.1 Overview

A summary of the overall work studied in Chapters 3 and 4 will be presented. In addition, the results will be evaluated in this chapter.

5.2 Results and Analysis

5.2.1 InsightFace and FaceNet

The results of the InsightFace and FaceNet experiments in Chapter 4 are first summarised.

Comparison of Experiment 1 and Experiment 4. A total of 100 faces were studied, comparing their respective top 5 similar results, and it was found that the average face retrieval accuracy of InsightFace (99.2%) would outperform that of FaceNet (94.4%).

Comparison of Experiment 2 and Experiment 5. A total of 100 faces were compared in their respective top 10 similarity results, and it was found that the average face retrieval accuracy of InsightFace (96.4%) would outperform that of FaceNet (92.2%).

Comparison of Experiment 3 and Experiment 6. A total of 100 faces were compared with their respective top 20 similarity results, and it was found that InsightFace (93.6%) outperformed FaceNet (88.0%) in terms of average face retrieval accuracy.

Through the comparison of the above 6 sets of experiments, it can be found that the average face retrieval accuracy of InsightFace is higher than that of FaceNet.

Gaussian noise was introduced to the photos in later experiments. The statistical retrieval accuracy was then calculated using two models, which were utilized to recover related photos individually. InsightFace still had a higher accuracy rating than FaceNet.

The analysis is based on the conclusions mentioned above. Facenet and insightface, including algorithm performance and effects; facenet utilizing the 18-year-old arcface, Python2 environment, based on the mxnet implementation; and insightface using the older softmax, Python3 environment. As seen in the challenging event megaface, using the most recent arcface is preferable than the relatively early facenet. Tests on earlier datasets show that insightface is superior to facenet in terms of performance.

5.2.2 Milvus and Python Loop

Ten rounds of experiments were conducted in Chapter 4 for Python Loop and Milvus, respectively. A comparison of the studies shows that for the same

dataset (7200 512-dimensional face feature vectors), the retrieval speed of Milvus (32ms) is approximately 20 times faster than that calculated directly using a python loop (547ms). It is concluded that the Python loop using purely violent computation does not outperform Milvus.

Because the first data set (7200) was not large enough, three more larger data sets (10,000, 50,000, and 100,000) were used for random data validation. The conclusions obtained are shown in the table.

Datasets / avg time(ms) / Method	Datasets / 10,000	Datasets /50,000	Datasets /100,000
Milvus	33	41	54
Python Loop	747	3503	7401

Based on the analysis of the experimental results in the previous section. The performance of the computation using milvus will be vastly superior to the violent computation using the python loop.

Milvus slices the inserted feature data and creates a very efficient ANN (Approximate Nearest Neighbor) index for these data slices. Based on the data slicing and efficient ANN indexing, Milvus is able to maximise the use of CPU

computational cores, allowing feature search tasks to be performed in parallel within the system, maximising retrieval efficiency.

Using a Python loop for direct brute force computation is analogous. Since no data sharding is done, better concurrency is not possible. Because there is no dedicated ANN index, it is unable to maximise CPU computational power. It is therefore logical that the computation speed is an order of magnitude lower than Milvus.

5.2.3 Milvus benchmark

Benchmark trials were also performed to confirm Milvus's stability. Three sets of tests, each taking one million, ten million, and one hundred million vectors, were performed using the sift1B dataset. For accuracy and effectiveness, different tests were run on each set of studies. The experiments' findings are as follows:Experimental results for accuracy and performance (1,000,000):

1,000,000 datasets	Topk=1	Topk=10	Topk=100	Topk=500
Three-times Avg Accuracy	98.2%	98.3%	98.4%	94.6%

NQ	Avg Time(S)
nq=1	0.0024
nq=10	0.0107
nq=100	0.0719

nq=500	0.3666
nq=1000	0.9332

Experimental results for accuracy and performance (10,000,000)

10,000,000 datasets	Topk=1	Topk=10	Topk=100	Topk=500
Three-times Avg Accuracy	97.9%	98.2%	98.2%	97.6%

NQ	Avg Time(S)
nq=1	0.0051
nq=10	0.0157
nq=100	0.3300
nq=500	1.4644
nq=1000	2.8720

Experimental results for accuracy and performance (100,000,000)

100,000,000 datasets	Topk=1	Topk=10	Topk=100	Topk=500
Three-times Avg Accuracy	96.4%	96.1%	95.2%	92.9%

NQ	Avg Time(S)
nq=1	0.0208

nq=10	0.0873
nq=100	0.5408
nq=500	2.4510
nq=1000	4.5279

Milvus has demonstrated excellent retrieval of massive data volumes in performance tests. Even with billions of bytes of data, millisecond returns are still attainable. On the other hand, it is feasible to attain such high retrieval performance while yet maintaining a highly respectable retrieval accuracy rate of over 95%. It is ideal for large-scale face retrieval in daily life because to its performance and accuracy.

5.3 Summary and Evaluation

Overall, I think the project's primary goals have been accomplished. Through experimental comparisons, it was intended to confirm the accuracy and performance under various approaches before choosing the most suitable method in light of the findings. The following section assembles a face retrieval system.

The identical data from the same dataset was used in the studies for both the FaceNet and InsightFace algorithms; the only factors that differed were the methods themselves and the number of datasets chosen. Because there is enough evidence to draw broad inferences, the conclusions are stronger.

Throughout the Milvus and Python Loop investigations. These two approaches are very different from one another. A frequent form of invocation is the Python loop. In contrast, Milvus is a feature library retrieval technique. The same dataset was used for performance tests, and the results were compelling enough.

However, the chosen solution has certain limitations because I am not familiar with feature library calling techniques. The principle adopted was to conduct

the test using the broadest comparison feasible. As a result, although it may be a somewhat optimal solution, this approach may not be the best one.

The dataset that was selected, which was primarily assembled by others, also has limitations. As a result, the data sets have numerous issues, including being too small, having low categorization quality, having unbalanced data, and having poor data set quality. As a result, the experiment may be more prone to bugs and the dataset issues may take longer to resolve. It could be challenging for the model to handle the images in the dataset if they are not in a specified format or if their values are outside of a certain range. Better outcomes would be attained if the photos had a specified aspect ratio or pixel values. As a result, the data must be cropped or stretched in order to conform to the format of the other samples.

Both the subsequent benchmark exam and the random data test were created to increase the relevance of the experiments to the actual world. These two tests provided evidence to support the conclusions. In terms of performance and accuracy, Milvus outperformed Python Loop, and the outcomes were in line with those of other trials.

6 Extensions

6.1 Overview

The face similarity percentage metric will be put out in this part as a way to compare two faces based on their Euclidean distance from each other. The results of the preceding section will subsequently be used to construct a face retrieval system.

6.2 Facial resemblance

It is necessary to establish a matching metric in order to compare two faces. The level of similarity between two faces can be determined using the metric of percentage facial similarity.

There are three phases involved in calculating the degree percentage of facial likeness.

1.Normalize the features of the face.

2. Identify the face feature of the Euclidean distance.

3. Determine the percentage of faces that are similar.

Below are the crucial formulas for these calculations.

-L2 Regularization

To prevent overfitting a regularization term needs to be added to the model to limit the complexity of the model so that the model balances complexity and performance.

N-dimensional original vector space: $R^{n}, n \in N$ (R denotes a real number and N denotes a non-zero natural number)

Original vector: $X = (x_1, x_2, ..., x_n), X \in R^{n}$

The L2 norm (modulus length) of vector X: $|X| = \sqrt{\sum_1^n x_i^2}$

Normalised vector: $X' = (x_1', x_2', ..., x_n'), X' \in R^{n}$

Calculate the L2 regularisation for each dimension: $x_i' = \frac{x_i}{|x|} = \frac{x_i}{\sqrt{\sum_1^n x_i^2}}$

After normalisation, the vector modulus length equals 1: $|X'| = 1$

- Inner product

The inner product of the vectors A, B: $p(A,B) = A \cdot B = \sum_1^n (a_i \times b_i)$

-Vector similarity calculation

Approximate Nearest Neighbor Searching (ANNS) is the current mainstream idea for vector search. The core idea is to compute and search only in a subset of the original vector space, thus speeding up the overall search.

Subsets of the original vector space: $\gamma, \gamma \subset R^n$

-Similarity of cosine

Cosine similarity of vectors A,B: $$\cos(A,B) = \frac{A \cdot B}{|A||B|}$$

Similarity is determined by the cosine: the larger the cosine value, the higher the similarity, i.e. $$TopK(A) = \arg\max_{B \in \gamma} \cos(A,B)$$

Suppose the normalised vectors A and B are A' and B' respectively, then

$$\cos(A,B) = \frac{A \cdot B}{|A||B|} = \frac{\sum_1^n (a_i \times b_i)}{|A||B|} = \sum_1^n (\frac{a_i}{A} \times \frac{b_i}{B}) = \cos(A',B')$$

Therefore, the cosine similarity between the two vectors remains the same after normalisation.

$$\cos(A',B') = \sum_1^n (\frac{a_i}{A} \times \frac{b_i}{B}) = A' \cdot B' = p(A',B')$$

Therefore, after normalisation, the inner product is equivalent to the formula for calculating cosine similarity.

- Euclidean distance

Euclidean distance of vectors A, B: $d(A,B)=\sqrt{\sum_1^n (a_i-b_i)^2}$

Similarity is judged by the Euclidean distance: the smaller the Euclidean distance, the higher the similarity. i.e. $TopK(A)=\arg\min_{B\in\gamma} d(A,B)$

Assuming that the vectors A, B are normalised, the above equation is further expanded:

$$d(A,B)=\sqrt{\sum_1^n (a_i-b_i)^2}=\sqrt{\sum_1^n (a_i^2-2a_ib_i+b_i^2)}=\sqrt{\sum_1^n a_i^2-2\sum_1^n a_ib_i+\sum_1^n b_i^2}=\sqrt{2-2\times p(A,B)}$$

Therefore, $d(A,B)^2=-2\times p(A,B)+2$

Thus, the square of the Euclidean distance is negatively related to the inner product. Whereas the Euclidean distance is a non-negative real number, the magnitude relationship between two non-negative real numbers is the same as the magnitude relationship between their own squares.

Therefore, after normalisation, for the same vector, the top K closest vector result sets returned by the Euclidean distance are equivalent to the top K most similar vector result sets returned by the inner product, given the same search space. Based on the formula above, it is possible to calculate the percentage of similarity between the two faces.

Noteworthy The final two features obtained are two vectors, and some choose to do this with cosine similarity, while others do it directly with Euclidean distance.

If the cosine similarity method is used, then the result will take the value 0-1 now, and if the test is done with cosine similarity, the training should be considered as close as possible.

A more popular strategy is to request feature normalisation before performing an equivalent Euclidean distance. Performance, however, is likely to be impacted by feature normalisation. Therefore, it is preferable to use the Euclidean distance in practise.The Euclidean distance is mapped to a value between 0 and 1 by modifying the sigmoid function.

$s = \frac{1}{1+e^{\alpha x+\beta}}$, where x is the distance. In this case, the entire dataset will have a plausible 0-1 score, which indicates the degree of similarity of the dataset.

6.3 Face Retrieval System

6.3.1 Overview

Through the findings of previous research, a face retrieval system will be built using MTCNN+InsightFace+Milvus.

Figure 6.1 depicts a system architecture diagram.

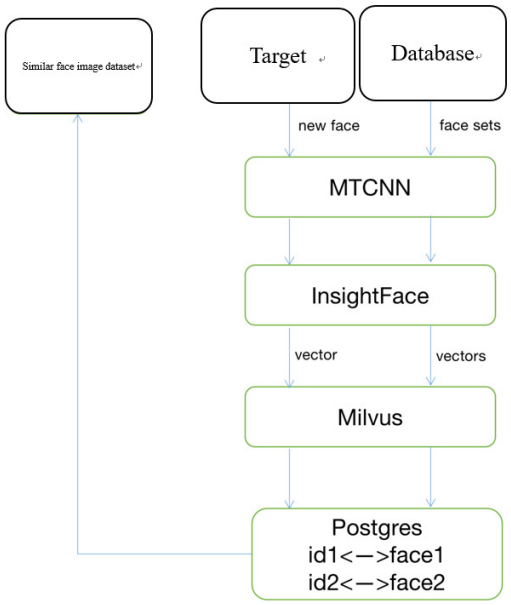


Figure 6.1

6.3.2 Implementation

Two data processing stages make up the overall face retrieval system. Target face retrieval and base library face entry.

The initial step is to classify the dataset's face photos into the basic library.

1. For each face image in the base library, the face is first detected using the MTCNN algorithm.
2. For the detected faces, extract the feature vectors of the faces using the InsightFace model. 3.
3. Store the extracted face feature vectors in the Milvus database.
4. Store the vector ID and face image name from the Milvus database into the relational database Postgres.

Phase 2: Target Face Retrieval

The following describes how data is processed whenever a fresh face image is retrieved or when a face image that is similar to the target face needs to be looked up in the base database.

1. Take this image of the face to be searched and detect the face using the MTCNN algorithm.
2. The feature vector of the face is extracted by the InsightFace model for the detected face.
3. take the extracted face feature vectors to the Milvus vector base library for similar vector retrieval and return the ids of the similar vectors.
4. Milvus returns the vector id returned by Milvus. Go to Postgres to find the corresponding face image name and return the specific image. Complete the whole face retrieval process.

6.3.3 System demo screenshots

The whole system is divided into two parts, which are the face entry module and the face recognition module. By calculating the similarity between the attendance face and the face in the base library, if there is a face with a similarity greater than 90%, it is judged to be the same person. On the contrary, recognition fails.

Face entry (Figures 6.2 and 6.3).

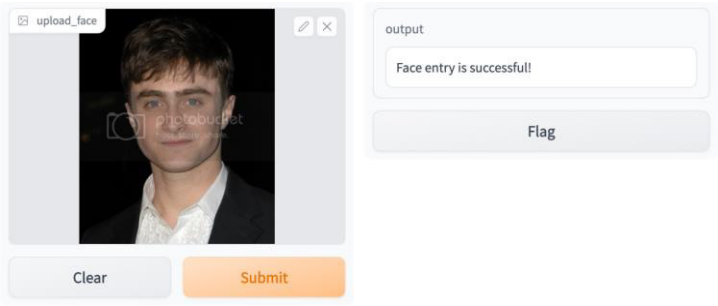


Figure 6.2

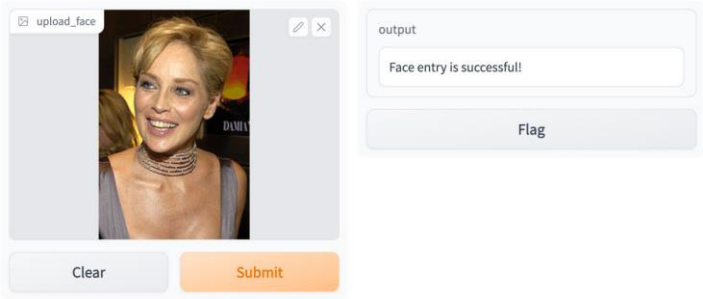


Figure 6.3

Face recognition (Figure 6.4 and 6.5). These are two images that have been recorded in advance. Recognition was successful.

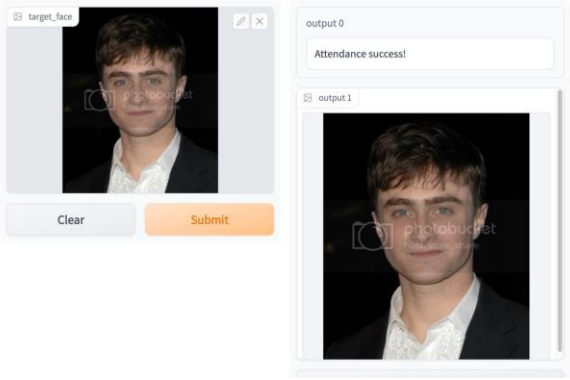


Figure 6.4

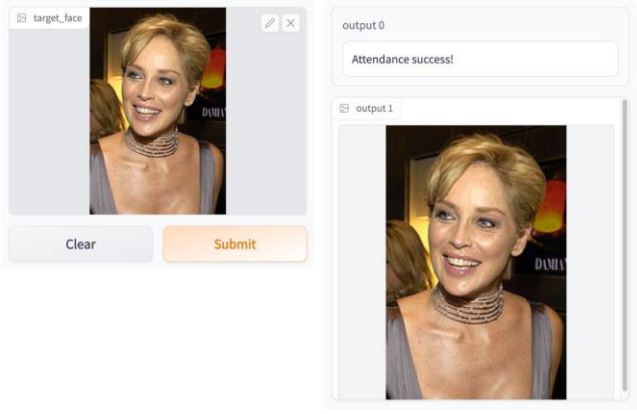


Figure 6.5

As this image is an unrecorded face, recognition failed (Figures 6.6 and 6.7).

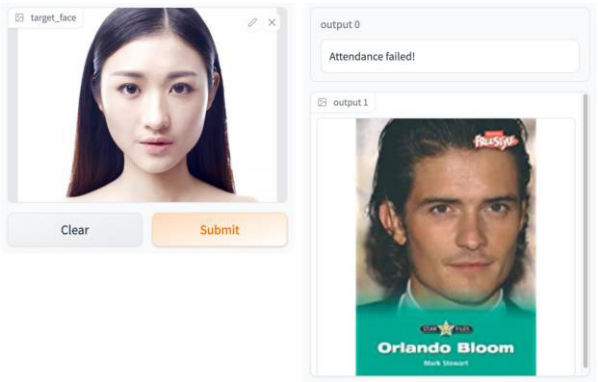


Figure 6.6

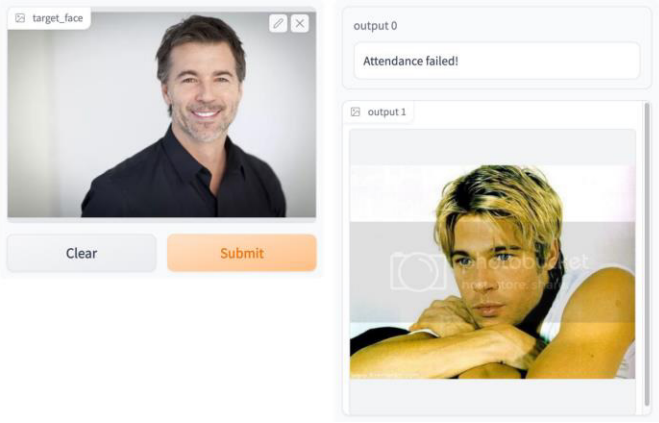


Figure 6.7

-Similar image search

First upload a face image from the base library. After that, enter the number of images to be retrieved. The results are shown below (Figures 6.8 and 6.9).

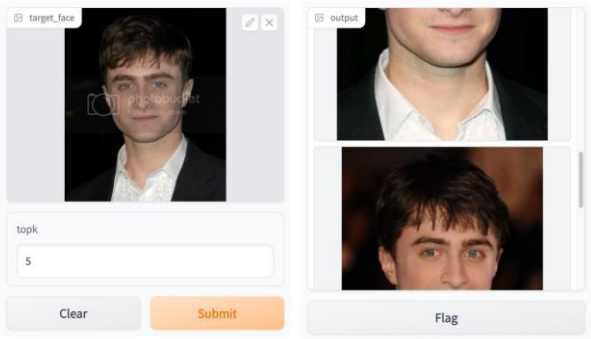


Figure 6.8

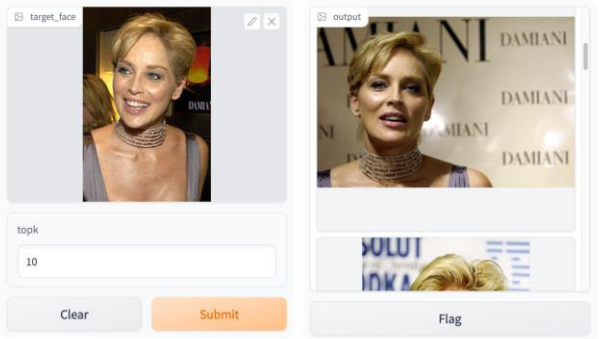


Figure 6.9

6.4 Summary

The method for obtaining face similarity percentages is presented in the earlier section of this one. Using this method, face similarity may be quickly determined in production practice and filtered and filtered based on face similarity to identify more compliant faces. The face retrieval system developed based on the research findings is shown in the latter portion, demonstrating the accuracy and viability of the selected tool and algorithm.

7 Conclusion and Future Work

7.1 Conclusion

This thesis analyses and investigates the accuracy of two face retrieval models, InsightFace and FaceNet, and comes to the conclusion that InsightFace has a greater face retrieval accuracy than FaceNet. It also suggests a way to check the correctness of face retrieval models. Future studies and comparisons of more face retrieval models can be based on the study approach presented in this publication.

This thesis also compares two approaches for retrieving face-feature values. Experimental results show that utilising a base database of 7,200 faces and Milvus, a vector database, the retrieval speed of Milvus is 20 times faster than python loop direct computing.

This article also benchmarked Milvus using an open source dataset to further confirm its viability for daily use. The tests demonstrated that, even with a data scale of hundreds of millions of records, Milvus can still ensure 95+% search accuracy and millisecond high performance search. This is ideal for daily requirements.

Calculating how similar two faces are can be useful in some situations. This thesis suggests a method for estimating the proportion of face similarity, B = -0.5A+1, based on the assumption that the Euclidean distance is constrained after normalising the facial features (where B is the percentage of face similarity and A is the Euclidean distance after normalisation of the face features).

Finally, based on the above research in this paper, an innovative MTCNN+InsightFace+Milvus architecture is used to build a similar face retrieval application system, where the user enters a face and then enters the number of similar faces they want to search for, and the corresponding number of similar face images can be searched from the underlying database. The similar face retrieval system can be used in a wide range of applications such as public security, community management, suspect capture and face control, etc. It has a high practical value.

7.2 Future Work

The content of several of the papers is still not excellent because of time and my own academic limitations. Future-oriented work will be the main topic of this section.

-Speeding up the extraction of facial features through the use of GPU.

The GPU benefits from its excellent image processing capabilities, and it is the face recognition that has to process the images of faces captured by the camera. So, with the help of the GPU, image preprocessing and feature extraction can be done with half the effort. The GPU's powerful computing power is put to good use in face image matching and recognition. With parallel computing, the search time for matching the feature data of a face image with the feature templates stored in the database is drastically reduced.

Face recognition technology relies on artificial neural networks under deep learning, and the rapid rise of deep learning is due to the use of GPUs, whose powerful parallel computing capabilities have led to a significant increase in the training speed of deep neural networks. With the GPU's unique floating point

computation, the computation time has been reduced by tens or even hundreds of times. As the user's appearance changes slightly every day, e.g., shaving, changing hairstyle, more glasses, changing expressions, may cause a comparison failure. Here, deep learning driven by the GPU is required. The input of a large amount of data related to the captured face image is used to validate the algorithm. The recognition's precision is always being increased. When the similarity surpasses this threshold, the corresponding outcome can be output until the system is able to determine an exact threshold.

-Improvement of face recognition accuracy.

There are still many difficulties and challenges in the facial recognition process. The quality of the video images is particularly bad because they are regularly recorded without the user's consent and frequently outside (or indoors, but under poorer acquisition conditions). Since they can be hidden or altered, the lighting and positioning of the video facial images may differ dramatically.

Second, face images are smaller: again due to less ideal acquisition conditions, video face images will often be lower than the predetermined size of a static image based face recognition system. The small size of the image will have an impact on the precision of face detection, segmentation, and key point localization, which will inevitably lead to a deterioration in the performance of the complete face recognition system.

Thirdly, environmental considerations: The lighting in the face recognition area is referred to as environmental factors. In general, light is needed in the recognition area of a face recognition system to ensure uniform illumination, absence of shadows, and flashing light. Although certain high-end products use less lighting, their costs will be rather high. Therefore, there is a requirement to increase the supplemental light equipment in order to increase the face recognition accuracy rate under unfavorable lighting situations.

The facial recognition system's camera and motherboard performance are referred to as the hardware component. Between 2 million and 4 million pixels

are employed in most face recognition cameras. Performance varies, according to various manufacturers of various models, and the number of pixels is not always a good indicator of performance. Additionally, at entrances and exits with a low traffic volume, general static face recognition can be used; however, to maintain the accuracy of face recognition in these situations, entrances and exits with high traffic volumes require the camera to have ultra-wide dynamic and motion compensation functions.

Another significant element affecting the facial recognition system's accuracy is the functionality of the main board of the control host. Therefore, to ensure that the accuracy of the face recognition system is increased, one must consider the motherboard computational capability, storage capacity, environmental adaptability, and software algorithm when choosing face recognition equipment.

The software factor refers to the enhancement and optimization of the facial recognition system. The face recognition method operates on the assumption that the system inputs a face image or sequence of face images containing an unidentified identity in order to identify a known identity in the face database. A series of similarity evaluations are then used to reveal the identity of the face that has to be recognised.

There are four commonly used biometric algorithms: algorithms using neural network-based algorithms, algorithms using facial expression feature points, algorithms using modular approaches on full-face pictures, and algorithms using templates. Several different sorts of algorithms can be used to further improve the accuracy of facial recognition systems.

8 Reflection

Through this project, I gained knowledge on the structure and operation of facial recognition systems. These systems' architecture. Through highly interesting routes, this project helped me get valuable knowledge and learn how to handle picture recognition issues. Furthermore, learning about the developments that are available in this region was one of the most crucial elements. This inspired me to

research the prospects for intelligent photo identification detection and their potential value going forward.

I learned that facial recognition accuracy is currently at a very high quality after reading a lot for my thesis. Therefore, continuing to work on increasing the accuracy would be incredibly difficult. As civilization advances, the use of facial recognition technologies will become increasingly vital since they significantly reduce labour costs and ease daily life. Examples include mobile payments and security checks. However, until better theories or algorithms are developed and maintained, the accuracy of recognition won't considerably improve. Otherwise, it will be quite difficult to achieve this. As a result of working on this project, I have increased my technical expertise and skill set in a variety of different computer science domains. I need more than simply the project's specifics to finish this project. I also had to pick up some new skills. For instance, I learned PostgreSQL, a totally distinct database from MySQL. I also became familiar with the Milvus vector database. All of these are areas that have long attracted my interest. Considering that I have had a genuine interest in these subjects for a long time. I knew very little about image detection and facial recognition systems when I initially learned about this project. My horizons were broadened as an outcome of learning more about the topic and all of its facets. With the help of my supervisor, I was able to better comprehend facial recognition by looking over other research articles. By reading other research papers, I have been able to distil this information into a useful report, so I sincerely hope you will read through it.

9 Reference

1. Chellappa, R., Wilson, L., Sirohey, S.(1995)‘Human and machine recognition of faces: A survey’, Proceedings of the IEEE, 83(5),pp705-741
2. Stone, Z., Zickler, T., Darrell, T.(2010)‘Toward large-Scale face recogniton using Social network context’, Proceedings of the IEEE, 98(8),pp1408-1415.
3. Schweinberger, R., Burton, M.(2003)‘Covert recognition and the neural system for face processing’, Cortex, 39(1),pp9-30.

4. Huang, K., Ren, W., Tan, T.(2014)‘Overview of image classification and detection algorithms’, Journal of Computer Science,37(6),pp1225-1240.

5. Zhao, L. et al.(2002)‘Nearest Feature Line (NFL) clustering algorithm based on key frame extraction for lens retrieval’, Journal of Computer Science,23(12)，pp1292-1296.

6. Lin, Z. et al.(2015)‘Background modeling based on uniform scale invariant local ternary model and its parallel implementation on Intel HD graphics cards’, Computer Applications, 35(8),pp2274-2279.

7. Wang, W., Chen, F., Yang, J.(2005)‘A feature extraction method based on singular value decomposition. Journal of Electronics and Information’, Journal of Electronics and Information, 27(2),pp294-297.

8. Liang, L. et al.(2001)‘Face detection based on template matching and artificial neural net confirmation’, Journal of Electronics, 29(6),pp744-747.

9. Yuan, C., Zhang, C.(2000)‘Automatic face detection based on multiple template matching’, Journal of Electronics, 28(3),pp95-98.

10. Zhou, Z. et al.(2001)‘Automatic face detection based on multiple template matching’, Computer Research and Development, 38(10),pp1204-1210.

11. Turk, M., Pentland, A.(1991)‘Eigenfaces for recogniton’, Journal of Cognitive Neuroscience, 3(1),pp71-86.

12. Grudin, A.,1997.Compact multi-level representation of human faces for recognition.Thesis.(PhD).Liverpool John Moores University.

13. Belhumeur, N., Hespanha, P., Kriegman, J.(1997)‘Eigenfaces vs. fisherfaces: Recognition using class specific linear projeceion’, 19(7).

14. Liu, C., Wechsler, H.(1998)‘Probalilistic reasoning models for face recognition. Proceedings of the 1998’, IEEE Computer Society Conference on CVPR, pp827-832.

15. Moghaddam, B., Pentland, A.(1997)‘Beyond linear Eigenfaces: Bayesian matching for face recognition. ’,Face Recognition from Theory to Application, pp 230-243.

16. Park, H.(2008)‘A comparison of generalized linear discriminant analysis algorithms’, Pattern Recognition, 41(3),pp1083-1097.

17. Yu, H., Yang, J.(2001)‘A direct LDA algorithm for high-dimensional data-with application to face recognition’, Pattern recognition, 34(10),pp2067-2070.

18. Guo, Y, Hastie, T., Tibshirani, R.(2007)‘Regularized linear discriminant analysis and its application in microarrays’,Biostatistics, 8(1),pp86-100.

19. Sharma, A., Paliwal, K.(2010)'Regularisation of eigenfeatures by extrapolation of scatter-matrix in face-recognition problem',Electronics Letters, 46(10),pp682-683.

20. Ojala, T., Pietikainen, M., Harwood, D.(1996)'A comparative study of texture measures with classification based on feature distributions',Pattern Recognition,29(1),pp51-59.

21. Ahonen,T., Hadid, A., Pietikainen, M.(2006)'Face Description with Local Binary Patterns: Application to Face Recognition',EEE Transactions on Pattern Analysis and Machine Intelligence,28(12).

22. Lee, S.(1996)'Image representation using 2D Gabor wavelets',IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(10),pp959-971.

23. Cao, L. et al.(2006)'Two-dimensional Gabor wavelet-based face recognition algorithm', Journal of Electronics,28(3),pp490-494.

24. Samaria, F., Young, S.(1994)'HMM based architecture for face identification', Image and Computer Vision, 12(8),pp537-543.

25. Lanitis, A., Taylor, J., Cootes, F.(1995)'Automatic face identification system using flexible appearance models', Image Vision Comput, 13(5),pp393-401.

26. Penev, P., Atick, J.(1996)'Local feature analysis: a general statistical theory for object representation', Network Computer, 7(3),pp477-500.

27. Wiskott, L. et al.(1997)'Face recognition by elastic bunch graph matching', IEEE Trans. on Pattern Analysis and Machine Intelligence, 9(7),pp775-779.

28. Luo, H.,2004.Face Recognition Technology Research.Thesis.(MA). Guizhou University of Technology.

29. Li, F., Hu, K.(2005)'Current status and outlook of non-rigid motion analysis methods', Chinese Journal of Graphical Graphics,10(1),pp11-17.

30. Cao, Q. et al.(2018)'Vggface2: A dataset for recognising faces across pose and age', In FG.

31. Parkhi, O., Vedaldi, A., Zisserman, A.(2015)'Deep face recognition',In BMVC

32. Sengupta, S. et al.(2016)'Frontal to profile face verification in the wild',In WACV.

33. Zheng, T., Deng, W.(2018)'Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments',Technical Report.

34. Moschoglou, S. et al.(2017)'Agedb: The first manually collected in-the-wild age database',In CVPR Workshop.

35. Zheng, T., Deng, W., Hu, J.(2017)'Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments',arXiv:

36. Shlizerman Ira et al.(2016) ‘The megaface benchmark: 1 million faces for recognition at scale.’ In CVPR,

37. Whitelam, C. et al.(2017)‘Iarpa janus benchmark-b face dataset’,In CVPR Workshop.

38. Maze, B. et al.(2018)‘Iarpa janus benchmark–c: Face dataset and protocol’,In ICB.

39. Zhang, K. et al.(2018)‘Joint face detection and alignment using multitask cascaded convolutional networks’,SPL.

40. Zhang, X. et al.(2017)‘Range loss for deep face recognition with long-tail’,In ICCV.

41. Wang, H. et al.(2018)‘Cosface: Large margin cosine loss for deep face recognition’,In CVPR.

42. Wang, F. et al.(2017)‘Normface: l 2 hypersphere embedding for face verification’,arXiv.

43. Ranjan, R., Castillo, C., Chellappa, R.(2017)‘L2-constrained softmax loss for discriminative face verification’,arXiv.

44. Wang, F. et al.(2018)‘Additive margin softmax for face verification’, SPL.

45. Liu, W. et al.(2016)‘Large-margin softmax loss for convolutional neural networks’,In ICML.

46. Zeiler, M., Fergus, R.(2013)‘Visualizing and understanding convolutional networks’, CoRR.

47. Szegedy, C. et al.(2014)‘Going deeper with convolutions’, CoRR.

48. Weinberger, Q., Blitzer, J., Saul, L.(2006) ‘Distance metric learning for large margin nearest neighbor classification’,In NIPS.

49. Ahmed, Osman H, Nandi A.(2019) ‘Linear Subspace Learning.’

50. Sukri, Syafiqah & Ruhaiyem, (2019) iFR: A New Framework for Real-Time Face Recognition with Deep Learning.’