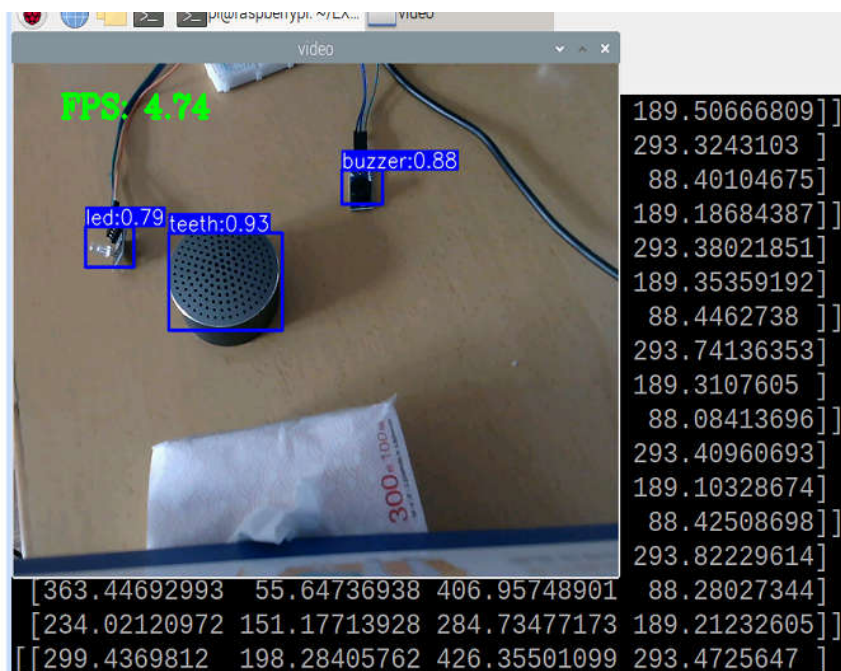


智能系统与控制

树莓派：YOLOV5-Lite 目标检测



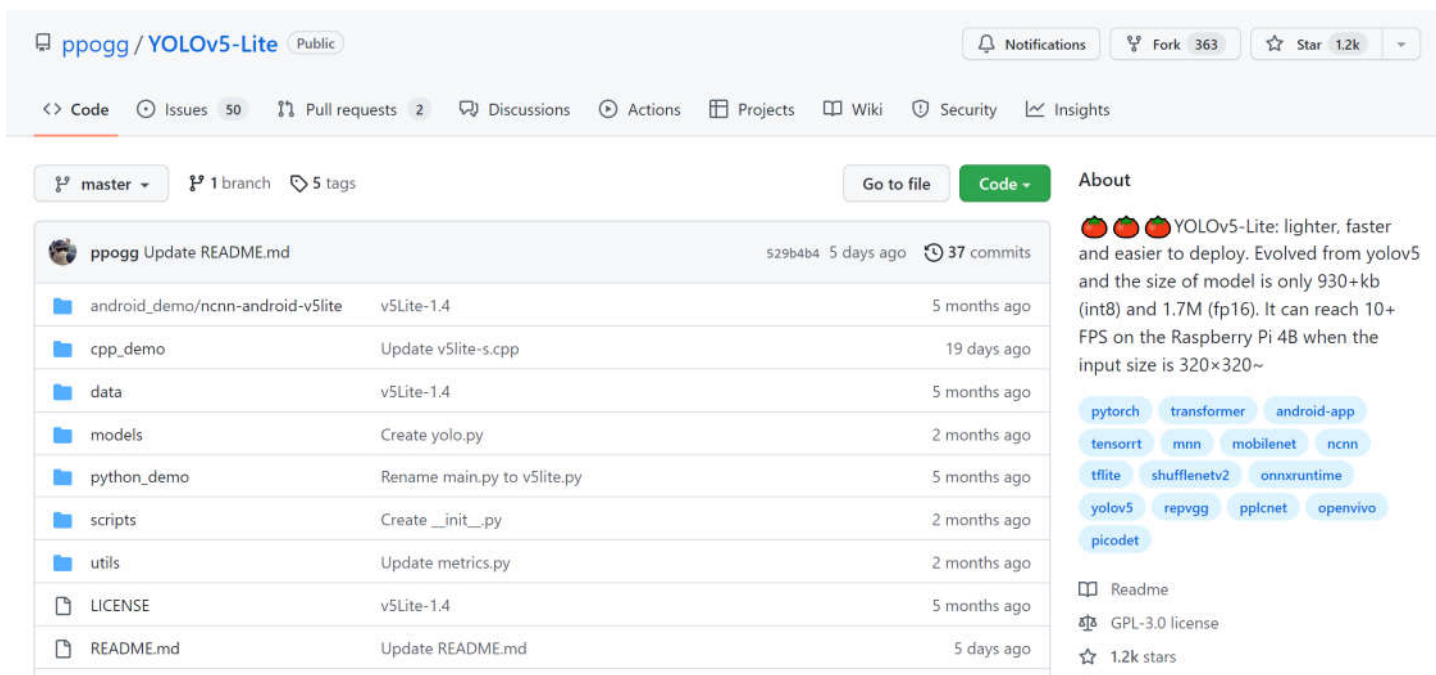
于泓

鲁东大学

信息与电气工程学院

2022.7.27

YoloV5-lite <https://github.com/ppogg/YOLOv5-Lite>



ppogg / YOLOv5-Lite Public

Notifications Fork 363 Star 1.2k

<> Code Issues 50 Pull requests 2 Discussions Actions Projects Wiki Security Insights

master 1 branch 5 tags

Go to file Code

ppogg Update README.md 529b4b4 5 days ago 37 commits

android_demo/ncnn-android-v5lite	v5Lite-1.4	5 months ago
cpp_demo	Update v5lite-s.cpp	19 days ago
data	v5Lite-1.4	5 months ago
models	Create yolo.py	2 months ago
python_demo	Rename main.py to v5lite.py	5 months ago
scripts	Create __init__.py	2 months ago
utils	Update metrics.py	2 months ago
LICENSE	v5Lite-1.4	5 months ago
README.md	Update README.md	5 days ago

About

🍅🍅🍅 YOLOv5-Lite: lighter, faster and easier to deploy. Evolved from yolov5 and the size of model is only 930+kb (int8) and 1.7M (fp16). It can reach 10+ FPS on the Raspberry Pi 4B when the input size is 320×320~

pytorch transformer android-app tensorrt mnn mobilenet ncnn tflite shufflenetv2 onnxruntime yolov5 repvgg pplcnet openvivo picodet

Readme GPL-3.0 license 1.2k stars

通过改变主干网络的结构
生成的轻量型模型

(1) 数据采集 与yolov5相同

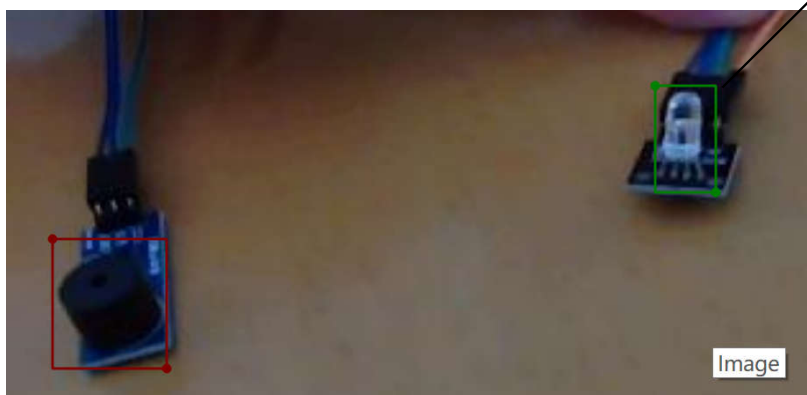
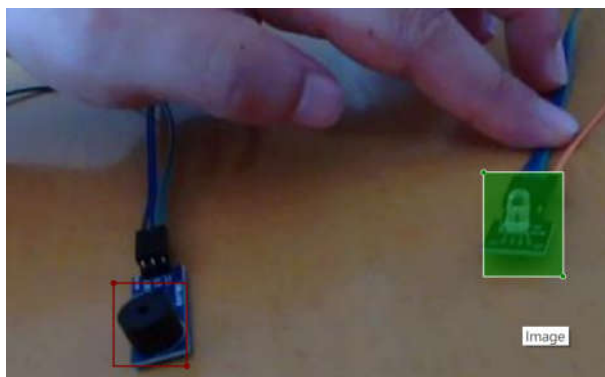
注意：

↑ > 此电脑 > Data (D:) > 工作相关 > 我设计的课程 > python与人工智能课程设计 > yolo > mydata > train >

名称	修改日期	类型
images	2022/7/11 20:24	文件夹
labels	2022/7/11 20:24	文件夹
labels.cache	2022/7/11 20:26	CACHE 文件

删除

标注时目标尽量贴合



(2) 预训练模型下载:

电脑 > Data (D:) > 工作相关 >

名称

argoverse_hd.yaml
coco.yaml
coco128.yaml
hyp.finetune.yaml
hyp.scratch.yaml
mydata.yaml
person.yaml
voc.yaml

```
# train and val data as 1) directory: path/images/, 2) fi
train: ../data_led_buzzer/train/images # 118287 images
val: ../data_led_buzzer/valid/images # 5000 images
test:
# number of classes
nc: 3

# class names
names: ['led', 'buzzer', 'teeth']
```

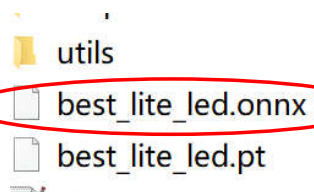
(3) 模型训练

```
python train.py --weights v5lite-s.pt
                  --cfg models/v5Lite-s.yaml
                  --img-size 320
                  --batch-size 16
                  --data data/mydata.yaml
                  --device cpu
```

结果存放: run/trian/exp/weights

(4) 模型导出 生成 onnx文件

```
python export.py --weights best_lite_led.pt
```

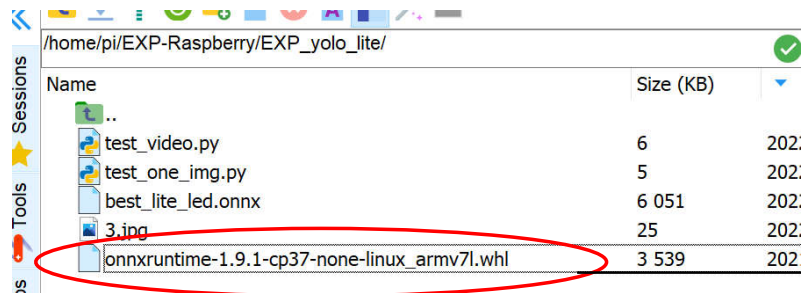


把onnx文件下载到树莓派中

(5) 在树莓派中安装 onnxruntime



Numpy版本1.21



pip3 install xxx.whl

```
if __name__ == "__main__":
```

```
    # 模型加载
```

```
    model_pb_path = "best_lite_led.onnx"
```

```
    so = ort.SessionOptions()
```

```
    net = ort.InferenceSession(model_pb_path, so)
```

```
    # 标签字典
```

```
    dic_labels= {0:'led',
                 1:'buzzer',
                 2:'teeth'}
```

```
    # 模型参数
```

```
    model_h = 320
```

```
    model_w = 320
```

```
    nl = 3
```

```
    na = 3
```

```
    stride=[8.,16.,32.]
    anchors = [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]]
    anchor_grid = np.asarray(anchors, dtype=np.float32).reshape(nl, -1, 2)
```

三层输出

每层3种锚框

每层缩放尺度

```
    # 进行推理
```

```
    img0 = cv2.imread('3.jpg')
```

```
    t1 = time.time()
```

```
    det_boxes,scores,ids = infer_img(img0,net,model_h,model_w,nl,na,stride,anchor_grid,thred_nms=0.4,thred_cond=0.5)
```

```
    t2 = time.time()
```

```
    print("%.2f"%(t2-t1))
```

```
    # 结果绘图
```

```
    for box,score,id in zip(det_boxes,scores,ids):
        label = '%s:%.2f'%(dic_labels[id],score)
```

```
        plot_one_box(box.astype(np.int), img0, color=(255,0,0), label=label, line_thickness=None)
```

```
    cv2.imshow('img',img0)
```

```
    cv2.waitKey(0)
```

```
import cv2
import numpy as np
import onnxruntime as ort
import math
import time
```

```
def infer_img(img0,net,model_h,model_w,nl,na,stride,anchor_grid,thred_nms=0.4,thred_cond=0.5):  
    # 图像预处理  
    img = cv2.resize(img0, [model_w,model_h], interpolation=cv2.INTER_AREA)  
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
    img = img.astype(np.float32) / 255.0  
    blob = np.expand_dims(np.transpose(img, (2, 0, 1)), axis=0)  
  
    # 模型推理  
    outs = net.run(None, {net.get_inputs()[0].name: blob})[0].squeeze(axis=0)  
  
    # 输出坐标矫正  
    outs = cal_outputs(outs,nl,na,model_w,model_h,anchor_grid,stride)  
  
    # 检测框计算  
    img_h,img_w,_ = np.shape(img0)  
    boxes,confs,ids = post_process_opencv(outs,model_h,model_w,img_h,img_w,thred_nms,thred_cond)  
  
    return boxes,confs,ids
```

```
def cal_outputs(outs,nl,na,model_w,model_h,anchor_grid,stride):  
  
    row_ind = 0  
    grid = [np.zeros(1)] * nl  
    for i in range(nl):  
        h, w = int(model_w/ stride[i]), int(model_h / stride[i])  
        length = int(na * h * w)  
        if grid[i].shape[2:4] != (h, w):  
            grid[i] = _make_grid(w, h)  
  
        outs[row_ind:row_ind + length, 0:2] = (outs[row_ind:row_ind + length, 0:2] * 2. - 0.5 + np.tile(  
            grid[i], (na, 1))) * int(stride[i])  
        outs[row_ind:row_ind + length, 2:4] = (outs[row_ind:row_ind + length, 2:4] * 2) ** 2 * np.repeat(  
            anchor_grid[i], h * w, axis=0)  
        row_ind += length  
    return outs  
  
def _make_grid( nx, ny):  
    xv, yv = np.meshgrid(np.arange(ny), np.arange(nx))  
    return np.stack((xv, yv), 2).reshape((-1, 2)).astype(np.float32)
```



```
def post_process_opencv(outputs,model_h,model_w,img_h,img_w,thred_nms,thred_cond):
    conf = outputs[:,4].tolist()
    c_x = outputs[:,0]/model_w*img_w
    c_y = outputs[:,1]/model_h*img_h
    w = outputs[:,2]/model_w*img_w
    h = outputs[:,2]/model_h*img_h
    p_cls = outputs[:,5:]
    if len(p_cls.shape)==1:
        p_cls = np.expand_dims(p_cls,1)
    cls_id = np.argmax(p_cls,axis=1)

    p_x1 = np.expand_dims(c_x-w/2,-1)
    p_y1 = np.expand_dims(c_y-h/2,-1)
    p_x2 = np.expand_dims(c_x+w/2,-1)
    p_y2 = np.expand_dims(c_y+h/2,-1)
    areas = np.concatenate((p_x1,p_y1,p_x2,p_y2),axis=-1)

    areas = areas.tolist()
    ids = cv2.dnn.NMSBoxes(areas,conf,thred_cond,thred_nms)
    return np.array(areas)[ids],np.array(conf)[ids],cls_id[ids]
```