上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

程序设计团队作业

项目名称：情绪识别歌单

姓名学号：董复庆（队长）

杜洋

王颜涛

专业：经济管理实验班

学院：安泰经济与管理学院

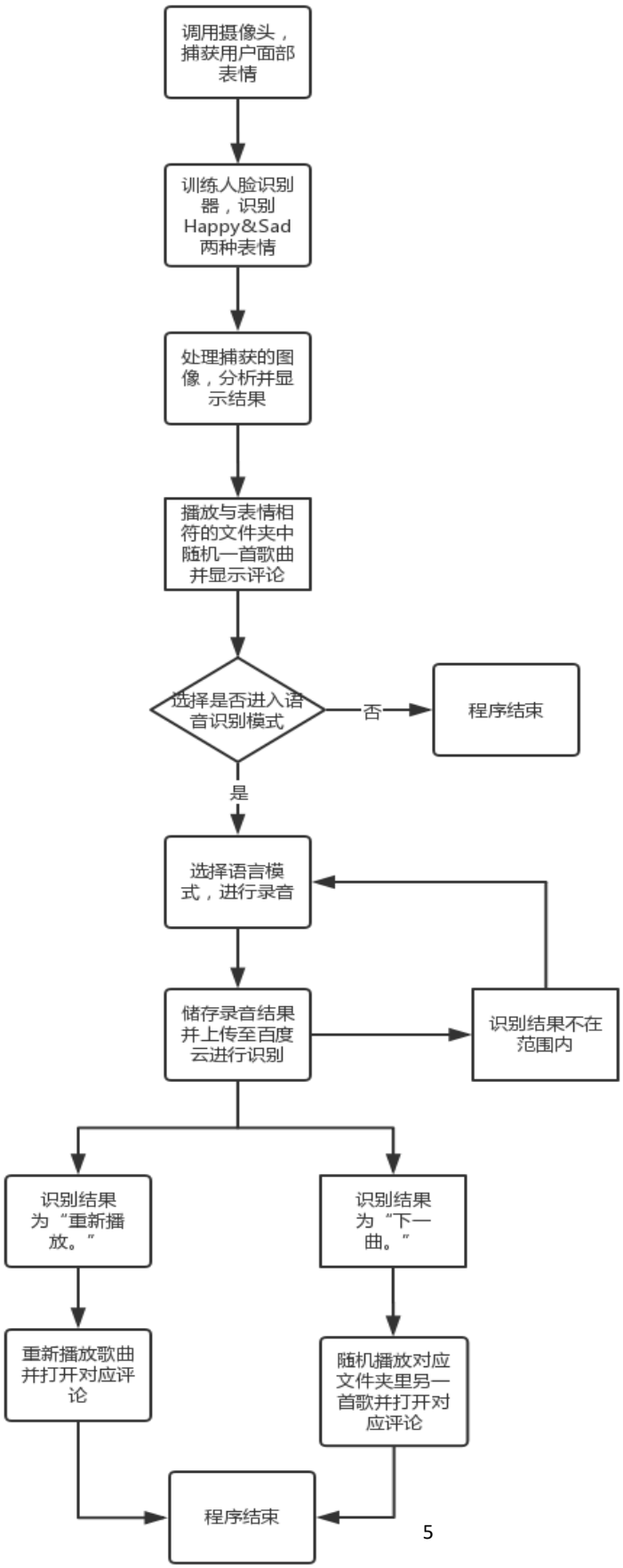
日期：2020.06.10

目录

一、简介 （杜洋）

“情绪识别歌单”是一款用于识别用户面部表情来随机匹配符合相应情绪的歌曲、评论的程序，并可以通过语音识别进行切歌等操作，致力于起到贴合用户情绪播放歌曲的惊喜效果。运行程序时，我们首先利用 OpenCV 与机器学习进行情绪预测训练，再捕捉用户面部照片进行情绪分析，利用分析结果打开相应歌曲和评论图片。若有切歌需要时，通过采集音频上传至百度云进行分析的方式得到语音结果，并匹配结果对应的操作模式，从而实现“重新播放”和“切歌”的功能。

流程图如下：


调用摄像头，捕获用户面部表情
训练人脸识别器，识别Happy&Sad两种表情
处理捕获的图像，分析并显示结果
播放与表情相符的文件夹中随机一首歌曲并显示评论
选择是否进入语音识别模式
否
程序结束
是
选择语言模式，进行录音
储存录音结果并上传至百度云进行识别
识别结果不在范围内
识别结果为“重新播放。”
识别结果为“下一曲。”
重新播放歌曲并打开对应评论
随机播放对应文件夹里另一首歌并打开对应评论
程序结束

二、问题缘起 （杜洋）

一方面，随着网络大数据的发展，人们更倾向于使用如网易云音乐软件中“每日歌曲推荐”这样的功能，实现一种发现符合自己喜好歌曲时的惊喜感。另一方面，听音乐已经成为现在年轻人进行情绪抒发、心情缓解的普遍方式，也会选择从歌曲评论中找到情绪共鸣。我们希望可以帮助用户根据实时的情绪来进行歌曲匹配，同时实现惊喜感与抒发情绪的功能，利用歌曲和评论为用户起到共鸣、安慰的效果。

三、函数库的使用 （董复庆）

1.Opencv

OpenCV 是一个基于 BSD 许可发行的跨平台计算机视觉库，实现了图像处理和计算机视觉方面的许多通用算法。本项目通过使用 OpenCV 实现调用计算机摄像头采集图像、读取图像、图像处理、人脸识别等功能。

OpenCV 中人脸检测分类器有多种，本项目选择基于 LBP 特征人脸识别器，能很好的避免光线明暗的影响

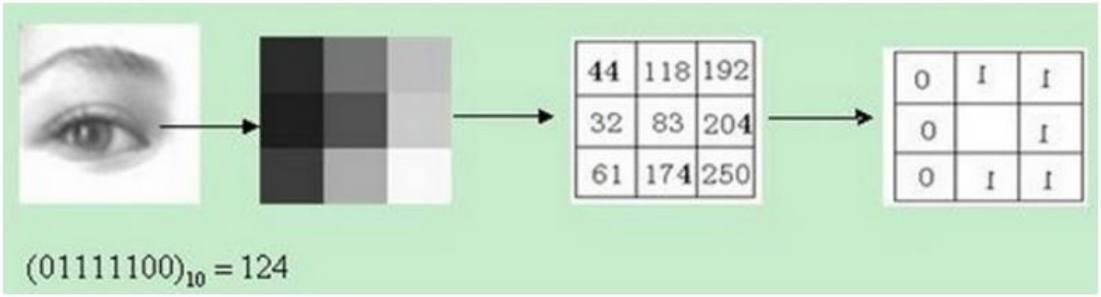
LBP 特征原理

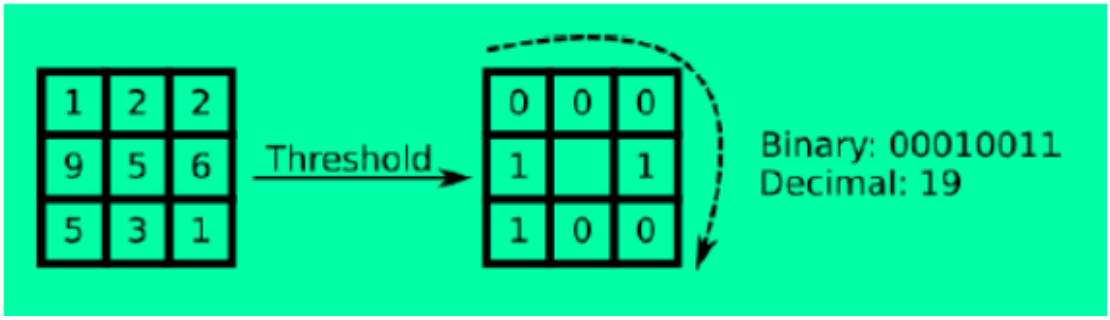
LBP(Local Binary Pattern)算法是一种描述图像特征像素点与各个像素点之间的灰度关系的局部特征的非参数算法，同时也是一张高效的纹理描述算法，纹理是物体表面的自然特性，它描述图像像素点与图像领域之间的灰度空间的分布

关系，不会因为光照强弱而改变图像的视觉变化。LBP 算法首次提出于 1994 年，主要是使用 8 邻域位置的局部关系，具有灰度不变性；随后在 2002 年提出了其改进版《Multiresolution gray-scale and rotation invariant texture classification with local binary patterns》 Timo Ojala, Matti Pietikaè inen,2002，主要引入了圆形模式、旋转不变性、和等价模式。这里只介绍灰度不变性基本 LBP 算法。

原始的 LBP 算子定义在像素 33 的邻域内，以邻域中心像素为阈值，相邻的 8 个像素的灰度值与邻域中心的像素值进行比较，若周围像素大于中心像素值，则该像素点的位置被标记为 1，否则为 0。

上述过程用图像表示为:





将上述过程用公式表示为：

Algorithmic Description A more formal description of the LBP operator can be given as:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases} \quad (15.1)$$

(xc,yc)为中心像素的坐标，p 为邻域的第 p 个像素，ip 为邻域像素的灰度值，ic 为中心像素的灰度值，s(x)为符号函数。

2.PIL

PIL（Python Image Library）是 python 的第三方图像处理库，但是由于其强大的功能与众多的使用人数，几乎已经被认为是 python 官方图像处理库了。它可以实现图像归档、图像展示、图像处理等功能。本项目调用 PIL 中的核心类——Image 类，实现读取并打开图片文件的功能。

3.requests

requests 库是一个常用的用于 http 请求的模块，它使用 python 语言编写。本项目通过使用 requests 中的 post 方法向百度云语音识别应用提交 post 请求。

4.pyaudio

pyaudio 库，是一个跨平台的音频 I/O 库，使用这个可以在 python 中实现录音、播放、生成 wav 文件等功能。本项目通过使用 pyaudio 库实现录音功能。

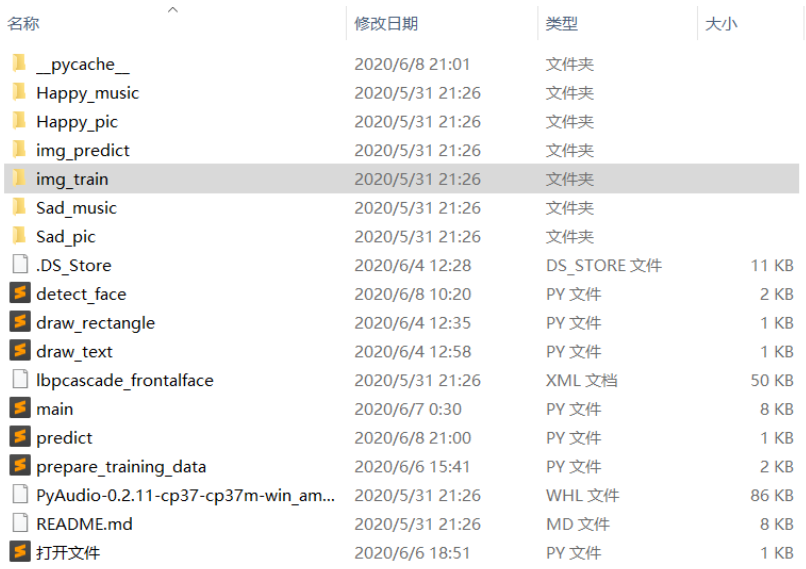
5.wave

wave 是 python 语言第三方库，主要作用是操作 wav 格式文件。本项目通过使用 wave 音频库实现从音频文件中读取数据并将数据写入声卡的功能。

6.numpy

Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。Numpy 是一个运行速度非常快的数学库，主要用于数组计算。本项目使用 numpy.array()命令创建一个 n 维数组对象。

四、代码运行 （董复庆 杜洋 王颜涛）



名称	修改日期	类型	大小
__pycache__	2020/6/8 21:01	文件夹	
Happy_music	2020/5/31 21:26	文件夹	
Happy_pic	2020/5/31 21:26	文件夹	
img_predict	2020/5/31 21:26	文件夹	
img_train	2020/5/31 21:26	文件夹	
Sad_music	2020/5/31 21:26	文件夹	
Sad_pic	2020/5/31 21:26	文件夹	
.DS_Store	2020/6/4 12:28	DS_STORE 文件	11 KB
detect_face	2020/6/8 10:20	PY 文件	2 KB
draw_rectangle	2020/6/4 12:35	PY 文件	1 KB
draw_text	2020/6/4 12:58	PY 文件	1 KB
lbpcascade_frontalface	2020/5/31 21:26	XML 文档	50 KB
main	2020/6/7 0:30	PY 文件	8 KB
predict	2020/6/8 21:00	PY 文件	1 KB
prepare_training_data	2020/6/6 15:41	PY 文件	2 KB
PyAudio-0.2.11-cp37-cp37m-win_am...	2020/5/31 21:26	WHL 文件	86 KB
README.md	2020/5/31 21:26	MD 文件	8 KB
打开文件	2020/6/6 18:51	PY 文件	1 KB

如上是工作文件夹，所有用到的图片、音乐、代码均以封装。

1.函数定义

定义识别人脸函数，使用 LBP 特征人脸识别器将图片中人脸识别出并返回面部区域的位置信息

```
def detect_face(img):
    #将图像转变成灰度图像，因为OpenCV人脸检测器需要灰度图像
    gray = cv2.cvtColor(img, cv2.COLOR_BAYER_BG2GRAY)

    #加载OpenCV人脸识别器，注意这里的路径是前面下载识别器时，你保存的位置
    face_cascade = cv2.CascadeClassifier(r'lbpcascade_frontalface.xml')

    #scaleFactor表示每次图像尺寸减小的比例，minNeighbors表示构成检测目标的相邻矩形的最小个数
    #这里选择图像尺寸减小1.2倍。minNeighbors越大，识别出来的人脸越准确，但也极易漏判
    #检测多尺度图像，返回值是一张脸部区域信息的列表（x,y,宽,高）
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=1)

    # 如果图中没有人脸，该图片不参与训练，返回None
    if len(faces) == 0:
        return None, None

    # 提取面部区域:(x,y)为左上点坐标，w、h是宽和高
    (x, y, w, h) = faces[0]

    #返回人脸及其所在区域
    return gray[y:y + w, x:x + h], faces[0]
```

之后进行机器学习训练前的数据准备，即把预先准备好的训练组照片进行面部识别，最后得到包含多个人脸矩阵的序列和它们对应的标签

```python
def prepare_training_data():

    #读取训练文件夹中的图片名称
    #os.listdir命令返回指定的文件夹包含的文件或文件夹的名字的列表
    dirs = os.listdir(r'./img_train')
    faces = [] #两个列表分别保存所有的脸部矩阵和标签
    labels = []
    for image_path in dirs:
        #浏览每一张图片
        #如果图片的名称以happy开头，则标签为11；sad开头，标签为2
        if image_path[0] == 'h':
            label = 1
        else:
            label = 2

        #建立图片路径
        image_path = './img_train/' + image_path

        #读取图像，返回灰度图，返回Mat对象
        image = cv2.imread(image_path,0)

        #以窗口形式显示图像，显示100毫秒
        #cv2.imshow("Training on image...", image)
        #cv2.waitKey(100) #在一个给定的时间内(单位ms)等待用户按键触发，返回按键的ASCII码;如果用户没有

        #用detect_face函数检测脸部
        face, rect = detect_face(image)
        if face is not None: #忽略未检测到的脸部
            faces.append(face) ##将脸添加到脸部列表并添加相应的标签
            labels.append(label)

    cv2.destroyAllWindows() #删除所有窗口
    #cv2.waitKey(1) #键盘绑定函数 1毫秒
    #cv2.destroyAllWindows() #删除所有窗口

    return faces, labels ##最终返回值为人脸和标签列表
```

首先利用 os 模块对训练组文件夹的文件名进行读取，创建两个空列表，利用 for 循环机制浏览每一张图片，对每一张图片进行面部区域检测，最后将面部区域存在 faces 列表下，用‘1’代表’ Happy’,用’ 2’代表‘Sad’,封装在 labels 列表下，完成数据准备。

predict 函数先是通过事先定义好的 detect_face 进行面部识别，之后用训练好的人脸识别器进行预测，将预测出的数字利用一个 subjects 列表与相应的情绪对应，并调用已经定义好的 draw_rectangle 和 draw_text 函数为图片框选人脸并添加文字，返回预测结果。

```python
def predict(test_img, face_recognizer):
    # 将标签1，2转换成文字
    subjects = ['', 'Happy', 'Sad']

    # 得到图像副本
    img = test_img.copy()

    # 从图像中检测脸部
    face, rect = detect_face(img) #调用dect_face函数

    # 使用我们的脸部识别器预测图像
    label = face_recognizer.predict(face)


    # 获取由人脸识别器返回的相应标签的名称
    label_text = subjects[label[0]]


    # 在检测到的脸部周围画一个矩形
    draw_rectangle(img, rect)
    # 在矩形周围标出人脸情绪
    draw_text(img, label_text, rect[0], rect[1] - 5)

    return img,label_text ##返回预测的图像，标签内容
```

```python
#添加文字 参数为（图片，添加的文字，左上角坐标，字体，字体大小，颜色，字体粗细）
def draw_text(img, text, x, y):
    cv2.putText(img, text, (x, y), cv2.FONT_HERSHEY_PLAIN, 1.5, (0, 255, 0), 2)
```

定义在图片上添加文字函数。用于将识别的结果添加到图片中人脸框的上方。

调用 cv2 中 putText()函数，其中参数分别为操作的图片、添加的文字、人脸框左上角的坐标、添加文字的字体、文字的大小、颜色和粗细。这里我们采用 cv2.FONT_HERSHEY_PLAIN 字体、大小为 1.5、颜色为红色、粗细为 2。

```python
def draw_rectangle(img, rect):
    (x, y, w, h) = rect #画出矩形
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
#图片 左上角位置 右下角位置 边框颜色 粗细
```

定义画出人脸矩形框的函数。用于在识别的人脸区域周围画出矩形框。首先将四维变量储存到 rect 这个变量名中，用于之后 predict.py 文件中。之后调用 cv2

中 rectangle()函数，这里将变量设置分别为操作的图片、识别的人脸区域左上角坐标、右下角坐标、红色、边框粗细为 2。

```
def music_and_pic(label_text):
    the_number_=random.randrange(10) #返回0—9之间随机一个数
    #指定根据表情决定的文件夹中随机数决定的一首歌
    music='./{}_music/{}.mp3'.format(label_text,the_number_)
    #打开音乐文件
    os.system(music) #system函数可以将字符串转化成命令在服务器上运行

    #指定根据表情决定的文件夹中随机数决定的与歌匹配的图片
    picture='./{}_pic/{}.png'.format(label_text,the_number_)
    #从文件加载图像
    image=Image.open(picture)
    image.show()

    return the_number_ #储存随机数的值
```

定义打开音乐和图片函数。用于随机打开情绪结果对应文件夹中的一个音乐以及其配套的图片。

首先利用 random 库中的 randrange()函数生成 0 到 9 中一个随机数。变量名 music 中储存的是情绪识别结果的对应文件夹中随机数决定的音乐文件的相对路径的字符串，之后利用 os 库中的 system 函数将 music 中的字符串转化为命令，在终端中运行，即打开音乐文件。

变量名 picture 中储存的是情绪识别结果的对应文件夹中随机数决定的图片文件的相对路径的字符串。之后利用 PIL 库中的 Image.open()函数和 Image.show()函数打开图片文件。最后储存随机数的值于 the_number_中。

```
#定义 将data中的数据保存到名字=变量filepath的WAV文件中 的函数
#通过wave模块从音频文件中读取数据，返回wave类。然后把读取的str数据通过pyaduio模块写到声卡里。
def save_wave_file(filepath, data):
    wf = wave.open(filepath, 'wb') #读取音频文件，返回instance对象类
    wf.setnchannels(channels) #设置音频文件的声道数，用在写音频流时
    wf.setsampwidth(sampwidth) #获得音频文件每个采样值得保存位数，用在读音频流时
    wf.setframerate(framerate) #获得采样率，用在读音频流时
    wf.writeframes(b''.join(data))
    wf.close()
```

定义保存音频文件函数。

调用了 wave 库的 open 函数，新建一个 wave 文件，‘wb’表示可以写入的打开方式，随后设置声道数、采样率、采样宽度，用 join 方法将参数写入 wave 文件，随后关闭文件。

```
def getToken(host): #获取token 使用post向授权服务地址https://aip.baidubce.com/oauth/2.0/token发送请求
    res = requests.post(host)

    return res.json()['access_token']
```

定义 getToken 函数获得百度云的授权。

此处 host 参数是组合后的 url，用 requests 的 post 函数，之后使用 json 函数，将 url 转化为 json 的数据格式，随后获取其中名为’access_token’的变量。

```python
#定义 录音识别 的函数
def speech2text(speech_data, token, dev_pid=1537):
    #设置格式
    FORMAT = 'wav' #语音文件的格式为wav（不压缩，pcm编码）
    RATE = '16000' #采样率 固定值16000
    CHANNEL = 1 #单声道 固定值1
    CUID = '*******' #
    SPEECH = base64.b64encode(speech_data).decode('utf-8') #base64编码格式

    #以字节格式读取文件之后进行编码
    data = {              #JSON 里包括的参数
        'format': FORMAT, #语音文件的格式
        'rate': RATE,     #采样率，16000，固定值
        'channel': CHANNEL, #声道数，仅支持单声道，请填写固定值 1
        'cuid': CUID,  #用户唯一标识，用来区分用户，计算UV值
        'len': len(speech_data), #本地语音文件的的字节数，单位字节
        'speech': SPEECH, #本地语音文件的的二进制语音数据 ，需要进行base64 编码。与len参数连一起使用。
        'token': token, #开放平台获取到的开发者[access_token]获取 Access Token
        'dev_pid':dev_pid #不同语种
    }
    url = 'https://vop.baidu.com/server_api' #使用JSON格式POST上传本地文件到这个网址：短语音识别请求地址
    headers = {'Content-Type': 'application/json'}
    print('正在识别...')

    #上传文件得到结果
    r = requests.post(url, json=data, headers=headers)

    #返回识别结果
    Result = r.json() #采用 JSON 格式封装，如果识别成功，识别结果放在 JSON的“result”字段中


    return Result['result'][0] #‘result’中第一项的值
```

定义录音识别的函数，用于识别和返回用户的指令。

函数的三个参数分别是之前保存的录音内容、Access_token 以及识别语言类别的代码。先定义了文件的格式、声道、采样率，使用了 base64 的库对录音内容进行编码。Data 作为 json 格式的数据格式，内部保存了格式、采样率、用户标识、声道、本地语音文件的数据、开放平台的权限申请、语种。设置 url 和 headers 两个参数，利用 requests.post 函数上传 json 格式的信息，再采用 json 格式将结果封装，返回 result 中的第一项，即识别出的用户指令。

2.主体函数

```
#主体：运行camera函数
print('Press s to catch the picture.')
print('Press q to exit.')
camera()

print("Preparing data...")
#调用之前写的函数，得到包含多个人脸矩阵的序列和它们对于的标签
faces, labels = prepare_training_data() #调用prepare_training_data.py
print("Data prepared")

print("Total faces: ", len(faces)) #打印脸的个数
print("Total labels: ", len(labels)) #打印标签个数
```

训练（LBP）人脸识别器，加载之前摄像头采集的图像，调用 predict 函数进行预测

```
#得到（LBPH）人脸识别器
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
#应用数据，进行训练
face_recognizer.train(faces, np.array(labels))

print("Predicting images...")

#加载预测图像
test_img = cv2.imread(r"./0.jpg",0)

#进行预测
predicted_img,label_text = predict(test_img, face_recognizer) #调用predict.py里的函数
```

通过 OpenCV 库中的 imshow 函数显示结果

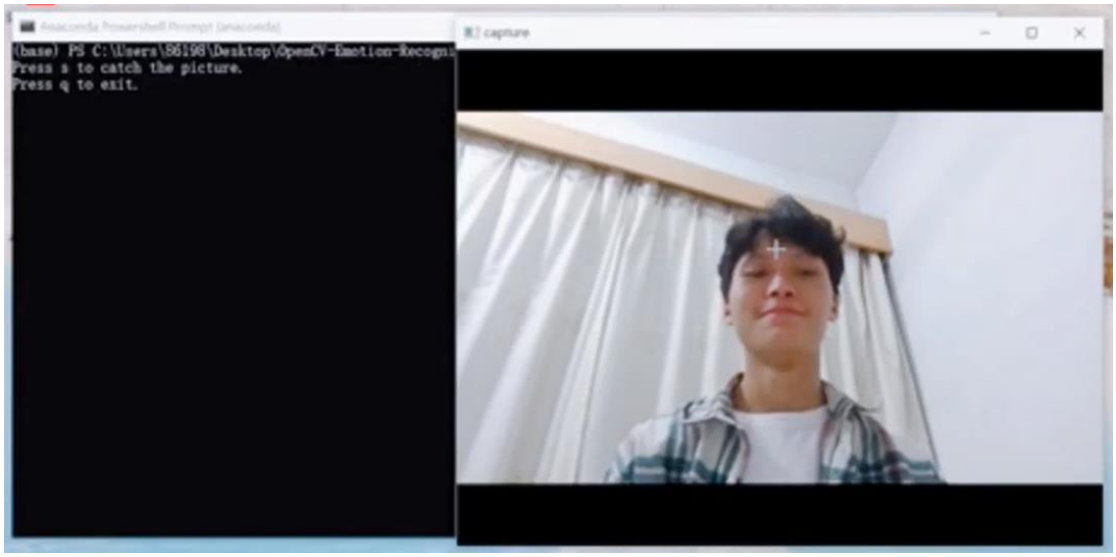
```
print("Prediction complete")

#显示预测结果和图像
cv2.imshow('the_result', predicted_img)
```

```
#主体：语音识别循环
while True:
    print('请输入数字选择语言：')
    devpid = input('1537:普通话(有标点),1737:英语\n') #输入语音选择
    my_record() #调用my_record函数
    TOKEN = getToken(HOST) #调用getToken函数得到百度云的授权
    speech = get_audio(FILEPATH) #调用get_audio函数得到录音文件数据
    result = speech2text(speech, TOKEN, int(devpid)) #调用speech2text函数得到语音识别结果
    print(result) #打印结果

    if result == '下一曲。': #输出结果为‘下一曲。’
        music_and_pic(label_text) #重新利用打开文件.py中的music_and_pic函数打开音乐的图片
        break #退出循环

    elif result == '重新播放。': #输出结果为‘重新播放。’
        #再次打开之前的文件和音乐 以下代码与打开文件.py中的music_and_pic函数中一部分相同
        music='C:/Users/86198/Desktop/OpenCV-Emotion-Recognition-master/{}_music/{}.mp3'.format(label_text,the_number_)
        os.system(music)


        picture='C:/Users/86198/Desktop/OpenCV-Emotion-Recognition-master/{}_pic/{}.png'.format(label_text,the_number_)
        image=Image.open(picture)
        image.show()
        break #退出循环

    else:
        print('Please input again!')
        continue #循环
```

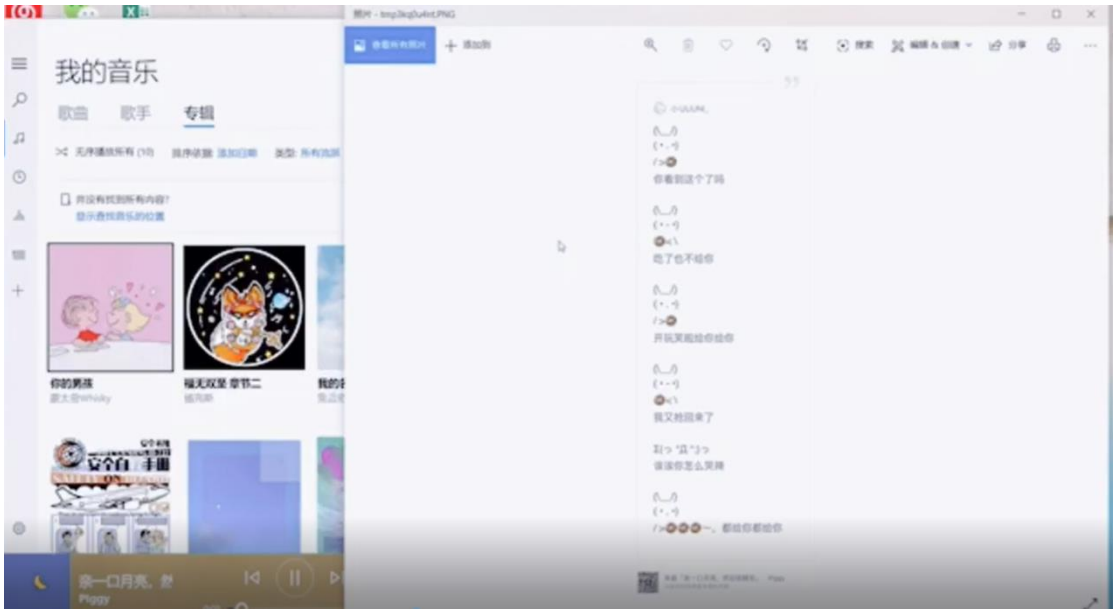
由用户选择所使用的语言，随后调用 my_record 函数进行录音的获取，得到百度云的权限后上传录音文件数据并返回语音识别的结果。打印出结果后，若为“下一曲”则重新调用函数随机播放音乐及打开对应图片；若为 “重新播放”则再次打开相同的音乐及图片。完成操作后等待用户后续指令。

五、运行演示 （董复庆）

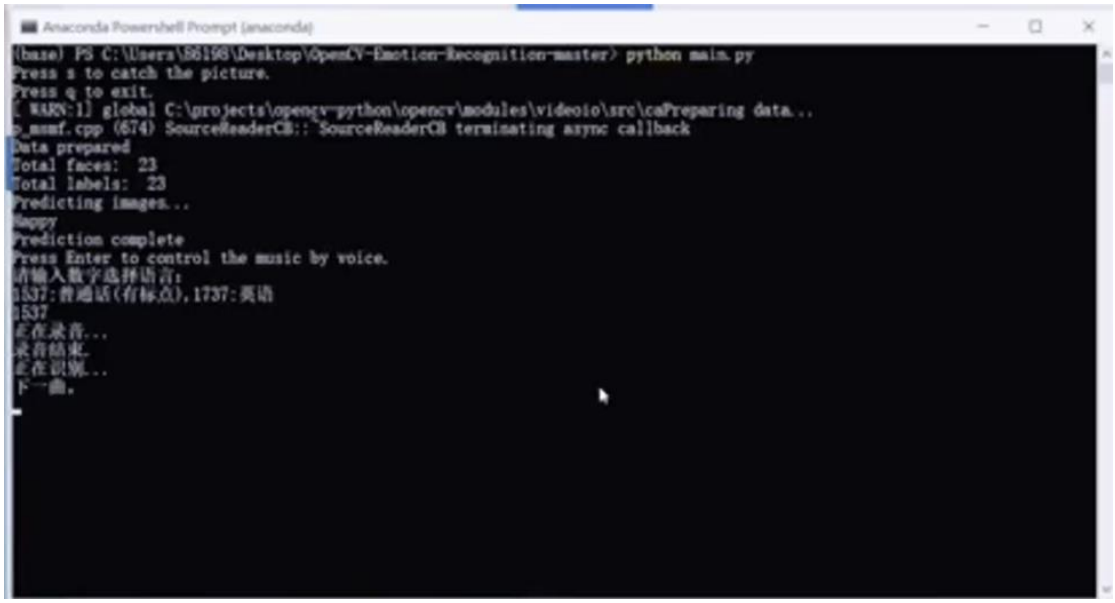
（1）运行代码，调用摄像头



（2）用户端得到结果，播放音乐，打开图片



（3）进入语音识别



（4）根据识别结果，完成切歌指令

六、局限和优化方向 （王颜涛）

目前项目整体比较完整，但是部分代码存在功能的重复（例如使用户表情变

成灰色图片部分），未来可以进行优化，使代码更加简洁清晰。此外，目前项目只能检测两种表情（快乐和悲伤），未来在机器学习方面，可以加入更多的表情，或者将表情细化，分解成不同程度，丰富项目的功能，给用户更好的体验。受制于团队成员目前对 python 的学习程度，目前我们的项目需要提前下载曲库以及评论图片。若未来对 python 有进一步的学习，可以考虑通过爬虫技术，在完成情绪识别后，从网页或音乐软件在线下载歌曲进行播放，使得用户的体验更加流畅。

七、总结 （王颜涛）

从音乐软件的功能出发，我们发现了用户通过歌曲来进行情绪抒发的需求，进而萌生了做这个项目的想法。在项目完成过程中，小组成员从各大网站、论坛上进行代码搜索和阅读，在自己的能力范围内争取将想法的还原度达到最高，代码运行流畅的同时丰富项目的功能，力求给予用户最好的体验。这一次的小组作业，为我们打开了 python 世界的大门，对于我们而言是一种收获：这样独立解决问题的机会，对我们的能力是一种磨练。期待未来对 python 的掌握能够更加深入，将其与生活联系得更加紧密，能够更好地帮助我们解决实际问题。

最后，感谢鲍杨老师一学期的指导教学！