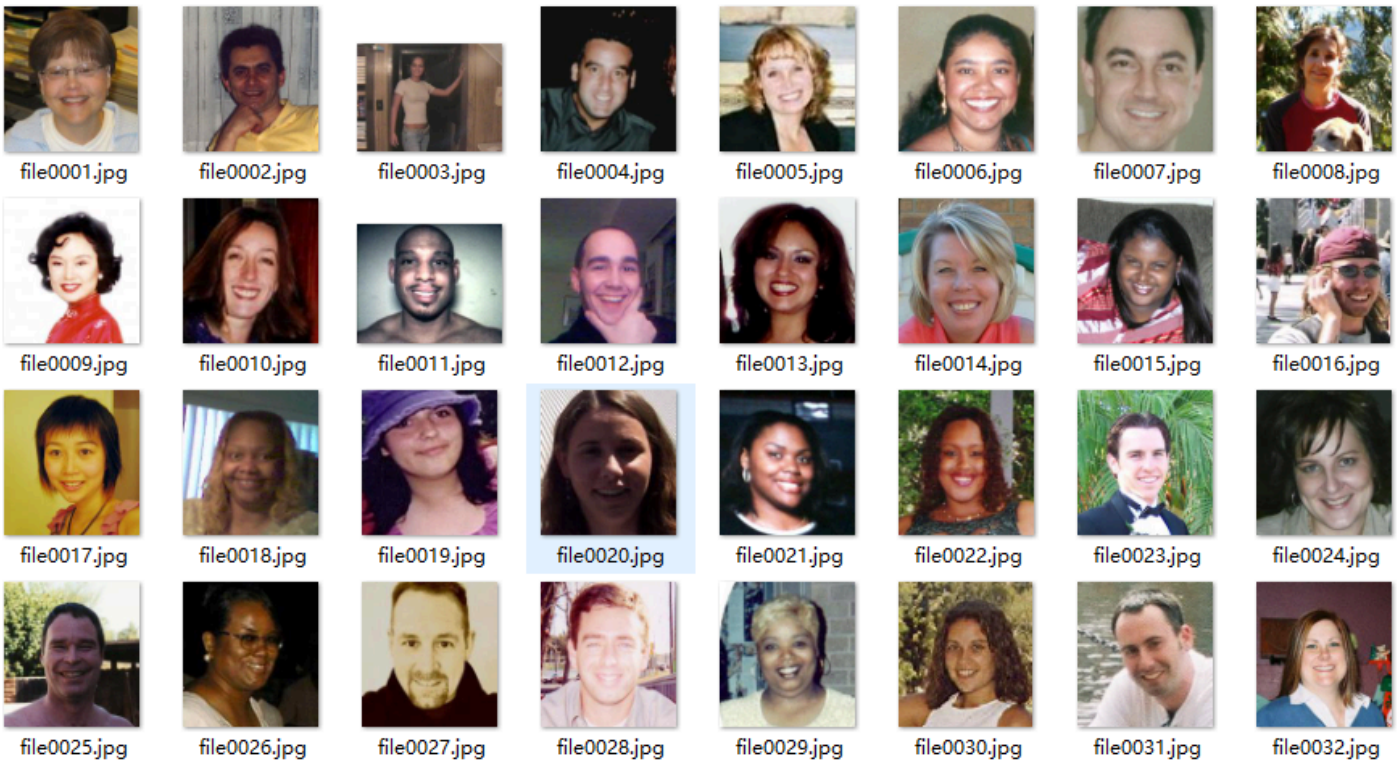
1. 任务

1.1 训练目的

使用Dlib提取人脸特征并训练二类分类器 (smile, nosmile) 来识别人脸微笑表情。

1.2 数据集

数据集为4000张照片，分为两类：微笑，不微笑；照片的格式是jpg，文件名为file+编号。



2. 编码

2.1 获取4000张人脸的特征点数据

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
import sys
import os
import dlib
```

第一步，读取图片文件，定义图片目录，预处理特征数据shape_predictor_68_face_landmarks.dat的目录。

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
predictor_path = "./train_dir/shape_predictor_68_face_landmarks.dat"
faces_folder_path = "./train_dir/face/"
```

第二步，加载 人脸检测　器。

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(predictor_path)
```

第三步，检测人脸68个特征点的数据并存储在文件中：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
fi=open("./train_dir/face_feature3.txt","a")
for i in range(4000):
```

```
    f=faces_folder_path+'file'+'{:0>4d}'.format(i+1)+".jpg"
    img = dlib.load_rgb_image(f)

    # 让检测器找到每个人脸的边界框。
    # 第二个参数中的 1表示我们应该对图像进行 1 次上采样。这个
    # # 将使一切变得更大，并允许我们检测更多的人脸。
    dets = detector(img, 1)
    #print("Number of faces detected: {}".format(len(dets)))
    won=False
    for k, d in enumerate(dets):
        # 得到的地标/用于面部在框d的部分。
        shape = predictor(img, d)
        rect_w=shape.rect.width()
        rect_h=shape.rect.height()
        top=shape.rect.top()
        left=shape.rect.left()
        fi.write('{},'.format(i))
        won=True
        for i in range(shape.num_parts):
            fi.write("{},{},".format((shape.part(i).x-left)/rect_w,(shape.part(i).y-top)/rect_h))
    if(won):
        fi.write("\n")

fi.close()
```

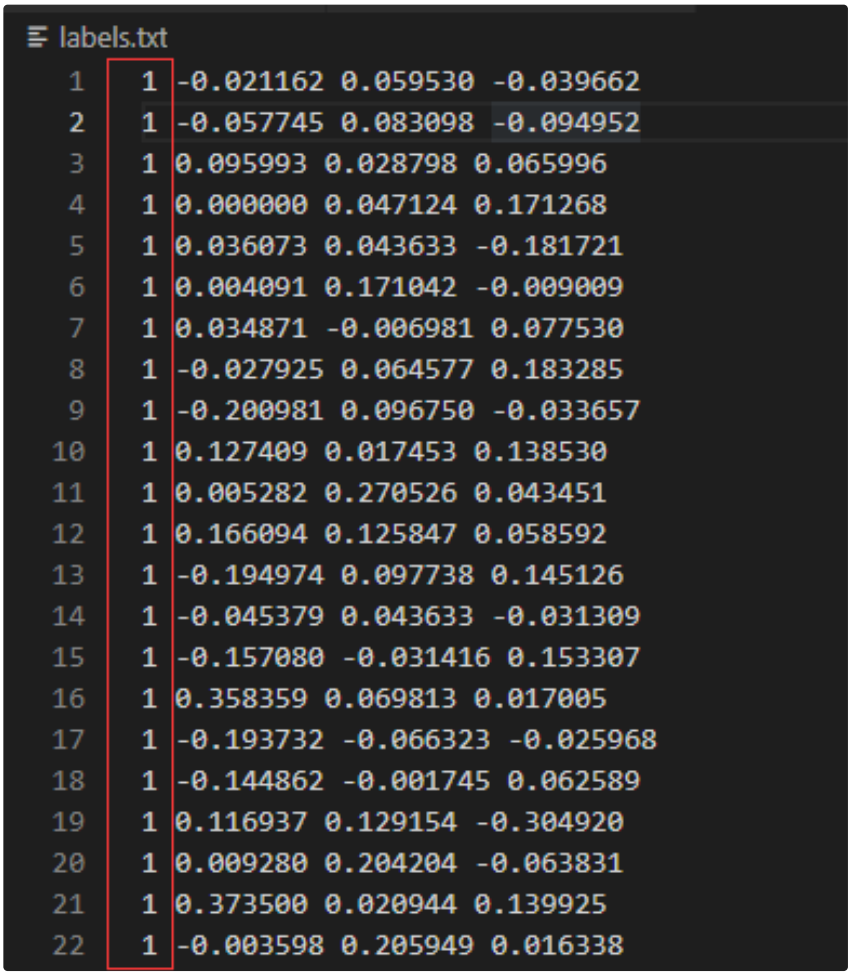
完整代码：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
import sys
import os
import dlib
predictor_path = "./train_dir/shape_predictor_68_face_landmarks.dat"
faces_folder_path = "./train_dir/face/"

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(predictor_path)

fi=open("./train_dir/face_feature3.txt","a")

for i in range(4000):
    f=faces_folder_path+'file'+'{:0>4d}'.format(i+1)+".jpg"
    img = dlib.load_rgb_image(f)

    # 让检测器找到每个人脸的边界框。
    # 第二个参数中的 1表示我们应该对图像进行 1 次上采样。这个
    # # 将使一切变得更大，并允许我们检测更多的人脸。
    dets = detector(img, 1)
    #print("Number of faces detected: {}".format(len(dets)))
    won=False
    for k, d in enumerate(dets):
        # 得到的地标/用于面部在框d的部分。
        shape = predictor(img, d)
        rect_w=shape.rect.width()
        rect_h=shape.rect.height()
        top=shape.rect.top()
        left=shape.rect.left()
        fi.write('{},'.format(i))
        won=True
        for i in range(shape.num_parts):
            fi.write("{},{},".format((shape.part(i).x-left)/rect_w,(shape.part(i).y-top)/rect_h))
    if(won):
        fi.write("\n")

fi.close()
```

2.2 处理4000张人脸的数据并将事先提供的 smile或者nosmile 数据添加到人脸数据集上

该列数据为1，表示是笑脸，为0表示为不笑：



labels.txt

```
1  1 -0.021162 0.059530 -0.039662
2  1 -0.057745 0.083098 -0.094952
3  1 0.095993 0.028798 0.065996
4  1 0.000000 0.047124 0.171268
5  1 0.036073 0.043633 -0.181721
6  1 0.004091 0.171042 -0.009009
7  1 0.034871 -0.006981 0.077530
8  1 -0.027925 0.064577 0.183285
9  1 -0.200981 0.096750 -0.033657
10 1 0.127409 0.017453 0.138530
11 1 0.005282 0.270526 0.043451
12 1 0.166094 0.125847 0.058592
13 1 -0.194974 0.097738 0.145126
14 1 -0.045379 0.043633 -0.031309
15 1 -0.157080 -0.031416 0.153307
16 1 0.358359 0.069813 0.017005
17 1 -0.193732 -0.066323 -0.025968
18 1 -0.144862 -0.001745 0.062589
19 1 0.116937 0.129154 -0.304920
20 1 0.009280 0.204204 -0.063831
21 1 0.373500 0.020944 0.139925
22 1 -0.003598 0.205949 0.016338
```

处理数据完整代码：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
import pandas as pd
import numpy as np

def dataprocess():
    iris = pd.read_csv('./train_dir/face_feature4.csv')
    result = pd.read_csv('./labels.txt',header=None,sep=' ')
    result.columns=['smile','2','3','4']
    smile=[]
    for k in iris['Column1']:
        smile.append(result['smile'][k])
    iris['Column138']=smile
    detectable=iris['Column1']
    iris.drop(columns=['Column1'],inplace=True)
    # 处理为二分类数据
    iris['Column138'].replace(to_replace=[1,0],value=[+1,-1],inplace=True)
    return iris
```

2.3 使用sklearn的svm对已有的数据进行训练和测试

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
from dataprocess import dataprocess
import numpy as np
import matplotlib.pyplot as plt

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
from sklearn.model_selection import train_test_split
data = dataprocess()
data = data.to_numpy()
#print(data)
x, y = np.split(data, (136,), axis=1)
x = x[:, :]
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1, train_size=0.6)
# 训练svm分类器
clf = SVC(C=0.8, kernel='rbf', gamma=1, decision_function_shape='ovr')
clf.fit(x_train, y_train.ravel())

#计算svc分类器的准确率
print(clf.score(x_train, y_train))  # 精度
y_hat = clf.predict(x_train)
#print(y_hat)
print(clf.score(x_test, y_test))
y_hat = clf.predict(x_test)
```

以下分别为训练集和测试集在最终决策模型上的准确度：

```
0.9414225941422594
0.9266409266409267
```

2.4 保存和读取模型

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
# 保存模型
import pickle
#lr是一个LogisticRegression模型
pkl_filename = "smile_detect.model"
with open(pkl_filename, 'wb') as file:
    pickle.dump(clf, file)
#读取模型
with open(pkl_filename, 'rb') as file:
    clf = pickle.load(file)
```

2.5 使用摄像头进行实时笑脸检测

进行实时笑脸检测，就是要将cv2图像转换为dlib detector能够检测的图像数组：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
#检测器
detector = dlib.get_frontal_face_detector()

...
# 抓取一帧视频
#cv2图像
ret, frame = video_capture.read()
#PIL Image图像
picture = Image.fromarray(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
#转换为numpy数组
img=np.array(picture)
#加载入检测器
dets = detector(img, 1)
```

然后使用dlib检测68个特征点数据：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
shape = predictor(img, d)
#68个特征点
```

```
for i in range(shape.num_parts):
            print("{},{},".format((shape.part(i).x-left)/rect_w,(shape.part(i).y-top)/rect_h))
```

完整代码：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
#视频检测
import cv2
import numpy as np
from PIL import Image, ImageDraw
import os
import time
import dlib
video_capture = cv2.VideoCapture(0)
j=0
detector = dlib.get_frontal_face_detector()
predictor_path = "./train_dir/shape_predictor_68_face_landmarks.dat"
predictor = dlib.shape_predictor(predictor_path)
faces=[]
while j<100:

     # 抓取一帧视频
    ret, frame = video_capture.read()

    picture = Image.fromarray(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
    img=np.array(picture)
    dets = detector(img, 1)
    face=[]
    for k, d in enumerate(dets):
        # 得到的地标/用于面部在框d的部分。
        shape = predictor(img, d)
        rect_w=shape.rect.width()
        rect_h=shape.rect.height()
        top=shape.rect.top()
        left=shape.rect.left()
        for i in range(shape.num_parts):
            #print("{},{},".format((shape.part(i).x-left)/rect_w,(shape.part(i).y-top)/rect_h))
            face=np.append(face,(shape.part(i).x-left)/rect_w)
            face=np.append(face,(shape.part(i).y-top)/rect_h)
    #img = dlib.convert_image(picture)
    print(type(face))
    print(len(face))
    if(len(face)!=0):
        faces.append(face)
        print(faces)
        j+=1
        smile=clf.predict(faces)
        issmile=""
        if(smile[0]==1):
            print("smile")
            issmile="smile"
        else:
            print("nosmile")
            issmile="nosmile"
        faces.clear()
        #cv2.imshow('face',cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))

        # 将图像从 BGR 颜色（OpenCV 使用的）转换为 RGB 颜色（face_recognition 使用的）
        rgb_frame = frame[:, :, ::-1]


        # 找到视频帧中的所有人脸和人脸编码
        face_locations = face_recognition.face_locations(rgb_frame)
        face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

        # 遍历这一帧视频中的每个人脸
```

```
        for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
            # 在脸部周围画一个框
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

            # 在人脸下方画一个带有名字的标签
            cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, issmile, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)


        # 显示结果图像
        cv2.imshow('笑脸检测', frame)

        # 按键盘上的“q”退出!
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# 释放网络摄像头的句柄
video_capture.release()
cv2.destroyAllWindows()
```

3. 完整代码

获取人脸68特征点 feature_process.py：

检测器数据 shape_predictor_68_face_landmarks.dat 从这个网站下载：http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2。

4000张照片放在./train_dir/face/里。

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
#!/usr/bin/python
# 这个文件的内容在公共领域。参见 LICENSE_FOR_EXAMPLE_PROGRAMS.txt
#
# 这个示例程序展示了如何在图像中找到正面人脸并
# 估计他们的姿势。姿势采用 68 个地标的形式。这些是
# 面部的点，例如嘴角、眉毛、眼睛等。
#
# 我们使用的人脸检测器是使用经典的定向直方图
# 梯度（HOG）特征结合线性分类器、图像金字塔、
# 和滑动窗口检测方案制成的。姿势估计器是由
# # 使用 dlib 的论文实现创建的：
##
# Vahid Kazemi 和 Josephine Sullivan，CVPR 2014
# 与回归树集合的一毫秒人脸对齐
# # 并在 iBUG 300-W 人脸地标数据集上进行了训练（参见# https://ibug.doc.ic.ac.uk /resources/facial-point-annotations/）：
#C. Sagonas、E. Antonakos、G、Tzimiropoulos、S. Zafeiriou、M. Pantic。
# 300 面临 In-the-wild 挑战：数据库和结果。
# Image and Vision Computing（IMAVIS），面部地标定位“In-The-Wild”特刊。2016.
# 你可以从以下位置获得训练好的模型文件：
# http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2。
# 请注意，iBUG 300-W 数据集的许可不包括商业用途。
# 所以你应该联系伦敦帝国理工学院，看看
#
#
# 另外，请注意，您可以使用 dlib 的机器学习

#
# 编译/安装 DLIB PYTHON 接口
# 您可以使用以下命令安装 dlib：
# pip install dlib
#
# 或者，如果您想自己编译 dlib，则进入 dlib
# 根文件夹并运行：
# python setup.py install
#
# 编译 dlib 应该可以在任何操作系统上运行，只要你有
# CMake 已安装。在 Ubuntu 上，这可以通过运行
```

```
# 命令
# # sudo apt-get install cmake
#
# 另请注意，此示例需要可以
# pip install numpy

import sys
import os
import dlib
import glob


predictor_path = "./train_dir/shape_predictor_68_face_landmarks.dat"
faces_folder_path = "./train_dir/face/"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(predictor_path)
#win = dlib.image_window()

fi=open("./train_dir/face_feature3.txt","a")
#i=0

#for f in glob.glob(os.path.join(faces_folder_path, "*.jpg")):
for i in range(4000):
    f=faces_folder_path+'file'+'{:0>4d}'.format(i+1)+".jpg"
    img = dlib.load_rgb_image(f)

    # 让检测器找到每个人脸的边界框。
    # 第二个参数中的 1表示我们应该对图像进行 1 次上采样。这个
    # # 将使一切变得更大，并允许我们检测更多的人脸。
    dets = detector(img, 1)
    won=False
    for k, d in enumerate(dets):
        # 得到的地标/用于面部在框d的部分。
        shape = predictor(img, d)
        #特征归一化处理
        rect_w=shape.rect.width()
        rect_h=shape.rect.height()
        top=shape.rect.top()
        left=shape.rect.left()

        fi.write('{},'.format(i))
        won=True
        for i in range(shape.num_parts):
            fi.write("{},{},".format((shape.part(i).x-left)/rect_w,(shape.part(i).y-top)/rect_h))
    if(won):
        fi.write("\n")

fi.close()
```

数据处理dataprocess.py：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
import pandas as pd
import numpy as np

def dataprocess():
    iris = pd.read_csv('./train_dir/face_feature4.csv')
    result = pd.read_csv('./labels.txt',header=None,sep=' ')
    result.columns=['smile','2','3','4']
    smile=[]
    for k in iris['Column1']:
        smile.append(result['smile'][k])
    iris['Column138']=smile
    detectable=iris['Column1']
    iris.drop(columns=['Column1'],inplace=True)
    # 处理为二分类数据
```

```
    iris['Column138'].replace(to_replace=[1,0],value=[+1,-1],inplace=True)
    return iris


#用自己的脸作为监测数据
def getmyself():
    myface = pd.read_csv('./train_dir/myfeatures.csv')
    myface.drop(columns=['Column1'],inplace=True)
    myface.drop(columns=['Column138'],inplace=True)
    return myface
```

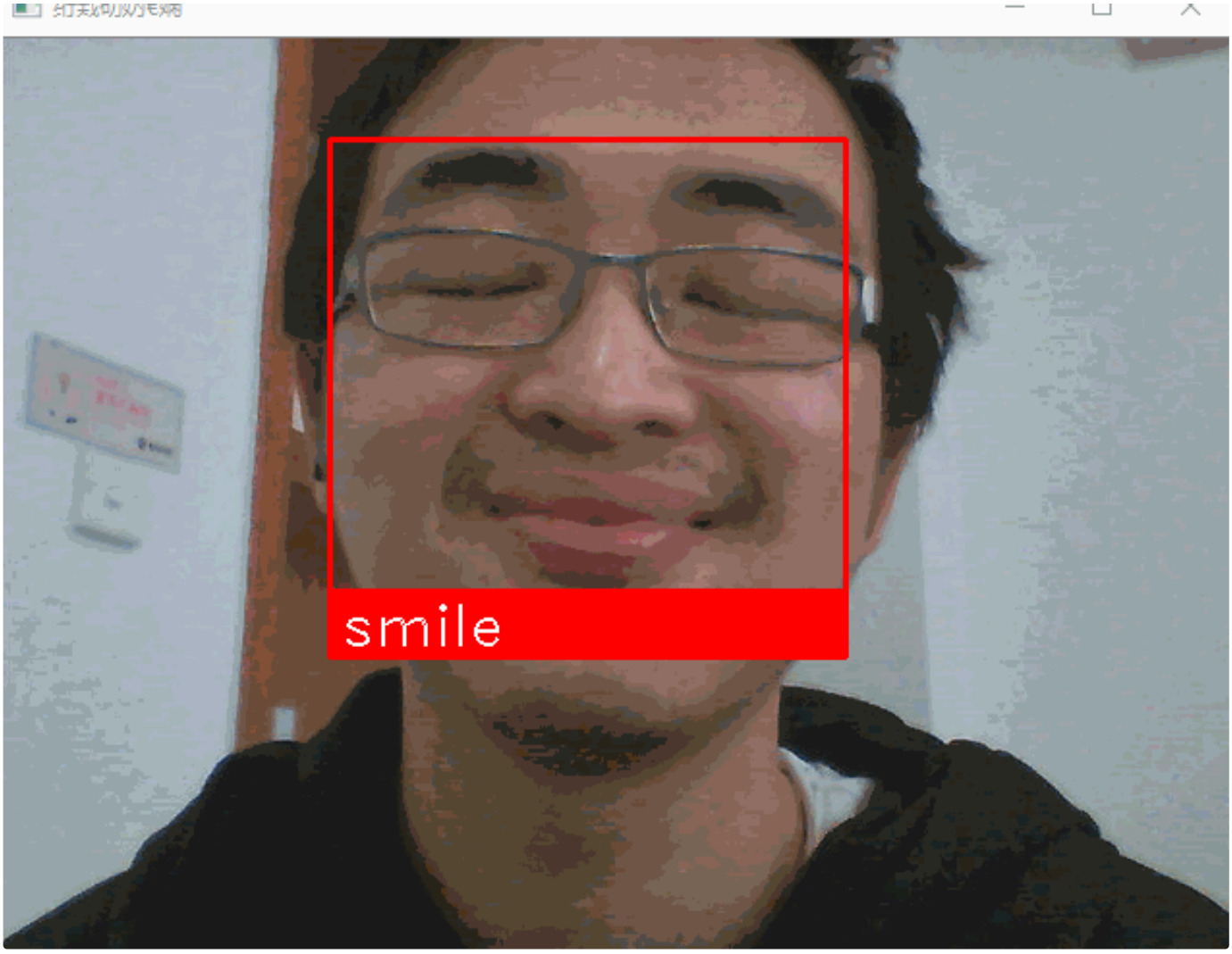
人脸检测模型训练以及实时人脸检测：

代码语言：javascript　　代码运行次数：0　　运行　　AI代码解释

```
from dataprocess import dataprocess
from dataprocess import getmyself
import numpy as np
import matplotlib.pyplot as plt
import face_recognition
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
#获取4000人脸数据
data = dataprocess()
data = data.to_numpy()

x, y = np.split(data, (136,), axis=1)
x = x[:, :]
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1, train_size=0.6)
# 训练svm分类器
clf = SVC(C=0.8, kernel='rbf', gamma=1, decision_function_shape='ovr')
clf.fit(x_train, y_train.ravel())
#计算svc分类器的准确率
print(clf.score(x_train, y_train))  # 精度
y_hat = clf.predict(x_train)

print(clf.score(x_test, y_test))
y_hat = clf.predict(x_test)


#用自己的人脸数据做检测
#myself=getmyself()
#myself=myself.to_numpy()
#print(myself)
#AmISmile=clf.predict(myself)
#print(AmISmile)

#视频检测
import cv2
import numpy as np
from PIL import Image, ImageDraw
import os
import time
import dlib
video_capture = cv2.VideoCapture(0)
j=0
detector = dlib.get_frontal_face_detector()
predictor_path = "./train_dir/shape_predictor_68_face_landmarks.dat"
predictor = dlib.shape_predictor(predictor_path)
faces=[]
while j<100:

     # 抓取一帧视频
    ret, frame = video_capture.read()

    picture = Image.fromarray(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
    img=np.array(picture)
```

```
        dets = detector(img, 1)
        face=[]
        for k, d in enumerate(dets):
            # 得到的地标/用于面部在框d的部分。
            shape = predictor(img, d)
            rect_w=shape.rect.width()
            rect_h=shape.rect.height()
            top=shape.rect.top()
            left=shape.rect.left()
            for i in range(shape.num_parts):

                face=np.append(face,(shape.part(i).x-left)/rect_w)
                face=np.append(face,(shape.part(i).y-top)/rect_h)

        print(type(face))
        print(len(face))
        if(len(face)!=0):
            faces.append(face)
            print(faces)
            j+=1
            smile=clf.predict(faces)
            issmile=""
            if(smile[0]==1):
                print("smile")
                issmile="smile"
            else:
                print("nosmile")
                issmile="nosmile"
            faces.clear()
            #cv2.imshow('face',cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))

            # 将图像从 BGR 颜色（OpenCV 使用的）转换为 RGB 颜色（face_recognition 使用的）
            rgb_frame = frame[:, :, ::-1]


            # 找到视频帧中的所有人脸和人脸编码
            face_locations = face_recognition.face_locations(rgb_frame)
            face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

            # 遍历这一帧视频中的每个人脸
            for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
                # 在脸部周围画一个框
                cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

                # 在人脸下方画一个带有名字的标签
                cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
                font = cv2.FONT_HERSHEY_DUPLEX
                cv2.putText(frame, issmile, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)


            # 显示结果图像
            cv2.imshow('笑脸检测', frame)

            # 按键盘上的“q”退出!
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break


    # 释放网络摄像头的句柄
    video_capture.release()
    cv2.destroyAllWindows()
```

检测效果：



代码下载

4. 参考

[1] sklearn 模型的保存与加载