图像识别技术文档

项目概述

- 项目名称：面部表情识别
- 项目背景：在与客户交流的过程中，通过客户的面部表情来判断用户对话题是否感兴趣，营销人员或者沟通人员可以从中找到客户感兴趣的方面，或者判断客户的购买欲望
- 图像识别是指利用计算机对图像进行处理、分析、理解，图像识别的过程可分为图像处理和图像分析两个过程
  图像处理是指对所需要分析的图像进行一系列的图形操作，包括拉伸缩放、旋转翻转、颜色变换、像素保留……，图像分析是指通过图像的特征来反馈图像信息并且分类，最简单的是基于统计的方法、最常用的是深度学习方法
- 技术支持
  图像处理：VoTT、OpenCV、PIL
  图像分析：TensorFlow、Keras、CNN

应用领域

- 总体目标
  a) 一级目标
  通过客户的面部表情变化，来抓住客户感兴趣的点，提高沟通效率
  b) 二级目标
  学习图像识别的技术
  c) 三级目标
  学习深度学习实现工具
  d) 四级目标
  了解图像识别技术实现的步骤，技术原理

实验步骤

- **资源**
  - 数据资源：本次实验为面部表情的二分类识别，情绪为高兴和沮丧，各表情图像5000张
  - 技术支持：Keras、TensorFlow、VoTT、OpenCV、PIL
    - Keras

Keras是一个由Python编写的开源人工神经网络库，可以作为Tensorflow、Microsoft-CNTK和Theano的高阶应用程序接口，进行深度学习模型的设计、调试、评估、应用和可视化，Keras的神经网络API是在封装后与使用者直接进行交互的API组件，在使用时可以调用Keras的其它组件。除数据预处理外，使用者可以通过神经网络API实现机器学习任务中的常见操作，包括人工神经网络的构建、编译、学习、评估、测试等。

- TensorFlow

  TensorFlow由谷歌人工智能团队谷歌大脑（Google Brain）开发和维护，拥有包括TensorFlow Hub、TensorFlow Lite、TensorFlow Research Cloud在内的多个项目以及各类[应用程序接口](API）[2]。自2015年11月9日起，TensorFlow依据阿帕奇授权协议（Apache 2.0 open source license）开放源代码，TensorFlow是一个基于数据流编程（dataflow programming）的符号数学系统，被广泛应用于各类机器学习（machine learning）算法的编程实现，谷歌大脑自2011年成立起开展了面向科学研究和谷歌产品开发的大规模深度学习应用研究，其早期工作即是TensorFlow的前身DistBelief。

- VoTT

  VoTT是一个用TypeScript编写的React + Redux Web应用程序。该项目是通过Create React App启动的，用于图像和视频资产的开源注释和标签工具，VoTT可以作为本机应用程序安装，也可以从源代码运行。VoTT也可以作为独立的Web应用程序使用，并且可以在任何现代Web浏览器中使用。

- OpenCV

  OpenCV是一个基于BSD许可（开源）发行的跨平台计算机视觉库，可以运行在Linux、Windows、Android和Mac OS操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

- PIL

  python图像处理库，这个库支持多种文件格式，并提供了强大的图像处理和图形处理能力。

方法

试验阶段

- 在项目开始初步试验阶段，通过人工百度搜索人像表情图像，利用VoTT工具进行裁切和标注，各表情100张，并按照7,2,1的比列把数据分为训练集、测试集、验证集。
- 利用Keras深度学历框架搭建简单的CNN进行训练，结果显示CNN在图像处理领域效果很不错，在模型未经过调整前准确率能超过50%。

中期工作

- 扩大数据集，利用爬虫批量获取人像图片，在爬取百度图片的过程中，如果直接获取那最多只能获取到30张图像，之后程序就自动停止了，采用的解决方法是，通过观察url发现可以添加page页码这个选择来达到滚轮向下刷新图片的作用，因此添加一个页码循环来爬取图像。

```
# 爬取图片
import re
import requests
from urllib import error
from bs4 import BeautifulSoup
import os
```

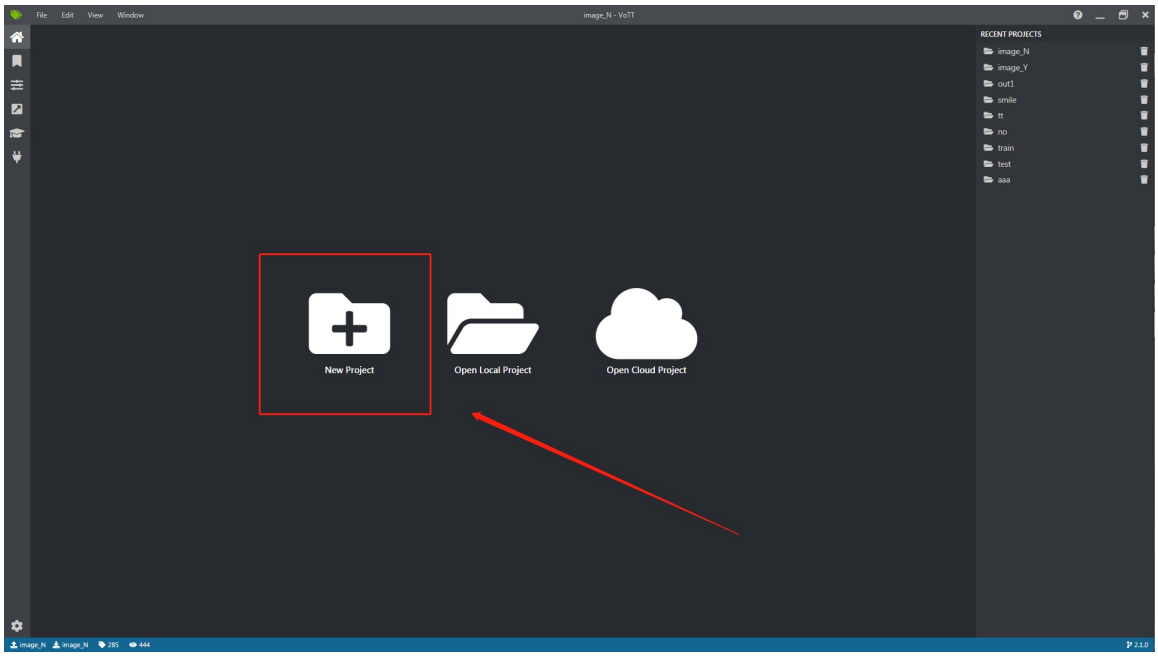
```
num = 0
numPicture = 0
file = ''
List = []

def Find(url):
    global List
    print('正在检测图片总数，请稍等.....')
    t = 0
    i = 1
    s = 0
    while t < 1000:
        Url = url + str(t)
        try:
            Result = requests.get(Url, timeout=7)
        except BaseException:
            t = t + 60
            continue
        else:
            result = Result.text
            pic_url = re.findall('"objURL":"(.*?)",', result, re.S)  # 先利用正
则表达式找到图片url
            s += len(pic_url)
            if len(pic_url) == 0:
                break
            else:
                List.append(pic_url)
                t = t + 60
    return s

def recommend(url):
    Re = []
    try:
        html = requests.get(url)
    except error.HTTPError as e:
        return
    else:
        html.encoding = 'utf-8'
        bsObj = BeautifulSoup(html.text, 'html.parser')
        div = bsObj.find('div', id='topRS')
        if div is not None:
            listA = div.findAll('a')
            for i in listA:
                if i is not None:
                    Re.append(i.get_text())
        return Re


def dowmloadPicture(html, keyword):
    global num
    # t =0
    pic_url = re.findall('"objURL":"(.*?)",', html, re.S)  # 先利用正则表达式找到
图片url
```

```
    print('找到关键词:' + keyword + '的图片，即将开始下载图片...')
    for each in pic_url:
#         print('正在下载第' + str(num + 1) + '张图片，图片地址:' + str(each))
        print('\r',"已下载"+str(num+1).ljust(10)+"张图片",end='')
        try:
            if each is not None:
                pic = requests.get(each, timeout=7)
            else:
                continue
        except BaseException:
            print('错误，当前图片无法下载')
            continue
        else:
            string = file + r'\\' + keyword + '_' + str(num) + '.jpg'
            fp = open(string, 'wb')
            fp.write(pic.content)
            fp.close()
            num += 1
        if num >= numPicture:
            return


if __name__ == '__main__':  # 主函数入口
    word = input("请输入搜索关键词(可以是人名，地名等): ")
    #add = 'http://image.baidu.com/search/flip?tn=baiduimage&ie=utf-
8&word=%E5%BC%A0%E5%A4%A9%E7%88%B1&pn=120'
    url = 'http://image.baidu.com/search/flip?tn=baiduimage&ie=utf-8&word=' +
word + '&pn='
    tot = Find(url)
    Recommend = recommend(url)  # 记录相关推荐
    print('经过检测%s类图片共有%d张' % (word, tot))
    numPicture = int(input('请输入想要下载的图片数量 '))
    file = input('请建立一个存储图片的文件夹，输入文件夹名称即可')
    words=input("请输入图片前缀")
    y = os.path.exists(file)
    if y == 1:
        print('该文件已存在，请重新输入')
        file = input('请建立一个存储图片的文件夹，)输入文件夹名称即可')
        os.mkdir(file)
    else:
        os.mkdir(file)
    t = 0
    tmp = url
    while t < numPicture:
        try:
            url = tmp + str(t)
            result = requests.get(url, timeout=10)
            print(url)
        except error.HTTPError as e:
            print('网络错误，请调整网络后重试')
            t = t+60
        else:
            dowmloadPicture(result.text, words)
```

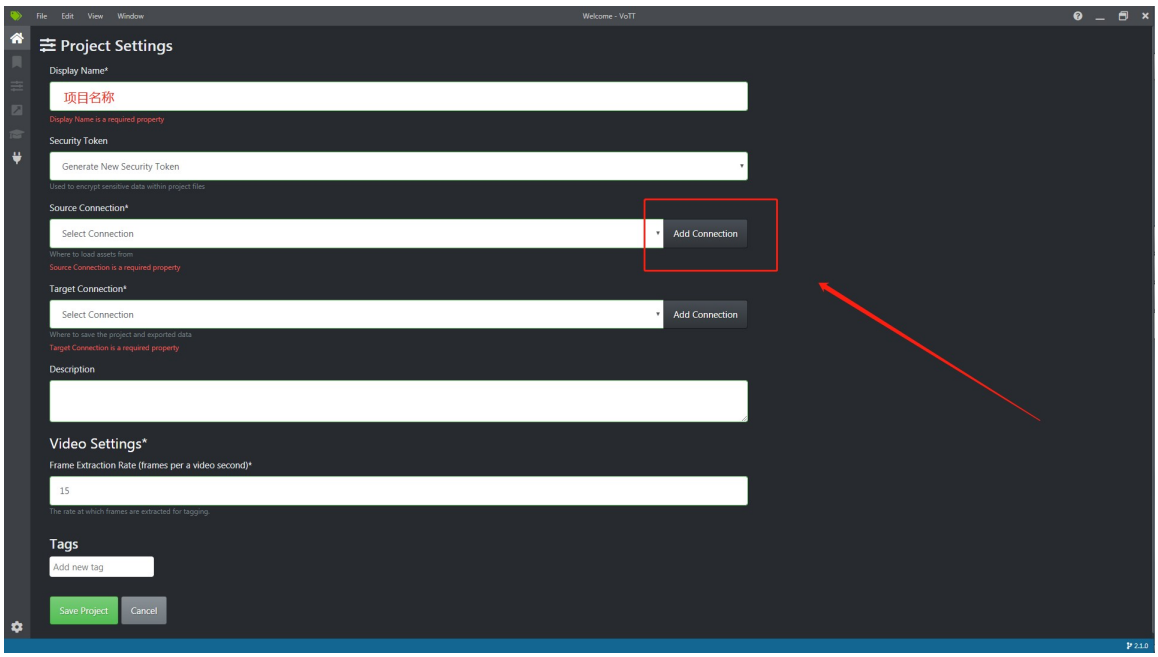
```
            t = t + 60
```

- 再利用VoTT工具时框选的图像只记录位置大小，并不能裁切保存图像，所以采用VoTT加Python的方式来完成图像标注和裁切，在VoTT新建项目，选择图像位置进行框选和标注，完成后保存导出json格式文件，在利用Python读取json文件，打开对应图像按照框选的大小进行裁切，然后转化为数组也就是像素保存，按照6,2,2的比例划分数据集保存CSV文件。

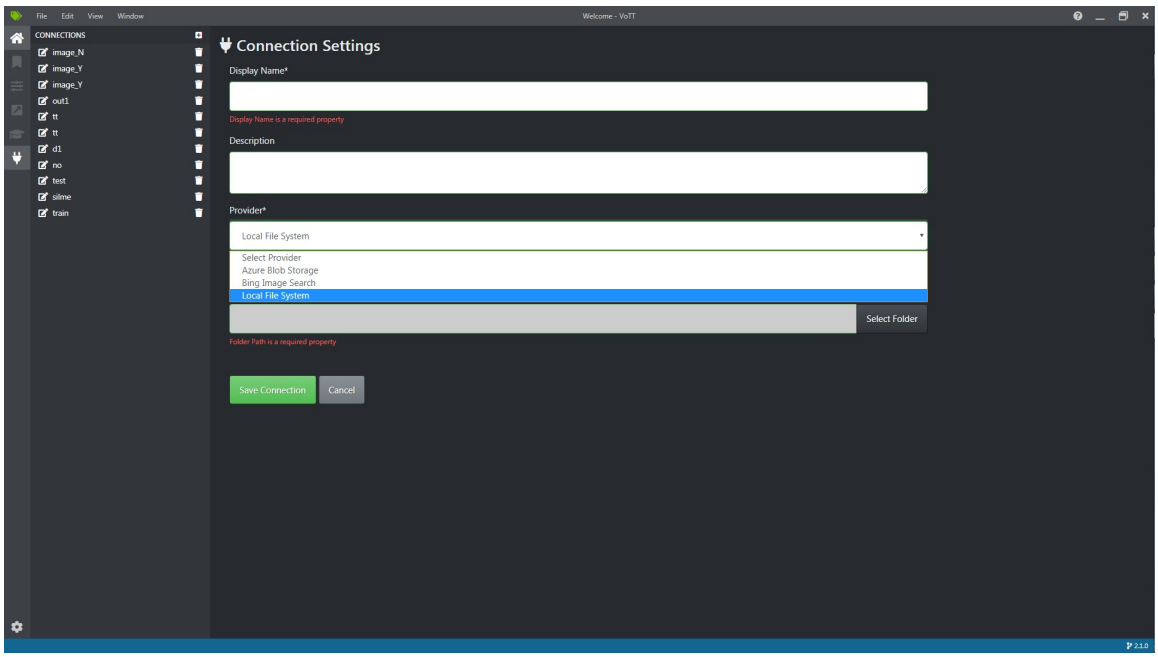
- 打开VoTT，选择新建项目



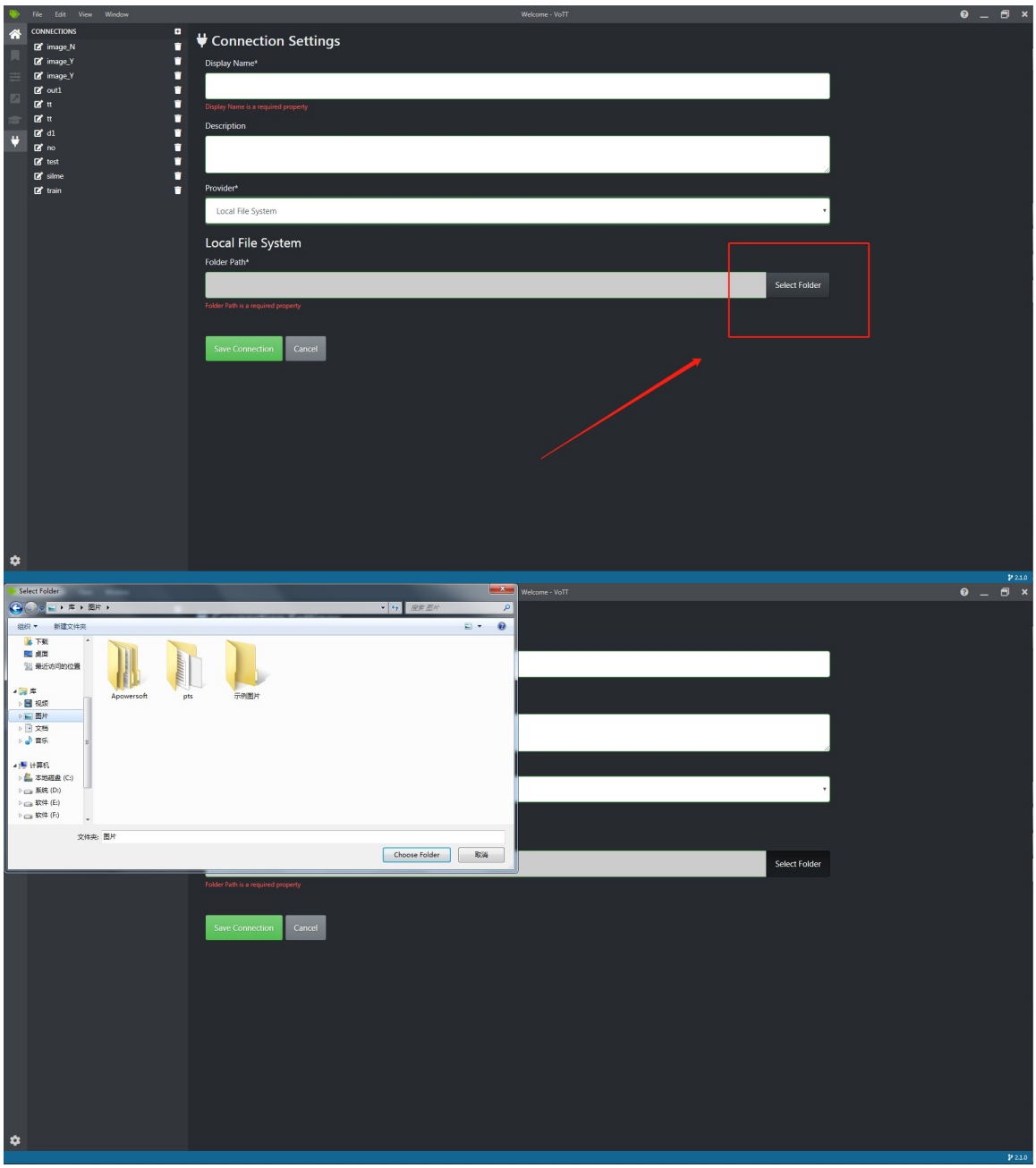
- 新建连接点



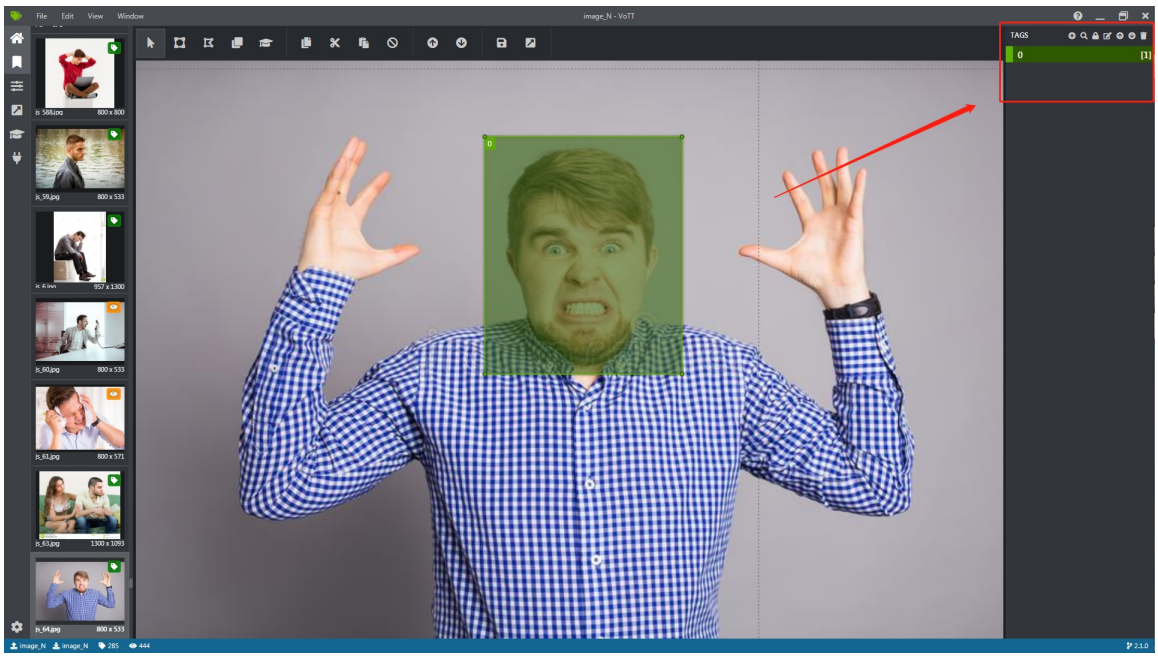
- 选择为添加位置


File
Edit
View
Window
Welcome - VoTT
CONNECTIONS
image_N
image_Y
image_Y
out1
tt
tt
d1
no
test
slime
train
Connection Settings
Display Name*
Display Name is a required property
Description
Provider*
Local File System
Select Provider
Azure Blob Storage
Bing Image Search
Local File System
Select Folder
Folder Path is a required property
Save Connection
Cancel
2.1.0

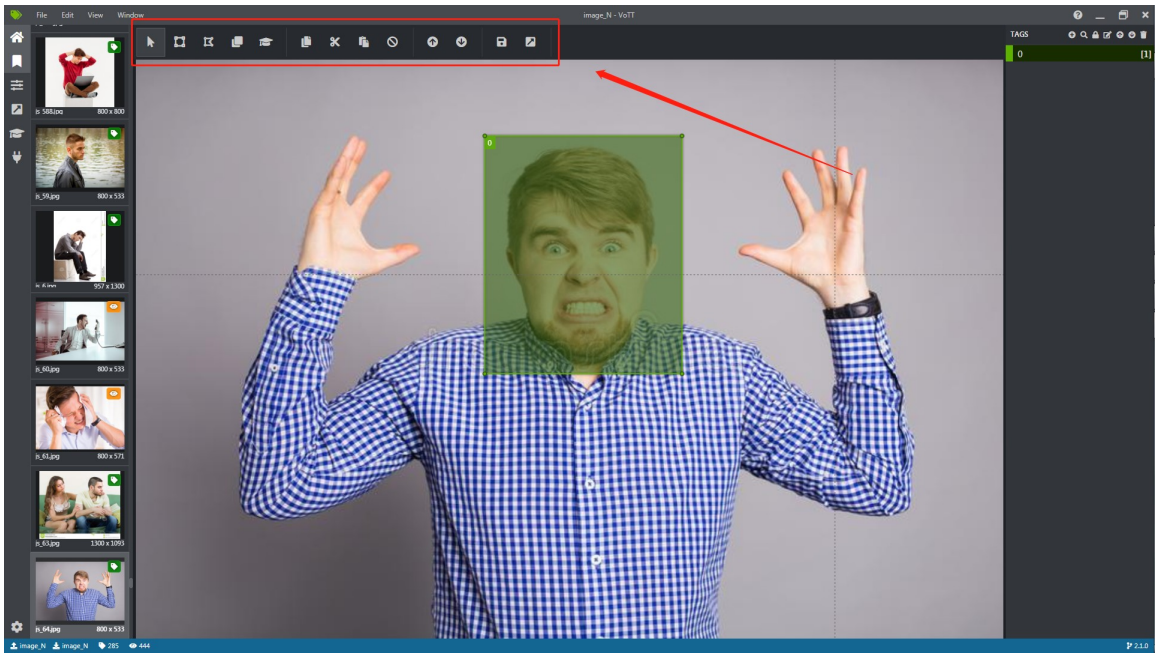
- 选择图像所在文件夹位置，添加确认



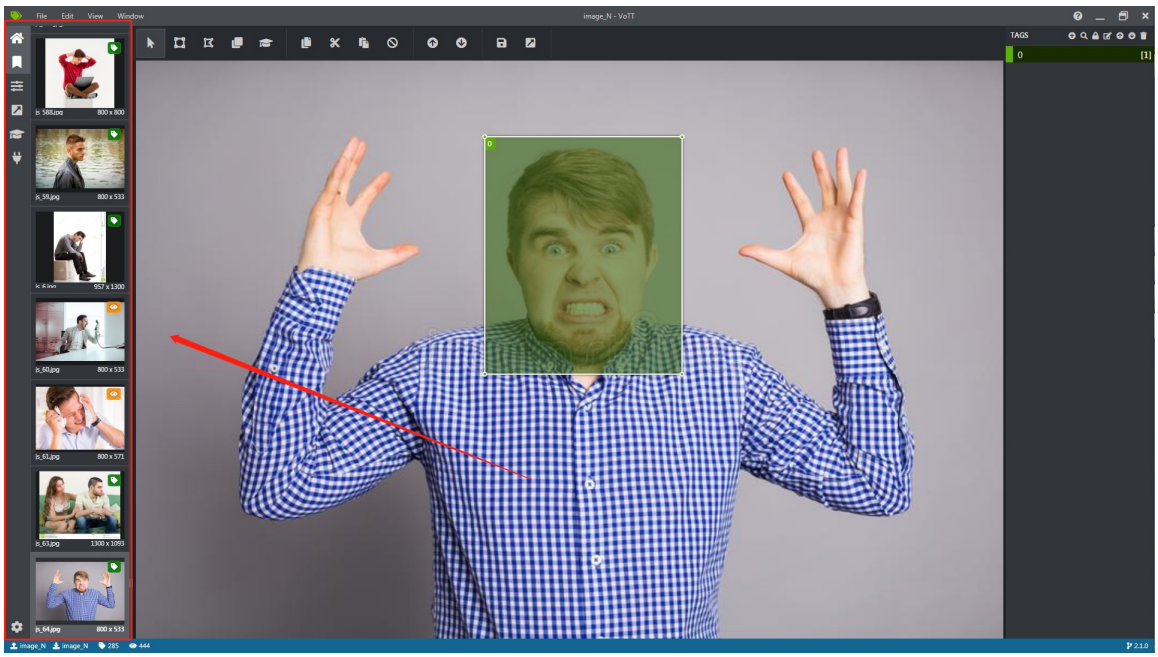
- 工作界面右上角区域为标签处理区域，可以进行标签的添加删除和位置更改操作



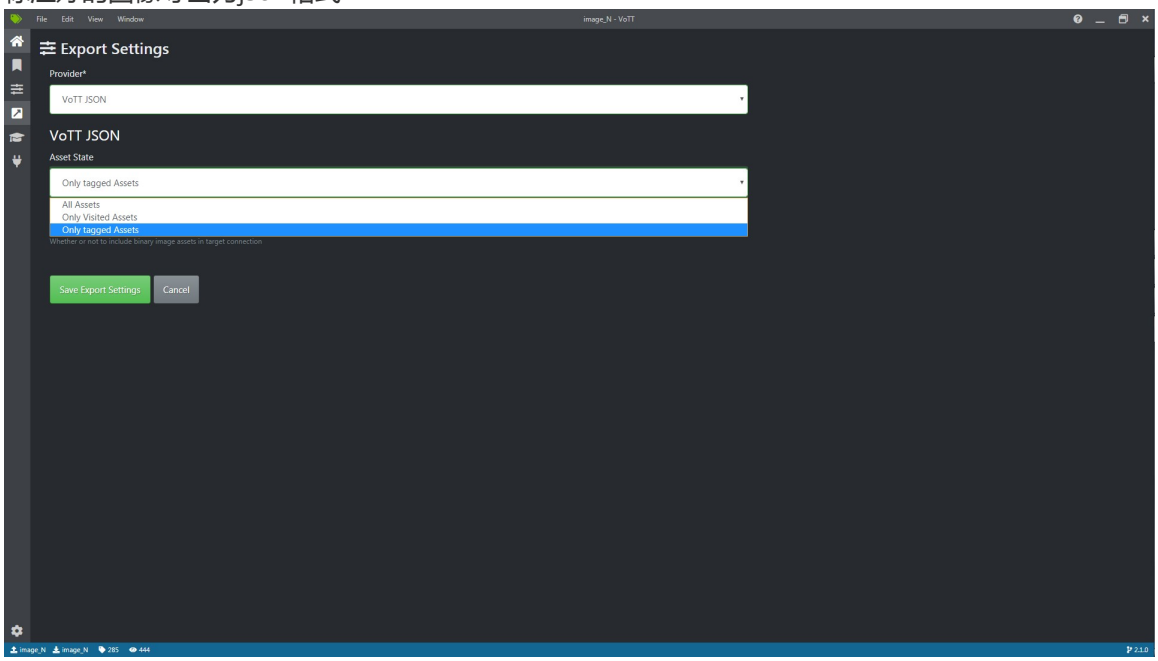
- 正上方区域为框选工作区域



- 左边为文件选项，可以选择框选图像已经导出



- 标注好的图像导出为json格式




```
#图像按照框选大小进行裁切保存
import pandas as pd
import os
import json
from PIL import Image

path=os.path.abspath(".")
path_image=path+"\\dat_20191202"
for i,j in zip(os.listdir(path_image),range(len(os.listdir(path_image)))):
    if i.endswith("json"):
        with open(path_image+"\\"+i,"r") as f:
```

```
            data2=json.load(f)
            path_i=data2["asset"]["name"]
            labe=data2["regions"][0]["tags"][0]
            box=(data2["regions"][0]["points"][0]["x"],data2["regions"][0]
["points"][0]["y"],data2["regions"][0]["points"][2]["x"],data2["regions"][0]
["points"][2]["y"])
            im=Image.open(path_image+"\\"+path_i).crop(box)
            try:
                im.save(path+"\\image\\0\\"+str(j)+".jpg")
            except:
                print("格式错误")
```

```
#讲图像转换为数据保存
import pandas as pd
import numpy as np
from PIL import Image
import os

path=os.path.abspath(".")
emotion_Yes=path+"\\image\\1"
emotion_No=path+"\\image\\0"

Y=[]
for i in os.listdir(emotion_Yes):
    im=Image.open(emotion_Yes+"\\"+i)
    im=im.resize((48,48))
    im=im.convert("L")
    Y.append(np.array(im))

N=[]
for i in os.listdir(emotion_No):
    im=Image.open(emotion_No+"\\"+i)
    im=im.resize((48,48))
    im=im.convert("L")
    N.append(np.array(im))

d=[]
for i in Y:
    d1=[]
    for j in i:
        for n in j:
            d1.append(n)
    d.append(str(d1).replace("[","").replace("]","").replace(",",""))

e=[]
for i in N:
    e1=[]
    for j in i:
        for n in j:
            e1.append(n)
    e.append(str(e1).replace("[","").replace("]","").replace(",",""))
```

```
dat1=pd.DataFrame({"emotion":[1]*len(d),"pixels":d})
jh1=["Training"]*int(dat1.shape[0]*0.7)+["PrivateTest"]*
(int(dat1.shape[0]*0.2))+["PublicTest"]*(int(dat1.shape[0]-
int(dat1.shape[0]*0.7)-int(dat1.shape[0]*0.2)))
#训练集、测试集、验证集=70%，20%，10%
dat1["Usage"]=jh1

dat2=pd.DataFrame({"emotion":[0]*len(e),"pixels":e})
jh2=["Training"]*int(dat2.shape[0]*0.7)+["PrivateTest"]*
(int(dat2.shape[0]*0.2))+["PublicTest"]*(int(dat2.shape[0]-
int(dat2.shape[0]*0.7)-int(dat2.shape[0]*0.2)))
dat2["Usage"]=jh2

data_x=pd.concat([dat1,dat2],ignore_index=True)
data_x.to_csv(path+"\\emotion_file.csv",index=False)

print(data_x.shape)
```

- 读取CSV文件，根据标签将图像划分为训练集、测试集、验证集三个文件夹，每个文件夹在根据图像的分类标签保存在所属标签的子文件夹下，在保存时要紧图像进行灰度处理，也就是只保留一个通道，这样的好处是可以减少计算量，加快处理速度。

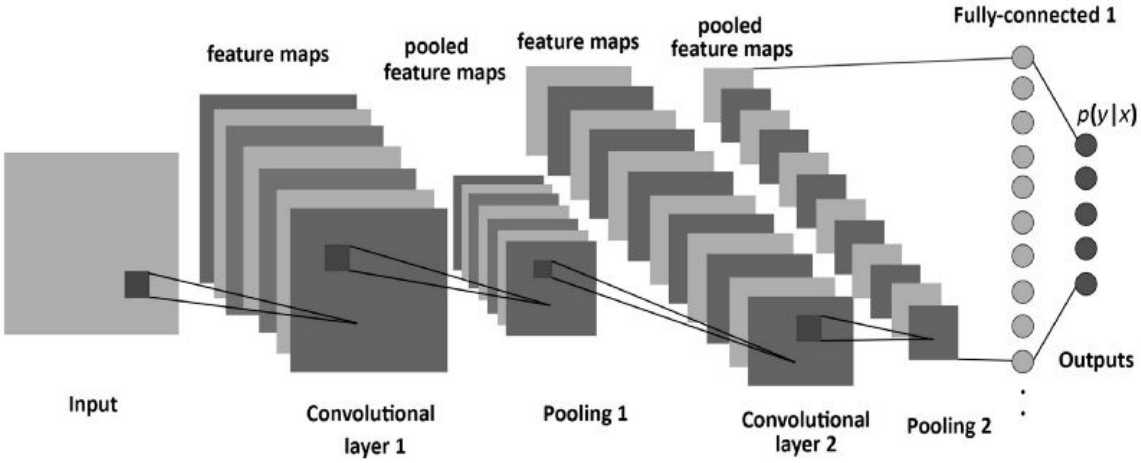
```
#将数据集根据根据标签分成训练集、测试集、验证集
import csv
import os

path = os.path.abspath(".")
database_path = os.path.abspath(".")
datasets_path = os.path.abspath(".")
csv_file = database_path+'emotion_file.csv'
train_csv = datasets_path+'train.csv'
val_csv = datasets_path+'val.csv'
test_csv = datasets_path+ 'test.csv'


with open(csv_file) as f:
    csvr = csv.reader(f)
    header = next(csvr)
    print(header)
    rows = [row for row in csvr]

    trn = [row[:-1] for row in rows if row[-1] == 'Training']
    csv.writer(open(train_csv, 'w+'),
lineterminator='\n').writerows([header[:-1]] + trn)
    print(len(trn))

    val = [row[:-1] for row in rows if row[-1] == 'PublicTest']
    csv.writer(open(val_csv, 'w+'),
lineterminator='\n').writerows([header[:-1]] + val)
    print(len(val))
```

```
    tst = [row[:-1] for row in rows if row[-1] == 'PrivateTest']
    csv.writer(open(test_csv, 'w+'),
lineterminator='\n').writerows([header[:-1]] + tst)
    print(len(tst))
```

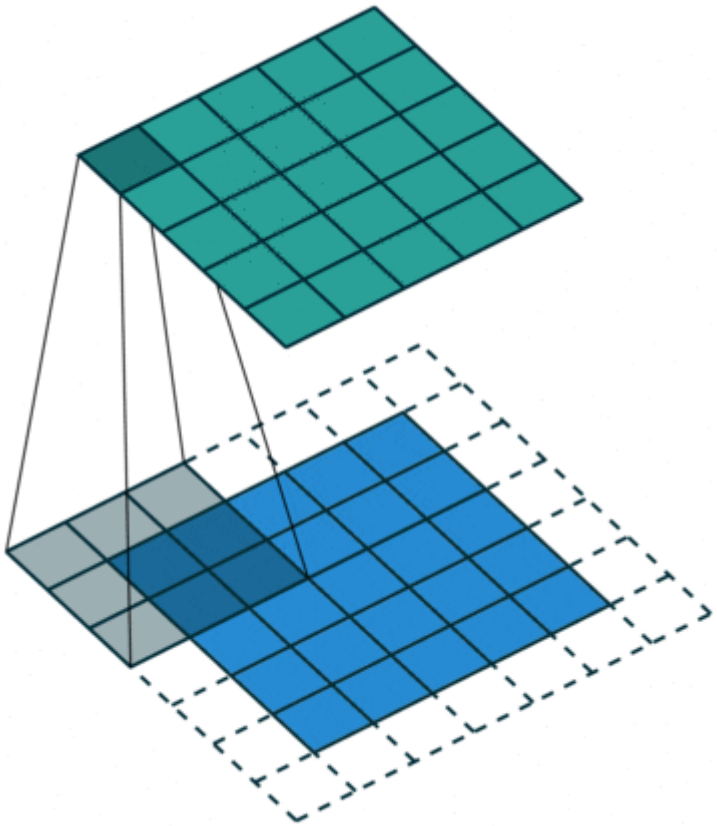
```
#图像转化为灰色通道并保存在子文件夹下
import csv
import os
from PIL import Image
import numpy as np

datasets_path = os.path.abspath(".")
train_csv = os.path.join(datasets_path, 'train.csv')
val_csv = os.path.join(datasets_path, 'val.csv')
test_csv = os.path.join(datasets_path, 'test.csv')

train_set = os.path.join(datasets_path, 'train')
val_set = os.path.join(datasets_path, 'val')
test_set = os.path.join(datasets_path, 'test')

for save_path, csv_file in [(train_set, train_csv), (val_set, val_csv),
(test_set, test_csv)]:
    if not os.path.exists(save_path):
        os.makedirs(save_path)

    num = 1
    with open(csv_file) as f:
        csvr = csv.reader(f)
        header = next(csvr)
        for i, (label, pixel) in enumerate(csvr):
            pixel = np.asarray([float(p) for p in pixel.split()]).reshape(48,
48)
            subfolder = os.path.join(save_path, label)
            if not os.path.exists(subfolder):
                os.makedirs(subfolder)
            im = Image.fromarray(pixel).convert('L')
            image_name = os.path.join(subfolder, '{:05d}.jpg'.format(i))
            im.save(image_name)
```

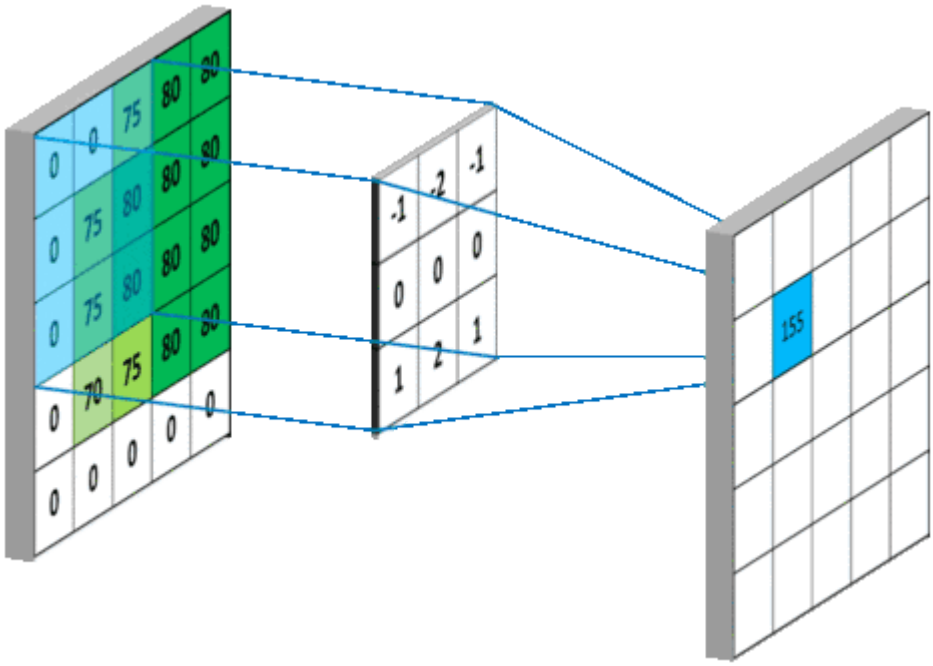
- **搭建CNN**
  - 卷积神经网络介绍，神经网络就是利用计算机来模拟人体大脑学历思考问题的方式和能力，不断向前传输计算值，向后反馈误差，不断学习，直到误差也就是损失最小时停止训练，将训练好的模型进行保存用于预测。当我们基于一个有限的固定特征集合解决分类问题的时候，这种方法是很奏效的——例如，我们根据足球运动员在比赛中记录的统计数据来预测他的位置。

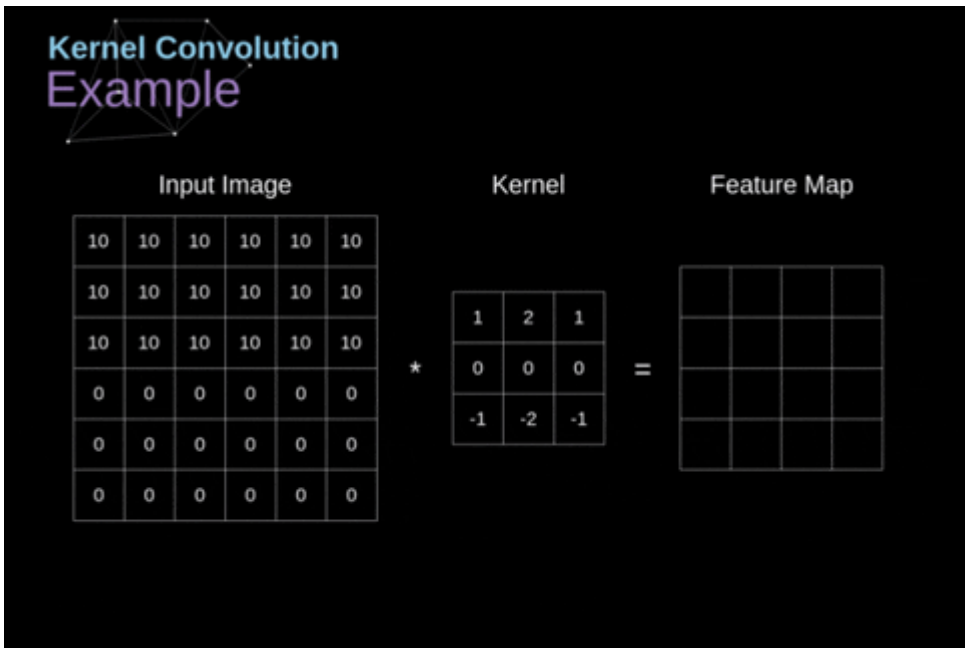
但是，当处理照片的时候，问题变得更加复杂。当然，我们可以把每个像素的亮度视作一个单独的特征，然后将它作为密集网络的输入传递进去。不幸的是，为了让它能够应付一张典型的智能手机照片，我们的网络必须包含数千万甚至上亿的神经元。另一方面，虽然我们可以将照片缩小，但是我们也会在这个过程中损失有价值的信息。所以我们马上就会发现，传统的策略是没有用的——我们需要一种新的聪明的方法，来尽可能多的利用数据，但同时还要减少必需的计算量和参数。这就是 CNN 发挥作用的时候了。



- 卷积神经网络和普通神经网络的区别主要在于“卷积”二字，卷积也就是我们说的内积计算方式，就是将矩阵相乘再相加，卷积的操作是为了提取区域的主要特征，也是一种减少维度的方法，并且在图像识别中这种方式也是很合理的，卷积这个操作最开始来自于计算机波纹处理，后来应用到计算机视觉领域得到了显著的作用。

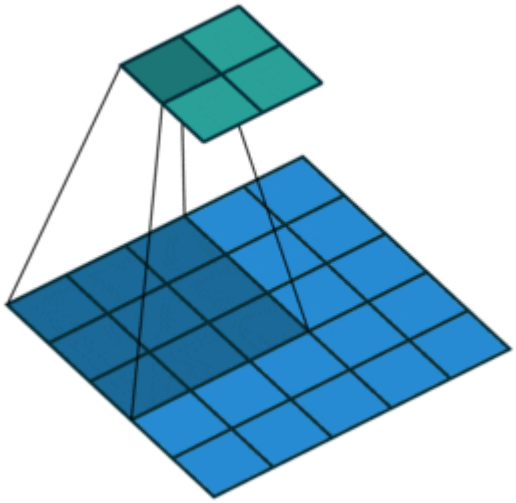


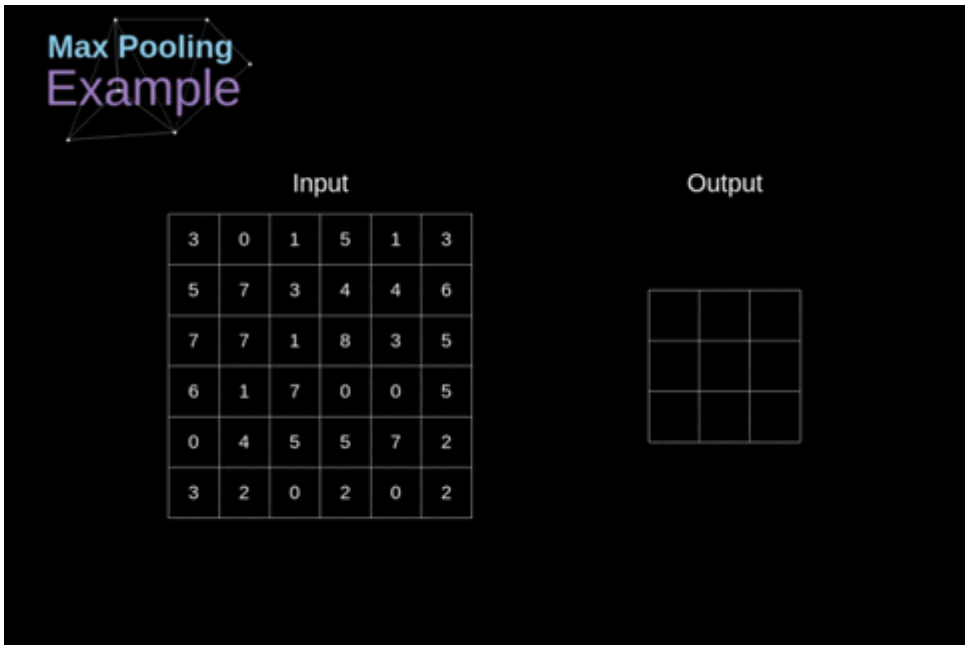




- 卷积操作可以作为物体的边缘检测，这在图像识别上提供了很大的帮助，在将我们的滤波器放在选中的像素上之后，我们将卷积核中的每一个数值和图像中对应的数值成对相乘。最后将乘积的结果相加，然后把结果放在输出特征图的正确位置上。我们在上边的动画中可以以一个微观的形式看到这个运算的过程，但是更有趣的是我们在整幅图像上执行这个运算得到的结果。

- “池化”操作是和“卷积”配合使用，池化操作也是一种提取重要特征的方式，但是在这一环节中没有权重产生，它的工作方式是指定池化的大小，对数据进行扫描保留区域内最大数值，这个操作也可以保留区域内的平均数值。除了卷积层，CNN 通常会用到所谓的池化层。它们最早被用来减小张量的大小以及加速运算。这些层是比较简单的——我们需要将我们的图像分成不同的区域，然后在每一个部分上执行一些运算。例如，对 Max Pool 层而言，我们会选择每个区域的最大值，并将它放到对应的输出区域。与卷积层的情况一样，我们有两个可用的超参数——滤波器大小和步长。最后但同样重要的一点是，如果你对一个多通道的图像执行池化操作，那么每一个通道的池化应该单独完成。





- “全连接”，此操作是将图像数组进行平压之后全连接，之后的操作和普通的神经网络原理一致，最后的输出层利用逻辑函数进行预测。
- 数据输入，在数据输入过程中添加Keras图像生成器模块，扩大数据集。

```
import keras
from keras.layers import Dense, Dropout, Activation, Flatten,Conv2D,
MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping
from keras.optimizers import SGD
batch_siz = 128
num_classes = 2
nb_epoch = 10
img_size=48
root_path=os.path.abspath(".")
```

```
import tensorflow as tf
import keras.backend.tensorflow_backend as KTF
KTF.set_session(tf.Session(config=tf.ConfigProto(device_count={'gpu':0})))

class Model:
    def __init__(self):
        self.model = None

    def build_model(self):
        self.model = Sequential()

        self.model.add(Conv2D(32, (3, 3), strides=1, padding='same',
input_shape=(img_size, img_size, 1)))
        self.model.add(Activation('relu'))
        self.model.add(Conv2D(32, (5, 5), padding='same'))
        self.model.add(Activation('relu'))
        self.model.add(MaxPooling2D(pool_size=(2, 2)))

        self.model.add(Conv2D(32, (3, 3), padding='same'))
        self.model.add(Activation('relu'))
        self.model.add(MaxPooling2D(pool_size=(2, 2)))

        self.model.add(Conv2D(64, (5, 5), padding='same'))
        self.model.add(Activation('relu'))
        self.model.add(MaxPooling2D(pool_size=(2, 2)))

        self.model.add(Flatten())
        self.model.add(Dense(2048))
        self.model.add(Activation('relu'))
        self.model.add(Dropout(0.5))
        self.model.add(Dense(1024))
        self.model.add(Activation('relu'))
        self.model.add(Dropout(0.5))
        self.model.add(Dense(num_classes))
        self.model.add(Activation('softmax'))
        self.model.summary()
    def train_model(self):
        sgd=SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
        self.model.compile(loss='categorical_crossentropy',
                optimizer=sgd,
                metrics=['accuracy'])
        #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.flow_from_directory(
                root_path+'/train',
                target_size=(img_size,img_size),
                color_mode='grayscale',
                batch_size=batch_siz,
                class_mode='categorical')#categorical
        val_generator = val_datagen.flow_from_directory(
                root_path+'/val',
                target_size=(img_size,img_size),
                color_mode='grayscale',
                batch_size=batch_siz,
                class_mode='categorical')
        eval_generator = eval_datagen.flow_from_directory(
                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_generator(
                train_generator,
                steps_per_epoch=800/(batch_siz/32),#28709
                nb_epoch=nb_epoch,
                validation_data=val_generator,
                validation_steps=500
                )
        history_predict=self.model.predict_generator(
                eval_generator,
                steps=500)
        with open(root_path+"\\model"+'/model_fit_log','w') as f:
            f.write(str(history_fit.history))
        with open(root_path+"\\model"+'/model_predict_log','w') as f:
            f.write(str(history_predict))
        print('model trained')
    def save_model(self):
        model_json=self.model.to_json()
        with open(root_path+"\\model"+"/model_json.json", "w") as json_file:
            json_file.write(model_json)
        self.model.save_weights(root_path+"\\model"+'/model_weight.h5')
        self.model.save(root_path+"\\model"+'/model.h5')
        print('model saved')

if __name__=='__main__':
    model=Model()
    model.build_model()
    print('model built')
    model.train_model()
    print('model trained')
    model.save_model()
    print('model saved')
```

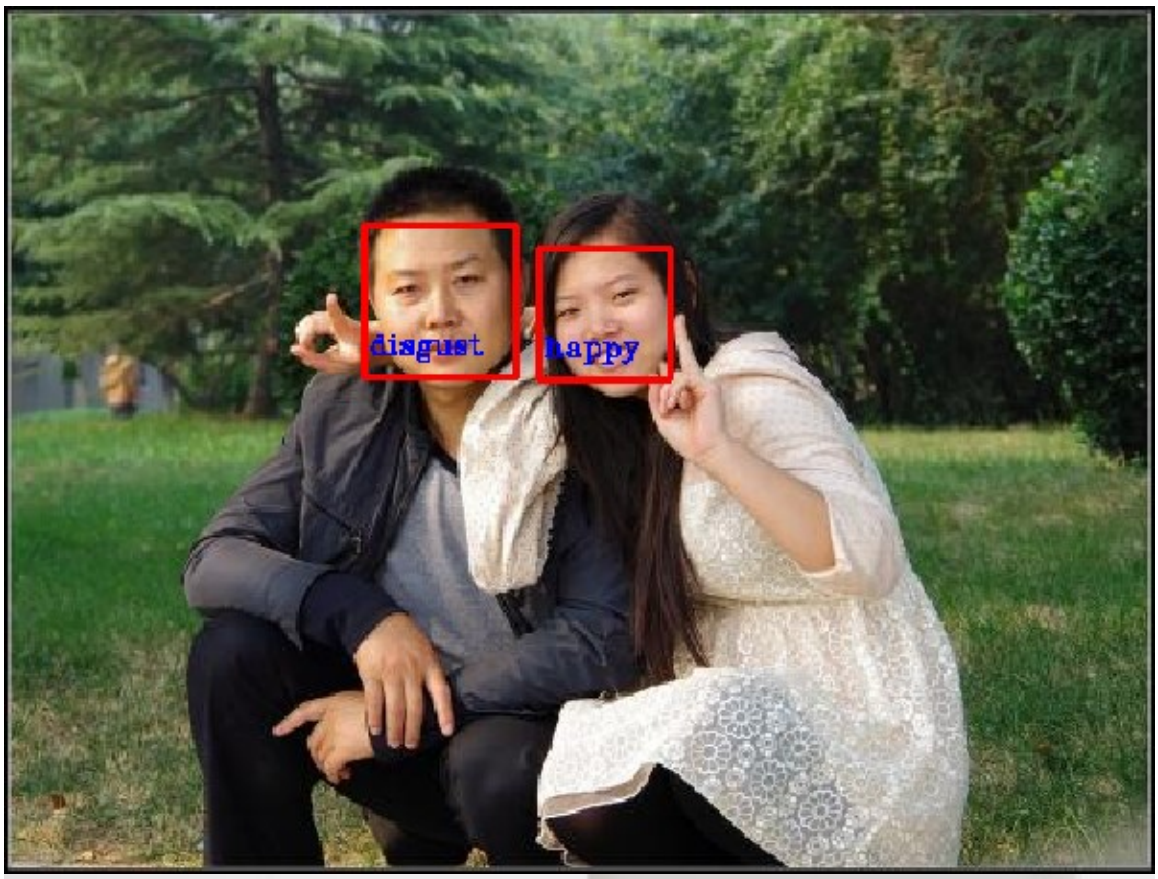
- **图像预测结果**

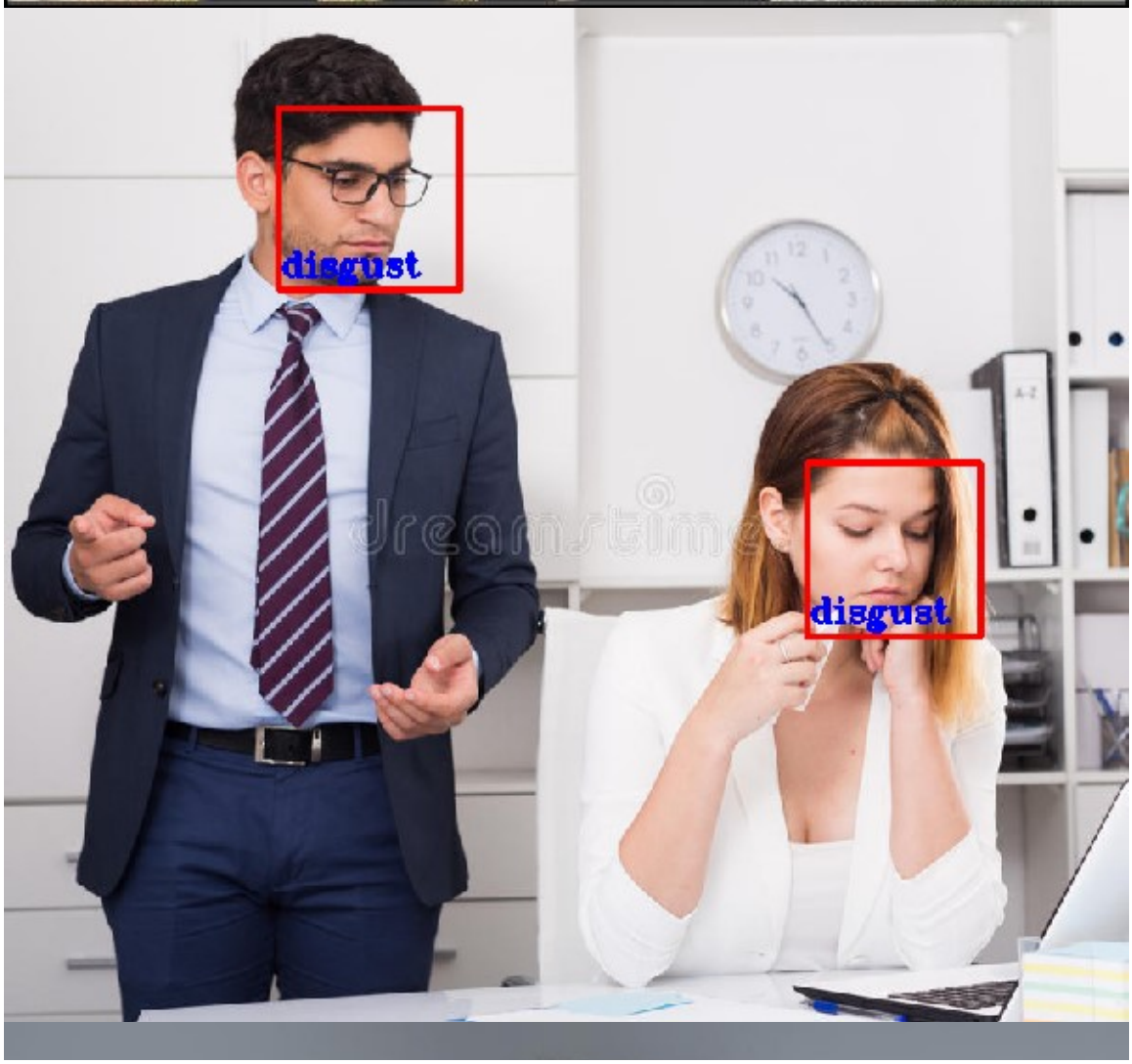
- 加载模型预测图像表情

```
import numpy as np
import cv2
import sys
import json
import time
import os
from keras.models import model_from_json
root_path=os.path.abspath(".")
model_path = root_path + '/model_50/'
img_size = 48
emotion_labels = ["happy",'disgust']
num_class = len(emotion_labels)
json_file = open(model_path + 'model_json.json')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights(model_path + 'model_weight.h5')

def predict_emotion(face_img):
    img_size=48
    face_img = face_img * (1. / 255)
    resized_img = cv2.resize(face_img, (img_size, img_size))
    results = []
    rsh_img=resized_img.reshape(1,img_size, img_size, 1)
    list_of_list = model.predict_proba(rsh_img, batch_size=32, verbose=1)
    result = [prob for lst in list_of_list for prob in lst]
    results.append(result)
    return results

def face_detect(image_path):
    cascPath = root_path+"/data/haarcascade_frontalface_alt2.xml"
    faceCasccade = CascadeClassifier(cascPath)
    img = imread(image_path)
    img_gray = cvtColor(img, COLOR_BGR2GRAY)
    faces = faceCasccade.detectMultiScale(
        img_gray,
        scaleFactor=1.1,
        minNeighbors=1,
        minSize=(10, 10)
    )
    return faces, img_gray, img

if __name__ == '__main__':
    images = []
    A=input("请输入待预测图片所在文件夹名称：")
    dir = root_path +"\\"+ A
    if isdir(dir):
        files = listdir(dir)
        for file in files:
            if file.endswith('jpg') or file.endswith('png') or
file.endswith('PNG') or file.endswith('JPG'):
```

```
                images.append(dir + '\\' + file)
    else:
        images.append(dir)

    for image in images:
        faces1, img_gray, img = face_detect(image)
        if faces1==():
            faces1=array([[0,0,img_gray.shape[1],img_gray.shape[0]]])
        faces=[]
        for i in faces1:
            x,y,w,h=i
            if w >= max(faces1[:,2])*0.8:
                j=(x,y,w,h)
                faces.append(j)
        spb = img.shape
        sp = img_gray.shape
        height = sp[0]
        width = sp[1]
        size = 600
        emo = ""
        face_exists = 0
        for (x, y, w, h) in faces:
            face_exists = 1
            face_img_gray = img_gray[y:y + h, x:x + w]
            results = predict_emotion(face_img_gray)  # face_img_gray
            result_sum = array([0]*num_class)
            for result in results:
                result_sum = result_sum + array(result)
            disgust,happy= result_sum
            print('disgust:', disgust,' happy:', happy)
            label = argmax(result_sum)
            emo = emotion_labels[label]
            print('Emotion : ', emo)
            t_size = 2
            ww = int(spb[0] * t_size / 300)
            www = int((w + 10) * t_size / 100)
            www_s = int((w + 20) * t_size / 100) * 2 / 5
            rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), ww)
            putText(img, emo, (x+2, y+h-10),FONT_HERSHEY_TRIPLEX,www_s, (255,
0, 0), thickness=www, lineType=1)
        if face_exists:
            HoughLinesP
            namedWindow(emo, 0)
            cent = int((height * 1.0 / width) * size)
            resizeWindow(emo, (size, cent))
            imshow(emo, img)
            k = waitKey(0)
            destroyAllWindows()
            if k & 0xFF == ord('q'):
                break
```

- 效果展示


disgust
happy


disgust
disgust


disgust


happy

- **模型评估**
  - 训练集

	0	1
0	4749	216
1	283	6933

评估	百分比
真阳率	96.98%
假阳率	3.02%
真阴率	5.62%
假阴率	94.38%
召回率	96.08%

  - 测试集

	0	1
0	562	64
1	76	803

评估	百分比
真阳率	92.62%
假阳率	7.38%
真阴率	11.91%
假阴率	88.09%
召回率	91.35%

- **实时预测**

  对面部表情进行实时识别需要计算机获取摄像头的信息，思路是通过获取计算机摄像头的权限，打开摄像头进行实时画面的捕捉，获取当前画面的一帧来做预测，根据刷新率不同识别的实时速率也不同

```
#-*- coding: utf-8 -*-

import cv2
import sys
import gc
import json
import numpy as np
from keras.models import Sequential
from keras.models import model_from_json
root_path=os.path.abspath(".")
model_path=root_path+'/model/'
img_size=48
emo_labels = ["happy", 'disgust']
num_class = len(emo_labels)
json_file=open(model_path+'model_json.json')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights(model_path+'model_weight.h5')

if __name__ == '__main__':
    if len(sys.argv) == 1:
        print(len(sys.argv))
        print("Usage:%s camera_id\r\n" % (sys.argv[0]))
        sys.exit(0)

    #框住人脸的矩形边框颜色
    color = (0, 0, 2555)

    #捕获指定摄像头的实时视频流
    cap = cv2.VideoCapture(0)

    #人脸识别分类器本地存储路径
    cascade_path ="C:/ProgramData/Anaconda3/lib/site-
packages/cv2/data/haarcascade_frontalface_alt.xml"
```

```
    #循环检测识别人脸
    while True:
        _, frame = cap.read()    #读取一帧视频

        #图像灰化，降低计算复杂度
        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        #使用人脸识别分类器，读入分类器
        cascade = cv2.CascadeClassifier(cascade_path)

        #利用分类器识别出哪个区域为人脸
        faceRects = cascade.detectMultiScale(frame_gray, scaleFactor = 1.1,
                                     minNeighbors = 1, minSize = (120, 120))
        if len(faceRects) > 0:
            for faceRect in faceRects:
                x, y, w, h = faceRect
                images=[]
                rs_sum=np.array([0.0]*num_class)
                image = frame_gray[y: y + h, x: x + w ]
                image=cv2.resize(image,(img_size,img_size))
                image=image*(1./255)
                images.append(image)
                images.append(cv2.flip(image,1))
                images.append(cv2.resize(image[2:45,:],(img_size,img_size)))
                for img in images:
                    image=img.reshape(1,img_size,img_size,1)
                    list_of_list = model.predict_proba(image,batch_size=32,verbose=1)
                    result = [prob for lst in list_of_list for prob in lst]
                    rs_sum+=np.array(result)
                print(rs_sum)
                label=np.argmax(rs_sum)
                emo = emo_labels[label]
                print ('Emotion : ',emo)
                cv2.rectangle(frame, (x - 10, y - 10), (x + w + 10, y + h + 10),
color, thickness = 2)
                font = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(frame,'%s' % emo,(x + 30, y + 30), font, 1, (255,0,255),4)
        cv2.imshow("识别朕的表情！", frame)

        #等待10毫秒看是否有按键输入
        k = cv2.waitKey(30)
        #如果输入q则退出循环
        if k & 0xFF == ord('q'):
            break

    #释放摄像头并销毁所有窗口
    cap.release()
    cv2.destroyAllWindows()
```

总结

- 面部表情识别的准确率和数据集的质量密切相关，在训练时最好选用高清无水印的人像表情进行训练，否则OpenCV在进行面部识别的时候会出现错误，以至于影响到预测结果
- 对于面部表情的二分类卷积神经网络能较为出色的完成，在于客户交流沟通的时候可以通过客户表情的反馈来不断更正我们的营销策略
- 面部表情还可以用于多方面，可作为一个技术支持来辅佐其它行业，需要继续探索和完善