AGX Orin搭建 yolov5

1. yolov5是基于pytorch实现的一套物体目标检测的案例，可以做简单的训练和推理，在Jetson平台跑起来需要CUDA，CuDnn，pytorch的支持
2. 在ARM机器上安装组件环境, 安装环境 CUAD-11.4, cuDNN-8.6

**apt update; apt install cuda-11-4 libfreeimage-dev libcudnn8 libcudnn8-dev libcudnn8-samples tensorrt tensorrt-dev tensorrt-libs python3-libnvinfer python3-libnvinfer-dev uff-converter-tf onnx-graphsurgeon graphsurgeon-tf deepstream-6.3 python3-pip libopenblas-dev ffmpeg libavutil-dev libavcodec-dev libavformat-dev libavdevice-dev libavfilter-dev libswscale-dev libswresample-dev libswresample-dev libpostproc-dev libjpeg-dev libpng-dev**

1. 安装python环境

python3 -m pip install aiohttp opencv-python jetson-stats scipy=='1.5.3' -i <https://pypi.tuna.tsinghua.edu.cn/simple>

1. 安装torch和torchvision

pip3 install ./torch-2.1.0a0+41361538.nv23.06-cp38-cp38-linux\_aarch64.whl -i <https://pypi.tuna.tsinghua.edu.cn/simple>

pip3 install ./torchvision-0.16.0-cp38-cp38-linux\_aarch64.whl -i https://pypi.tuna.tsinghua.edu.cn/simple

备注：关于pytorch包放在百度网盘

链接: <https://pan.baidu.com/s/1Xu4b6JEz52KjfLAHplsvJQ> 提取码: 89w4

1. 验证torch torchvision是否安装成功

import torch

>>> print(torch.\_\_version\_\_)

>>> print('CUDA available: ' + str(torch.cuda.is\_available()))

>>> print('cuDNN version: ' + str(torch.backends.cudnn.version()))

>>> a = torch.cuda.FloatTensor(2).zero\_()

>>> print('Tensor a = ' + str(a))

>>> b = torch.randn(2).cuda()

>>> print('Tensor b = ' + str(b))

>>> c = a + b

>>> print('Tensor c = ' + str(c))

>>> import torchvision

>>> print(torchvision.\_\_version\_\_)

1. 下载yolov5网络模型:

git clone <https://github.com/ultralytics/yolov5.git>

安装yolov5运行依赖的环境:

pip3 install gitpython==3.1.30 numpy==1.22.2 setuptools==65.5.1 -i <https://pypi.tuna.tsinghua.edu.cn/simple>

pip3 install onnx onnxruntime opencv-python ultralytics -i <https://pypi.tuna.tsinghua.edu.cn/simple>

备注：由于git clone下载不下来的问题，yolov5代码放在百度网盘

链接: <https://pan.baidu.com/s/1Xu4b6JEz52KjfLAHplsvJQ> 提取码: 89w4

1. 下载模型：

wget https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt

【备注：yolov5s.pt, yolov5m.pt, yolov5l.pt,yolov5x.pt, yolov5n.pt是YOLOV5的不同变体,表示不同大小和复杂性的模型】

【 yolov5s-seg.pt, yolov5m-seg.pt表示实例分割是指将图片中属于物体类别的像素识别出来并作分类】

1. 运行目标检测案例

# python3 detect.py --weights yolov5s.pt //利用yolov5s.pt模型进行目标检测

【运行结果如下, 生成文件在runs/detect】

9.自定义训练数据集

a. 下载数据集图片素材，然后通过labelImg或labelme工具进行不分类标注, 使用标注工具标记图像后，将标签导出为YOLO格式，

每张\*.txt图像一个文件（如果图像中没有对象，则不需要\*.txt文件）

生成images/和labels/:

--> imaegs/:里面存放素材图片

--> labels/:里面存放标注的\*.txt文件，

--> 文件\*.txt规格,包含class x\_center y\_center width height(类 标注x坐标 标注y坐标 宽度 高度)

b. 准备custom.yaml,放置yolov5/data/目录下: yaml里面指定YOLOv5的images,labels目录位置以及有关我们的自定义类的信息

Example(data/custom.yaml):

path: ../datasets/hard-hat

train: train/images

val: valid/images

test: test/images

nc: 3

names: ['head', 'helmet', 'person']

【备注：另外一种方法直接在Roboflow网站上下载标注好的数据集，里面有已经生成好的images和labels，data.yaml(重命名custom.yaml)】

c. 接下来为自定义对象检测器编写模型配置文件,选择了最小、最快的 YOLOv5 基础模型-yolov5s.pt

cp yolov5/models/yolov5s.yaml yolov5/models/custom.yaml //拷贝标准网络结构，来设置自定义的网络结构

编辑网络的结构, 修改custom.yaml里面的【nc:】, 和data/custom.yaml的nc一致, 适配自定义数据集

d. 准备好以上文件后就可以开始训练

python3 train.py --data=data/custom.yaml --img 640 --epochs 300 --cfg=models/custom-yolov5s.yaml --weights weights/yolov5s.pt

--data 设置我们数据集yaml文件的路径

--img 定义输入图像大小

--epochs 定义训练时期的数量

--cfg 指定我们的模型yaml配置路径

--weights 指定权重的自定义路径

--name 结果名称

--nosave 只保存最后的检查点

--cache ram or disk 缓存图像以加快训练速度,需要大量 RAM/磁盘资源