**目 录**

[内容摘要 I](#_Toc105159426)

[1 导言 1](#_Toc105159427)

[1.1 研究背景及意义 1](#_Toc105159428)

[1.2 国内外研究概述 2](#_Toc105159429)

[1.3 研究内容与方法 2](#_Toc105159430)

[2 算法设计 4](#_Toc105159431)

[2.1 试剂盒检测算法设计 4](#_Toc105159432)

[2.1.1 神经网络结构设计 4](#_Toc105159433)

[2.1.2 网络损失函数定义 5](#_Toc105159434)

[2.2 试剂盒检测结果识别算法 6](#_Toc105159435)

[2.2.1 检测区域图像校正 6](#_Toc105159436)

[2.2.2 结果识别算法 7](#_Toc105159437)

[3 算法训练与结果分析 9](#_Toc105159438)

[3.1 试剂盒检测算法训练 9](#_Toc105159439)

[3.1.1 数据集准备 9](#_Toc105159440)

[3.1.2 训练集图片处理流程 9](#_Toc105159441)

[3.1.3 训练方法 10](#_Toc105159442)

[3.2 结果分析 11](#_Toc105159443)

[4 结论 12](#_Toc105159444)

[参考文献 13](#_Toc105159445)

[附录 14](#_Toc105159446)

内容摘要：新冠疫情出现以来，快速检测并发现感染者作为一种有效防控手段，在疫情防控中发挥了重要作用。本文提出一种针对新冠病毒抗原检测试剂盒识别的算法，采用多层卷积神经网络定位试剂盒区域，通过图像处理算法识别检测结果。本文通过在事先采集的试剂盒图像上进行标注，采用随机梯度下降算法对神经网络进行训练。本文提出的算法在验证集上达到了定位精度大于99%，综合识别准确率在95%以上。结论证明本文算法可以运用于疫情防控领域，能够有效实现抗原检测结果的信息化采集，减轻数据统计工作量，助力提升疫情早发现能力。

关键词：新冠病毒抗原试剂盒；目标检测；图像处理

1. 导言
   1. 研究背景及意义

当前，新型冠状病毒肺炎（Corona Virus Disease 2019，COVID-19）疫情仍在持续流行，奥密克戎（Omicron）变异毒株引发的新一轮疫情正在席卷全球，数十亿人的生命健康面临严重威胁。截至2022年6月1日，全球超过210多个国家及地区出现新冠确诊病例，累计超过5.3亿人感染新冠病毒，累计629万人因感染新冠病毒死亡。自疫情发生以来，我国坚持人民至上，生命至上的疫情防控总方针，坚持外防输入、内防反弹，坚持动态清零，因时因势不断调整防控措施，疫情防控取得了重大战略成果。目前全球疫情仍处于高位，病毒还在不断变异，疫情的最终走向还存在很大不确定性，疫情防控形势仍然不可放松。我国是人口大国，老龄人口多，地区发展不平衡，医疗资源总量不足，放松防控势必造成大规模人群感染、出现大量重症和病亡，经济社会发展和人民生命安全、身体健康将受到严重影响。针对控制新冠病毒传播实施的动态清零政策注重对早期感染者的精准识别与及时隔离治疗，因此研究如何准确且快速的对新冠病毒感染进行识别是一项十分重要的研究课题。

目前对于新冠病毒感染主要的识别方法有荧光聚合酶链式反应（PCR）识别,俗称核酸检测[1]，以及肺部影响识别和特异性抗原检测试剂识别。表1-1为各种检测方法的指标对比。

表1-1 新冠病毒感染识别方法对比

|  |  |  |  |
| --- | --- | --- | --- |
| 指标\检测方法 | 核酸检测 | 影像检测 | 抗原检测 |
| 误检率 | 较低 | 低 | 较高 |
| 检测时间 | 4~6小时 | 5分钟 | 5分钟 |
| 成本 | 5~16元 | 50~100元 | 8~20元 |
| 病情信息 | 较少 | 较多 | 较多 |
| 安全性 | 较高 | 较低 | 较高 |
| 是否可自测 | 否 | 否 | 是 |

2022年3月14日，抗原检测试剂盒作为早期检测病毒感染的手段进入《新型冠状病毒肺炎诊疗方案（试行第九版）》中。国家卫健委发布的《新冠病毒抗原检测应用方案（试行）》明确指出，抗原检测适用人群包括三类：第一类是基层医疗卫生机构就诊，伴有呼吸道、发热等症状且出现症状5天以内的人员；第二类是隔离观察人员，包括居家隔离观察、密接和次密接、入境隔离观察、封控区和管控区内的人员；第三类是有抗原自我检测需求的社区居民。抗原检测具有快速出结果和操作简单等优势，其阳性结果可以用于对疑似人群的早期分流与快速管理。抗原检测试剂盒成本低廉，本身不具有数据采集上报功能，在大规模检测应用中只能采取人工识别、手动上报等操作，影响了对于疑似感染者的发现效率。本文提出一种基于卷积神经网络的新冠病毒抗原检测试剂盒结果识别方法，能够有效解决大规模检测实施过程中的结果统计与快速响应需求。

* 1. 国内外研究概述

随着深度学习技术的不断发展，卷积神经网络（Convolutional Neural Netw-orks，CNN）凭借其优秀的性能表现已经成为目标检测中的一种流行技术[2]。多种基于CNN的神经网络结构被用于目标检测与图像分类，例如 AlexNet[3]、VGGNet和ResNet[4]。卷积神经网络广泛的应用于解决视频或者图片中的目标物体的检测与定位识别问题。FasterR-CNN[5]和YOLO-V3[6]是两种性能卓越的CNN目标检测算法，它们具有检测精度高、检测速度快的优点。Faster R-CNN是Fast R-CNN的改进，该算法执行四步目标检测，即目标区域提取、特征提取、目标分类和检测框回归[7]。其采用了区域生成网络（Region Proposal Network， RPN）,直接使用RPN网络生成候选检测框，极大的提高了检测框的生成速度和精度。YOLO-V3 算法将分类和定位直接结合成一个任务识别目标，以整张图片为兴趣区域。结合三个尺度的预测特征层融合，利用FPN算法融合不同尺度的特种，增强小目标的检测效果。目前CNN在医学检测领域有着广泛应用。

* 1. 研究内容与方法

本文需要研究检测的抗原试剂盒为白色长方体塑料盒，其中有结果显示区域与试剂存放区域，常见的抗原检测试剂盒及其测试结果显示区域如图1-1所示。抗原检测试剂盒可得出三种检测结果如图1-2所示，其判别方式为：

阳性结果：”C”和”T”处均显示红色条带；

阴性结果：”C”处显示红色条带，”T”处未显示条带；

无效结果：”C”处未显示条带。

电子设备

描述已自动生成

图1-1 抗原检测试剂盒及其结果显示区域

图表

描述已自动生成

图1-2 抗原检测结果识别

本文提出一种能够快速准确检测出目标图片中的检测结果的算法。主要研究思路如下：

1. 采用卷积神经网络和目标检测算法，定位试剂盒在图片中的具体位置；
2. 找到检测区域的四角关键点的位置，利用透视变换对结果区域进行标准化处理；
3. 定位显色条带位置，分析识别结果。
4. 算法设计
   1. 试剂盒检测算法设计
      1. 神经网络结构设计

本文采用了一种基于残差结构的深度卷积神经网络，其基本构造由三部分组成。其一部分称为骨干网络（backbone），用于提取图像中的基础特征，其二部分称为颈部网络（Neck），用于连接底层特征与高层特征，利于检测不同尺度目标的检测，同时可以使后续网络获取更多底层特征信息，提高检测的精度，其三是检测头网络（Detector head）,用于输出目标分类、回归目标位置框、回归特征点等任务，通过更改检测头可以使网络适应不同的目标检测任务。

本文提出的骨干网络采用残差模块（Resblock）进行构建，其主要思想是通过在卷积神经网络中添加跳层连接解决深层卷积网络的梯度消失问题。多层残差模块可以构成残差网络，其中残差单元的基本结构如图2-1所示：

图示

描述已自动生成

图2‑1 残差模块的基本结构

图2-1中的残差单元定义为：

是残差单元的第层的输入和输出，和是输入层的长宽维度，是通道数，是时间。是权重矩阵，是表示卷积运算的残差函数。本文采用每两个残差模块为一组，共四组模块串联，每组输出的通道数分别为24、48、48、80。采用Relu函数[8]作为激活函数。

Neck部分网络采用基于特征金字塔FPN网络[9]改进来的PAFPN网络[10]，FPN网络在骨干网络的四组输出后，对不同层级的特征图进行上采样并进行级联运算，从而使不同层次尺度的特征信息在网络中更好的传播。PAFPN在FPN的基础上将最后一层自上而下传播改为自下而上传播从而使信息的流动更加高效，提高了神经网络的整体性能，如图2-2所示。

图表, 散点图

描述已自动生成

图2‑2 PAFPN与FPN网络结构

本文采用的PAFPN的输出通道数为16，输出3个节点用于连接下层网络。

本文算法采用的检测头网络为三分支网络结构，其中一个分支用于预测检测框bbox的分类置信度，一个分支用于预测bbox相对于其锚点anchor(x,y,w,h)的偏移量，一个分支用于预测抗原试剂盒检测区域的四个角点坐标。

* + 1. 网络损失函数定义

本文算法的分类分支采用了基于Focal Loss改进的GFL[11]损失函数，其中Focal Loss的公式为：

其中，为缩放系数 。

GFL 是FL损失函数的扩展形式，它改进了定位质量估计与分类score直接 的关联性，通过将每个类别的分数变为与真实样本之间的重叠区域。GFL损失函数有一下优点：

（1）消除额外的质量估计分支在训练和测试时的差异，提出简单且高效的联合预测策略。

（2）很好地对预测框的真实分布进行灵活建模，提供更多的信息以及准确的位置预测。

（3）在引入额外开销的情况下，能够提升所有一阶检测算法的性能。

分类框的预测损失采用了DIoU损失，其公式为：

其中和代表预测框的中心点和真实框的中心点, 表示欧式距离，表示包含两个框的最小矩阵的对角线长度。的计算公式为：

其中 和代表预测框和真实框的面积。

DIOU解决了直接最小化先验框anchor与目标框之间的归一化距离，能够达到更快的收敛速度。

关键点检测分支采用了SmoothL1Loss，是基于L1范数损失改进的平滑版本损失，能够防止发生梯度爆炸，能够适用于关键点回归任务。

本文算法总损失函数是对上述三部分损失函数加权求和得来，其权重分别为0.25，0.5，0.25。

* 1. 试剂盒检测结果识别算法
     1. 检测区域图像校正

由摄像头等图像采集设备采集的真实试剂图像往往不能保证一定的角度，为了保证后续结果分类的鲁棒性，需要对真实图像进行校正，使目标区域对齐在统一的矩形平面内。本文采用透视变换的方法对图像进行校正。透视变换是一种将一个平面通过一个投影矩阵投影到指定平面上，其公式为：

其中和是原始图片，参数等于1，通过透视变换后得到的图片坐标和，其中, 。

根据上述公式可以得知，需要求解透视变换矩阵的8个参数，所以要由4个角点的坐标对求解8个方程。原始图像的四个角点坐标由神经网络算法的关键点预测分支输出，变换后的角点坐标分别设定为（0， 0）、（16，0）、（0，96）、（16，96）四个点，对应的目标图像尺寸为16x96像素。原始图像与校正后的图像如图2-3所示。

白板

中度可信度描述已自动生成

图2-3 原始与透视变换后的结果区域图像

* + 1. 结果识别算法

本文采用了图像处理算法来对结果区域的图像进行增强处理，根据检测试剂的预设特征作为判断条件，综合比对得出检测结果信息。算法首先对图片进行灰度化处理并进行黑白反色，使信号条更清晰显示。然后对图像进行高斯模糊处理，去除图片中的噪声干扰，再对图像进行以（3，10）为核心的开运算，用于消除细纹和强化横向纹理。图像增强结果如图2-4所示。

文本

描述已自动生成

图2-4 原始图像与增强后图像

对增强后的图像像素以横轴为方向进行累加，可以得到具有信号条识别信息的曲线，对给定的C、T线区域内的累加值求均值，若超过非信号区域的均值，即可认定为存在该信号条。如图2-5所示，根据检测试剂识别规则可以确定样本图片属于阴性、阳性或者无效结果。

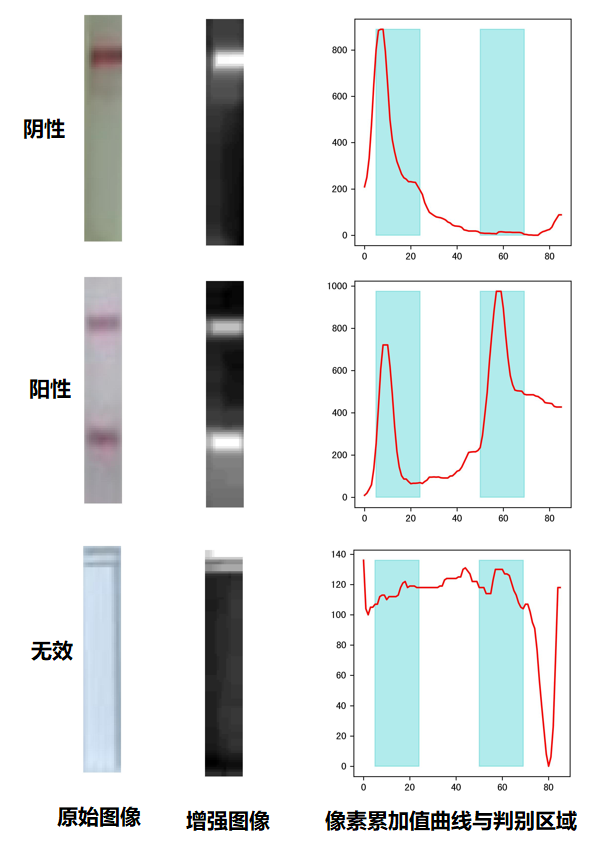


图2‑5 图像经处理后的结果判别依据

1. 算法训练与结果分析
   1. 试剂盒检测算法训练
      1. 数据集准备

本文中用于训练试剂盒检测算法的数据集来源于手机摄像头采集，部分来源于网络公开图片。数据集图片共214张，其中摄像头采集200张，网络搜集14张。其中包含不同角度，不同检测结果，不同品牌，不同背景下的试剂盒图片。数据集使用图像标注软件进行标注，标注抗原检测试剂的目标框以及结果显示区域的四个角点坐标，目标框覆盖整个试剂盒区域并选取该区域的左上和右下点坐标作为目标框。数据集中选取90%作为训练集，10%作为验证集。数据集部分图像如图3-1所示。

图形用户界面

中度可信度描述已自动生成

图 3‑1 抗原检测试剂盒数据集中的部分图片

* + 1. 训练集图片处理流程

由于采集数据集数量较小，直接使用数据集训练会产生过拟合现象，本文在上述数据集的基础上对训练数据集部分进行了数据增强与扩充。同时，为了适应神经网络的输入需求，训练数据还会进行尺度变换和归一化操作。具体的操作流程如下：

1. 对图像进行随机方形裁剪，按照0.3~2.0倍宽度，对输入图像进行裁剪；
2. 按比例缩放图片大小至640x640分辨率；
3. 图像随机增强，对图像进行随机操作，包括随机亮度、随机对比度调整、将颜色从 BGR 转换为 HSV、随机饱和度、随机色调、将颜色从 HSV 转换为 BGR、随机交换通道；
4. 将图片进行归一化

数据增强后的部分训练集图片如图3-2所示：

手机屏幕截图

中度可信度描述已自动生成

图3‑2 数据增强后的图片

* + 1. 训练方法

在准备好训练数据集后，使用pytorch框架搭建神经网络模型并进行训练。在构建好基础的模型结构和损失函数之后，将训练数据集及其真实标签送入神经网络进行训练，采用反向传播和随机梯度下降算法（SGD）对神经网络权重进行优化，使其能够满足试剂盒检测需求。SGD算法采用的学习率为0.01，动量为0.9。训练的batchsize为32，轮次为80。训练过程中的损失函数变化曲线如图3-3所示。

图形用户界面, 文本

中度可信度描述已自动生成

图3-3 训练过程中的损失变化图

训练过程中的mAP变化曲线如图3-4所示。

形状

中度可信度描述已自动生成

图3-1 mAP指标变化曲线

* 1. 结果分析

本文将训练后的模型在验证集上进行验证测试并评估试剂盒检测的mAP指标，经验证，该模型在验证集上的指标可达99%以上，具有一定的检测精度。同时，将检测结果采用上述结果识别算法中，识别结果见表3-1.，其识别准确率可达95%以上。

表3-1 验证集结果

|  |  |  |
| --- | --- | --- |
| **验证集图片** | **识别值** | **真实值** |
| val\_0 | Negative | Negative |
| val\_1 | Negative | Negative |
| val\_2 | Invalid | Invalid |
| val\_3 | Negative | Negative |
| val\_4 | Invalid | Invalid |
| val\_5 | Negative | Negative |
| val\_6 | Negative | Negative |
| val\_7 | Invalid | Invalid |
| val\_8 | Negative | Invalid |
| val\_9 | Invalid | Invalid |
| val\_10 | Invalid | Invalid |
| val\_11 | Negative | Negative |
| val\_12 | Negative | Negative |
| val\_13 | Negative | Negative |
| val\_14 | Negative | Negative |
| val\_15 | Invalid | Invalid |
| val\_16 | Positive | Positive |
| val\_17 | Invalid | Invalid |
| val\_18 | Positive | Positive |
| val\_19 | Positive | Positive |
| val\_20 | Positive | Positive |

1. 结论

本文提出了一种基于卷积神经网络的新冠病毒抗原检测试剂盒识别算法，通过构建试剂盒目标检测算法，对试剂盒检测区域进行定位，然后通过透视变换校正检测区域，最后通过图像处理算法结合检测区域识别规则对识别结果进行解析，最终验证了该算法能够完成抗原检测试剂的结果识别需求。同时，算法整体的识别准确率大于95%，可与其他算法联合判别，做到精准识别检验结果。

参考文献

1. 陈玫君,宋江勤,曹伟伟,康娜,吕文静,周君阳.新冠肺炎患者应用核酸联合抗体检测的临床应用[J].医学食疗与健康,2020,18(24):165+202.
2. Shin H C, Roth H R, Gao M, et al. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning[J]. IEEE transactions on medical imaging, 2016, 35(5): 1285-1298.
3. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.
4. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
5. Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. Advances in neural information processing systems, 2015, 28.
6. Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.
7. Ren S, He K, Girshick R, et al. Object detection networks on convolutional feature maps[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(7): 1476-1481.
8. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks[C]//Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2011: 315-323.
9. Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.
10. Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8759-8768.
11. Li X, Wang W, Wu L, et al. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection[J]. Advances in Neural Information Processing Systems, 2020, 33: 21002-21012.
12. Guo J, Deng J, Lattas A, et al. Sample and computation redistribution for efficient face detection[J]. arXiv preprint arXiv:2105.04714, 2021.
13. Chen K, Wang J, Pang J, et al. MMDetection: Open mmlab detection toolbox and benchmark[J]. arXiv preprint arXiv:1906.07155, 2019.

附录

**from** tools.scrfd **import** SCRFD

**import** numpy **as** np

**import** os.path **as** osp

**import** cv2

**import** matplotlib.pyplot **as** plt

In [ ]:

MODLE\_PATH **=** "scrfd\_x00m\_bnkps\_fast\_covid-kit\_resnet\_shape640x640.onnx"

detector **=** SCRFD(model\_file **=** MODLE\_PATH)

In [ ]:

detector**.**prepare(**-**1) *#cpu run*

In [ ]:

val\_list **=** "covid-kit/val.txt"

*# val\_list = "covid-kit/train.txt"*

val\_image\_path **=** []

**with** open(val\_list, "r") **as** f:

**for** line **in** f**.**readlines():

splits **=** line**.**split(" ")

**if** splits[0] **==** "#":

val\_image\_path**.**append(splits[1])

print("val images:", len(val\_image\_path))

val images: 21

In [ ]:

**def** transform\_target(img, kps, bbox):

x1, y1, x2, y2, score **=** bbox**.**astype(np**.**int32)

xmin **=** max(0, x1)

ymin **=** max(0, y1)

xmax **=** min(img**.**shape[1], x2)

ymax **=** min(img**.**shape[0], y2)

w **=** xmax **-** xmin **+** 1

h **=** ymax **-** ymin **+** 1

img\_box **=** np**.**zeros((h, w, 3), dtype**=**np**.**uint8)

img\_box**=** img[ymin:ymax**+**1, xmin:xmax**+**1, :]**.**copy()

kps **=** kps**.**astype(np**.**int32)

new\_x1, new\_y1 **=** kps[0][0] **-** xmin, kps[0][1] **-** ymin

new\_x2, new\_y2 **=** kps[1][0] **-** xmin, kps[1][1] **-** ymin

new\_x3, new\_y3 **=** kps[2][0] **-** xmin, kps[2][1] **-** ymin

new\_x4, new\_y4 **=** kps[3][0] **-** xmin, kps[3][1] **-** ymin

points1 **=** np**.**float32([[new\_x1, new\_y1], [new\_x2, new\_y2], [new\_x3, new\_y3], [new\_x4, new\_y4]])

points2 **=** np**.**float32([[0, 0], [16, 0], [16, 96], [0, 96]])

M **=** cv2**.**getPerspectiveTransform(points1, points2)

dst **=** cv2**.**warpPerspective(img\_box, M, (16, 96))

**return** dst

In [ ]:

c\_line\_area **=** [5,25]

t\_line\_area **=** [50,70]

other\_area **=** [35,45]

c\_thresh **=** 40

t\_thresh **=** 15

**def** hist\_anysis(hist):

c\_hist **=** hist[c\_line\_area[0]:c\_line\_area[1]]

t\_hist **=** hist[t\_line\_area[0]:t\_line\_area[1]]

other\_hist **=** hist[other\_area[0]:other\_area[1]]

c\_mean **=** np**.**mean(c\_hist)

t\_mean **=** np**.**mean(t\_hist)

other\_mean **=** np**.**mean(other\_hist)

c\_diff **=** c\_mean **-** other\_mean

t\_diff **=** t\_mean **-** other\_mean

*# print("c\_diff:", c\_diff, "t\_diff:", t\_diff)*

*# print("c\_mean:", c\_mean, "t\_mean:", t\_mean, "other\_mean:", other\_mean)*

**if** c\_diff **>** c\_thresh **and** t\_diff **>** t\_thresh:

**return** "Positive"

**elif** c\_diff **>** c\_thresh **and** t\_diff **<** t\_thresh:

**return** "Negative"

**else**:

**return** "Invalid"

In [ ]:

hist\_offset **=** 10

hist\_len **=** 86

hist\_sum **=** np**.**zeros((hist\_len,), dtype**=**np**.**float32)

**for** idx, img\_path **in** enumerate(val\_image\_path):

img **=** cv2**.**imread(img\_path)

bboxes, kpss **=** detector**.**detect(img, 0.5, input\_size**=**(640, 640))

img\_view **=** img**.**copy()

**for** i **in** range(bboxes**.**shape[0]):

bbox **=** bboxes[i]

x1, y1, x2, y2, score **=** bbox**.**astype(np**.**int32)

cv2**.**rectangle(img\_view, (x1, y1), (x2, y2), (0, 255, 0), 2)

**if** kpss **is** **not** **None**:

kps **=** kpss[i]

**for** kp **in** kps:

kp **=** kp**.**astype(np**.**int32)

cv2**.**circle(img\_view, (kp[0], kp[1]), 2, (0, 0, 255), **-**1)

*# plt.imshow(img\_view[:, :, ::-1])*

*# plt.show()*

processed\_img **=** transform\_target(img, kps, bbox)

*# plt.imshow(processed\_img[:, :, ::-1])*

*# plt.show()*

gray **=** cv2**.**cvtColor(processed\_img, cv2**.**COLOR\_BGR2GRAY)

reverse\_gray **=** 255 **-** gray

gray **=** reverse\_gray

blur **=** cv2**.**GaussianBlur(gray, (5, 5), 0)

*# \_, binary = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY | cv2.THRESH\_OTSU)*

*# binary = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY\_INV, 11, 2)*

*# plt.imshow(binary, cmap='gray')*

*# plt.show()*

*# plt.imshow(gray, cmap='gray')*

*# plt.show()*

*# plt.imshow(reverse\_gray, cmap='gray')*

*# plt.show()*

*# plt.imshow(blur, cmap='gray')*

*# plt.show()*

kernel **=** np**.**ones((3, 10), dtype**=**np**.**uint8)

*# erosion = cv2.erode(binary, kernel, iterations=1)*

*# plt.imshow(erosion, cmap='gray')*

*# plt.show()*

opening **=** cv2**.**morphologyEx(blur, cv2**.**MORPH\_OPEN, kernel, 3)

*# plt.imshow(opening, cmap='gray')*

*# plt.show()*

hist **=** opening**.**sum(axis**=**1)

hist **=** hist **-** hist**.**min()

hist **=** hist[hist\_offset:hist\_offset**+**hist\_len]

hist\_sum **+=** hist

*# plt.plot(hist)*

*# plt.show()*

resutl **=** hist\_anysis(hist)

*#subplot img,processed\_img,opening and hist*

plt**.**figure(figsize**=**(16, 4))

plt**.**subplots\_adjust(wspace**=**0.1)

plt**.**subplot(1, 4, 1)

plt**.**axis('off')

plt**.**imshow(img\_view[:, :, ::**-**1])

plt**.**title("img")

plt**.**subplot(1, 4, 2)

plt**.**axis('off')

plt**.**imshow(processed\_img[:, :, ::**-**1])

plt**.**title("detected area")

plt**.**subplot(1, 4, 3)

plt**.**axis('off')

plt**.**imshow(opening, cmap**=**'gray')

plt**.**title("detected area after enhance")

plt**.**subplot(1, 4, 4)

plt**.**plot(hist, color**=**'r')

*# plt.plot(hist.mean()\*np.ones(hist.shape), color='r')*

*# fill red from c\_line\_area[0] to c\_line\_area[1]*

hist\_max **=** np**.**zeros(hist**.**shape, dtype**=**np**.**float32)

hist\_max[:] **=** hist**.**max()

plt**.**fill\_between(range(c\_line\_area[0], c\_line\_area[1]), hist\_max[c\_line\_area[0]:c\_line\_area[1]], color**=**'c', alpha**=**0.3)

plt**.**fill\_between(range(t\_line\_area[0], t\_line\_area[1]), hist\_max[t\_line\_area[0]:t\_line\_area[1]], color**=**'c', alpha**=**0.3)

plt**.**title(resutl)

*# plt.savefig(osp.join("/home/bits/insightface/detection/scrfd\_lpr/covid-kit/result\_output", osp.basename(img\_path)), dpi=300)*

plt**.**show()

print("{} Result: "**.**format(idx), resutl)

*# print("val\_" + str(idx) + "," + resutl + "," + resutl)*

plt**.**plot(hist\_sum)

plt**.**show()

电脑萤幕

低可信度描述已自动生成

0 Result: Negative

图片包含 文本

描述已自动生成

1 Result: Negative

文本

描述已自动生成

2 Result: Invalid

文本

低可信度描述已自动生成

3 Result: Negative

文本

描述已自动生成

4 Result: Invalid

电脑屏幕的照片

低可信度描述已自动生成

5 Result: Negative

图片包含 文本

描述已自动生成

6 Result: Negative

图片包含 文本

描述已自动生成

7 Result: Invalid

文本

低可信度描述已自动生成

8 Result: Negative

图片包含 文本

描述已自动生成

9 Result: Invalid

文本

中度可信度描述已自动生成

10 Result: Invalid

电脑的屏幕

低可信度描述已自动生成

11 Result: Negative

文本

低可信度描述已自动生成

12 Result: Negative

文本

低可信度描述已自动生成

13 Result: Negative

电脑萤幕

低可信度描述已自动生成

14 Result: Negative

电脑截图

低可信度描述已自动生成

15 Result: Invalid

电脑屏幕的照片

低可信度描述已自动生成

16 Result: Positive

文本

中度可信度描述已自动生成

17 Result: Invalid

图片包含 游戏机, 电脑, 冰箱

描述已自动生成

18 Result: Positive

电脑萤幕

中度可信度描述已自动生成

19 Result: Positive

屏幕上写着字

描述已自动生成

20 Result: Positive

图表, 折线图

描述已自动生成

In [ ]:

mask **=** np**.**zeros((96, 16), dtype**=**np**.**uint8)

plt**.**imshow(mask, cmap**=**'gray')

plt**.**show()

形状

低可信度描述已自动生成

In [ ]: