

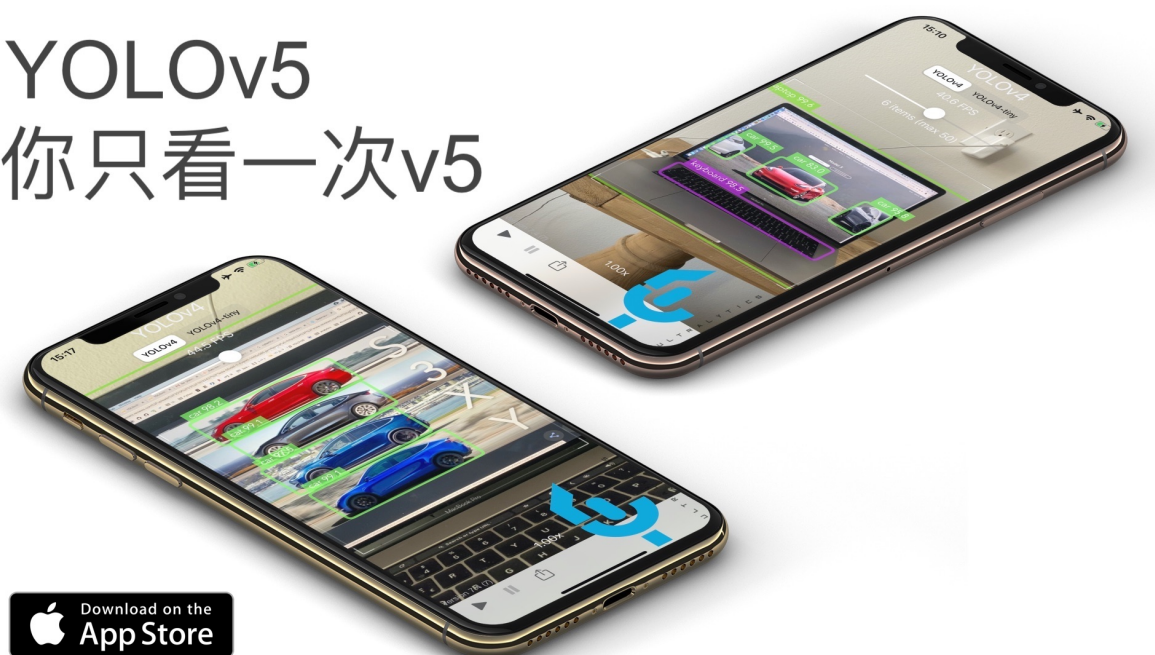
# YOLO v5在医疗领域中消化内镜目标检测的应用

## YOLO v5训练自己数据集详细教程

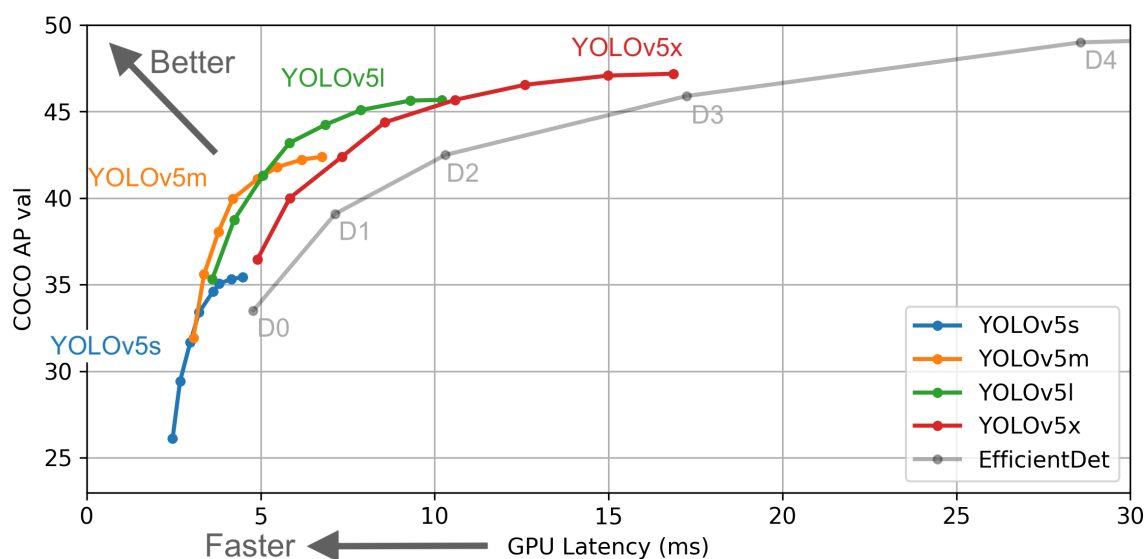
Xu Jing

- YOLOv4还没有退热，YOLOv5已经发布！
- 6月9日，Ultralytics公司开源了YOLOv5，离上一次YOLOv4发布不到50天。而且这一次的YOLOv5是完全基于PyTorch实现的！
- YOLO v5的主要贡献者是YOLO v4中重点介绍的马赛克数据增强的作者

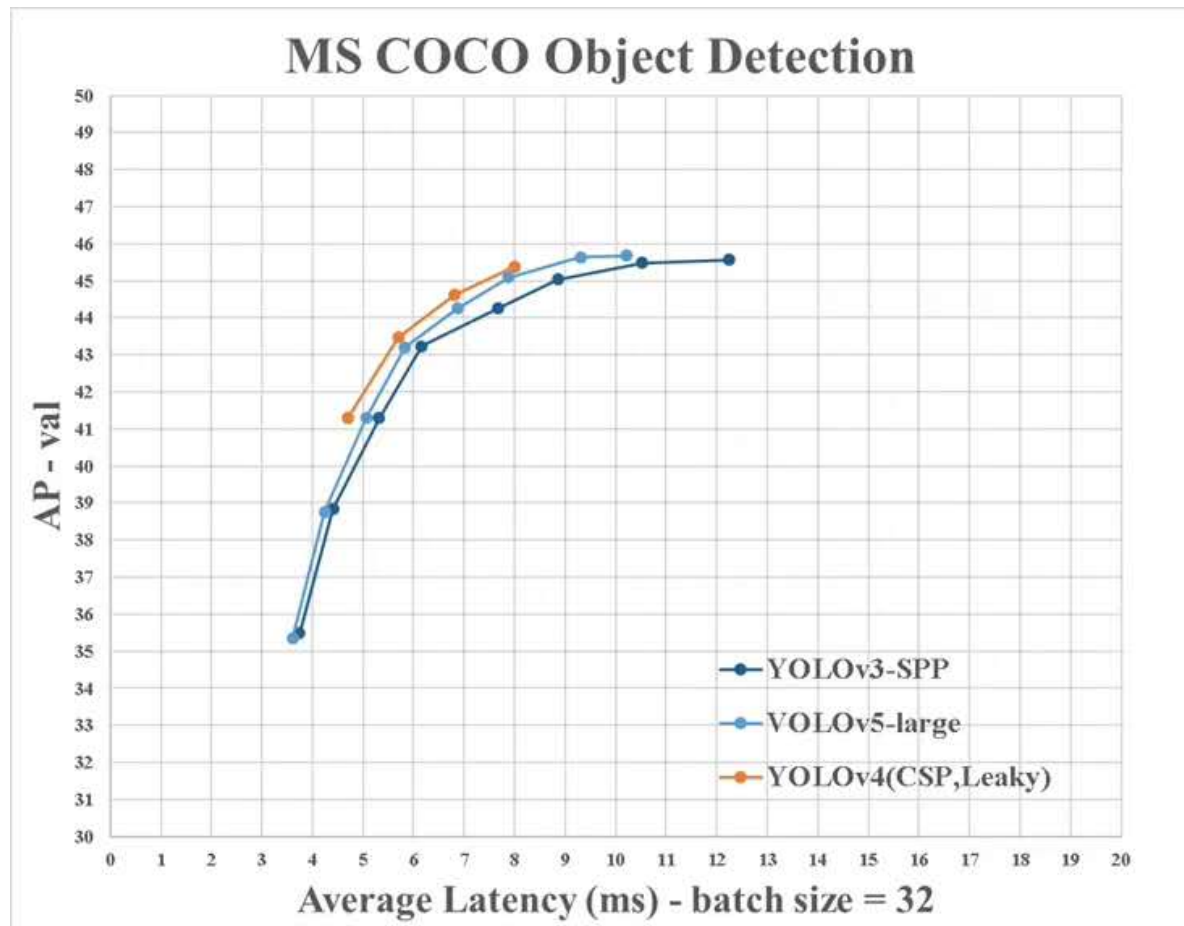
## YOLOv5 你只看一次v5



本项目描述了如何基于自己的数据集训练YOLO v5



但是YOLO v4的二作提供给我们的信息和官方提供的还是有一些出入：



## 0.环境配置

安装比较的python package和配置相关环境

```
# python3.6
# torch==1.3.0
# torchvision==0.4.1

# git clone yolo v5 repo
git clone https://github.com/ultralytics/yolov5 # clone repo
# 下载官方的样例数据（这一步可以省略）
python3 -c "from yolov5.utils.google_utils import gdrive_download;
gdrive_download('1n_oKgR81BJtqk75b00eAjdvd03qVCQn2f','coco128.zip')" # download dataset
cd yolov5
# 安装必要的package
pip3 install -U -r requirements.txt
```

## 1.创建数据集的配置文件dataset.yaml

[data/coco128.yaml](#)来自于COCO train2017数据集的前128个训练图像，可以基于该yaml修改自己数据集的yaml文件

```
# train and val datasets (image directory or *.txt file with image paths)
train: ../datasets/score/images/train/
val: ../datasets/score/images/val/

# number of classes
nc: 30

# class names
names: ['QP', 'NY', 'QG']
```

## 2.创建标注文件

可以使用LabelImg, Labme, Labelbox, CVAT来标注数据, 对于目标检测而言需要标注bounding box即可。然后将标注转换为和darknet format相同的标注形式, 每一个图像生成一个\*.txt的标注文件 (如果该图像没有标注目标则不用创建\*.txt文件)。创建的\*.txt文件遵循如下规则:

- 每一行存放一个标注类别
- 每一行的内容包括class x\_center y\_center width height
- Bounding box 的坐标信息是归一化之后的 (0-1)
- class label转化为index时计数是从0开始的

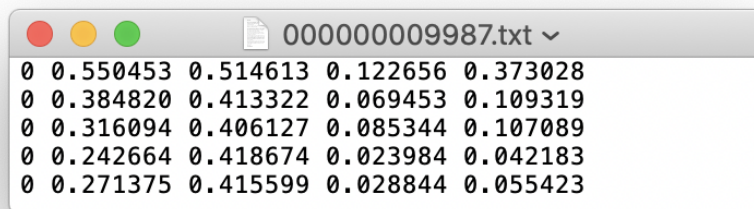
```
def convert(size, box):
    '''
    将标注的xml文件标注转换为darknet形的坐标
    '''
    dw = 1./(size[0])
    dh = 1./(size[1])
    x = (box[0] + box[1])/2.0 - 1
    y = (box[2] + box[3])/2.0 - 1
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x*dw
    w = w*dw
    y = y*dh
    h = h*dh
    return (x,y,w,h)
```

每一个标注\*.txt文件存放在和图像相似的文件目录下, 只需要将/images/\*.jpg替换为/labels/\*.txt即可 (这个在加载数据时代码内部的处理就是这样的, 可以自行修改为VOC的数据格式进行加载)

例如:

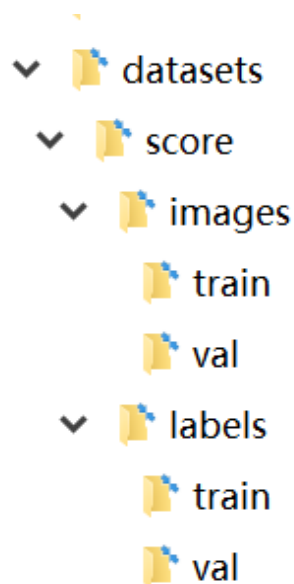
```
datasets/score/images/train/000000109622.jpg # image
datasets/score/labels/train/000000109622.txt # label
```

如果一个标注文件包含5个人物了别 (person在coco数据集中是排在第一的类别因此index为0):



### 3.组织训练集的目录

将训练集train和验证集val的images和labels文件夹按照如下方式进行存放



至此数据准备阶段已经完成，过程中我们假设算法工程师的数据清洗和数据集的划分过程已经自行完成。

### 4.选择模型backbone进行模型配置文件的修改

在项目的./models文件夹下选择一个需要训练的模型，这里我们选择yolov5x.yaml,最大的一个模型进行训练，参考官方README中的[table](#),了解不同模型的大小和推断速度。如果你选定了一个模型，那么需要修改模型对应的yaml文件

```
# parameters
nc: 3 # number of classes <----- UPDATE to match your dataset
depth_multiple: 1.33 # model depth multiple
width_multiple: 1.25 # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23] # P3/8
  - [30,61, 62,45, 59,119] # P4/16
  - [116,90, 156,198, 373,326] # P5/32

# yolov5 backbone
backbone:
  # [from, number, module, args]
```

```

[[-1, 1, Focus, [64, 3]], # 1-P1/2
[-1, 1, Conv, [128, 3, 2]], # 2-P2/4
[-1, 3, Bottleneck, [128]],
[-1, 1, Conv, [256, 3, 2]], # 4-P3/8
[-1, 9, BottleneckCSP, [256]],
[-1, 1, Conv, [512, 3, 2]], # 6-P4/16
[-1, 9, BottleneckCSP, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 8-P5/32
[-1, 1, SPP, [1024, [5, 9, 13]]],
[-1, 6, BottleneckCSP, [1024]], # 10
]

# yolov5 head
head:
[[-1, 3, BottleneckCSP, [1024, False]], # 11
[-1, 1, nn.Conv2d, [na * (nc + 5), 1, 1, 0]], # 12 (P5/32-large)

[-2, 1, nn.Upsample, [None, 2, 'nearest']],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 1, Conv, [512, 1, 1]],
[-1, 3, BottleneckCSP, [512, False]],
[-1, 1, nn.Conv2d, [na * (nc + 5), 1, 1, 0]], # 17 (P4/16-medium)

[-2, 1, nn.Upsample, [None, 2, 'nearest']],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 1, Conv, [256, 1, 1]],
[-1, 3, BottleneckCSP, [256, False]],
[-1, 1, nn.Conv2d, [na * (nc + 5), 1, 1, 0]], # 22 (P3/8-small)

[[], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
]

```

## 5.Train

```

# Train yolov5x on score for 300 epochs

$ python3 train.py --img-size 640 --batch-size 16 --epochs 300 --data
./data/score.yaml --cfg ./models/score/yolov5x.yaml --weights weights/yolov5x.pt

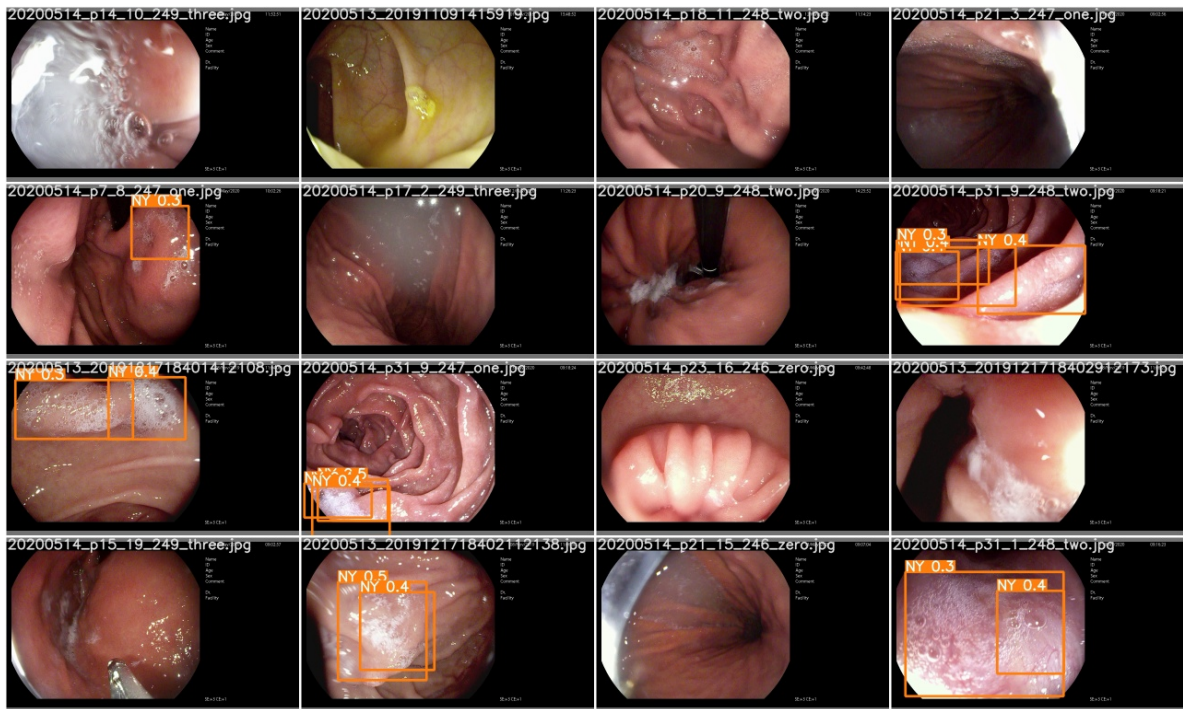
```

## 6.Visualize

开始训练后，查看train\*.jpg图片查看训练数据，标签和数据增强，如果你的图像显示标签或数据增强不正确，你应该查看你的数据集的构建过程是否有问题

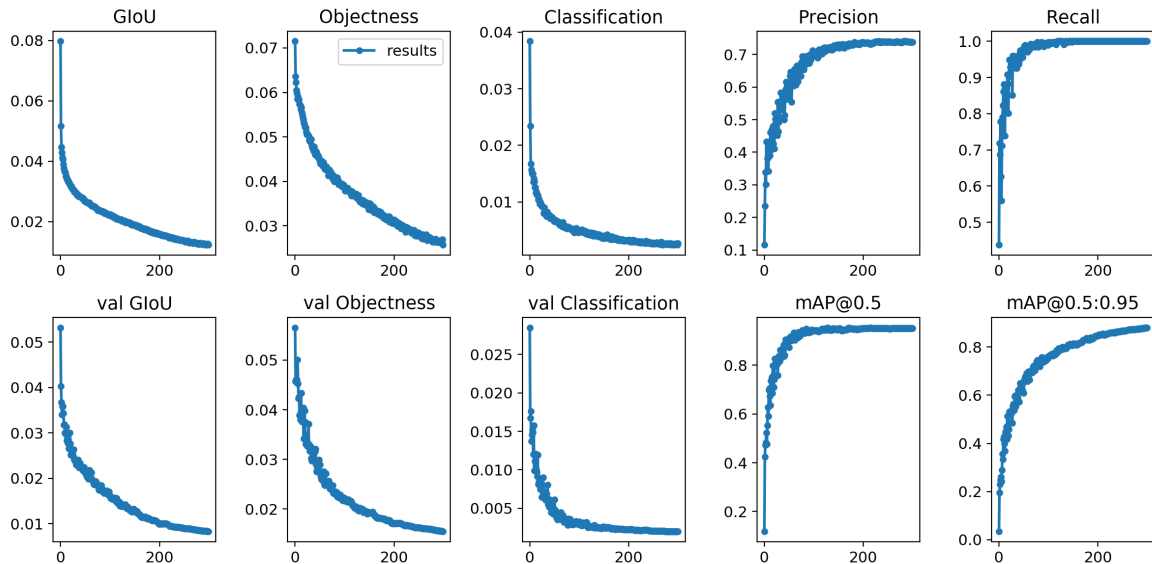






训练的losses和评价指标被保存在Tensorboard和results.txtlog文件。results.txt在训练结束后会被可视化成results.png

```
>>> from utils.utils import plot_results
>>> plot_results()
# 如果你是用远程连接请安装配置Xming:
https://blog.csdn.net/akuoma/article/details/82182913
```



## 7.推断

```
$ python3 detect.py --source file.jpg # image
                        file.mp4 # video
                        ./dir # directory
                        0 # webcam

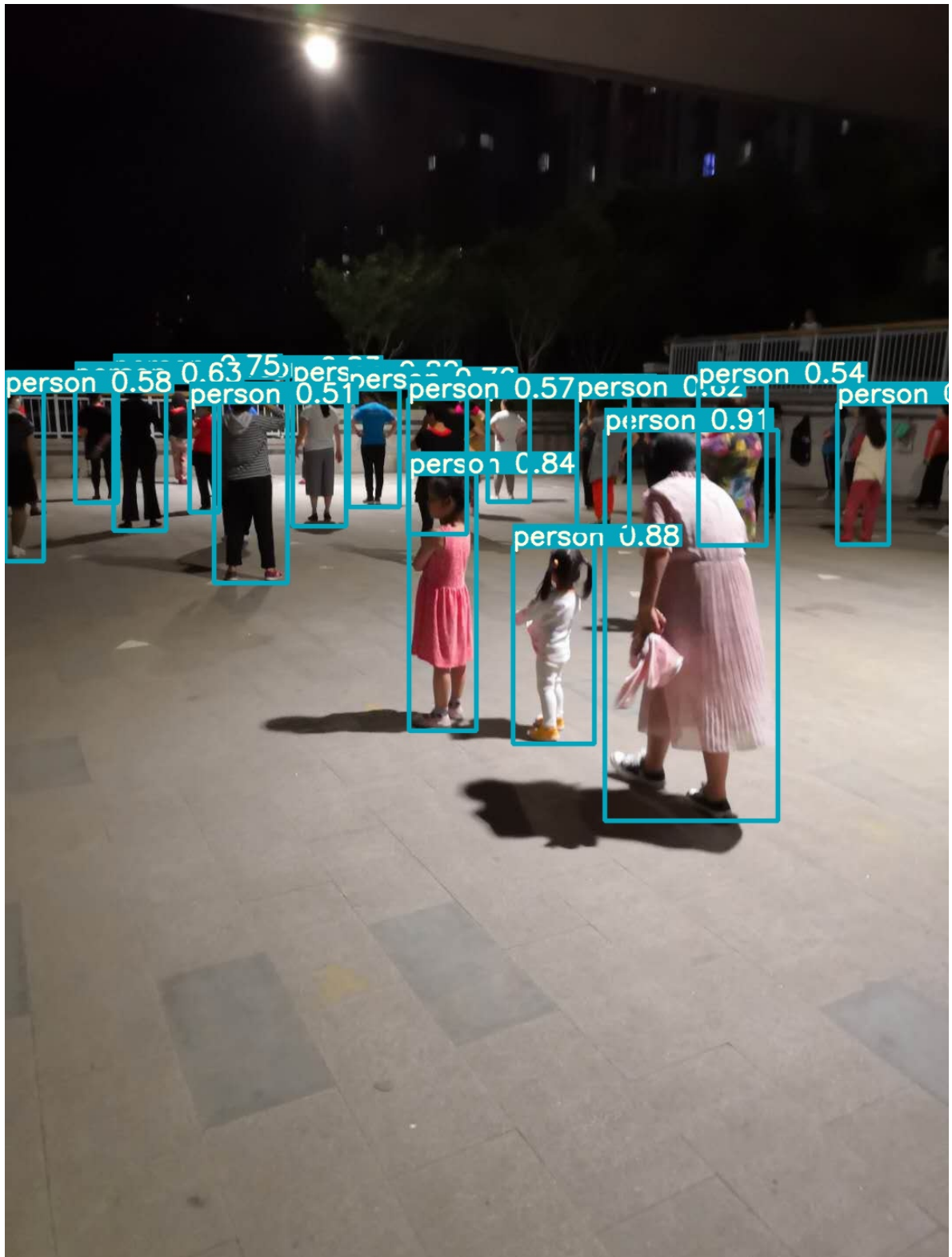
rtsp://170.93.143.139/rtplive/470011e600ef003a004ee33696235daa # rtsp stream
http://112.50.243.8/PLTV/88888888/224/3221225900/1.m3u8 #
http stream
```



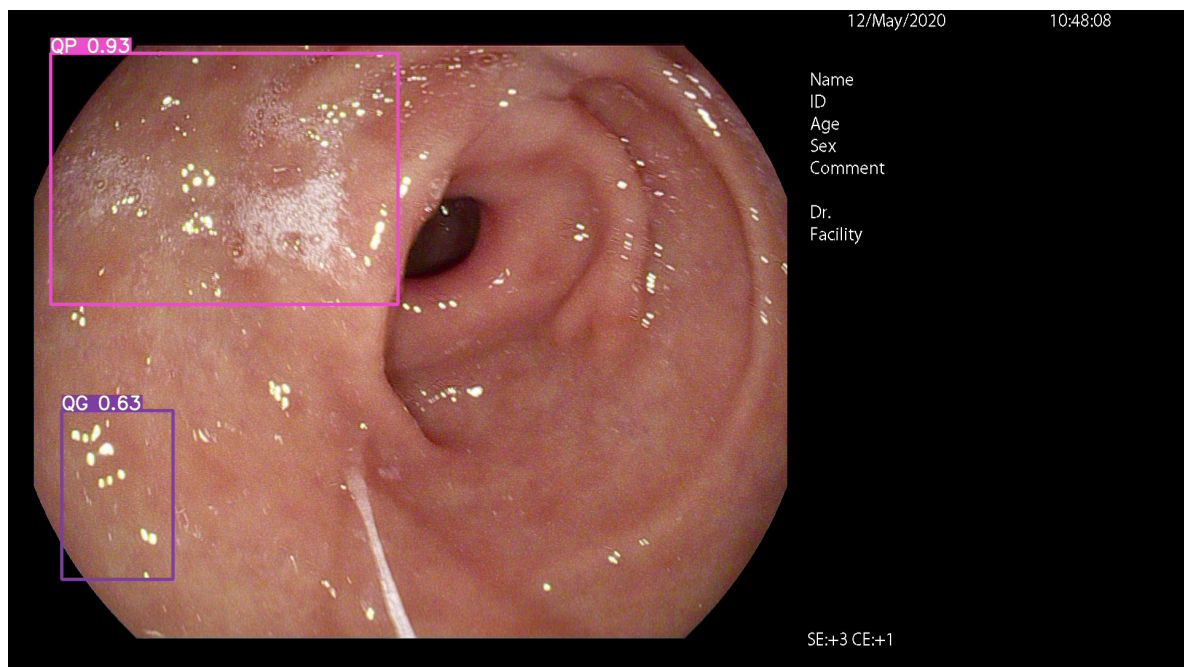
```
# inference /home/myuser/xujing/EfficientDet-Pytorch/dataset/test/ 文件夹下的图像
$ python3 detect.py --source /home/myuser/xujing/EfficientDet-Pytorch/dataset/test/ --weights weights/best.pt --conf 0.1

$ python3 detect.py --source ./inference/images/ --weights weights/yolov5x.pt --conf 0.5

# inference 视频
$ python3 detect.py --source test.mp4 --weights weights/yolov5x.pt --conf 0.4
```







## Reference

- [1].<https://github.com/ultralytics/yolov5>
- [2].<https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data>