北京郵電大學

Queen Mary University of London

Undergraduate Project Report
2018/19

Design and Implementation of Face Recognition System

Name:	Chen Ziming
School:	International School
Class:	2015215118
QM Student No.:	151006342
BUPT Student No.:	2015213363
Programme:	Internet of Things Engineering

Date: 05-05-2019

Table of Contents

Abstract

Face recognition technology is a very classic and popular biometric technology. By analysing and comparing people's facial features, face recognition can accurately and effectively identify different identities. Nowadays, with the rapid development of various computer technologies such as machine learning, face recognition technology has reached a very advanced level. At the same time, the revolution of mobile Internet and the popularization of smart mobile devices, have greatly brought down the barriers to utilize face recognition technology, expanded the scenario of applying face recognition, and further promoted the development of face recognition based on mobile devices. Based on machine learning techniques, mobile application development and wireless communication, this project successfully designed and implemented a mobile application with face recognition function and good interactive experience. The whole system is divided into mobile-end and server-end. The mobile-end has a friendly user interface, which can collect face images, send them to the server and present prediction results. The server-end loads the face recognition model, analyses faces, obtains the prediction results and returns them to the mobile-end. This report will introduce the background and relevant techniques as well as the design and implementation process of the project in detail. Also, the demonstration of the system and the experimental analysis of the system, will be described in detail in this report. Finally, this report will summarize the work and achievements of the project, and look forward to possible future extensions and improvements.

Keywords: Face recognition, mobile application development, machine learning

摘要

人脸识别技术是一项经典和热门的生物识别技术，通过分析和比较人的脸部特征，人脸识别能够对不同身份进行精准有效的识别。当下，随着机器学习等计算机技术的蓬勃发展，人脸识别技术已经达到了非常先进的水平。同时，移动互联网的革命和智能移动设备的普及，大大降低了使用人脸识别技术的门槛，拓展了人脸识别的适用场景，进一步推动了基于移动设备的人脸识别的发展。基于机器学习技术，移动应用开发和无线通信，本项目成功设计并实现了一款拥有人脸识别功能和良好交互体验的移动应用。整套系统分为移动端和服务器端，移动端具备友好的界面，能够采集人脸图像，发送给服务器端，并呈现人脸识别的结果；服务器端加载人脸识别模型，对人脸进行分析，并得出预测结果，返还给移动端。本报告将详细介绍项目的背景与相关技术，以及设计与实现过程。并且，对系统的演示和实验分析，本报告也将详细描述。最后，本报告总结了该项目的工作和成果，并对未来可能的拓展和改进作出了展望。

关键字：人脸识别，移动应用开发，机器学习

Chapter 1: Introduction

1.1 Technical Context and Motivation

Face recognition is relatively mature at present, but there is still a huge demand and development space for face recognition based on mobile devices. Since the face recognition application is on the mobile platform, the face recognition technology is more flexible and can be utilized more easily under many circumstances. Here are some possible applicable conditions:

First, the application can be used for remote self-service. For example, in conjunction with the Public Security Bureau to upload identity information, you can upload your face information remotely by using the mobile application. Compared with traditional fingerprint recognition that a special machine is needed for face-to-face data entry and verification, face recognition technology with mobile devices break through the geographical constraints.

Second, the application can be used for company attendance management. Similar face recognition applications have appeared on the market, and employees only need to use their own mobile devices to perform face recognition sign in and location check-in as well as viewing historical attendance data. The company can also more easily grasp the conditions of all employees, and reduce employee management costs.

Third, it can be used for the door control system to identify people who enter and go out. For example, it is no longer necessary to install additional equipment like cameras to implement the home access control system. All the users need to do is to connect their mobile phones to the home Wi-Fi, and use the face recognition application to unlock the door.

Finally, the application can be used for access control and help protect confidential information. For example, when you want to look for some confidential information related to your work or bank information stored in your mobile device, only if your face is stored on the back-end will you get permission. Moreover, the authorization of many transactions at present is implemented by using password. If the password is broken or stolen, security cannot be guaranteed. However, by using biometrics, the digital identity and real identity of the parties on the Internet can be unified, which greatly increases the reliability.

1.2 Purpose of the Project

The aim of the project is to design and implement a face recognition system that allows users

to use mobile application to use face recognition function. There are mainly four tasks in this project:

- Design and implement a face recognition algorithm with OpenCV or other machine learning algorithms
- Design and implement a face recognition application on iOS/Android platform
- Conduct experiment analysis on the proposed face recognition algorithm
- Design and implement a simple application to illustrate the prediction result

1.3 Functions of the System Realized

Based on machine learning techniques, mobile application development and wireless communication, this project successfully designed and implemented a face recognition system that allows users to experience face recognition technology using their mobile phones. The whole system is divided into mobile-end and server-end, which communicate with each other through wireless network. Figure 1 is a simple illustration of the system.

The mobile-end has a friendly user interface, allowing users to take selfies and start face recognition. The mobile application can also present results to users and guide them to give appropriate response.

The server-end loads the face recognition model, analyses faces by extracting features, calculates the similarity of the captured face image with pre-stored images, and obtains the prediction results.

The mobile-end and server-end can communicate with each other through wireless network. The mobile-end can send images to the server-end and the server-end can return prediction results to the mobile-end. Then the user can give feedback or response in terms of the face recognition prediction results. If the prediction is correct, the system will receive the signal and start updating the database. If the predicted similarity value is less than the threshold or if the prediction result is wrong, the user will be asked to give the correct name so as to help the back-end system correctly update the database.

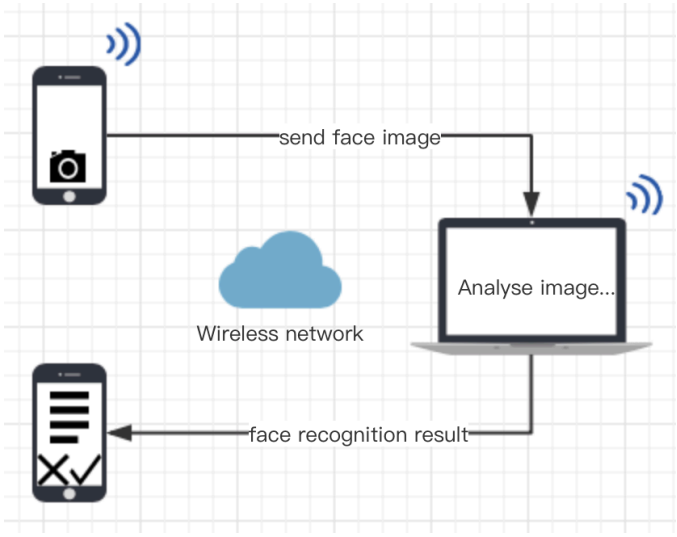


Figure 1 Simple illustration of the system

1.4 Report Structure

To begin with, the introduction part of this report describes the motivation and purpose of this project, and introduces the functions that the system finally realizes.

In Chapter 2, this paper introduces the background of this project, which focuses on the relevant background information and technology, including the concept of face recognition, the mechanisms and techniques to realize face recognition. Also, the related open source library and dataset that are used in the project are introduced.

In Chapter 3, the design and implementation process of the whole system is discussed thoroughly and clearly in this report, including the system architecture, system flowchart, the face recognition model, the mobile application and the communication mechanism.

Then in Chapter 4, a demonstration of the system and the experimental analysis of the performance of the system, are discussed in detail.

Finally, this report summarizes the work and achievements of the project, and looks forward to possible future extensions and improvements.

1.5 Development Environment

Operating System: macOS 10.14.3

Back-end development environment:

- Programming language: Python 3.3
- Integrated Development Environment: PyCharm

- Open source library: dlib

Front-end development environment:

- Programming language: Java, Extensible Markup Language (XML)
- Integrated Development Environment: Android Studio
- Software Development Kit: Android SDK

Mobile device: Redmi Note 2

Chapter 2: Background

2.1 Concept and Advantages of Face Recognition

Face recognition is a special computer technology that analyses and compares the facial features, which are often defined as high-dimensional vectors, to identify the identity of different people. The advantage of face recognition lies in its naturalness and characteristics that are not easily perceived by the individual being tested. Naturalness means that the recognition method is similar to the way humans adopt for individual identification. For example, humans also distinguish and identify different people by observing and comparing different faces. The high-concealment feature of face recognition is also important for identification, which makes the recognition method unobjectionable and not easily fooled because it is not easily noticeable.

2.2 Mechanisms of Face Recognition

In general, a face recognition system mainly consists of three steps: face detection, face feature extraction, and face classification. First, we need to detect faces in the image and ignore irrelevant part of the image, then we can use OpenCV or machine learning algorithms to extract face features and encode the faces into high-dimensional vectors. Finally, we identify the face image by calculating the Euclidean Distance or train a classifier to get the classification result.

The input of the face recognition system is usually one or a series of face images waiting to be identified. The system will compare the unidentified pictures with the pre-stored face images that have already been labelled with names in the face database. The output of the system is a series of similarities as well as the corresponding names, indicating the probability that the unknown person is believed to be a certain person.

2.2.1 Face detection

The method used in this project to detect faces is a very classic algorithm called Histogram of Oriented Gradients (HOG) [1]. It is a feature descriptor used for detecting targets. HOG will calculate the gradient direction of the local region of the image to form the feature. First, we look at the pixels and those pixels directly surrounding it, and we need to compare the lightness between the current pixel with these pixels that surround it.

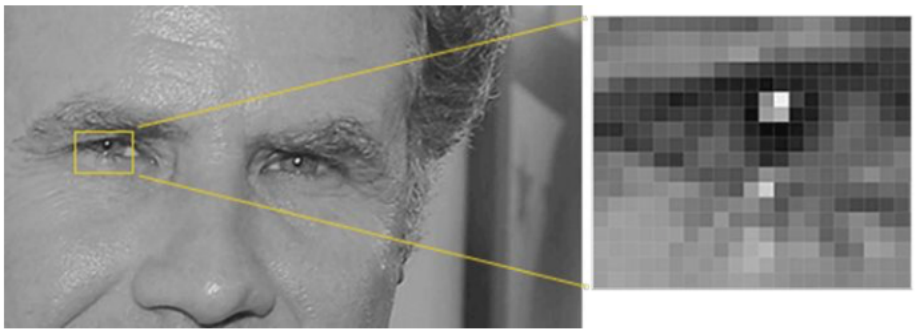


Figure 2 Compare the lightness of the current pixel and its surrounding pixels

After that, this algorithm will replace every pixel in the image with an arrow that show the flow from light to dark across the entire image, these arrows are called gradients.

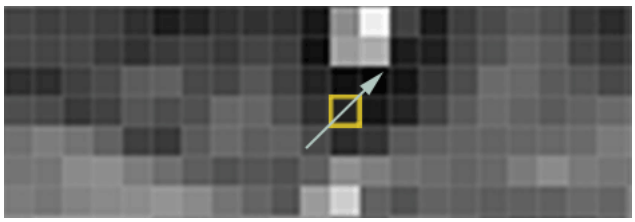


Figure 3 Image is getting darker towards the upper right

Finally, we can turn the original image into a simple representation that captures the basic structure of a face. In this way, to find a face in the HOG image, we just need to find the section which resembles most to a known HOG pattern extracted from a variety of other training faces:

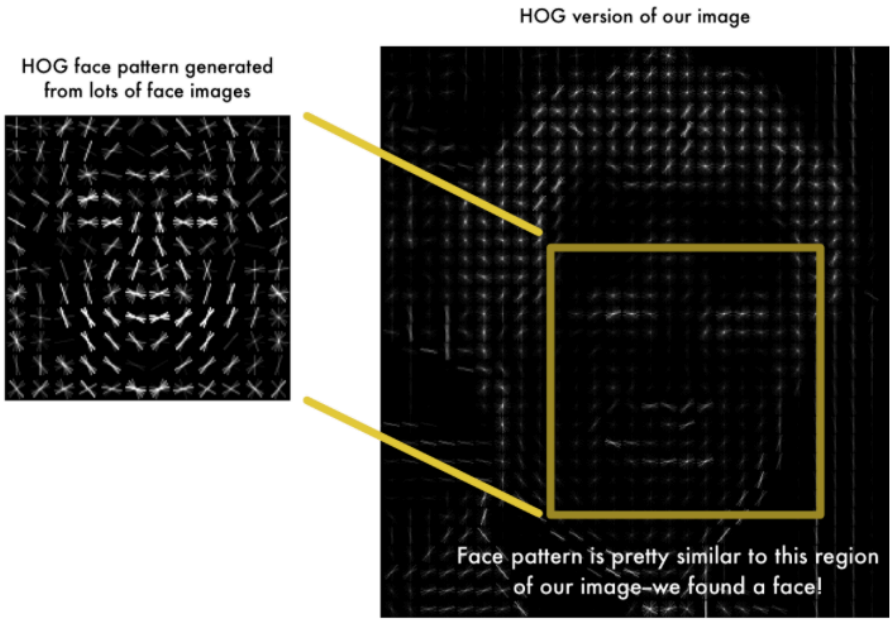


Figure 4 Find face in the HOG image

2.2.2 Face feature extraction

To extract features from faces, there are mainly two ways. First, we can use some API in OpenCV to process the images and extract features. Or we can use machine learning algorithms to train a model how to extract features. The algorithm I chose is Deep Residual Network (ResNet) proposed by He Kaiming et al in 2015 [2], which is a kind of deep convolutional neural network. This project makes use of the API of ResNet model provided in dlib library to extract features from faces. Here, this report will briefly introduce ResNet and how the face recognition model is trained.

Recent studies in the field of computer vision have shown that the depth of the network is an important factor in achieving good performance. As the number of network layers continues to increase, the accuracy of the model continues to improve. However, for deep plain network, when the number of network layers increases to a certain degree, the training error and test error increase rapidly, which means that when the network becomes deep, it will be difficult to train. This is because the neural network continuously propagates the gradient during the back propagation process, and when the number of network layers is deepened, the gradient gradually disappears during the propagation process, resulting in an inability to effectively adjust the weight of the previous network layer. This is also called the problem of degradation.

Suppose the existing shallow network has reached the saturation accuracy, then add several identity mapping layers (that is, y=x, the output is equal to the input). The depth of the network is increased, and the minimum error does not increase, that is, the deeper network should not bring about an increase in the error on the training set. The idea of using the identity mapping to directly pass the output of the previous layer to the back is the inspiration for the famous deep residual network, ResNet. ResNet introduces a residual network structure. This residual network structure allows a very deep network layer and also brings satisfying final classification result. The basic structure of the residual network is shown in the figure below.

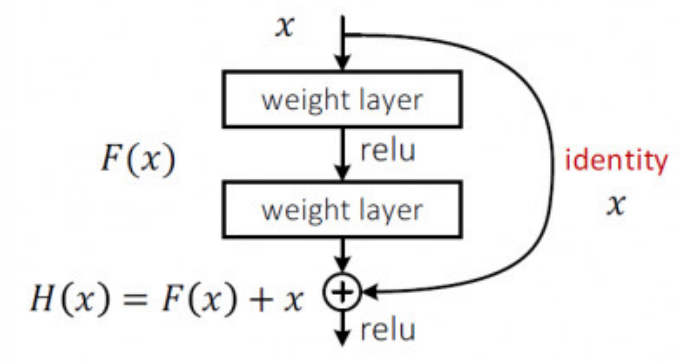


Figure 5 Basic structure of the residual network

The residual network draws on the idea of a cross-layer link in the Highway Network. Suppose that the input of a certain neural network is x, and the expected output is H(x), that is, H(x) is the expected complex potential mapping. If it is to learn such a model, the training difficulty will be relatively large;

Recalling the previous hypothesis, if we have learned relatively saturated accuracy (or when the underlying error is found to be large), then the next learning goal is transformed into the learning of the identity mapping, that is, the input x is approximated to the output H(x), to maintain the level in the back does not cause a drop in accuracy.

In the residual network structure diagram of the above figure, the input x is directly passed to the output as the initial result by the "shortcut connections" method, and the output result is H(x)=F(x)+x. When F(x) = 0, then H(x) = x, which is the identity mapping mentioned above. Therefore, ResNet is equivalent to changing the learning target, instead of learning a complete output, but the difference between the target value H(X) and x, which is called the residual F(x) := H(x) -x, therefore, the latter training goal is to approximate the residual result to 0, so that the accuracy does not decrease as the network deepens.

This kind of residual jump structure breaks the convention that the output of the traditional neural network n-1 layer can only give the n layer as an input, so that the output of a certain layer can directly cross several layers as the input of the latter layer. The significance of this is that it provides a new direction for the problem of superimposing the multi-layer network and making the error rate of the whole learning model not fall.

At this point, the number of layers of the neural network can exceed the previous constraints, reaching dozens of layers, hundreds of layers or even thousands of layers, providing feasibility for high-level semantic feature extraction and classification.

2.2.3 Face classification

To get the classification result of the unknown face image, a common method is to calculate the Euclidean Distances between the feature vectors of unknown image and that of pre-stored images. The Euclidean Distance is calculated as follows:

$$d(x,y) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \tag{1}$$

In this project, each face is encoded to output a 128-dimensional high-dimensional vector. We measure the distance of different vectors in high-dimensional space by calculating the

Euclidean Distance between them. The smaller the Euclidean Distance is, the greater the similarity between the two faces is, and vice versa.

2.3 Dlib Open Source Library

Like Open Source Computer Vision Library (OpenCV), dlib is a very useful computer vision library [3]. It is a general machine learning and data analysis library based on C++ language. Compared to OpenCV, dlib contains many of the latest algorithms, especially some advanced machine learning algorithms. Many applications can be implemented using these algorithms, such as face detection, vehicle detection, target tracking, semantic segmentation, and more.

After careful consideration and investigation, this project finally makes use of the dlib open source library to build a face recognition model. The reason is that this project requires the application to recognize people in the wild fastly and accurately, not only the people who have been recognized by the system before but also people who are brand new to the system. Consequently, the relatively mature face detection and feature extraction models in dlib library can help the system achieve ideal face recognition performance, especially under the circumstance that a new user begins to use the system.

2.4 Labeled Faces in the Wild dataset

The main dataset used in this project is called Labeled Faces in the Wild (LFW) dataset [4], which is a very famous public dataset used for evaluating the face recognition algorithm. LFW is one of the most important datasets in the field of face recognition. It was created to investigate the face recognition problem in an unrestricted environment. This collection contains more than 13,000 face images, all from the internet, not the lab environment. Each face is standardized by a person's name. Among them, about 1680 people have more than two faces. LFW dataset mainly tests the accuracy of face recognition.

Chapter 3: Design and Implementation

3.1 User requirements analysis

This project aims at designing and implementing a face recognition system that enables users to use mobile applications to experience and utilize face recognition technology. After carefully investigating and analysing the possible requirements of this mobile phone-based face recognition application, the user requirements of the system are finally identified as follows:

- The software should have a user-friendly graphic user interface (UI), allowing users to easily operate the software with common sense or simple instructions.
- Users will have a comfortable and non-harm experience using the system and get a fantastic feeling about being recognized by the system.
- Users can take a photo and see the preview of the image.
- Users can choose to retake the photo if they are not satisfied.
- Users can confirm the photo when they are ready and start the recognition.
- Users would not wait too long to get the identification results, so the process should be as quick as possible.
- As a new user, the system can properly guide him or her to upload required information.
- New users can be recognized next time after they have uploaded their information.
- Users will get simple and clear recognition results from the system.
- Users can give appropriate feedbacks to the system according to the recognition results.

3.2 Analysis and design of the overall system

Figure 6 shows the overall architecture of the system, which consists of two major parts: the server-end on PC and the front-end on mobile application.

The mobile application has 4 key modules. The main interface module displays the guidance information, and allows users to open the camera. The module will process the photo and sends the photo to the server. After receiving the classification results, the module will activate the results display module or the new user module. The camera module allows users to use system camera to capture face photo, and returns the image to main interface module, which will then store the image and send it to the server through wireless network communication. The results

display module is responsible for displaying the face recognition results on the user interface of the mobile application, and providing ways for users to give feedback about the results. The new user module is responsible for getting the correct name of the user and sending the name to the server if a new user is using the system or the user finds that the classification result is wrong.

As for the server-end, there are 3 key modules. The face recognition module contains multiple key functions including face feature extraction, face classification, and system update. The face recognition model is loaded on this module and will analyse the receiving image. The encoding operation module contains a series of functions related to encoding processing operations, including feature import, feature output and feature update. The communication module is responsible for receiving image and feedback from users, and sending classification results and confirmation signals to the front-end.

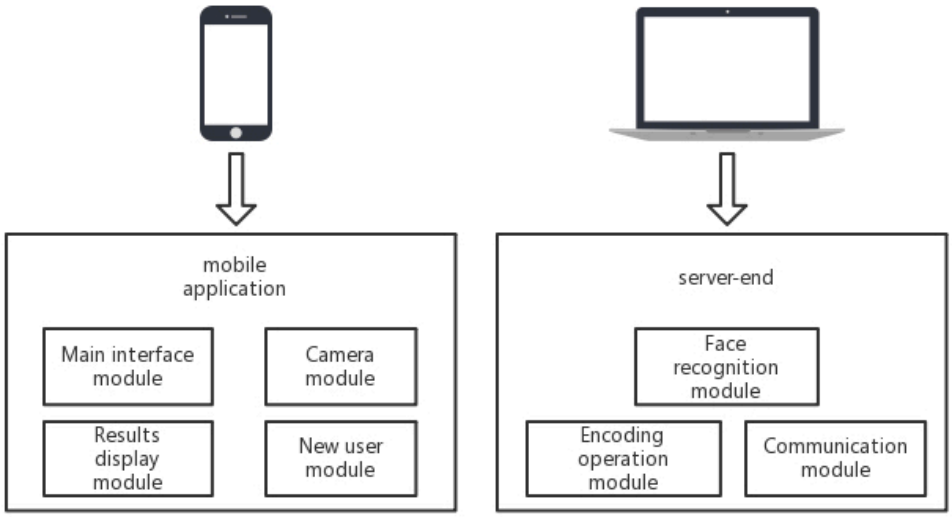


Figure 6 System architecture

3.3 How users interact with the system

Figure 7 shows the system flowchart and explains how users interact with the system.

i) To begin with, the user needs to open the face recognition app, and click the corresponding button to turn on the camera. The camera will be called to capture the face of the user. The user can choose to retake the photo or confirm the photo.

ii) Once the photo is confirmed, the face image will be transmitted to the server, which is the back end, to activate the running of the face recognition algorithm. The algorithm will extract features of the photo waited to be matched, and compare the unknown

encodings with the encodings of the pre-stored images, and get the classification results. Then the server will return the classification results, including the name as well as the similarity it calculates, to the mobile end.

iii) The mobile application will receive the classification result returned by the server, and present it on the user interface. If the classification result is correct, the user can confirm the result, so the match is successful and the system will update local data files on the server. If the classification result is wrong, the user can inform the application, and enter the correct name of him/her. Finally, the photo to be matched will be included into the corresponding local data file.

iv) If the similarity it calculates is below a certain threshold, the algorithm will consider the input image a brand-new data, i.e. this is a new user, and ask the user to offer the required information, e.g. his/her name. Then the face recognition system will first check if the information provided by the user can be found on the server, if not, then it will create a new local folder and include the new face image into the local folder. Also, the system will create a text file in this folder to store the encodings. As a result, when the new user uses the system next time, the system will be able to output his/her name and the similarity it calculates.

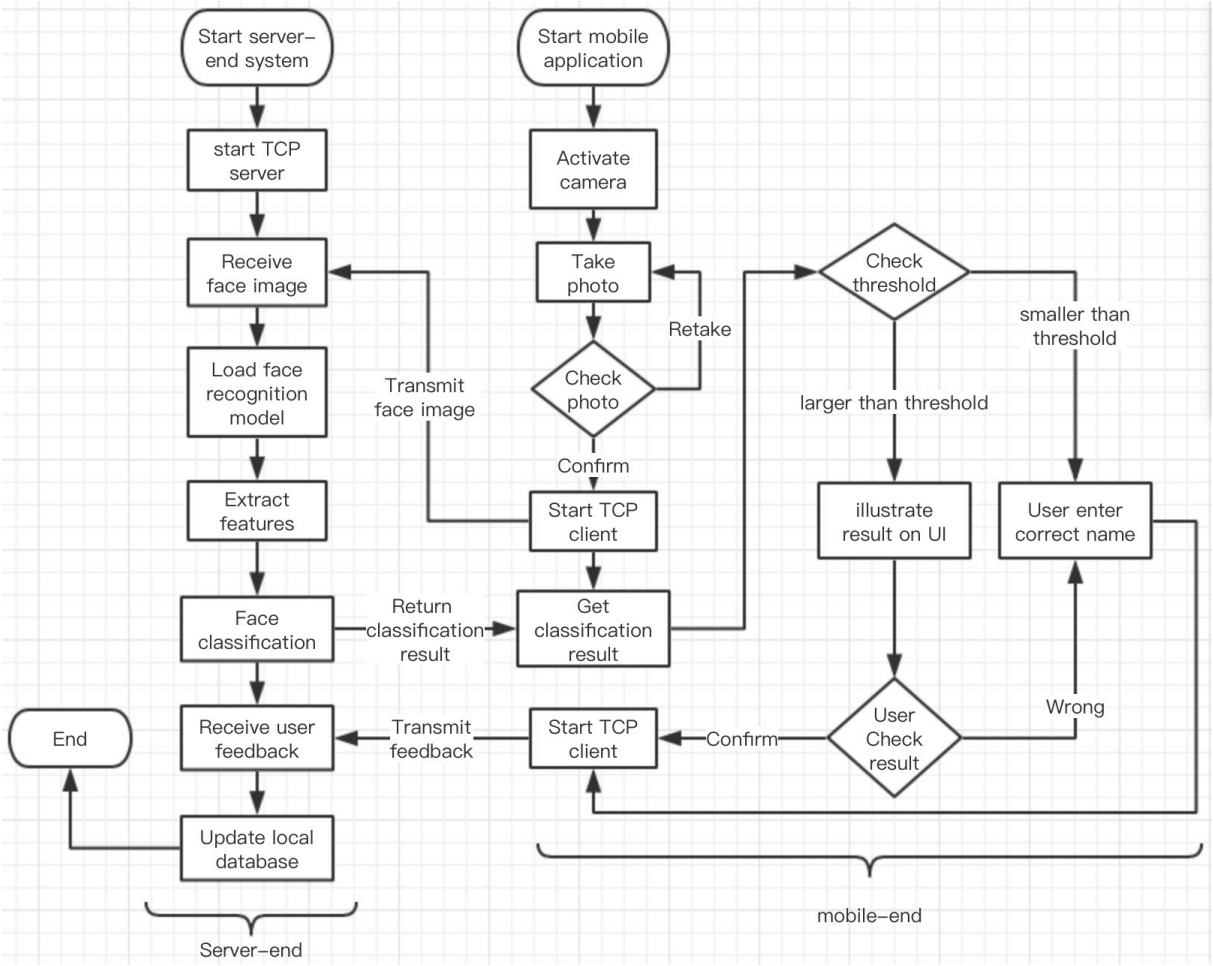


Figure 7 Flowchart of the overall face recognition system

3.4 Design and implementation of face recognition model

3.4.1 Feature extraction

This project uses the HOG algorithm and the ResNet model in the dlib open source library to detect faces and extract features. By extracting features of a certain face, we can map the feature vector of the face into a 128-dimensional high-dimensional space to quantify the face, which is convenient for comparing the differences between different faces. Specifically, first, we load the detection operator using HOG for face detection in the dlib library:

```
# Create a HOG face detector using the built-in dlib class
face_detector = dlib.get_frontal_face_detector()
```

Figure 8 Load the detection operator

Then the detection operator is used to detect the image and get the boundary of the face in the image:

```
# Run the HOG face detector on the image data.
# The result will be the bounding boxes of the faces in our image.
detected_faces = face_detector(image, 1)
```

Figure 9 Detect image

After that, the shape predictor is used to find the landmarks of the face, the algorithm is called face landmark estimation. The main idea is to find 68 specific points that are common to people's faces – including the outline of eyes, eyebrows, chin, lips, etc. Here the landmark model provided by dlib library is used to get the pose of the face and localize the face.

```
face_pose_predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
pose_landmarks = face_pose_predictor(image, face_rect)
```

Figure 10 Landmark estimation

Then the pre-trained ResNet model is loaded to get prepared for the following analysis of the face image. The ResNet model provided in dlib library is open in the public domain and free to download.

```
facerec = dlib.face_recognition_model_v1('dlib_face_recognition_resnet_model_v1.dat')
```

Figure 11 Load ResNet model

After a series of processing, the loaded face recognition model is used to analyse the face image and calculate a 128-dimensional feature vector. As shown in the figure, face_descriptor is a 128-dimensional feature vector deeply and comprehensively describing the features of the face image analysed and calculated by the face recognition model.

```
face_descriptor = facerec.compute_face_descriptor(image, shape)
print(face_descriptor)
```

Figure 12 Analyse face image using the model

After encapsulating the above operations, I wrote a module called *face_recog_system.py* in Python as the core module for implementing the back-end face recognition system, where there are several crucial functions in this module as show in Figure 8:

```
class FaceRecognition:
    def __init__(self):...

    def face_feature_extraction(self, face):...

    def face_classification(self, pic_name):...

    def update_system(self, feedback, unknown_encoding, result_name, unknown_path):...

    def update_face(self, result_name, unknown_path):...

    def create_new_person(self, new_name):...

    def system_test(self):...
```

Figure 13 Main functions of the face_recog_system module

Among them, function *face_feature_extraction()* will uniformly call the operations encapsulated above and return the feature vector output of the unknown image after the face recognition model extracts the features.

```
def face_feature_extraction(self, face):
    unknown_image = face_recognition.load_image_file(face)
    unknown_encoding = face_recognition.face_encodings(unknown_image)[0]
    # print(unknown_encoding)
    return unknown_encoding
```

Figure 14 Function of feature extraction

3.4.2 Feature output and import

This part is designed to improve the speed greatly by extracting and saving the feature values of the images. Previously, the process of extracting the features and comparing the images is required for each match, which costs a large amount of time since the feature extraction process costs a lot of calculation. Thus, the system is improved by extracting the features of the local images, processing the feature values, and outputting the features into a text file called *name_encodings.txt* in the appropriate format. In this way, the model would not need to do feature extraction for a large number of pictures every time. The only thing the model needs to do is to import the feature values into the dictionary from the encoding file and then uses the features for classification. To implement the feature output and import part, a particular module called *encoding_operation.py* is written, which contains a series of functions related to the encoding operations.

```
class EncodingOperation:
    # encoding_operation module: this class includes methods related to encoding operations
    def __init__(self, path_faces):
        self.all_face_encodings = defaultdict(lambda: [0])
        self.face_encodings = []
        self.local_faces = path_faces
        # self.output_encodings(self.path_faces)
        # self.import_encodings()

    def import_encodings(self):...

    def update_encodings(self, unknown_encoding, update_person):...

    def write_encodings(self, path, face_encodings):...

    def output_encodings(self):...

    def check_encoding_file(self, name_face, faces):...
```

Figure 25 Main functions of the encoding_operation module

Among these functions, import_encodings() and output_encodings() can import and output the feature values appropriately. Specifically, a data structure called list is used to store the feature values of a particular face image, and another special data structure in Python called dictionary

is used to temporarily store the face encodings of all people in the database. The format of the dictionary that is used to store the encodings is like this: {name1: [[], [], [], …, []], name2: [[], [], [], …, []], …, nameN: [[], [], [], …, []]}. Each name corresponds to a nested list, in which each list represents a 128-dimensional feature vector that quantifies a certain face image. The function output_encodings() will convert the list that stores a person's all face encodings into string format, and write them all into a text file. The function import_encodings() will repeatedly read every text file that corresponds to a certain person, and import them into the dictionary. After importing the features in this way, it would be much easier to conduct face classification then.

3.4.3 Face classification

To implement face classification, a function called face_classification() in the module *face_recog_system.py* is written to calculate the distances between the unknown image and the pre-stored images by using Euclidean Distance. Since a certain person may have multiple face images in the local database, the similarity is derived by calculating the average similarity. After comparing the unknown face image with all the prestored faces, the system will get a dictionary called name_sim that contains all the names and their corresponding average similarities. Finally, the system will select out the name with the highest average similarity and compare the similarity with the threshold. If the similarity is less than threshold, the system will return String "new_user" to the mobile application, indicating that the system believes the unknown image is probably a new user. Otherwise, the system will combine the name and the average similarity and return the classification result.

```
def face_classification(self, pic_name):
    self.unknown_path = pic_name
    self.unknown_encoding = self.face_feature_extraction(self.unknown_path)
    self.all_face_encodings = self.encoding.import_encodings()
    keys = sorted(self.all_face_encodings.keys())
    print(keys)
    name_sim = defaultdict(lambda: 0)
    for key_name in keys:
        results = face_recognition.face_distance(self.all_face_encodings[key_name], self.unknown_encoding)
        avg_sim = 1 - results.mean()
        print("\nname: " + key_name + "\ndistances: " + str(results) + '\naverage similarity: ' + str(avg_sim))
        name_sim[key_name] = avg_sim

    self.result_name = max(name_sim, key=name_sim.get)
    num = name_sim[self.result_name]
    if num < 0.6:
        max_name_sim = 'new_user'
    else:
        max_name_sim = self.result_name + "_" + str(round(num, 4))

    return max_name_sim
```

Figure 36 Function of face classification

Here is an example that shows the similarity between an unknown image and the person called langcheng. Each value in the list represents the distance between the unknown image and a single face image that belongs to langcheng.

```
name: langcheng
distances: [0.68756386 0.61913074 0.65005501 0.65869888]
similarity: 0.3461378766517391
```

Figure 47 An example of the face classification result

3.4.4 System update

After the user gives feedback about the face recognition results, the backend system will start updating according to the feedback.

To be specific, if the feedback is “confirm”, then the system just needs to first update the feature values of the corresponding person in the local database. The specific function that implements this function is called *update_encodings()* in the *encoding_operation.py* module. It will first import the feature values from the encoding text file into a list, then append the unknown encoding of the current image onto the list. After that, the list is written into the text file again. The second step is to call *update_face()*, which will move the current face photo to the folder of the corresponding person.

Otherwise, the feedback given by the user is a new name. The system will first create a new folder by calling *create_new_person()* to store the information of the user, then move the image into this folder by calling *update_face()*, and finally output the encoding into an encoding file that is created in the folder by calling *write_encodings()*, which is a function in the *encoding_operation.py* module.

```
def update_system(self, feedback, unknown_encoding, result_name, unknown_path):
    if feedback == 'confirm':
        print("\nOkay! start updating...")
        self.encoding.update_encodings(unknown_encoding, result_name)
        self.update_face(result_name, unknown_path)
    else:
        new_name = feedback
        encoding_file_path = self.local_faces + new_name + '/' + new_name + '_encodings.txt'
        new_encoding = []
        new_encoding.append(unknown_encoding.tolist())
        self.create_new_person(new_name)
        self.update_face(new_name, unknown_path)
        self.encoding.write_encodings(encoding_file_path, new_encoding)
```

Figure 18 Function of system update

3.5 Design and Implementation of Mobile Application

3.5.1 Prototype of the application

To design the mobile application, I first designed the prototype of the application, i.e. all the interfaces of the application and the corresponding layout and elements. I divided the application into four parts, the main interface, the camera interface, the results display interface, and the user feedback interface.

3.5.2 Activities and methods

After designing the prototype of the application, I followed [5] to learn how to use Android Activity class to develop and implement specific application interfaces and functions. The activity represents a single screen with a user interface, like a Java window. Activity is an important component of Android. It provides a visual interface for user interaction. The user interacts with the application through the Activity to complete relevant operations. An Activity is a component of an application that provides an area on the screen where users can do interactive operations such as sending text messages, viewing maps, or taking photos. In general, an app allows multiple activities, and there will be a main activity usually. When the user opens the application for the first time, the application will present the main interface. An activity can trigger some other activities, such as taking a photo or entering personal information, so as to complete different interactions.

The Android system initializes its program by starting calling the onCreate() callback in the activity. The life cycle of the Activity is shown as Figure 19:

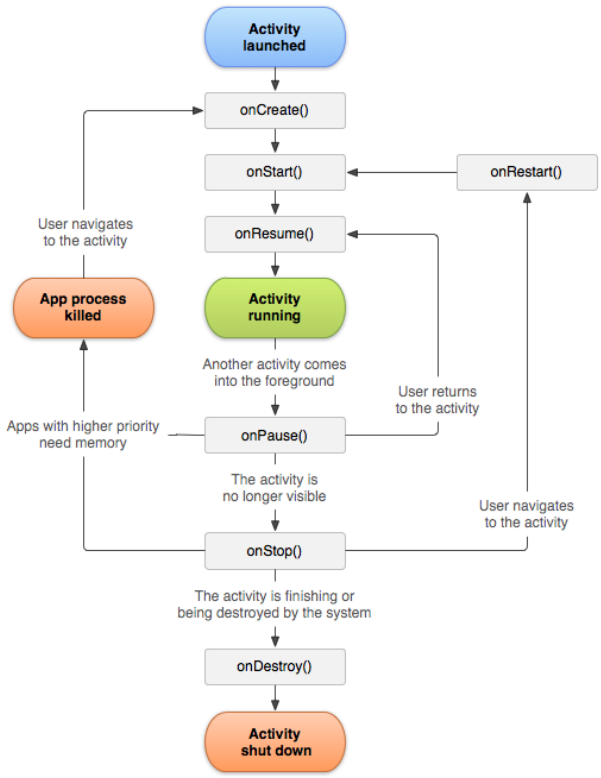


Figure 19 Life cycle of the Activity

Specifically, there are finally three activities designed in the face recognition mobile application.

The first activity is the main activity, which is called “MainActivity” and presents the main interface of the application, including welcome messages, application guidance information, and buttons for taking pictures. Users start using this face recognition application from the main interface. They can take a selfie, and confirm or retake the photo. The application is able to send the captured photo to the server-end for analysis, receive the classification results from the server, and check the results for further operations. The main activity contains the following key methods:

```
protected void onCreate(Bundle savedInstanceState) {...}
private void initViews() {...}
private void dispatchTakePictureIntent() {...}
protected void onActivityResult(int requestCode, int resultCode, Intent data) {...}
private void runTcpClient(final Bitmap bitmap) {...}
public void sendImgMsg(DataOutputStream out, Bitmap bitmap) throws IOException {...}
public String recvMsg (DataInputStream dataIn) throws IOException{...}
protected void checkResults(String predResults){...}
protected void newUser() {...}
protected void showPredResults(String picPath, String predResults){...}
```

Figure 20 Key methods of the main activity

Methods onCreate() and initViews() will set the content view of the main activity, including the camera, the image and the guidance information. dispatchTakePictureIntent() will create an intent to call the Camera API that allows users to use the system camera to take photos. After taking the photo, onActivityResult(), runTcpClient() and sendImgMsg() will send the photo to the server. After receiving the classification results in recvMsg(), checkResults() will check the message and determine which activity to activate next, by calling either newUser() if the system believes this is a new user, or showPredResults() if the similarity is above the threshold.

The second activity is designed to display the prediction results, which is called "ShowSmilarity". The user can review the photo just taken, the predicted similarity as well as the recognition name given by the system. The similarity value represents the degree of confidence of the system in the recognition result. Also, the user can give corresponding feedback according to the result. If the user chooses "Yes, it's me.", the feedback will be transmitted to and received by the server-end, who then knows that the user confirms the prediction result and will update the system. If the user chooses "No, it's not me.", the third activity will be activated to get the real name of the user. The second activity mainly includes the following key methods:

```
protected void onCreate(Bundle savedInstanceState) {...}
protected void initViews() {...}
protected void setViews() {...}
protected void extractString() {...}
protected void buttonNoAct() {...}
protected void buttonYesAct() { confirmResults(); }
protected void confirmResults(){...}
public void sendTextMsg(DataOutputStream out, String msg) throws IOException {...}
public String recvMsg (DataInputStream dataIn) throws IOException{...}
```

Figure 21 Key methods of the second activity

Similar as other activities, the first three methods are used to set the content view of the activity, including the text, image and buttons. Method extractString () will process the classification results including the similarity and name returned by the server, so setViews() can display them on the interface. buttonNoAct() is called when the user thinks the results are not correct and clicks the "No, it's not me" button, and then the third activity will be activated. buttonYesAct() is called when the user confirms the results. Methods confirmResults(), sendTextMsg() and recvMsg() are the communication methods that enable the activity to send the confirmation

feedback to the server-end and ensure the feedback has been successfully received by the server-end. A TCP client will be created to connect to the server-end.

The third activity is designed to get the real name of the user. This activity will be activated under two circumstances. The first condition is that the similarity is less than the threshold, so the system believes this is probably a new user, and will directly activate the third activity to ask the user to input his or her name instead of activating the second activity that shows the classification result. The second condition happens when the user denies the results on the second activity, so the user still needs to input the correct name in order to help the system update the database correctly. The key methods of the third activity are shown in Figure.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_new_user);

    initViews();
    setViews();
}

protected void initViews() {...}

protected void setViews() {...}

protected void sendUpdateResults(final String realName){...}

public void sendTextMsg(DataOutputStream out, String msg) throws IOException {...}

public String recvMsg (DataInputStream dataIn) throws IOException{...}
```

Figure 22 Key methods of the third activity

The first three activities will set the content view of the activity, including the text, text box and button. setViews() will set click listener onto the button. The last three activities are the communication methods that will be called when the user enters the name and clicks the “OK” button, and the real name given by the user will be sent to the server. recvMsg() is used to receive the signal from the system to ensure the feedback given by the user has been successfully sent to the server, and an alert dialog will appear on the interface.

3.5.3 User Interface (UI) layout

The layout determines the visual framework of a UI, for example, a user interface of an activity. We can declare UI elements in XML. Android provides concise XML vocabulary for the View class and its subclasses, such as vocabulary for widgets and layouts; we can leverage the XML vocabulary in Android in the same way that we create web pages with a series of nested elements in HTML to quickly design the UI layout as well as the screen elements it contains. The advantage of declaring a UI in XML is that we can better isolate the appearance of the application from the code that controls the behaviour of the application. The UI description is

outside the application code, which means we don't have to modify the source code and recompile when modifying or adjusting the description. For example, we can create XML layouts suitable for different screen orientations and sizes. In addition, declaring layouts in XML makes it easier to display the structure of the UI, simplifying the problem debugging process.

Figure 23 is the user interface of the first activity. I used the ConstraintLayout provided in Android Studio development to design the size and relative position of each element. As shown in the figure, each element is relatively limited to a certain area, so that it can adapt to different sizes of mobile phone screens. The text is the guide information, and the user clicks the camera button below to trigger the corresponding event and open the system camera.

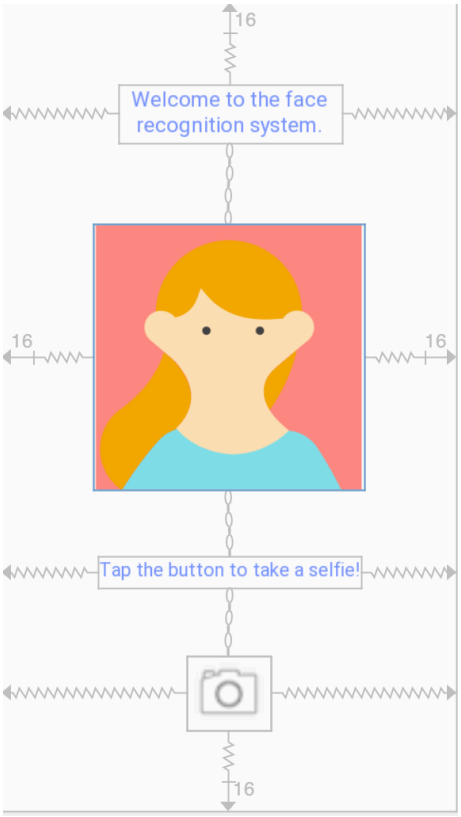


Figure 23 User interface of the first activity

Figure 24 is the user interface of the second activity. Each element is relatively limited in a certain position. For example, the picture and text are centred, and the top of the picture to the parent is symmetrical to the bottom of the picture to the text. The image in the interface will show the photo that the user just took, and the similarity and name will show the results of the classification just returned by the server. The two buttons below are provided for the user to give feedback. If the user denies the result, then he will be directed to the interface of the third activity. If the user confirms the result, an alert dialog will appear to inform the user that the

feedback has been received by the system.

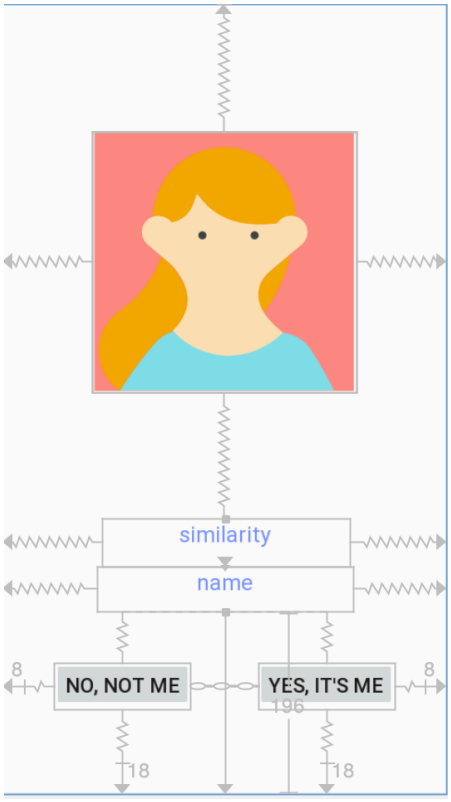


Figure 24 User interface of the second activity

Figure 25 is the user interface of the third activity. The text part is the guidance information. The user can enter his own name according to the instructions. Clicking the OK button will send the feedback to the server, and an alert dialog will appear to inform the user that the feedback has been received by the system.

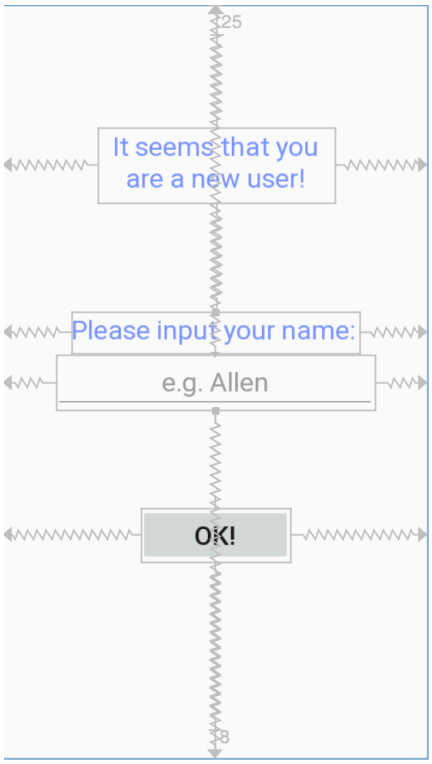


Figure 25 User interface of the third activity

3.6 Communication between the Front-end and Back-end

Since the overall system is divided into two parts, i.e. the front-end and the back-end, the communication mechanism becomes an essential component or module of the system. The following part will introduce the details of the communication process, including the methods used, the image transmission, the classification result and user feedback transmission. Some of the points worth mentioning in implementing the communication mechanism will also be discussed in detail.

3.6.1 Socket programming

What is socket? A socket is a communication mechanism by which client-server communication can be performed either locally or across a network [6]. Sockets are the cornerstone of communication and are the basic operating unit for network communications that support the TCP/IP protocol. It is an abstract representation of endpoints in the process of network communication. It contains five kinds of information necessary for network communication: the protocol used for connection, the IP address of the local host, the protocol port of the local process, the IP address of the remote host, and the protocol port of the remote process.

When the application layer communicates data through the transport layer, TCP encounters the

problem of providing concurrent services for multiple application processes at the same time. Multiple TCP connections or multiple application processes may need to transfer data over the same TCP protocol port. To distinguish between different application processes and connections, many computer operating systems provide a socket interface for applications to interact with the TCP/IP protocol. The application layer can communicate with the transport layer through the Socket interface to distinguish communication from different application processes or network connections, and implement concurrent services for data transmission.

In order to establish a Socket connection, at least one pair of sockets is required, one of which runs on the client side, called ClientSocket, and the other runs on the server side, called ServerSocket. The connection process between sockets is divided into three steps: server listening, client request, and connection confirmation.

Server listening: Server-side sockets do not locate specific client sockets, but are in a state of waiting for a connection, monitoring the network status in real time, waiting for a connection request from the client.

Client request: client's socket make a connection request, and the target to be connected is the server-side socket. To do this, the client's socket must first describe the socket of the server it is connecting to, indicating the server-side socket's address and port number, and then make a connection request to the server-side socket.

Connection confirmation: When the server-side socket listens to or receives a connection request from the client socket, it responds to the request of the client socket, creates a new thread, and sends a description of the server-side socket. To the client, once the client confirms the description, the two parties formally establish a connection. The server-side socket continues to be listening and continues to receive connection requests from other client sockets.

Figure 26 is a typical socket workflow. The left column represents the server-side and the right column represents the client-side. First, the server-side initializes the Socket, then binds to the port, and listens to the port, then calls accept() and blocks, waiting for the client to connect. The client first initializes a Socket and then calls connect() to establish a connection to the server and initiate a three-way handshake. The handshake step ensures that the client and server can reach each other. If the connection is successful, then the client-server connection is established. The client and server then use send() and recv() to exchange data. The client sends a data request, and the server receives the request and processes the request, and then sends the response data to the client, and the client reads the data. At the end, the client and server close their respective

sockets, and one interaction ends.

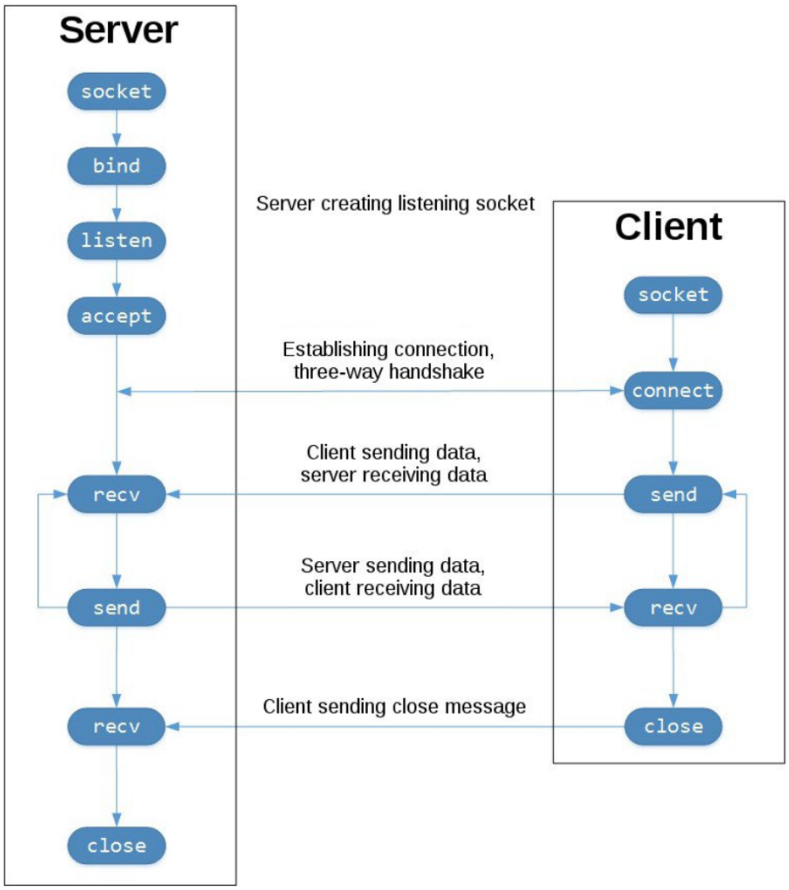


Figure 26 A typical socket work flow

In this project, specifically, the communication mechanism is implemented by making the back-end system as the TCP server end and the mobile application as the TCP client end. The ServerSocket runs on the server and the ClientSocket runs on the mobile application.

First, the server calls socket.socket() to create the socket.

```
class Server:
    def __init__(self):
        self.host = '193.169.1.107'
        self.port = 9999    # Arbitrary non-privileged port
        # 创建服务器套接字，AF_INET 表示连接使用ipv4地址族 SOCK_STREAM表示用流式套接字
        self.tcpSerSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket_bind(self.host, self.port, self.tcpSerSocket)
        self.data_length = 0
```

Figure 27 Create socket

Next, the socket calls bind() to bind the host IP and port to the socket, and calls listen() to listen on the port.

```
def socket_bind(self, host, port, tcp_ser_socket):
    tcp_ser_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    # 将套接字绑定该地址
    tcp_ser_socket.bind((host, port))
    # 开始监听TCP传入连接。参数指定在拒绝连接之前，操作系统可以挂起的最大连接数量。
    tcp_ser_socket.listen(4)

    hostname = socket.gethostname()
    print('hostname: ', hostname)
    print('host ip: ', self.get_ip_addrs(hostname))
    self.wait_connection(tcp_ser_socket)
```

Figure 28 Bind socket

The socket then calls accept() to wait for the client's connection. The returned address parameter is the client's IP and port, which uniquely identifies the client process on the network.

```
def wait_connection(self, tcp_ser_socket):
    while True:
        print('\nWaiting for connection...')
        tcp_cli_socket, address = tcp_ser_socket.accept()
        print('Connected with...', address, " >--- mobile-end")
```

Figure 29 Wait for connection

On the client side, the socket is first created and a connection request is made to the server-side socket. It should be noted that the IP of the client and server need to be in the same LAN. After successfully establishing a connection with the server, tcpclient creates an output stream and an input stream, and calls sendImgMsg( ) to send the image data with the output stream, and calls recvMsg () to receive the classification results returned by the server with the input stream. After the image is sent, we need to call the Socket shutdownOutput() method to close the output stream, thus informing the end of the data flow. Note that socket.getInputStream().close() cannot be called, because it will cause the socket to be closed.

```
private void runTcpClient(final Bitmap bitmap) {
    Thread thread = new Thread() {
        public void run(){
            try {
                Socket tcpCliSocket = new Socket( host: "193.169.1.107",  port: 9999);
                DataOutputStream dataOut = new DataOutputStream(tcpCliSocket.getOutputStream());
                DataInputStream dataIn = new DataInputStream(tcpCliSocket.getInputStream());

                sendImgMsg(dataOut, bitmap);
                // shutdown the output stream to inform the end of the data flow
                tcpCliSocket.shutdownOutput();
                // receive the similarity and name returned by the server
                String predResults = recvMsg(dataIn);
                // check the results to display the corresponding activities
                checkResults(predResults);
                // close connection
                tcpCliSocket.close();
            } catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    };
    thread.start();
}
```

Figure 30 TCP client

In the server-end, after receiving the image data by calling receive_image(), the server sends the image to the FaceRecognition module, and calls the face_classification() function to get the classification results returned by the face recognition model, i.e. max_name_sim, and then sends the results to the client by calling send_data().

```
self.receive_image(tcp_cli_socket, pic_file)
pic_file.close()
print("received successfully.\n")

start_recognition = FaceRecognition()
max_name_sim = start_recognition.face_classification(pic_path_name)
self.send_data(tcp_cli_socket, max_name_sim)
tcp_cli_socket.close()
```

Figure 31 Server receives image and sends results

3.6.2 Image transmission

Images are different from ordinary text data. Text can be converted directly into byte stream for transmission, and the data size is usually small. The image is usually large and cannot be transmitted directly. Therefore, special processing is required when transmitting images. In order to transfer image data, first we need to represent the image as a bitmap format and compress it. Bitmap is an image represented by a pixel array. Then we pass the bitmap to runTcpClient(), which will call sendImgMsg() to send the image data.

```
// take out the locally saved image and store it in bitmap format
Bitmap bitmap = BitmapFactory.decodeFile(
        pathName: Environment.getExternalStorageDirectory() + "/image.jpg");
// compress image
Bitmap newBitmap = ImageTools.zoomBitmap
        (bitmap, width: bitmap.getWidth() / SCALE, height: bitmap.getHeight() / SCALE);
// start tcp client thread, transmit newBitmap
runTcpClient(newBitmap);
// recycle memory to avoid out of memory error
bitmap.recycle();
```

Figure 32 Represent image in bitmap format

Next, when sending the image data, we need to convert the bitmap to a byte array, and get the length or size of the stream. After that, we first send the size of the packet (type long with 8 bytes), and send the image data next.

```
public void sendImgMsg(DataOutputStream out, Bitmap bitmap) throws IOException {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    // read bitmap into ByteArrayOutputStream
    bitmap.compress(Bitmap.CompressFormat.PNG, quality: 100, baos);
    // get the size of the data stream
    long len = baos.size();
    byte[] bytes = baos.toByteArray();
    out.writeLong(len); // send data size
    out.write(bytes); // send image data
    Log.i( tag: "TcpClient", msg: "sent: " + bytes);
    out.flush();
    // out.close();  一定不能close! tcp是全双工，输出流close会把socket也关闭
}
```

Figure 33 Client sends image data

When receiving an image on the server side, we need to pay attention to the fact that it is not received at one time, which is related to the TCP protocol. The default internal sending buffer size of Socket is about 8K, but no matter how big the packet is, Ethernet will be limited to 46-1500 bytes. If this amount is exceeded, TCP will set offset for IP datagram and transmit the packet in fragments. Therefore, we need to repeatedly receive the image data until there is no data.

```
def receive_image(self, tcp_cli_socket, pic_file):
    print(tcp_cli_socket)
    i = 0
    # use recv(param) to receive the length of the bytes stream first, which is 8 bytes
    data = tcp_cli_socket.recv(8)
    self.data_length = struct.unpack('>q', data)[0]  # read length first
    print("data size: " + str(self.data_length) + " Bytes")
    # repeatedly receive the content of the image
    while len(data) != 0:
        print("order: " + str(i))
        # then read content according to data length
        data = tcp_cli_socket.recv(self.data_length)
        pic_file.write(bytearray(data))
        i = i + 1
```

Figure 34 Server receives image data

3.6.3 Classification results and feedback transmission

The classification results sent by the server and the feedback sent from the mobile application all belong to String text, so they can be sent more easily compared with the transmission of the image. The first step is to simply convert the String into bytes by calling encode() function, then we can use sendall() to send the results to the front-end.

```
def send_data(self, tcp_cli_socket, max_name_sim):
    print("\n" + max_name_sim)
    send_data = max_name_sim.encode()
    print(send_data)
    tcp_cli_socket.sendall(send_data)
    print("server sent data: " + str(send_data))
```

Figure 35 Server sends classification results

When receiving the results in the front-end, the first step is to read the bytes from the data input stream, and then convert the bytes into String, so we can successfully obtain the results. The feedback transmission is in the same way.

```
public String recvMsg (DataInputStream dataIn) throws IOException{
    Log.i( tag: "TcpClient",  msg: "\nstart receiving message");
//      创建这个长度的字节数组
    byte[] bytes = new byte[1024];
//      read应该类似于recv，有阻塞机制，返回了bytes长度
    int n = dataIn.read(bytes);
    System.out.println(n);
//      将字节数组转为String
    String msg = new String(bytes,  offset: 0, n);
    Log.i( tag: "TcpClient",  msg: "received: " + msg);
    return msg;
}
```

Figure 36 Client receives classification results

Chapter 4: Results and Discussion

This project implemented a face recognition system that consists of a front-end mobile application and a back-end server. Users can make use of the system to experience face recognition technology with clear instructions and simple operations. Even for new users, the system can guide them to upload their information and will store the information in the database, so they will be recognized by the system next time. To show the outcome of the project more clearly, here a demonstration of the use of the system will be given first, then an experimental analysis of the performance of the face recognition system will be introduced and discussed.

4.1 Demonstration of the system

Assume that a new user starts to use the system. First, click on the camera button on the main interface and the camera interface will be opened. We can retake the photo by selecting the "X" in the lower left corner or confirm the photo by selecting "√" in the lower right corner. If the photo is confirmed, the app will send the photo to the server for face recognition.

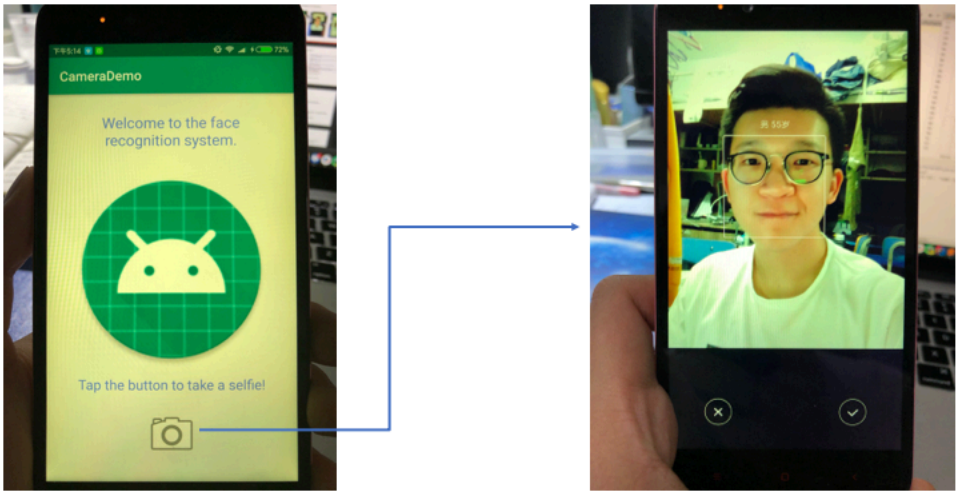


Figure 37 Main interface and camera interface

Since this is a new user, the highest similarity analysed by the system is less than the threshold (the threshold of this project is set to 0.6), so the new user interface will be activated. The user will be prompted to enter his own name. Clicking the "OK!" button will appear an alert dialog informing the user that the system has received feedback.

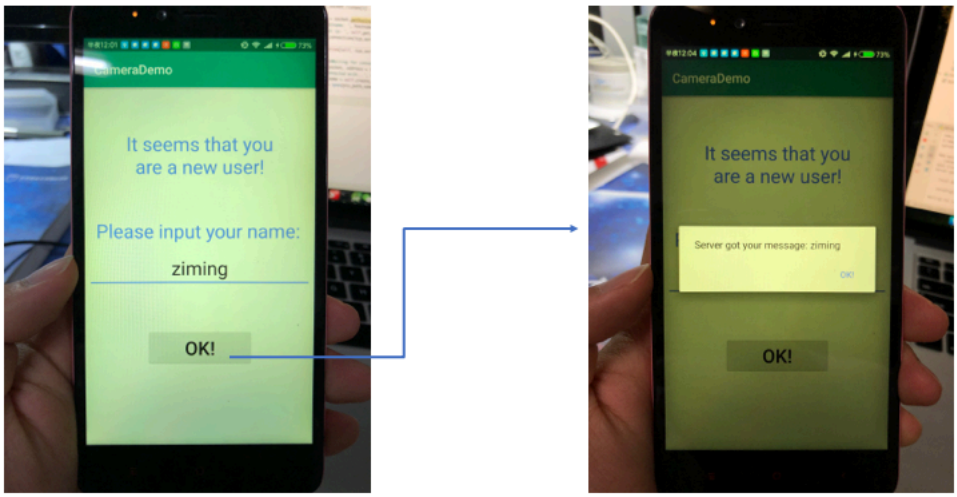


Figure 38 New user enters the correct name

The second time the user uses the app, the user opens the main interface and takes a photo of himself, just like the previous operations. The difference is that after confirming the photo, the system will be able to identify the user and return the name and similarity of the face recognition. The app will present the results on the interface. As shown in figure 39, the system predicts a similarity of 82%, and the name is ziming. If the user confirms the result, the alert dialog that the system has received the feedback will appear. Otherwise, the user will be redirected to the interface in Figure 38 and enter the correct name.

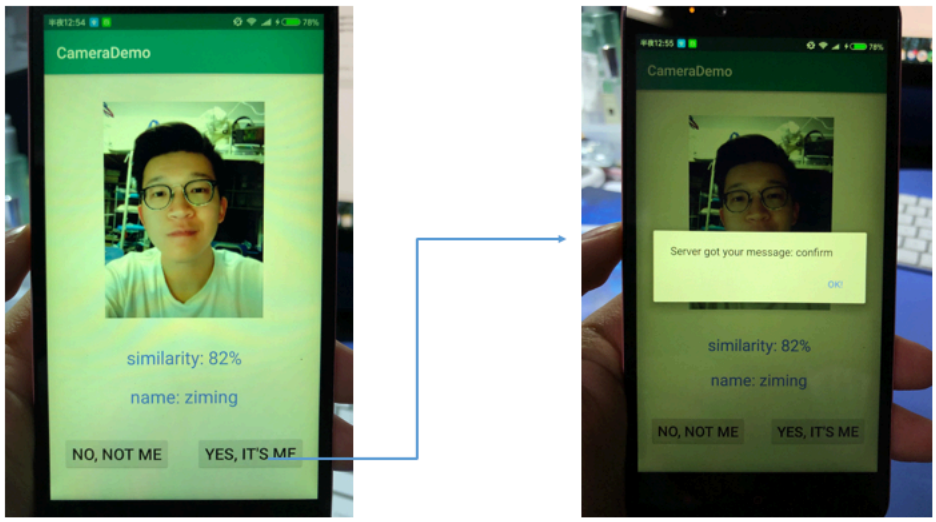


Figure 39 Classification results display interface

As for the server, first it is waiting for the connection from the client. We can see from Figure 40 that the server is successfully connected with the mobile application. And the size of the image is 84KB.

```
Waiting for connection...
Connected with... ('192.168.31.204', 53497) >--- mobile-end
<socket.socket fd=6, family=AddressFamily.AF_INET, type=Socke
data size: 84355 Bytes
```

Figure 40 Server is connected with the mobile application

After receiving the image and analyses the image by loading the face recognition model, the server obtains the classification result, which is “new_user”, indicating that this is a new user using the system probably. Then the server sends the result to the client.

```
new_user
b'new_user'
server sent data: b'new_user'

Now waiting for feedback...
```

Figure 41 Server obtains the classification result “new_user”

Now the server is waiting for feedback. It receives the feedback “ziming” from the client, and begins to update the system, storing the information of this new user.

```
Now waiting for feedback...
...connected with: ('192.168.31.204', 40089) >--- mobile-end
data size: 6
Server got your message: ziming

...local databases successfully updated.

...encodings successfully updated.

Waiting for connection...
```

Figure 42 Server receives feedback and updates the system.

Now the user begins to use the system the second time, and the system gets the classification result that a person called “ziming” has the highest average similarity 0.81, which is above the threshold. Then the server sends the result to the client and waits for feedback.

```
name: ziming
distances: [0.18483528]
average similarity: 0.815164722252408

ziming_0.8152
b'ziming_0.8152'
server sent data: b'ziming_0.8152'

Now waiting for feedback...
```

Figure 43 Server obtains classification result “Ziming_0.8152”

We can see that the server receives “confirm” feedback, and successfully updates the system.

```
Now waiting for feedback...
...connected with: ('192.168.31.204', 48814)  >--- mobile-end
data size: 7
Server got your message: confirm

Okay! start updating...

...encodings successfully updated.

...local databases successfully updated.

Waiting for connection...
```

Figure 44 Server receives confirmation and updates the system.

In the local database, we can see that a new folder containing two images just took by “ziming” has been created. Also, an encoding text file called “ziming_encodings.txt” has been created, which stores the feature values of these two face images that belong to “ziming”.

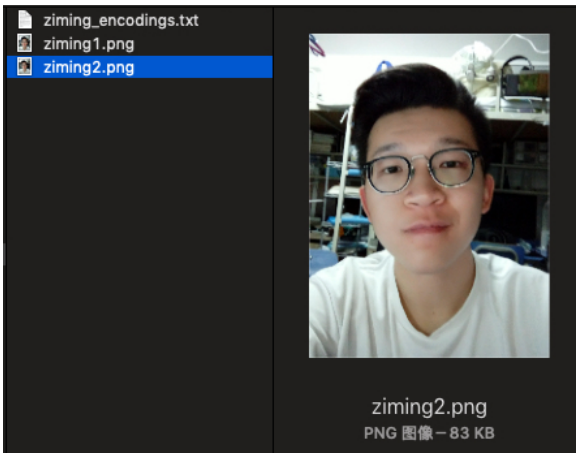


Figure 45 Two images just taken by the user and the encoding file have been stored.

4.2 Experimental analysis of the system performance

There are two datasets used for testing the performance of the system in this project. The first one is the Labeled Faces in the Wild (LFW) dataset, which is introduced in Chapter 2. The second dataset is a collection of BUPT students. 200 face images from 50 people are selected from the LFW dataset and are used for testing the system performance. The size of the second dataset is 30 people with 120 images.

Performance metrics of the system include two aspects, face verification and face identification. Verification means giving a pair of images each time, let the algorithm determine whether the two images are the same person. Identification means that each time a test image is input, making the system find the most similar face in a database containing multiple people. The identification is considered correct if the maximum similarity is greater than the threshold and the corresponding name is correct, or the maximum similarity is less than the threshold and the face is indeed a new data.

When performing face verification, we generally consider the following two indicators:

False Accept Rate (FAR): refers to the proportion of faces that should not be matched are considered from the same person.

$$FAR = NFA/NIRA * 100\% \quad (2)$$

False Reject Rate (FRR): refers to the proportion of faces that should be matched are considered from different persons.

$$FRR = NFR/NGRA * 100\% \quad (3)$$

In equations (2) and (3), NIRA is the total number of inter-class tests (different categories), and NGRA is the total number of in-class tests (same category). NFA is the number of false acceptances and NFR is the number of false rejections.

When performing face identification, we use Identification Rate (IR) to represent the ratio of the number of correctly identified times to the total number of recognition times.
Table 1 shows the experiment results.

Table 1: Experimental analysis

	Identification Rate	False Accept Rate	False Reject Rate
LFW dataset	100%	0.5%	0%
BUPT dataset	100%	1.3%	1.1%

From Table 1, we can see the Identification Rate of both datasets is 100%. It means that for every input image, the image can be correctly identified as a new user or a particular person. The FAR of BUPT dataset is 1.3%, meaning for every 75 pairs of different images, 1 pair of images will be mistakenly verified as the same person. The FRR of BUPT dataset is 1.1%, meaning for every 90 pairs of images from the same person, 1 pair of images will be mistakenly verified as different persons.

Overall, the face recognition effect of the system has reached a very advanced performance.

Chapter 5: Conclusion and Further Work

5.1 Conclusion of the project

This project achieved all the required tasks:

1. Designed and implemented a face recognition algorithm using Python and dlib open source library.
2. Designed and implemented a face recognition application on Android platform using Java and Android Studio.
3. Conducted experiment analysis on the proposed face recognition algorithm.
4. Designed and implemented a simple application to illustrate the prediction result.

Also, there are three special features of the system:

1. High recognition speed. The system is improved by extracting and saving the feature values of the local images. Previously, feature extraction is needed for each match. I improved the model by extracting the features of the local images, processing the feature values, and saving the features into a text file in the appropriate format. In this way, the model would not need to do feature extraction for a large number of pictures every time.
2. Applicable in real environment, recognize people in the wild. This system is able to guide new users to use it, and automatically create local dataset for those new faces. If no face above a certain similarity is detected in the local dataset, the face waited to be matched will be regarded as a new data, and a local dataset will be created for it. Next time if there is a face belonging to the same person, the model will be able to identify the face.
3. High accuracy and increasing robustness. For a successfully matched face, the corresponding dataset will continue to expand, and the average similarity of the matching results will be output, which enhances the robustness of the model. This is due to the reason that there may be some bias on the matching of some images, causing a negative impact on the correct match. With the increasing use of the model, and expansion of the local dataset, the matching result will be more robust.

5.2 Future work and improvements

Currently, I use average similarity of the matching results to represent the extent to which the unknown face resembles the local faces. This classification method is intuitive and simple, but

it has limitations. For example, the output average similarity will be limited to a certain value with the expansion of the dataset. In the future, I plan to use SVM to train a classifier that can output the classification results more accurately.

References

[1] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)* (Vol. 1, pp. 886-893). IEEE Computer Society.

[2] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, *10*(Jul), 1755-1758.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[4] Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*.

[5] Rogers, R., Lombardo, J., Mednieks, Z., & Meike, B. (2009). Android application development: Programming with the Google SDK. O'Reilly Media, Inc.

[6] Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer networks fifth edition. In Pearson Education, Inc., Prentice Hall.

Acknowledgement

It’s quite a long journey. I truly grow up a lot from the process of studying, designing, implementing and finally finishing the project. I would like to say thank you to all the people who helped me in this journey, especially my advisor and my roommates. My advisor kept guiding me in achieving the project with huge patience, clear instructions and useful tips. My roommates discussed with me a lot about our projects, from which I could get useful inspirations and suggestions and also learn from them.

I feel that this is an end, but it is also a brand-new beginning.

Stay hungry, stay foolish.

Appendix

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 1 – Supervisor

论文题目 Project Title	Design and implementation of face recognition system		
题目分类 Scope	Software Development	Implementation	Software
主要内容 Project description	Face recognition is a research hotspot in biometric identification area. The popularity and the increasing performance of smart device makes mobile phone based face recognition possible. The main steps of face recognition includes three parts: face detection, face feature extraction and feature identification. This project aims to design and implement a mobile phone based face recognition application. The application has three main functions: (1) capture face pictures; (2) calculate similarity of the captured face picture with pre-stored reference pictures; (3) automatic classification according to face similarity.		
关键词 Keywords	machine learning, face recognition, mobile application		
主要任务 Main tasks	**1** Design and implement a face recognition algorithm with openCV or other machine learning algorithms		
	2 Design and implement a face recongnition application on iOS/Android platform		
	3 Conduct experiment analysis on the proposed face recognition algorithm		
	4 Design and implement a simple application to illustrate the prediction result		
主要成果 Measurable outcomes	**1** Core code of face recognition algorithm		
	2 A mobile application with user-friendly UI		
	3 A detail experimental analysis of the performance of the proposed approach		

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 2 - Student

学院 School	International School	**专业 Programme**	**Internet of Things Engineering**		
姓 Family name	Chen	**名 First Name**	Ziming		
BUPT 学号 BUPT number	2015213363	**QM 学号 QM number**	151006342	**班级 Class**	2015215118
论文题目 Project Title	Design and implementation of face recognition system				
论文概述 Project outline **Write about 500-800 words** **Please refer to Project Student Handbook section 3.2**	**An initial analysis of user requirements:** The software should have a user-friendly graphic user interface (UI), allow users to easily operate the software with common sense or simple instructions. Users will have a comfortable and non-harm experience using the system and get a fantastic feeling about being recognized by the system. As a new user, the system can properly guide him or her to upload required information. Basic restrictions need to considered: users need to show a complete and clear face enough for the system to analyze and get result. **How data will be collected:** A dataset containing a sufficient number of human faces needs to be collected, so as to be offered to the face recognition algorithm to get a well-trained face recognition model. The data will be collected through the collection of face images of BUPT students, plus the open source labeled faces dataset. Noting that the face images of BUPT students will be strictly protected. **The algorithms, methodologies and other techniques to be employed:** OpenCV and some machine learning algorithms including Convolutional Neural Network and ResNet need to be employed for the training of the face recognition model and the realization of face detection and recognition. Also, certain open source libraries related to face recognition, like dlib (C++ face recognition library), may be used. **An initial specification of how users will interact with the system (implementation):** To begin with, the user needs to open the face recognition app, and click the corresponding button to turn on the camera. The camera will be called to capture the face of the user. Then the face image will be transferred into the back end to activate the running of the face recognition algorithm. The algorithm will compare the imported image with the pre-stored images, select and display the most similar one, output its name as well as the similarity it calculates. One point worth mentioning is that, once the similarity it calculates is below a certain threshold(tolerance), the algorithm will consider the input image a brand new data, i.e. this is a new user, and ask the user to confirm the condition and offer the required information, including his/her name. Then the back end algorithm will include the new face image into its database, and start a training. When the new user uses the system next time, the system will output his/her name and the similarity with the most similar one.				

	Experiments that should be done to prove the project hypotheses (research): Experiments should be conducted to test and analyze the performance of the proposed face recognition algorithm. There are mainly two parts: first, use a part of the pre-stored image faces that can be well recognized to conduct the experiment, and calculate the number of pictures that are correctly matched as well as the time it costs to finish the recognition process, so to get the accuracy and speed; second, find a group of participants to use the system, including old users and new users, and calculate the number of participants that are correctly recognized and the time it costs. **Programming language / database/ software package and hardware to be used:** This project may require using programming language including Python/C/C++ (for face recognition algorithm), Java/Swift (for Android/iOS platform development). **A list of background material consulted including World Wide Web pages:** [1] Mankar, Vijay & G Bhele, Sujata. (2012). A Review Paper on Face Recognition Techniques. International Journal of Advanced Research in Computer Engineering & Technology. 1. 339-346. [2] "Face recognition system", https://en.wikipedia.org/wiki/Facial_recognition_system [3] A. Özdil and M. M. Özbilen, "A survey on comparison of face recognition algorithms," 2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT), Astana, 2014, pp. 1-3.
道德规范 **Ethics**	Please confirm that you have discussed ethical issues with your Supervisor using the ethics checklist on QMPlus. [YES/NO] YES

	Summary of ethical issues: (put N/A if not applicable) The data collected from participants will be stored securely and in an anonymous form.
中期目标 **Mid-term target.** **It must be tangible outcomes, E.g. software, hardware or simulation.** **It will be assessed at the mid-term oral.**	The mid-term target is as follows: 1. The basic face recognition algorithm is implemented and tested through simulation on computer. 2. Finish the analysis and design of the face recognition application

Work Plan (Gantt Chart)

Fill in the sub-tasks and insert a letter X in the cells to show the extent of each task

	Nov	Dec	Jan	Feb	Mar	Apr	May
Task 1 break down							
Make research about the current state-of-the-art face recognition algorithms	X	X	X				
Compare different algorithms and list the pros and cons of them	X	X	X				
Select the algorithms to be used for face recognition and implement them in simulation environment			X	X	X		
Test the algorithms and ensure the correct running of them			X	X	X	X	
Task 2 break down							
Find and identify user requirements of the mobile application			X	X			
Analysis and design of the software system				X	X		
Study the mobile development and implement the prototype of the application				X	X	X	
Refine and test the application						X	X
Task 3 break down							
Collect data and construct dataset used for training the model				X			
Identify the desired output of the experiments and design the corresponding experiments				X	X		
Using the dataset collected to train the face recognition model					X	X	
Get results and make a detailed and comprehensive analysis on the performance of the algorithm						X	X
Task 4 break down							
Identify the procedure of using the application to get the prediction result			X				
Design the user interface of the illustration of prediction result			X	X			
Implement the user interface of the application to show the result				X	X	X	X
Test the process of displaying the prediction result						X	X

北京邮电大学 本科毕业设计（论文）初期进度报告

Project Early-term Progress Report

学院 School	International School	**专业 Programme**	**Internet of Things Engineering**		
姓 Family name	Chen	**名 First Name**	Ziming		
BUPT 学号 BUPT number	2015213363	**QM 学号 QM number**	151006342	**班级 Class**	2015215118
论文题目 Project Title	Design and implementation of face recognition system				

已完成工作 Finished work:

- What material was read or researched?

1. Michael Ruhl (2018), "How we created our Face-Recognition model", Available at: https://www.novatec-gmbh.de/en/created-face-recognition-model/

I studied this material recommended by my supervisor, where the project detects faces using the Vision-API and runs the extracted face through a CoreML-model to identiy the specific persons. And I tried to follow the instructions of the tutorial to train my own face recognition model. The procedures of the project are as follows:

a) Trained a model in the AWS using Nvidia DIGITS
b) Took a couple of hundred pictures of each person, and extracted the faces
c) Also added an "unknown" category with different faces.
d) Used a pretrained model fine-tuned for face-recognition.

2. Adam Geitgey (2016), "Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning", Available at: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

I read this article to learn how to use deep learning to train a face recognition model. This article introduces the basic algorithms and principles of face recognition.

3. "ageitgey/face_recognition", https://github.com/ageitgey/face_recognition

I studied this web page, which provides and introduces the *face_recognition* project that can recognize and manipulate faces from Python or from the command line. The project is built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

4. "guoyilin/FaceDetection_CNN", https://github.com/guoyilin/FaceDetection_CNN

This web page provided a fine-tune AlexNet using AFLW dataset.

5. "NovatecConsulting/FaceRecognition-in-ARKit", https://github.com/NovatecConsulting/FaceRecognition-in-ARKit

This web page provided a simple showcase project as described in [1]

6. "Labeled Faces in the Wild", http://vis-www.cs.umass.edu/lfw/

This is the homepage of the "Labeled Faces in the Wile" dataset. I read the homepage, downloaded the dataset, and checked the testing results of different algorithms using the LFW dataset.

7. Gary B. Huang and Erik Learned-Miller, *Labeled Faces in the Wild: Updates and New Reporting Procedures*, UMass Amherst Technical Report UM-CS-2014-003, 5 pages, 2014.

Since LFW dataset was released in 2007 in an effort to spur research in face recognition, a remarkably wide variety of innovative methods have been developed to overcome the challenges presented in this database. This paper reviewed the progress and contributions to LFW dataset.

- What work was done?

1. Based on the Gantt chart I planned before, I investigated various algorithms for face recognition and their performance on the LFD dataset. My conclusion is, it is better to use CNN-based machine learning algorithms. Currently, I have tried to use AlexNet and fine tuned AlexNet, and have already downloaded the pre-trained caffemodel. On the other hand, I am trying to implement the face recognition project provided by the third material mentioned above. This open source project is based on the deep learning model of the dlib open source library. The model is based on ResNet and has an accuracy of 99.38% on the LFW dataset.

2. I have collected several dataset, including Labeled Face in the Wild Database, Yale Face Database, The Facial Recognition Technology (FERET) database.

3. I followed the tutorials for the FaceRecognition-in-ARKit project to try to train a face recognition model with the collected dataset and pre-trained caffemodel on the Amazon AWS platform.

4. I studied the tutorial of the face_recognition project written by Adam Geitgey, and understanded the idea and procedure of the project to implement face recognition. I also analyzed and considered the feasibility of implementing face recognition on the mobile-end.

- What problems were faced?

The problems I faced when implementing the AR project were:

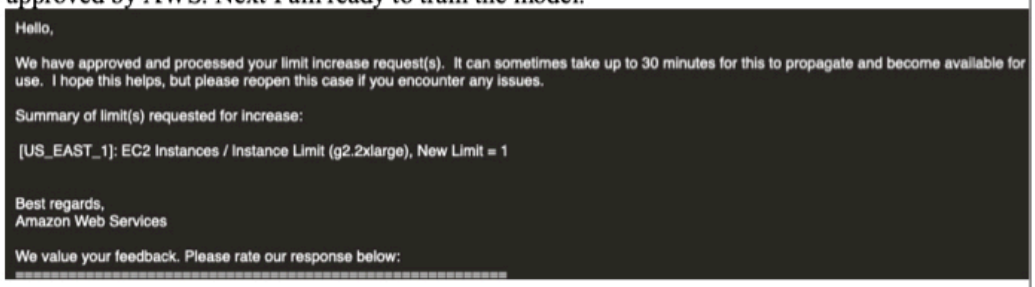
1. I tried to train the model with Nvidia DIGITS and AWS, but I was limited by instances allocation, I couldn't use the g2.2xlarge instance as normal.
2. The training of the project is on the AWS platform, and requires caffe environment and GPU, so the training cannot be implemented on the mobile application, and the training and model loading is separate (asynchronous).
3. According to the previous descriptions of the final project I want to implement, the mobile application can implement the function of adding new users. Consequently, the mobile application can only load a very mature model that can cope with the real environment and requirements, and it is difficult for real-time training for new users.
4. The training amount for each face needs 300 pictures, which is too much, and no data set is provided.

The problem I faced when implementing the face_recognition project was:

1. The project needs Python 3.3+ or Python 2.7, macOS or Linux, but does not support ios system, so there is a problem with how to deploy the project or model to the mobile application.

- What solutions were found?

1. AWS has limited instance resources for different users, so I applied for an instance and was approved by AWS. Next I am ready to train the model.


Hello,

We have approved and processed your limit increase request(s). It can sometimes take up to 30 minutes for this to propagate and become available for use. I hope this helps, but please reopen this case if you encounter any issues.

Summary of limit(s) requested for increase:

[US_EAST_1]: EC2 Instances / Instance Limit (g2.2xlarge), New Limit = 1

Best regards,
Amazon Web Services

We value your feedback. Please rate our response below:

2. I Looked for a mature face recognition model that can directly output high quality feature vectors, which would be stored for comparison afterwards. The deep learning model in the dlib open source library is relatively mature. A better feature vector can also be output for new data.

3. I have searched for various datasets myself.

4. For mobile deployment problems, the current analysis is to look for such a possibility: We made PC as a server or back-end to store the existing face dataset. The face recognition model is loaded on the server, and the face image sent by the mobile terminal is received by server and sent to the model, the feature vector is output. Then the comparison with the existing data is completed, and the classification result is returned to the mobile application. As the front end, the mobile application uses camera to capture user face and collects the face image, then it sends the image to the server and receives the classification result returned by the server, and presents it on the User Interface.

是否符合进度？On schedule as per GANTT chart?

[YES/NO] Yes

下一步 Next steps:

1. Continue to try to implement the AR project and train a face recognition model.
2. Implement the dlib-based face_recognition project on the computer.
3. Try to use computer as the back end, the mobile application as the front end, and explore the possibility of establishing collaboration and communication between them.

北京邮电大学 本科毕业设计（论文）中期进度报告

Project Mid-term Progress Report

学院 School	International School	**专业 Programme**	**Internet of Things Engineering**		
姓 Family name	Chen	**名 First Name**	Ziming		
BUPT 学号 BUPT number	2015213363	**QM 学号 QM number**	151006342	**班级 Class**	2015215118
论文题目 Project Title	Design and Implementation of Face Recognition System				

是否完成任务书中所定的中期目标？Targets met (as set in the Specification)?
[YES/NO] YES

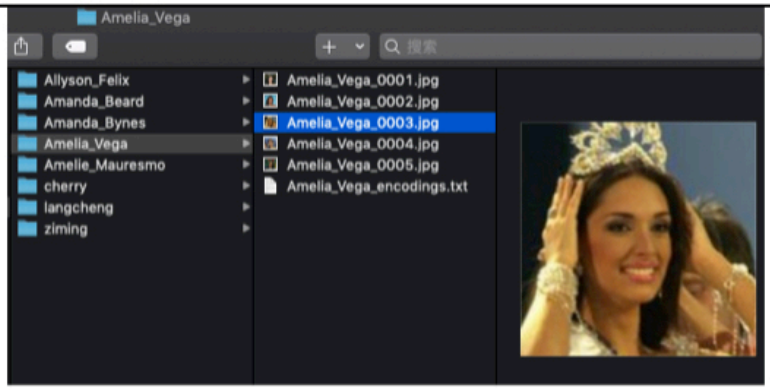
已完成工作 Finished work:

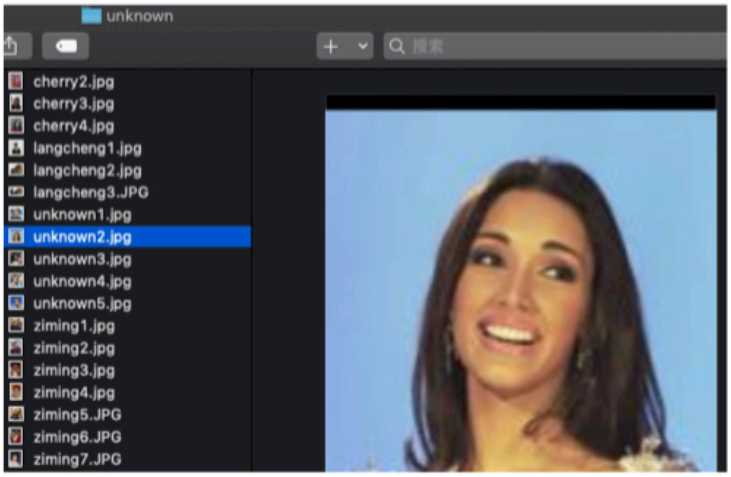
I have implemented the core face recognition model:

1. I investigated algorithms with OpenCV or machine learning algorithms for face recognition and their performance on the LFW dataset. To achieve the effect that the model can work well in real life and identify faces in the wild, it is better to use CNN-based machine learning algorithms. Finally, I decided to choose the dlib open source library, and used the deep learning model that is based on ResNet, to encode faces, and implement my face recognition algorithm.
2. Successfully built the main dlib library
3. Installed the Python extensions and related dependencies for the dlib library: python3, homebrew, etc.
4. Basic implementation of the face recognition algorithm based on the dlib open source library. The algorithm can implement the function of face detection, face feature extraction and feature identification
5. Tested the algorithm and ensured the correct running of the algorithm

I conducted and finished the initial experiment analysis of the face recognition algorithm:

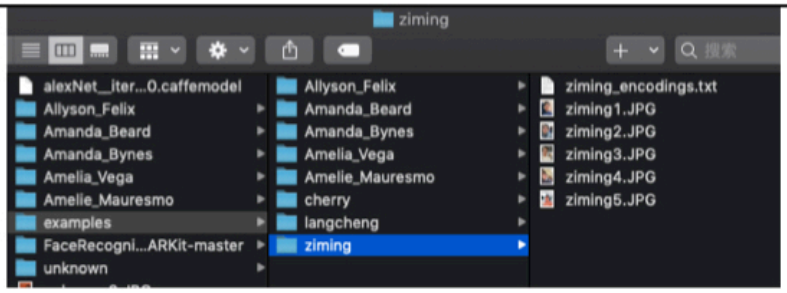
1. I collected several public face dataset, including Labeled Face in the Wild (LFW) Database, Yale Face Database, The Facial Recognition Technology (FERET) database, as well as real face images of BUPT students for experiment.
2. I selected a part of the LFW dataset to conduct experiment. For example, Amelia Vega in the screenshot below belongs to the LFW dataset. First, I used the face recognition model I have implemented to extract the features of the faces, and stored the output encodings into a text file. After that, I tested if the unknown image that is Amelia Vega can be correctly matched. The results suggest that the people called Amelia_Vega has the highest similarity to the unknown image, so the image is correctly identified. Up until now, the faces I selected from LFW have a match accuracy (percentage that unknown images are correctly matched) of 100%.

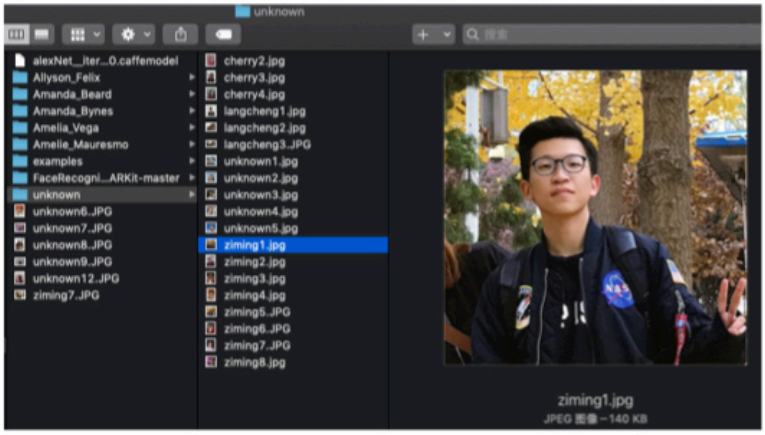





```
name: Amanda_Bynes
distances: [0.7884127  0.81719868 0.80406705 0.84029807]
similarity: 0.18750587441801092

name: Amelia_Vega
distances: [0.44852381 0.51595237 0.42143431 0.42943161 0.4916261 ]
similarity: 0.5386063595235446

name: Amelie_Mauresmo
distances: [0.87305023 0.90705228 0.92917    0.94208887 0.93531356]
similarity: 0.08266501222960088

name: cherry
distances: [0.67931396 0.66518243 0.65403863 0.66559693 0.66509133]
```

3. Also, I constructed a real-life dataset containing faces of BUPT students and my friends. The dataset is strictly protected and will not be released or published. Similar as above, first, I used the face recognition model to extract features of the faces, such as the five face images of ziming in the screenshot below. After feature extraction, I stored the output encodings into a text file. Then I tested if different unknown images that belong to ziming can be correctly matched. The results suggest that people called ziming in the local dataset has the highest similarity to the unknown image, so the image is correctly identified. Up until now, all the faces in the real-life dataset I collected have a match accuracy of 100%.






```
distances: [0.91765278 0.93332316 0.9557531  0.94307803 0.94327003]
similarity: 0.06138458015404358

name: cherry
distances: [0.6253597  0.65753816 0.67457382 0.6774269  0.68601837]
similarity: 0.33581660994765183

name: langcheng
distances: [0.68756386 0.61913074 0.65005501 0.65869888]
similarity: 0.3461378766517391

name: ziming
distances: [0.43758095 0.41113522 0.3367452  0.36039525 0.38287793]
similarity: 0.6142530915955582
```

I made improvements to the model:

1. Extracted and saved the feature values of the local images. Previously, I needed to extract and compare the images for each match. I improved the model by extracting the features of the local images, processing the feature values, and saving the features into a text file in the appropriate format. In this way, the model would not need to do feature extraction for a large amount of pictures every time.
2. Implemented automatic creation of a local dataset for strange faces. If no face above a certain similarity is detected in the local data, the face waited to be matched can be regarded as a new data, and a local data set will be created for it. Next time if there is a face belonging to the same person, the model will be able to identify the face.
3. For a successfully matched face, the corresponding dataset will continue to expand, and the average similarity of the matching results will be output, which enhances the robustness and accuracy of the model. This is due to the reason that matching of some images may be biased, causing a negative impact on the correct match. With the increasing use of the model, and expansion of the local dataset, the matching result will more robust.

4. In summary, the improvement of the model improves the speed and accuracy of face recognition.

I finished the basic analysis and design of the face recognition system:

1. Identified user requirements of the application:
   a. The software should have a user-friendly graphic user interface (UI), allowing users to easily operate the software with common sense or simple instructions.
   b. Users will have a comfortable and non-harm experience using the system and get a fantastic feeling about being recognized by the system.
   c. The user would not wait too long to get the identification result, the process should be as quick as possible.
   d. As a new user, the system can properly guide him or her to upload required information.
   e. Basic restrictions need to considered: users need to show a complete and clear face enough for the system to analyze and get result.

2. Finished the analysis and design of the system, including how users will interact with the system to get the prediction result using the face recognition application:
   a. To begin with, the user needs to open the face recognition app, and click the corresponding button to turn on the camera. The camera will be called to capture the face of the user. The user can choose to retake the photo or confirm the photo.
   b. If the photo is confirmed, the face image will be transmitted to the server, which is the back end, to activate the running of the face recognition algorithm. The algorithm will extract features of the photo waited to be matched, and compare the unknown encodings with the encodings of the pre-stored images using classifier, and get the classification result. Then the server will return the name as well as the similarity it calculates into the mobile end.
   c. The mobile application will receive the classification result returned by the server, and present it on the user interface. If the classification result is correct, the user can confirm the result. If the classification result is wrong, the user can inform the application, and enter the correct name of him/her. Finally, the photo to be matched will be included into the corresponding local data file.
   d. If the similarity it calculates is below a certain threshold, the algorithm will consider the input image a brand-new data, i.e. this is a new user, and ask the user to offer the required information, e.g. his/her name. Then the face recognition system will first check if the local data has information of this user, if not, then it will create a new local folder and include the new face image into the local folder. When the new user uses the system next time, the system will output his/her name and the similarity.

3. Designed the prototype of the APP User Interface

尚需完成的任务 Work to do:

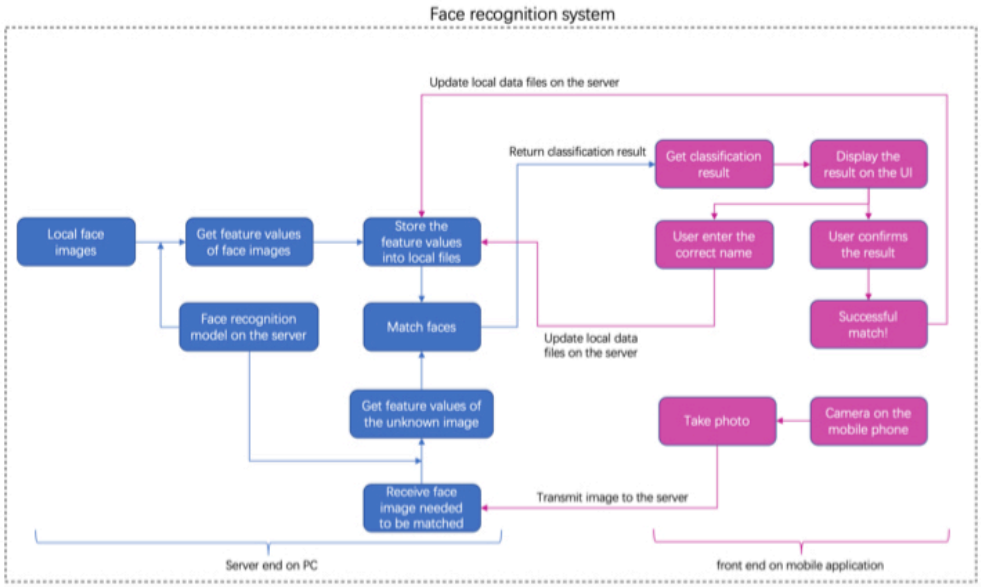
1. Study the mobile development and implement the front-end, i.e. the mobile application, to display the identification result.
2. Implement the communication and transmission of image information between the front-end and the back-end.
3. Further conduct experimental analysis of the model, in addition to testing the match accuracy, test the average time required to finish the classification.

存在问题 Problems:

1. The face recognition model needs Python 3.3+, macOS or Linux, but does not support mobile system, so there is a problem with how to deploy the project or model to the mobile application.
2. Some faces are not well recognized and may be misidentified.

拟采取的办法 Solutions:

1. For mobile deployment problems, I designed the structure of the face recognition system as the picture below: There are two parts. First, the PC is a server or back-end to store the existing face dataset. The face recognition model is loaded on the server, and the face image sent by the mobile terminal is received by server and processed by the model, and the feature vector is output. Then the comparison with the existing data is completed, and the classification result is returned to the mobile application. As the front end, the mobile application uses camera to capture user face and collects the face image, then it sends the image to the server and receives the classification result returned by the server, and presents it on the User Interface.



2. Currently, I use average similarity of the matching results to represent the extent to which the unknown face resembles the local faces. Later I plan to use SVM to train a classifier that output the classification result.

论文结构 Structure of the final report:

Abstract and Keywords

1. Introduction
   - 1.1 Definition of Face Recognition
   - 1.2 Purpose of the Project
2. Background
   - 2.1 Mechanism of Face Recognition
   - 2.2 Well-known Face Recognition Algorithms
3. Design and Implementation
   - 3.1 Structure of the Overall System
   - 3.2 Design and Implementation of Face Recognition Model
   - 3.3 Design and Implementation of Mobile Application
   - 3.4 Communication between the front-end and back-end

北京邮电大学 本科毕业设计（论文）教师指导记录表

Project Supervision Log

学院 School	International School	**专业 Programme**	**Internet of Things Engineering**		
姓 Family name	Chen	**名 First Name**	Ziming		
BUPT 学号 BUPT number	2015213363	**QM 学号 QM number**	151006342	**班级 Class**	2015215118
论文题目 Project Title	Design and Implementation of Face Recognition System				

Please record supervision log using the format below:

Date: dd-mm-yyyy
Supervision type: face-to-face meeting/online meeting/email/other (please specify)
Summary:

Date: 25-10-2018
Supervision type: email
Summary: discussed the purpose of the project

Date: 15-11-2018
Supervision type: face-to-face meeting
Summary: discussed the content and purpose of the project

Date: 20-11-2018
Supervision type: WeChat
Summary: discussed the possible architecture of the project

Date: 21-11-2018
Supervision type: WeChat
Summary: discussed the project specification

Date: 23-11-2018
Supervision type: WeChat
Summary: reported and refined the project specification

Date: 5-12-2018
Supervision type: WeChat
Summary: discussed the selection of the face recognition algorithms

Date: 15-12-2018
Supervision type: WeChat
Summary: reported the current progress and the problems faced

Date: 28-12-2018
Supervision type: WeChat
Summary: discussed the design function of mobile application

Date: 09-01-2018
Supervision type: WeChat

Summary: discussed the early-term progress

Date: 11-01-2019
Supervision type: face-to-face meeting
Summary: discussed the early-term progress and the report

Date: 20-02-2019
Supervision type: WeChat
Summary: reported the current progress

Date: 23-02-2019
Supervision type: WeChat
Summary: discussed the mid-term progress report

Date: 09-03-2019
Supervision type: WeChat
Summary: reported the mid-term progress and discussed the mid-term presentation

Date: 12-04-2019
Supervision type: face-to-face meeting
Summary: discussed the late-term progress, everyone summarized the current work that have been finished, the supervisor gave effective and accurate feedbacks on every student's project

Date: 16-04-2019
Supervision type: WeChat
Summary: discussed the draft report with the supervisor, including the structure and format of the report and the content

Risk Assessment

Risk description	Impact description	Level of likeliood	Seriousness of consequence	Preventative actions
Network failure	Time wasted to find out the problem	1	1	Use local area network with good condition
Underestimate the difficulty of some functions	Not finished the tasks in time	2	3	Study and prepare as early as possible
Have no experience developing some applications	Slow in software development	4	3	Invest more time in achieving the project and prepare early

Environmental Impact Assessment

This project is an implementation project focusing on designing and implementing a software system, and there is no negative impact on the environment consequently. The energy cost is little, and there is almost no cost of manufacture.