在疫情期间的疫情数据的查询及口罩佩戴辨别

--Python编程与实践--

孙浩峰 2019211119 2019210588

上交时间：

摘要：在中国新冠疫情爆发后，所有人都十分关注国内疫情的相关数据及变化，中国各地区也都使用各种方法来阻断新冠疫情的传播，互联网在中国的普及率极高，各基层也都使用了互联网来辅助各级部门管理群众控制疫情。在当前的网络媒介发达，人工智能深度学习兴起的时代，人们都在研究如何更快、更好的使用计算机来代替人类完成一些工作。于是我基于Python编写了对中国疫情数据的获取，查询及相应的可视化处理，以方便我们对相关数据的获取与研究。同时由于新冠病毒的特性，我们在外出时需要时刻佩戴口罩，在商场等人员密度大的地区，仅仅依靠人工检测效率低，人员检测容易失误，所以我们需要使用计算机来代替人类完成这项工作。因此我又基于Tensorflow，使用了AIZOO开源的经过训练过的模型设计了可以使用照片或实时使用摄像头对单个或多个人像进行口罩检测的程序。通过爬虫爬取新冠疫情数据，Tensorflow检测目标是否佩戴口罩这两方面的应用可以帮助基层部门管理群众，减轻压力，提高办事效率。

正文：

1. 项目框架与设计
   1. 目的

基于Python编写了对中国疫情数据的获取，查询及相应的可视化处理，以方便我们对相关数据的获取与研究。基于Tensorflow，使用了AIZOO开源的经过训练过的模型设计了可以使用照片或实时使用摄像头对单个或多个人像进行口罩检测的程序。通过爬虫爬取新冠疫情数据，Tensorflow检测目标是否佩戴口罩这两方面的应用可以帮助基层部门管理群众，减轻压力，提高办事效率。

* 1. 实现描述

a)使用requests库获取目标网站的数据，这里选用” [https://view.inews.qq.com/g2/getOnsInfo?name=disease\_h5&callback=&\_=%d](https://view.inews.qq.com/g2/getOnsInfo?name=disease_h5&callback=&_=%25d)”作为数据爬取的主要网站。然后使用json.loads()对获取到的数据进行加载处理。然后对获取的数据进行处理就可以得到我们所需要的省份、城市的信息。然后对数据进行查询或者使用matlabplot、pyecharts和seaborn对获取到的数据进行处理得到一些直方图、柱状图，或者使用chromdriver打开本地HTML得到一个全国疫情统计图。

b)使用AIZOO开源的可以对目标进行是否佩戴口罩的“face\_mask\_detection.pb”模型文件。在本项目中使用了 SSD 类型的架构，为了让模型可以实时的跑在浏览器以及终端设备上，将模型设计的非常小，只有 101.5 万个参数。本模型输入大小为 260x260，主干网络只有 8 个卷积层，加上定位和分类层，一共只有 24 层（每层的通道数目基本都是 32\64\128），所以模型特别小，只有 101.5 万参数。模型对于普通人脸基本都能检测出来，但是对于小人脸，检测效果肯定不如大模型。于是我们可以使用Tensorflow对输入的照片、视频对每一帧进行分割成许多个小部分，把每个小部分里具有的特征提取出来（也就是识别每个小部分），再将这些小部分具有的特征汇总到一起，然后完成机器识别图像的过程。

1. 项目实现

2.1 技术应用与环境

a) 随着网络的迅速发展，万维网成为大量信息的载 体，如何有效的提取并利用这些信息成为一个巨大挑战。我们利用网络爬虫可以有效的对web上的信息进行提取。网络爬虫按照系统结构和实现技术，大致可以分为以下几种类型：通用网络爬虫、聚焦网络爬虫、增量式网络爬虫、深层网络爬虫。在这里我们主要是为了获取与新冠疫情相关的数据，所以我们将会选择聚焦网络爬虫。聚焦网络爬虫，又称主题网络爬虫，是指选择性地爬行那些与预先定义好的主题相关页面的网络爬虫，聚焦爬虫只需要爬行与主题相关的页面，极大的节省了硬件和网络资源，保存的页面也由于数量少而更新快，还可以很好的满足一些特定人群对特定领域信息的需求。于是我们可以通过网络爬虫对新冠疫情的相关数据进行定向爬取，然后对获取的信息进行提取与整合，这样可以帮助人们更快更好的得到自己想要的数据，避免了时间、空间等资源的浪费。

b) 目标检测作为图像处理和计算机视觉领域中的经典课题,在交通监控、图像检索、人机交互等方面有着广泛的应用。它旨在一个静态图像(或动态视频)中检测出人们感兴趣的目标对象。传统的目标检测算法中特征提取和分类决策分开进行，对特征选取的要求就更加严格，在面对复杂场景的时候很难得到理想效果。自Hinton教授提出深度学习理论，越来越多的研究人员发现在目标检测领域应用深度学习，可以有效提高检测效果和性能，于是深度学习在实时视频的目标检测开始获得大规模的应用。时至今日，其检测效率和精度已经有了极大提高。在本次的设计中我使用Opencv与Tensorflow获取的人像进行检测并且与pb模型进行对比以判断目标人群是否佩戴口罩。

c)在这次设计使用python 3.7，包含numpy，time，PIL，tensorflow2.X，json，requests，jsonpath，pyecharts，seaborn，pandas，matplotlib等包构成基本环境，同时要求计算机上有摄像头以实现实时目标检测的功能。

2.2 实现方式

a)使用requests库获取页面，使用json库对获取的页面进行解析获取数据。

b)对人像进行识别时使用卷积神经网络。RCNN（Region with CNN feature）是卷积神经网络应用于目标检测问题的一个里程碑的飞跃。CNN具有良好的特征提取和分类性能，采用RegionProposal方法实现目标检测问题。算法可以分为三步

（1）候选区域选择。

（2）CNN特征提取。

（3）分类与边界回归。

接下来我将详细的介绍一下每一过程如何实现：

（1）候选区域选择：区域建议Region Proposal是一种传统的区域提取方法，基于启发式的区域提取方法，用的方法是ss，查看现有的小区域，合并两个最有可能的区域，重复此步骤，直到图像合并为一个区域，最后输出候选区域。然后将根据建议提取的目标图像标准化，作为CNN的标准输入可以看作窗口通过滑动获得潜在的目标图像，在RCNN中一般Candidate选项为1k~2k个即可，即可理解为将图片划分成1k~2k个网格，之后再对网格进行特征提取或卷积操作，这根据RCNN类算法下的分支来决定。然后基于就建议提取的目标图像将其标准化为CNN的标准输入。

（2）CNN特征提取：标准卷积神经网络根据输入执行诸如卷积或池化的操作以获得固定维度输出。也就是说，在特征提取之后，特征映射被卷积和汇集以获得输出。

（3）分类与边界回归：实际上有两个子步骤，一个是对前一步的输出向量进行分类（分类器需要根据特征进行训练）; 第二种是通过边界回归框回归（缩写为bbox）获得精确的区域信息。其目的是准确定位和合并完成分类的预期目标，并避免多重检测。在分类器的选择中有支持向量机SVM，Softmax等等；边界回归有bbox回归，多任务损失函数边框回归等 。

2.3核心代码

a) url = 'https://view.inews.qq.com/g2/getOnsInfo?name=disease\_h5&callback=&\_=%d'  
data = json.loads(requests.get(url=url).json()['data'])  
num = data['areaTree'][0]['children']

获取页面，进行解析

T o t a l \_ d a t a = { }  
f o r I t e m I n n u m :  
 I f I t e m [ ' n a m e ' ] n o t I n t o t a l \_ d a t a :  
 t o t a l \_ d a t a . u p d a t e ( { I t e m [ ' n a m e ' ] : 0})  
 f o r c I t y \_ d a t a I n I t e m [ ' c h I l d r e n ' ]:  
 t o t a l \_ d a t a [ I t e m [ ' n a m e ' ] ] + = I n t ( c I t y \_ d a t a [' t o t a l ']['c o n f I r m ' ])  
n a m e s = t o t a l \_ d a t a . k e y s ( )   
n u m s = t o t a l \_ d a t a . v a l u e s ( )

提取数据

I m p o r t m a t p l o t l I b . p y p l o t a s p l t   
  
p l t . f I g u r e ( f I g s I z e = [ 1 0 , 6] )  
p l t . r c P a r a m s [ ' f o n t . s a n s – s e r I f ' ] = [ ' S I m H e I '] # 用来正常显示中文标签  
p l t . r c P a r a m s [ ' a x e s . u n I c o d e \_ m I n u s ' ] = False # 用来正常显示负号  
  
# -----------------------------1.绘制确诊数据-----------------------------------  
p1 = p l t . s u b p l o t(221)  
  
# 获取数据  
n a m e s = t o t a l \_ d a t a . k e y s ( )   
n u m s = t o t a l \_ d a t a . v a l u e s ( )  
p l t . b a r ( n a m e s , n u m s , w I d t h = 0 . 3 , c o l o r = ' g r e e n ')  
  
# 设置标题  
p l t . y l a b e l ( " 确 诊 人 数 " , r o t a t I o n = 9 0 )  
p l t . x t I c k s ( l I s t ( n a m e s ) , r o t a t I o n = - 6 0 , s I z e = 8 )   
# 显示数字  
f o r a , b I n z I p ( l I s t ( n a m e s ) , l I s t ( n u m s ) ) :  
 p l t . t e x t ( a , b, b, h a= ' c e n t e r ', v a=' b o t t o m ', s I z e =6)  
plt.sca(p1)

绘制疫情数据图像

n = time.strftime("%Y-%m-%d") + "-{}-4db.csv".format(a)  
fw = open(n, 'w', encoding='utf-8')  
fw.write('province,type,data\n')  
i = 0  
while i < len(names):  
 fw.write(names[i] + ',confirm,' + str(num1[i]) + '\n')  
 fw.write(names[i] + ',dead,' + str(num3[i]) + '\n')  
 fw.write(names[i] + ',heal,' + str(num4[i]) + '\n')  
 fw.write(names[i] + ',new\_confirm,' + str(num5[i]) + '\n')  
 i = i + 1  
else:  
 fw.close()  
  
# ------------------------------------------------------------------------------  
# 调用Seaborn绘制柱状图  
# 读取数据  
n = time.strftime("%Y-%m-%d") + "-{}-4db.csv".format(a)  
data = pd.read\_csv(n)  
  
# 设置窗口  
fig, ax = plt.subplots(1, 1)  
# 设置绘图风格及字体  
sns.set\_style("whitegrid", {'font.sans-serif': ['simhei', 'Arial']})  
  
# 绘制柱状图  
g = sns.barplot(x="province", y="data", hue="type", data=data, ax=ax,  
 palette=sns.color\_palette("hls", 8))  
  
# 设置Axes的标题  
ax.set\_title('{}疫情最新情况'.format(a))  
  
# 设置坐标轴文字方向  
ax.set\_xticklabels(ax.get\_xticklabels(), rotation=-60)  
  
# 设置坐标轴刻度的字体大小  
ax.tick\_params(axis='x', labelsize=8)  
ax.tick\_params(axis='y', labelsize=8)  
  
plt.show()

绘制省份疫情柱状图

# 绘制全国疫情地图  
def map\_cn\_disease\_dis() -> Map:  
 c = (  
 Map()  
 .add('中国', ls\_province\_cfm, 'china')  
 .set\_global\_opts(  
 title\_opts=opts.TitleOpts(title='全国新型冠状病毒疫情地图（确诊数）'),  
 visualmap\_opts=opts.VisualMapOpts(is\_show=True,  
 split\_number=6,  
 is\_piecewise=True, # 是否为分段型  
 pos\_top='center',  
 pieces=[  
 {'min': 10000, 'color': '#7f1818'}, #不指定 max  
 {'min': 1000, 'max': 10000},  
 {'min': 500, 'max': 999},  
 {'min': 100, 'max': 499},  
 {'min': 10, 'max': 99},  
 {'min': 0, 'max': 5} ],  
 ),  
 )  
 )  
 return c

绘制全国疫情地图

def catch\_cn\_disease\_dis():  
 timestamp = '%d'%int(time.time()\*1000)  
 url\_area = ('https://view.inews.qq.com/g2/getOnsInfo?name=disease\_h5'  
 '&callback=&\_=') + timestamp  
 world\_data = json.loads(requests.get(url=url\_area).json()['data'])  
 china\_data = jsonpath.jsonpath(world\_data,  
 expr='$.areaTree[0].children[\*]')  
 ls\_province\_names = jsonpath.jsonpath(china\_data, expr='$[\*].name')  
 ls\_confirm\_vals = jsonpath.jsonpath(china\_data, expr='$[\*].total.confirm')  
 ls\_province\_confirm = list(zip(ls\_province\_names, ls\_confirm\_vals,))  
 return ls\_province\_confirm, world\_data

获取全国疫情地区分布（各省确诊病例）

b)

def load\_tf\_model(tf\_model\_path):  
 *'''  
 Load the model.* ***:param*** *tf\_model\_path: model to tensorflow model.* ***:return****: session and graph  
 '''* detection\_graph = tf.Graph()  
 with detection\_graph.as\_default():  
 od\_graph\_def = tf.GraphDef()  
 with tf.gfile.GFile(tf\_model\_path, 'rb') as fid:  
 serialized\_graph = fid.read()  
 od\_graph\_def.ParseFromString(serialized\_graph)  
 tf.import\_graph\_def(od\_graph\_def, name='')  
 with detection\_graph.as\_default():  
 sess = tf.Session(graph=detection\_graph)  
 return sess, detection\_graph

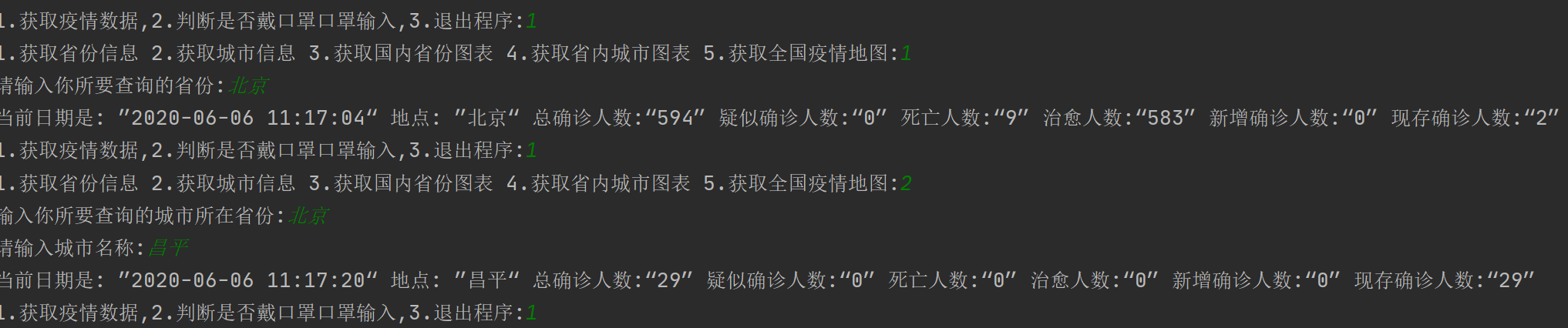
def decode\_bbox(anchors, raw\_outputs, variances=[0.1, 0.1, 0.2, 0.2]):  
 *'''  
 根据锚点解码实际 bbox。.  
 the anchor value order is:[xmin,ymin, xmax, ymax]* ***:param*** *anchors: numpy array with shape [batch, num\_anchors, 4]* ***:param*** *raw\_outputs: numpy array with the same shape with anchors* ***:param*** *variances: list of float, default=[0.1, 0.1, 0.2, 0.2]* ***:return****:  
 '''* anchor\_centers\_x = (anchors[:, :, 0:1] + anchors[:, :, 2:3]) / 2  
 anchor\_centers\_y = (anchors[:, :, 1:2] + anchors[:, :, 3:]) / 2  
 anchors\_w = anchors[:, :, 2:3] - anchors[:, :, 0:1]  
 anchors\_h = anchors[:, :, 3:] - anchors[:, :, 1:2]  
 raw\_outputs\_rescale = raw\_outputs \* np.array(variances)  
 predict\_center\_x = raw\_outputs\_rescale[:, :, 0:1] \* anchors\_w + anchor\_centers\_x  
 predict\_center\_y = raw\_outputs\_rescale[:, :, 1:2] \* anchors\_h + anchor\_centers\_y  
 predict\_w = np.exp(raw\_outputs\_rescale[:, :, 2:3]) \* anchors\_w  
 predict\_h = np.exp(raw\_outputs\_rescale[:, :, 3:]) \* anchors\_h  
 predict\_xmin = predict\_center\_x - predict\_w / 2  
 predict\_ymin = predict\_center\_y - predict\_h / 2  
 predict\_xmax = predict\_center\_x + predict\_w / 2  
 predict\_ymax = predict\_center\_y + predict\_h / 2  
 predict\_bbox = np.concatenate([predict\_xmin, predict\_ymin, predict\_xmax, predict\_ymax], axis=-1)  
 return predict\_bbox

对获取的照片进行分析，根据锚点解码实际 bbox。

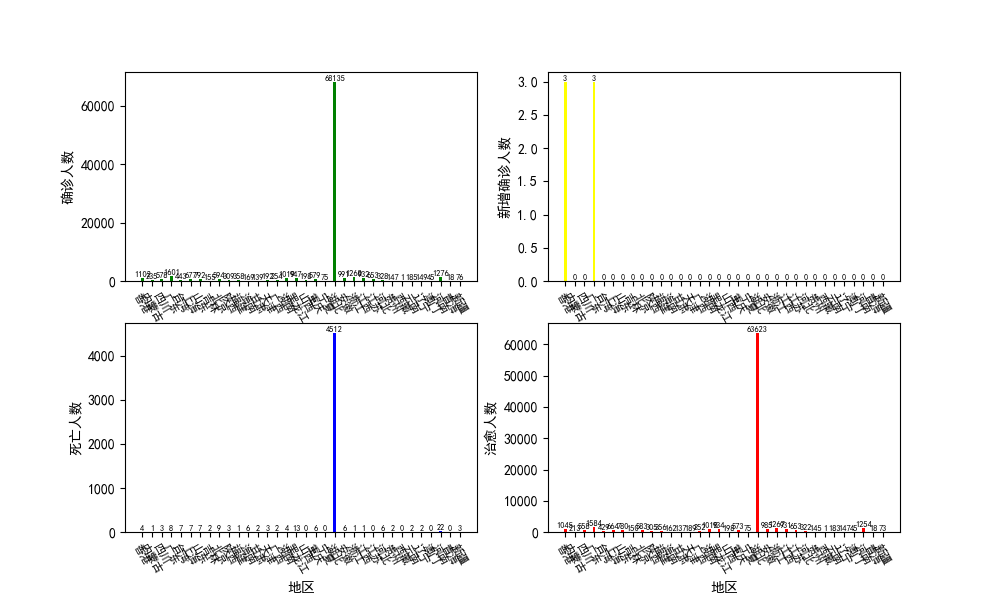
1. 结果

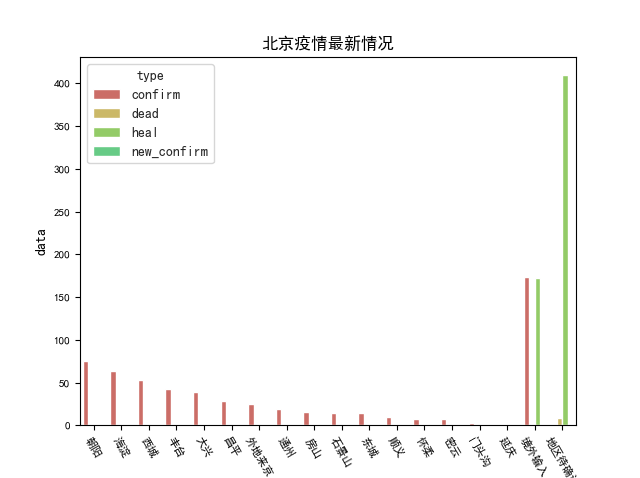
可以获取到我们想要获得的疫情相关数据

1.可查询文字数据

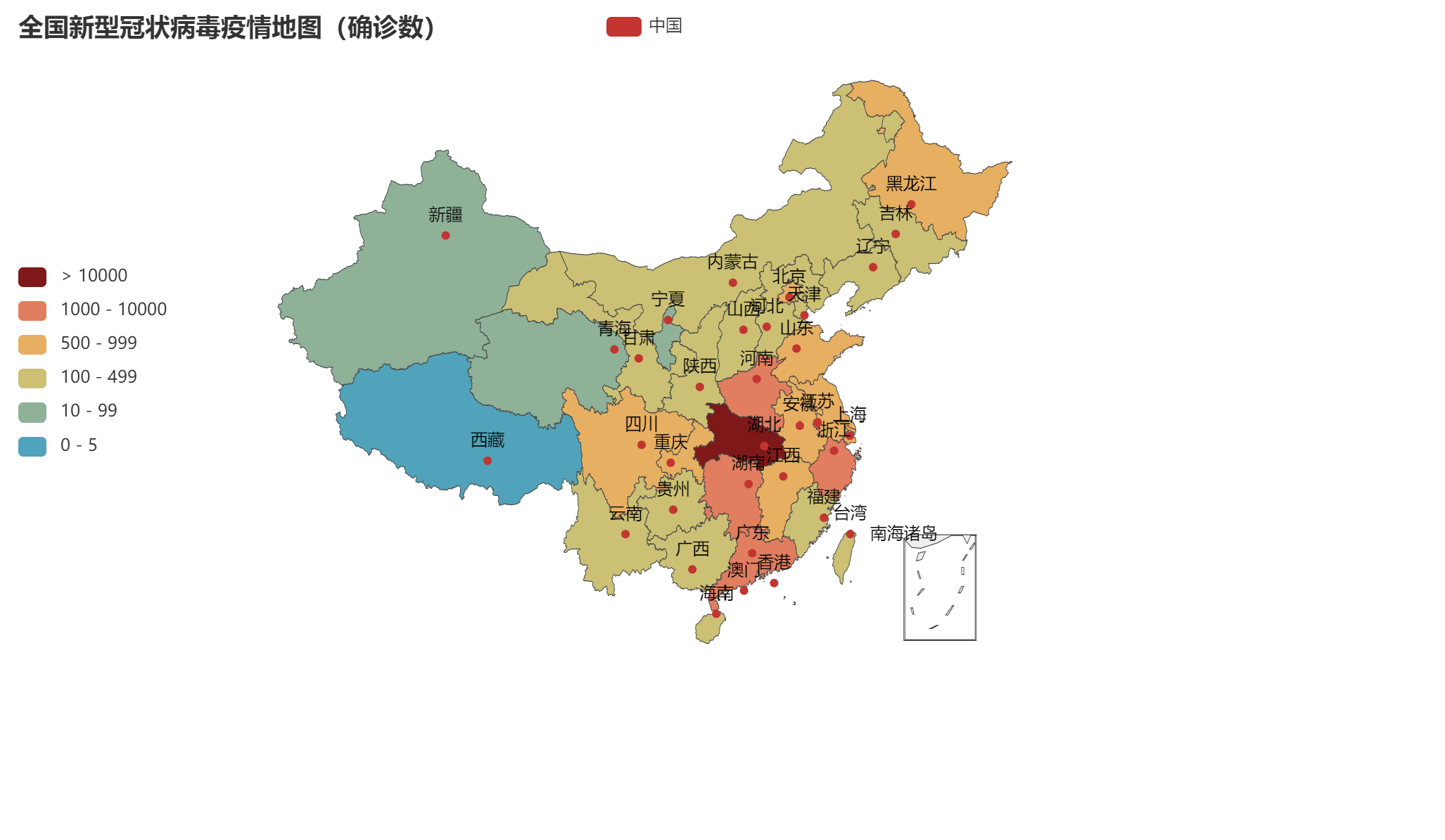


2.可查询图表类数据

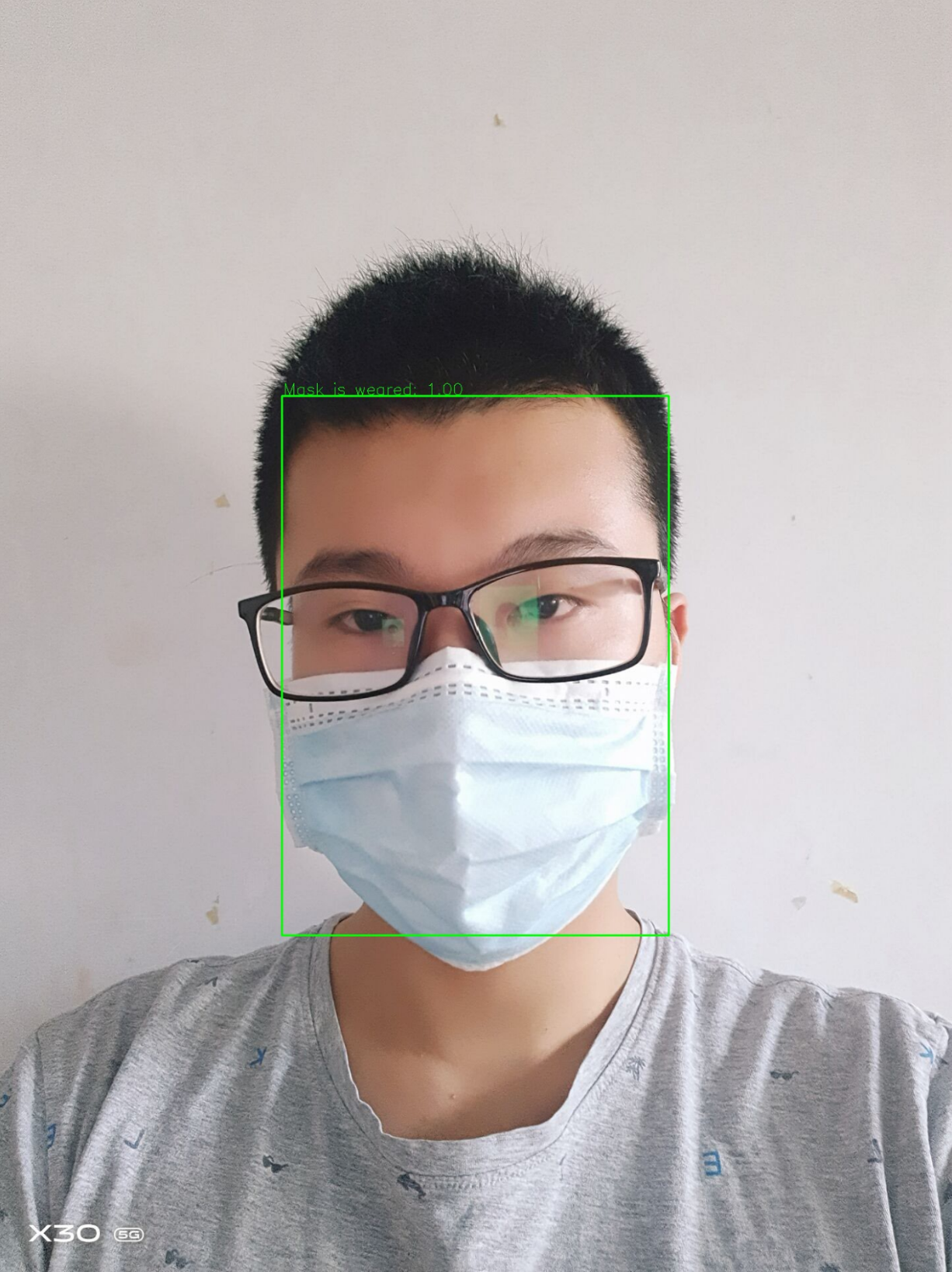


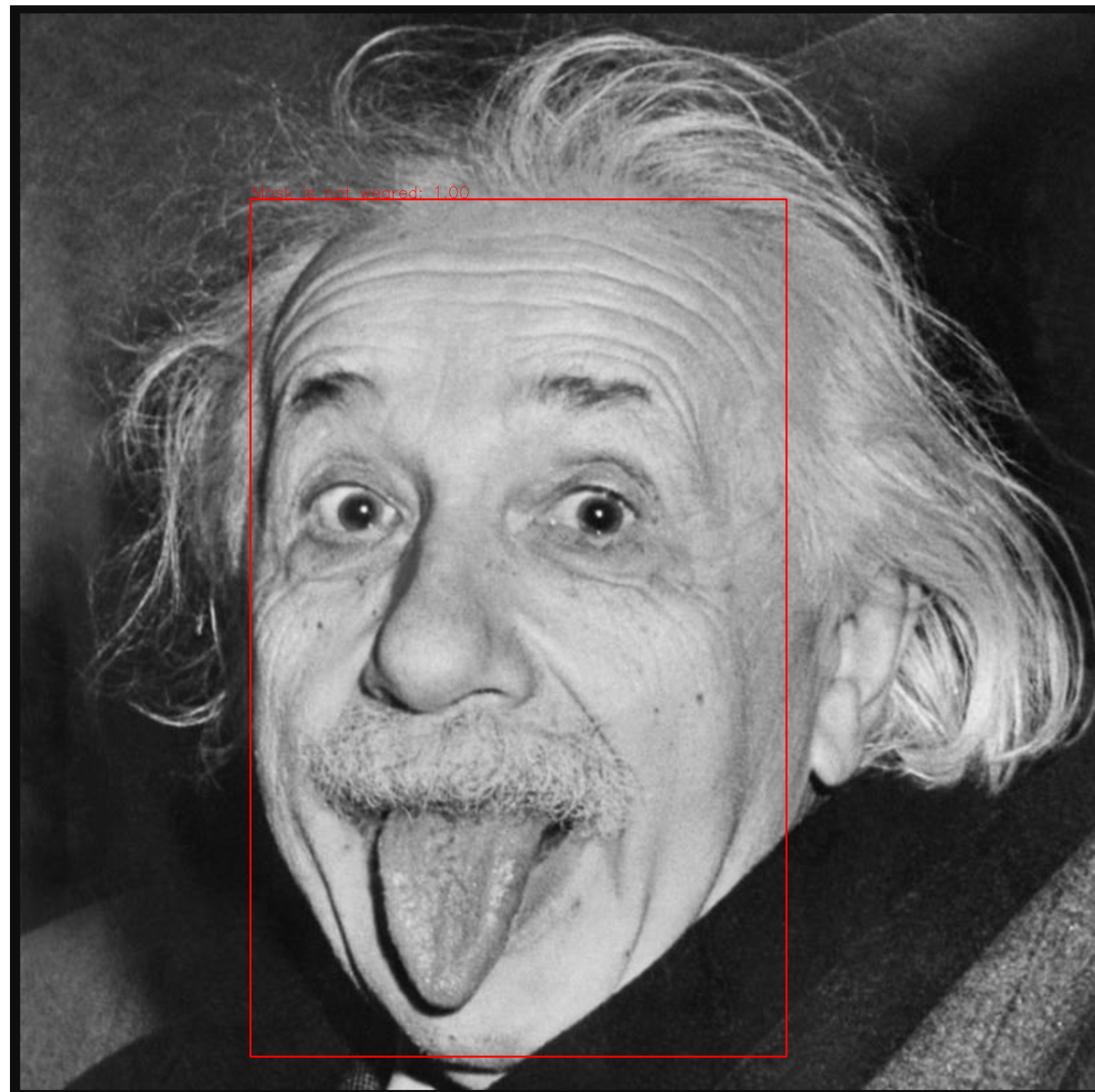


3.可查询全国疫情地图



4.对照片的口罩识别





5．对视频的实时口罩识别结果为该文件夹的video.mp4

1. 未来的展望或讨论
   1. 在网络爬虫部分，目前只能对特定的几个网站进行爬取，得到的数据相对单一，对于数据的汇总也存在着一定的问题，未来要把所有的数据汇总到一个本地网页上。之前一直想要做有关疫情的舆情分析但是限于时间与能力的问题现在没有完成，暑假我会继续完善。
   2. 在口罩识别部分，tensorflow使用的是CPU部分的，受到CPU并行处理数据速度慢的原因，代码速度就会有所下降，使用摄像头进行实时处理时，往我把提取图像的间隔调整为0.01s时，显示就会出现花屏，红屏。因此以后需要使用tensorflow的GPU部分，以提升并行处理的速度，增强代码处理的强度。

参考文献：

解析深度学习：卷积神经网络原理与视觉实践 魏秀参 著

TensorFlow:实战google深度学习框架

TensorFlow官方文档

OpenCV3计算机视觉python语言实现

OpenCV官方文档