Universe Scientific Publishing
Computer System Networking and Telecommunications
计算机系统网络和电信，2019,1(2)
ISSN:2661-3719 (Print); 2661-3727 (Online)

Research on Expression Recognition Technology based on OpenCV

Xinyu YANG Zhengde BAO Yawen TANG
School of Computer and Software, Jincheng College, Sichuan University, Chengdu, 611731

Abstract

As the main language of artificial intelligence (AI), Python has shown a strong momentum in recent years. The usage of python has increased linearly and has become one of the most popular programming languages. This paper is based on Python language, using OpenCV as carrier and object detection technology to realize facial expression recognition. At the same time, the importance of expression recognition and the concrete process of expression recognition are expounded.

Key Words

Video Image Processing, Target Detection, Expression Recognition

DOI:10.18686/jsjxt.v1i2.705

基于 OpenCV 的表情识别技术研究

杨鑫雨 鲍正德 唐娅雯
四川大学锦城学院计算机与软件学院，四川成都，611731

摘 要

Python 作为人工智能的主要使用语言，近几年呈现出强劲的势头，python 的使用率呈线性增长，已经成为最受欢迎的程序设计语言之一。本文基于 Python 语言，以 OpenCV 为载体，借助目标检测技术，实现人脸表情识别。同时阐述了表情识别的重要性以及表情识别实现的具体过程。

关键字

视频图像处理；目标检测；表情识别

1.引言

随着计算机视觉市场的持续扩大，人们对人机交互的需求迅速增长以及研究机器学习的层次越来越深入，表情识别就是众多研究之一——计算机获取到静态的图像或者动态的视频并分离出表情特征，从而分析被检测目标的情绪。OpenCV 在人脸识别领域也是关键的工具，它大大提高了计算机处理的速度，所以基于 OpenCV 实现表情识别也是对人脸识别领域的进一步探索。

2.OpenCV 的概念以及应用

OpenCV 是一个由 C 语言编写的，起到了简化计算机程序作用的开源计算机可视化库[1]。事实证明 OpenCV 大大提升了代码编写的执行速度[2]。Opencv 在人脸识别、人机交互、图像分割等领域都发挥了巨大的作用，简化了编程工作。除了 OpenCV，还有其他主流的视觉函数库比如 LTI,VXL,OpenCV+IPP，其中性能最好的是利用 IPP 加速的 Opencv。

3.表情识别在人类社会中的应用

表情识别是让摄像头或计算机模拟人的眼睛，实现机器提取出人脸面部特征并判断出人类心理状态的技术,可能听上去非常的多此一举，人眼识别的精确度明显高于机器识别的精确度，为何要浪费人力物力去研究虚无缥聊的表情识别？其实表情识别这项技术已经潜移默化地应用在人类社会中的众多领域。比如去年很火的 FaceU 拍照软件，这款软件会根据你的面部表情变换来切换拍摄特效，比如张嘴就会出现吐舌头的特效，

闭上嘴就会消失。正是因为运用到 dlib 技术，dlib 在人脸上标注了 68 个特征点，这些特征点连接的走向以及弯曲程度可以判断人的脸部特征和表情变换。dlib 比起 OpenCV 的结果更精确一些，也不需要训练模型，操作起来比较简单易懂，但是 OpenCV 更适合多脸检测，在遮挡或是不同方向的情况下，OpenCV 的效果更好，而且在 CPU 上运行的速度也快，更适合用于做多脸识别的项目。表情识别最主要是应用在智能机器人的研发上，智能机器人已经可以与人交流并作出相应的反应，著名的阿尔法狗就给人类社会带来了不小的惊喜，这一项研发也正在改变着人与计算机的关系，能够使计算机更好地服务于人类。而今表情识别在其他领域也发光发热，在刑侦方面，摄像头可以记录罪犯面部表情变化来判断其心理情绪，极大的帮助了警方破案，也给社会也带了积极的力量。在游戏领域，也可以制作虚拟人物的表情变化，使人物角色更加鲜活[5]。那么随着技术发展的洪流，在教育领域是否可以制作一个学生表情识别的系统呢

4.表情识别的实现

4.1 项目及工具介绍

本文中的实验是要实现一个教学质量检测系统即根据课堂上学生表情变化反应课堂质量好坏，老师可以根据学习状态调整教学方式，也对学生起了监督作用。本项目以 Windows10 为系统,Python 为设计语言，使用了 Anoconda3 与 JetBrains PyCharm 社区版两个开发工具，Python 和 OpenCV 都可以在 Anoconda3 中下载，主要在 Anoconda3 中训练模型，在 PyCharm 中做表情识别，并且使用 HTML5 制作前端页面，再与 PyCharm 制作的后端交互，实现了一个功能较为完整的表情识别网页。

4.2 OpenCV 之人脸检测

说到人脸检测就要涉及到目标检测的知识，目标检测一路发展而来经历了很多阶段，滑动窗口就是目标检测的最原始的方式，是使用不同大小的框在图像上滑动，然后经过卷积计算与已经训练好的分类器共同辨别存在物体的可能性，这种方法简单易懂，但是十分耗时，为了提高检测效率，从而产生了 RCNN 等快速的检测方法,RCNN 是使用选择性搜索的方法提取候选框，再利用 CNN 提取特征向量，然后使用 SVM 分类器判断物体是否存在，末了用回归函数精修位置。滑窗法与 RCNN 都是基础的物体检测方法，往后还有 Fast R-CNN，Faster-RCNN，SSD 等高效的方法，OpenCV 人脸检测就和 RCNN 目标检测法很相似。OpenCV 怎样在一段视频里框出出人脸的位置并获取[3]？首先我们要将视频分帧取成一个图片集，对每一张图片进一步处理成卷积网络能识别的数据结构。OpenCV 利用选择性搜索的物体检测方法生成了 1k-2k 个候选框，对候选框使用深度卷积网络进行特征提取，然后使用 OpenCV 的人脸检测分类器判断候选区域是否是人脸，如果是人脸，就可以使用训练好的模型进一步做表情识别。

人脸检测

```
def face_detect(imgdata):
    emotion_labels = ['angry', 'disgust:', 'fear',
'happy', 'sad', 'surprise', 'neutral']#表情标签
    num_class = 7
    face_cascade =
cv2.CascadeClassifier("haarcascade_frontalface_d
efault.xml")
    emotion = {'angry':0, 'disgust':0, 'fear':0,
'happy':0, 'sad':0, 'surprise':0, 'neutral':0}
        #转换成灰度图
        img_data = base64.b64decode(imgdata)
        nparr = np.frombuffer(img_data, np.uint8)
        #人脸检测
        faces =
face_cascade.detectMultiScale(gray,1.1,5)
        for (x,y,w,h) in faces:
            cv2.rectangle(img, (x,y),
(x+w,y+h),(0,0,255),2)
            img_fer=gray[y:y+w,x:x+h]
            new_im =
cv2.resize(img_fer,(48,48),0,0)
            result=predict_emotion(new_im)
            result_sum = np.array([0]*num_class)
            result_sum = result_sum +
np.array(result[1])
            angry, disgust, fear, happy, sad,
surprise, neutral = result_sum
            label = np.argmax(result_sum)
```

```
                emo = emotion_labels[label]
                emotion[emo]+=1

    img=cv2.putText(img,emo,(x,y-5),font,1.0,(255,255,
    255),2)
    return emotion
```

4.3 训练模型

训练模型究竟是什么意思呢？通俗一点来说就是根据已知的数据训练出一个规则机器，然后新输入一个未知的参数，机器也能按照规则输出一个与真实值相近的结果，那么我们就需要训练一个表情识别的机器来判断人的情绪。本文使用 TensorFlow 深度学习框架和 fer2013 数据库来训练网络结构是 AlexNet 的模型。Fer2013 数据库里并不是图片，而是一个 csv 文件，我们需要将 csv 文件转化成图片格式分为训练集（train），测试集（text），验证集（val），fer2013 有 0-6 七个情绪 angry, disgust, fear, happy, sad, surprise, neutral，训练集里每张图片已经打上情绪标签。再说说 AlexNet 网络结构，它是一个比较简单的网络结构，总共有 8 层，包括五个卷积层和三个全连接层，卷积层用于特征的抽象和提取，全连接层用于逻辑判断和去除空间信息。将训练集中的图片输入到卷积网络中，卷积层就可以获取特征然后开始学习，将 7 个情绪作为 label，最后把训练好的模型保存起来备用，然后将训练好的模型使用测试集测试，观察准确率，一般模型的的准确率不超过人眼判断的概率（65%）。

```
        #训练模型数据
        def train_model(self)
        sgd=SGD(lr=0.01,decay=1e-6,momentum=0.9,
    nesterov=True)
        self.model.compile(loss='categorical_crossentro
    py',optimizer=sgd,
        ,metrics=['accuracy'])[4]
            #自动扩充训练样本
            train_datagen = ImageDataGenerator(
            rescale = 1./255,
            shear_range = 0.2,
            zoom_range = 0.2,
            horizontal_flip=True)
            #归一化验证集
            val_datagen = ImageDataGenerator(
                rescale = 1./255)
            eval_datagen = ImageDataGenerator(
                rescale = 1./255)
            #以文件分类名划分 label
            train_generator = train_datagen.f1 (
                root_path+'/train',
                target_size=(img_size,img_size),
                color_mode='grayscale',
                batch_size=batch_siz,
                class_mode='categorical')
            val_generator = val_datagen.f1(
                root_path+'/val',
                target_size=(img_size,img_size),
                color_mode='grayscale',
                batch_size=batch_siz,
                class_mode='categorical')
            eval_generator = eval_datagen.f1(
                root_path+'/test',
                target_size=(img_size,img_size),
                color_mode='grayscale',
                batch_size=batch_siz,
                class_mode='categorical')
            early_stopping =
    EarlyStopping(monitor='loss',patience=3)
            history_fit=self.model.fit_g1 (
                train_g1,

    steps_per_epoch=800/(batch_siz/32),#28709
                nb_epoch=nb_epoch,
                validation_data=val_g1,
                validation_steps=2000,
                #callbacks=[early_stopping]
                )
            history_eval=self.model.evaluate_g1(
                eval_generator,
                steps=2000)
            history_predict=self.model.predict_g1(
                eval_generator,
                steps=2000)
```

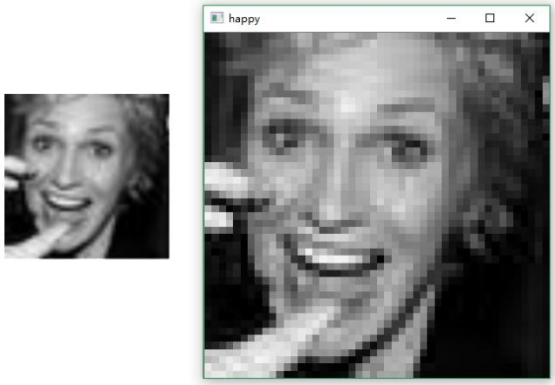
```
            with open(root_path+'/model_fit_log','w')
as f:
                f.write(str(history_fit.history))
            with
open(root_path+'/model_predict_log','w') as f:
                f.write(str(history_predict))
            print("%s: %.2f%%" %
(self.model.metrics_names[1], history_eval[1] *
100))
            print('model trained')
```

```
        #保存模型
        def save_model(self):
            model_json=self.model.to_json()
            with open(root_path+"/model_json.json",
"w") as json_file:
                json_file.write(model_json)

self.model.save_weights(root_path+'/model_weight.
h5')
            self.model.save(root_path+'/model.h5')
            print('model saved')
```

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 48, 48, 32)	320
activation_8 (Activation)	(None, 48, 48, 32)	0
max_pooling2d_4 (MaxPooling2	(None, 16, 16, 32)	0
conv2d_6 (Conv2D)	(None, 16, 16, 64)	18496
activation_9 (Activation)	(None, 16, 16, 64)	0
conv2d_7 (Conv2D)	(None, 16, 16, 64)	36928
activation_10 (Activation)	(None, 16, 16, 64)	0
max_pooling2d_5 (MaxPooling2	(None, 8, 8, 64)	0
dropout_3 (Dropout)	(None, 8, 8, 64)	0
conv2d_8 (Conv2D)	(None, 8, 8, 128)	73856
activation_11 (Activation)	(None, 8, 8, 128)	0
conv2d_9 (Conv2D)	(None, 8, 8, 128)	147584
activation_12 (Activation)	(None, 8, 8, 128)	0
max_pooling2d_6 (MaxPooling2	(None, 4, 4, 128)	0
dropout_4 (Dropout)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 2048)	4196352
activation_13 (Activation)	(None, 2048)	0
dropout_5 (Dropout)	(None, 2048)	0
dense_5 (Dense)	(None, 1024)	2098176
activation_14 (Activation)	(None, 1024)	0
dropout_6 (Dropout)	(None, 1024)	0
dense_6 (Dense)	(None, 7)	7175
activation_15 (Activation)	(None, 7)	0

（1）AlexNet 网络结构图



（2）原图和灰度对比结果

4.4 表情分类

在此之前已经做好了图像预处理和模型训练的前

期工作，接下来就是候选框里的人脸输入到模型中进行判断，模型的输出结果是列举出一张特征图七个情绪的概率，其中概率值最大的情绪就作为这个图的情绪。一张图片中可能有多张人脸，如果每一张人脸都单独输出结果，页面会显得繁琐,难免会有重复，显得结果不够直观，所以后端对图片中不同人脸的相同情绪做了一个统计，输出每个情绪的总数。

表情统计：

```
def controller():
    imgdata = request.get_data()
    imgdata = imgdata[37:]
    if (len(imgdata) % 3 == 1):
        imgdata += b"=="
    elif (len(imgdata) % 3 == 2):
        imgdata += b"="
    result = face_detect(imgdata)
    count  =  0    #总人数
    for i in result:
        count += result[i]
        result[i]=str(result[i])
    result['count']=str(count)
    return jsonify({'result': result})
```

```
['0.jpg', '1.jpg', '10.jpg', '11.jpg', '12.jpg', '13.jpg', '14.jpg', '15.jpg', '16.jpg', '17.jpg', '18.jpg', '19.jpg', '2.jpg', '20.jpg',
'21.jpg', '22.jpg', '23.jpg', '24.jpg', '25.jpg', '26.jpg', '27.jpg', '28.jpg', '29.jpg', '3.jpg', '30.jpg', '31.jpg', '32.jpg', '33.jpg',
'34.jpg', '35.jpg', '36.jpg', '37.jpg', '38.jpg', '39.jpg', '4.jpg', '40.jpg', '41.jpg', '42.jpg', '43.jpg', '44.jpg', '45.jpg', '46.jpg',
'47.jpg', '48.jpg', '49.jpg', '5.jpg', '50.jpg', '6.jpg', '7.jpg', '8.jpg', '9.jpg']
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 5ms/step
1/1 [==============================] - 0s 5ms/step
[0.25632903, 0.010830225, 0.04850967, 0.0028956695, 0.40190548, 0.0070463284, 0.27248356]
[0.28887635, 0.008345305, 0.023705162, 0.009220594, 0.26473296, 0.0030575686, 0.40206203]
[0.14924143, 0.013588456, 0.030094968, 0.01512248, 0.6109976, 0.0016887634, 0.1792662]
[0.69444682 0.03276399 0.1023098  0.02723874 1.27763605 0.01179266
 0.85381179]
angry: 0.6944468170404434 disgust: 0.032763986848294735  fear: 0.10230979882180691  happy: 0.027238742914050817  sad: 1.2776360511779785  s
urprise: 0.011792660341598094  neutral: 0.8538117855787277
Emotion :  sad
1/1 [==============================] - 0s 6ms/step
1/1 [==============================] - 0s 6ms/step
1/1 [==============================] - 0s 4ms/step
[0.023271082, 0.00012016544, 0.018603353, 0.08901693, 0.1858965, 0.0008397604, 0.68225217]
[0.021138912, 0.000121058496, 0.029120728, 0.08800087, 0.35094127, 0.0014306377, 0.5092466]
```

（3）后端输出结果图

4.5 实验结果

为了能够直观清晰的观察表情的变化，本项目实现了前后端交互，其使用到了 python 的 Flask 库和 Ajax 异步响应， flask 适用于小型网站开发，拓展性强，灵活多变，比较适合本项目，前端页面使用 Html5,在 javascript 中利用 Ajax 发送请求，后端将数据返回到前端 url 地址并将数据实时反映到前端的图表和页面上。

（4）网页效果图

4.6 项目面临的机会与挑战

在信息爆炸与计算机技术快速发展的年代，表情识别技术在学术界已经有很多探索与发现，而且依然有许多学者投身于人脸识别的研究，本项目借鉴了前人的经验与优点，简单完成了多人表情识别，能够实现课堂教学检测的功能，在教育领域算是一个创新的发展，未来

会有很大的需求市场。首先本项目的功能方便简洁，后期运维简单，不需要大量人力却能减轻老师的工作。其次，本项目能够通过学生表情判断课堂质量好坏，对学生起到了监督作用，大大提高了教学的效率。但是项目的内容和功能还有待完善，表情识别的精确度还需要提升，可以参考与学习像 VGG 这样更深层次的网路结构来增强机器学习的效果,还要增加逻辑判断直观反映教学质量，而且前端界面需要美化。所以往后依然会继续深入研发这个项目，抓住发展的机会，解决面临的困难与挑战，使其成为一个优秀的学情分析系统并运用到实践中，使教育更加智能和高效。

5.结束语

如今的表情识别还未普及到日常生活中，但是研究深度学习的学者越来越多，对表情识别的认识而越来越深刻，学者们正在不同领域进行着表情识别的实践，而且随着 OpenCV 的发展与升级，代码的处理时间会越来越短，代码编写也会更加简洁，表情识别技术实时反馈信息的速度也会加快，未来的表情识别极有可能作为重要的功能运用到各个领域，甚至实现机器的高度拟人化，。总之，表情识别技术的发展前景一片大好，Python 语言与 OpenCV 也是大势所趋，就像一辆一往无前的列车，跑在了时代的前端。

参考文献

[1]季云峰, 朱玲, 沈晏妮. 慕基于 OpenCV 的比赛图片中的乒乓球球体识别[J]. 微型电脑应用, 2016, 32(4):68-70.
[2] 吴俊. 基于自然场景的立体视觉实现研究[D]. 2014.
户,2018,25(03):16-18+12.
[3] 蔡兴桐. 移动机器人视觉导航研究. Diss. 2010.
[4]赵轶 . 深度学习在硅钢钢片缺陷识别中的应用[J]. 数字技术与应用, 2017.
[5] 惠阳. 基于中国京剧脸谱的网络体感游戏人物表情设计研究[D]. 哈尔滨工业大学, 2014.

作者简介

第一作者：杨鑫雨（1998-5），女，汉族，成都，本科，四川大学锦城学院，研究方向：J2EE
第二作者（通讯作者）：鲍正德（1989 年），男，汉族，黑龙江哈尔滨市人，研究生，讲师，四川大学锦城学院，研究方向：电子商务
第三作者：唐娅雯（1999-），女，汉，四川省资阳市，本科，四川大学锦城学院，研究方向：信息管理、J2EE