

# Python程序设计报告

设计题目：口罩检测软件

学生姓名：付炎平

专 业：物联网工程

班 级：物联网19-2班

学 号：2019217819

指导教师：马学森

完成日期：2021/1/1

# （一）需求和规格说明

**问题描述：**疫情到来之际，这是一块用来检测图片中人戴口罩的客户端软件，建立服务端，电脑可以向服务端发送图片，服务端对图片进行目标检测，然后将目标检测的结果返回到一个网址，软件再在该网址进行一个抓取，抓取之后在软件界面进行显示，用户就可以得到图片的检测结果。

**编程任务：**

1. 建立服务端用来监听用户发送图片
2. 在软件部分利用爬虫抓取服务端返回的图片
3. 在服务端后端利用机器学习训练出一个检测图片中人戴口罩的模型。
4. 当用户向服务端发送图片请求时，前端调用模型进行推理，并且把检测之后的图片返回到另一个网址中。
5. 建立图形界面和用户进行交互。

# （二）设计

## 1．设计思想

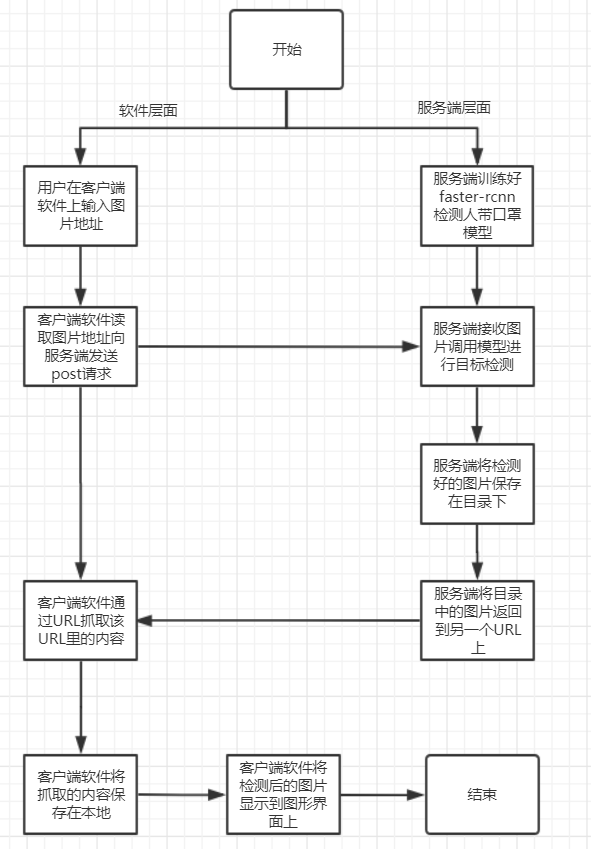
本题主要利用用户和软件交互，让用户通过输入框输入图片地址向服务器发送一张图片，服务器立马能够进行接收，并且能够把该图片保存在服务器所在的目录里，然后利用机器学习先利用训练集训练出一个检测人带口罩的模型，可以训练一个faster-rcnn的模型。

模型训练好之后，保存在服务器所在目录下，当服务器接收到来自用户发送的图片时，调用训练好的faster-rcnn模型进行目标检测，并且把目标检测的结果保存在服务器所在目录里，然后服务器把目录里完成检测的图片返回到一个URL上。

用户软件层面再利用爬虫技术，抓取服务器返回检测图片的URL里的内容，然后保存在本地，最后软件再读取爬取后本地的图片，返回到图形界面上。

## 2. 设计表示

流程图



界面图：



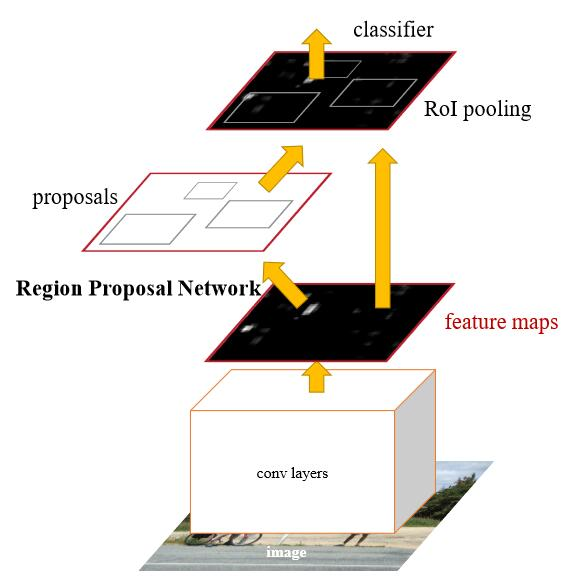
## 核心算法

本程序主要是客户端向服务端发送请求，上传图片，然后服务端进行目标检测，将检测后的图片返回到URL上，客户端进行抓取URL的过程。

本程序的核心算法是在机器学习模型Faster-rcnn的神经网络上：

对于一张输入的彩色图片，首先经过CNN层进行特征提取。在Faster R-CNN的原始论文中，他们使用的是预训练好的VGG16网络用于特征提取。VGG网络去掉全连接层，只保留卷积的部分。这个对应的就是图中的conv layer。

卷积层提取完特征之后，对于特征图的数据，分成两部分，分别进入RPN网络（候选区域选择网络），和ROI pooling网络。对于RPN而言，这个网络所做的工作是挑选出图片中可能的候选区域，区分前景和背景。这些信息用来辅助最终的目标检测的决策。ROI Pooling所要做的是收集输入的feature maps和proposals，综合这些信息后提取proposal feature maps，送入后续全连接层判定目标类别。



# 用户手册

用户首先在客户端软件输入框输入要检测的图片地址，然后按下旁边的按钮“开始检测”，然后很快就会在客户端软件上的图形界面上显示检测后的图片。

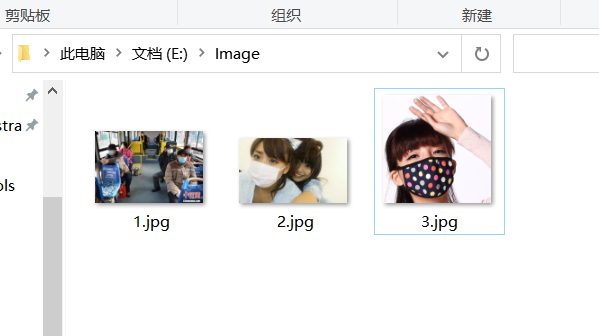
用户注意：

1.输入框里的图片地址是用户要检测的图片在用户电脑里的绝对路径并且加上图片的名称和后缀，例如：E:/Image/1.jpg

2.检测后的图片是保存在用户客户端软件的同一目录下，并且图片名称和上传的图片名称一致。

# （四）调试及测试

## 1. 测试数据：









## 2. 测试结果：





## 3. 进一步改进：

这次程序在界面方面处理方面还有待加强，界面还不够美观，与用过户交互的功能也不够多，并且服务端模型还有待优化，同时在体系这个功能体系还有待晚上，比如能够把戴口罩的人的数量和没有戴口罩的人的数量都给显示出来。

其次，图形界面与用户进行交互也不够完善，目前只能通过输入地址来上传图片，应该向着通过按钮来选择图片上传这样更加符合与用户的交互。

# 感想

我这道题比较综合，综合了很多知识，综合了Flask服务端，tkiner图形界面，爬虫，机器学习里面的知识，我之前主要使用Python来写项目的，发现Python很好用，主要是因为Python写起来比较简便，有很多功能强大的第三方库，而且Python在人工智能方向有很大的应用。

我在这次题目里也学会了很多，学会来如何综合使用多个方面的知识并且怕他们联系起来，我认为这个很锻炼人的能力。

# 附录：

源代码：

**服务端**：

# -\*- coding: utf-8 -\*-

import os,time

from gevent import pywsgi

import torchvision

import cv2

import torchvision

import torch

from PIL import Image

import numpy as np

import torchvision.transforms as T

from torchvision.ops import roi\_pool

from flask import Flask, render\_template, send\_from\_directory, request, jsonify

# device = torch.device('cuda') if torch.cuda.is\_available() else torch.device('cpu')

device=torch.device('cpu')

model=torch.load('model.pth', map\_location=lambda storage, loc: storage)

# model = torch.load('model.pth')

model.to(device)

model.eval()

COCO\_INSTANCE\_CATEGORY\_NAMES = [

"background","face","face\_mask"

]

app = Flask(\_\_name\_\_)

UPLOAD\_FOLDER = 'upload'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER # 设置文件上传的目标文件夹

basedir = os.path.abspath(os.path.dirname(\_\_file\_\_)) # 获取当前项目的绝对路径

ALLOWED\_EXTENSIONS = set(['txt', 'png', 'jpg', 'xls', 'JPG', 'PNG', 'xlsx', 'gif', 'GIF']) # 允许上传的文件后缀

# 判断文件是否合法

def allowed\_file(filename):

return '.' in filename and filename.rsplit('.', 1)[1] in ALLOWED\_EXTENSIONS

def change\_image(filepath,filename):

img\_path = filepath

img = Image.open(img\_path)

transform = T.Compose([T.ToTensor()]) # Defing PyTorch Transform

img = transform(img) # Apply the transform to the image

img = img.numpy().tolist()

image = torch.tensor([img]).to(device)

pred = model(image) # Returns predictions

head = ['background', 'face', 'face\_mask']

pred\_class = [head[i] for i in list(pred[0]['labels'].cpu().numpy())]

pred\_boxes = [[(i[0], i[1]), (i[2], i[3])] for i in list(pred[0]['boxes'].detach().cpu().numpy())]

pred\_score = list(pred[0]['scores'].detach().cpu().numpy())

# 过滤分数较低的预测

threshold = 0.3

pred\_t = [pred\_score.index(x) for x in pred\_score if x > threshold][-1]

pred\_boxes = pred\_boxes[:pred\_t + 1]

pred\_class = pred\_class[:pred\_t + 1]

# x, y = img.shape[0:2]

# img= cv2.resize(img, (int(y / 2), int(x / 2)))

img = cv2.imread(img\_path)

img = cv2.cvtColor(img, cv2.IMREAD\_COLOR)

number=pred\_t+1

for index, boxes in enumerate(pred\_boxes):

text\_bottom\_left = (int(boxes[0][0] - 10), int(boxes[0][1] - 10))

cv2.rectangle(img, boxes[0], boxes[1], color=(0, 255, 0), thickness=3)

cv2.putText(img, pred\_class[index], text\_bottom\_left, cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), thickness=2)

newImagepath="download/"+filename

cv2.imwrite(newImagepath,img)

return number

# 具有上传功能的页面

@app.route('/',methods=['GET'])

def upload\_test():

return render\_template("photo.html")

@app.route('/api/upload', methods=['POST'], strict\_slashes=False)

def api\_upload():

file\_dir = os.path.join(basedir, app.config['UPLOAD\_FOLDER']) # 拼接成合法文件夹地址

if not os.path.exists(file\_dir):

os.makedirs(file\_dir) # 文件夹不存在就创建

f=request.files['myfile'] # 从表单的file字段获取文件，myfile为该表单的name值

# if f and allowed\_file(f.filename): # 判断是否是允许上传的文件类型

fname=f.filename

path1=os.path.join(file\_dir, fname)

f.save(path1) #保存文件到upload目录

number=change\_image(path1,fname)

number=str(number)

return number

# else:

# return jsonify({"errno": 1001, "errmsg": "上传失败"})

@app.route("/<path:filename>")

def downloader(filename):

dirpath=os.path.join(app.root\_path,'download')

return send\_from\_directory(dirpath,filename)

if \_\_name\_\_ == '\_\_main\_\_':

server = pywsgi.WSGIServer(('0.0.0.0', 8888), app)

server.serve\_forever()

**客户端**：

import requests

from urllib.request import urlretrieve

import re

from tkinter import \*

from PIL import ImageTk

from tkinter import ttk

import cv2

import base64

def getByte(path):

with open(path, 'rb') as f:

img\_byte = base64.b64encode(f.read()) # 二进制读取后变base64编码

img\_str = img\_byte.decode('ascii') # 转成python的unicode

return img\_str

def get\_img():

filepath = e1.get()

with open(filepath, 'rb') as f:

filename = f.name.split('/')[-1]

response = requests.get(url=url+filename)

data = response.content

path = 'change.jpg'

with open(path, mode='wb') as f:

f.write(data)

img=cv2.imread(path)

img=cv2.resize(img,(500,400))

cv2.imwrite(path,img)

bm=ImageTk.PhotoImage(file=path)

l2=Label(root,image=bm)

l2.bm=bm

l2.grid(row=2,column=0,columnspan=10)

def post\_img():

filepath = e1.get()

f = open(filepath, 'rb')

filename = f.name

f.close()

file = {'myfile': (filename, open(filepath, 'rb'), 'image/jpeg')}

response=requests.post(url+"api/upload/",files=file)

print(response.status\_code)

print(response.text)

get\_img()

url='http://127.0.0.1:8888/'

root=Tk()

root.geometry("550x500")

root.title('图片检测')

l1=Label(root,text='图片地址',font=('微软雅黑',20),fg='blue')

l1.grid(row=0,column=0)

e1=Entry(root,width=25,font=('微软雅黑',20))

e1.grid(row=0,column=1)

b1=Button(root,text='开始检测',font=('微软雅黑',20),command=post\_img)

b1.grid(row=1,column=0)

root.mainloop()