嵌入式 ARM 无人机的 Deep-SORT-yolov5 识别与追踪

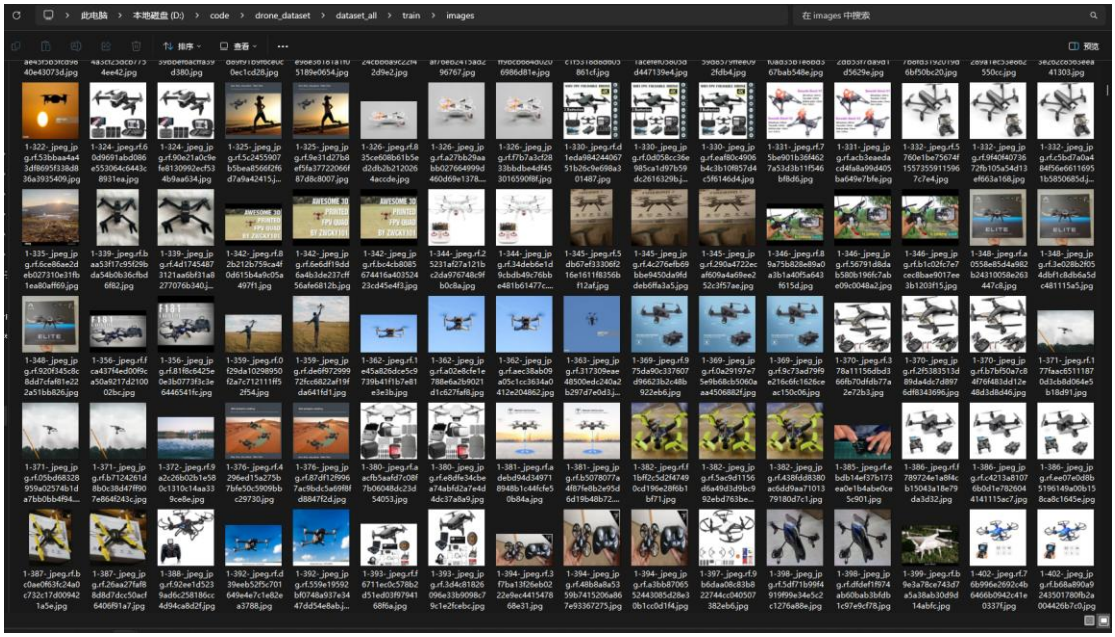
该项目以**“嵌入式 ARM 无人机 Deep-SORT-YOLOv5 识别与追踪”**为主题，完成了从数据集准备、模型训练，到算法融合及嵌入式部署，期望实现对无人机目标的实时检测与多目标追踪。

技术层面	具体组件 / 工具	作用
深度学习框架	PyTorch + CUDA、Tensorflow	GPU 训练 YOLOv5 模型
目标检测	**YOLOv5s**（以 COCO 权重为起点，迁移学习）	训练定制化“无人机”检测器
多目标追踪	**Deep-SORT**	利用卡尔曼滤波 & IoU 关联，实现 ID 一致性追踪
数据集管理	**Roboflow Universe**	在线检索并下载 Drone 数据集；配合 data.yaml 重映射标签
计算机视觉工具	OpenCV、NumPy、Pandas	图像/视频处理与结果分析
嵌入式硬件	**Orange Pi 3B （Rockchip RK3566）**	部署推理；Ubuntu arm64 系统
系统与包管理	Ubuntu 镜像烧录、pip + 清华源	ARM 环境下依赖安装与加速

下面从几个方面介绍工作内容：

基于数据集训练了关于无人机的训练模型

通常情况下，要靠自己收集成百上千张含有无人机的图像是非常困难的，于是我们需要在网上搜取用于训练的大量图片。而 Roboflow 是一个开源的数据集平台，平台上有数据集。登录网址：https://universe.roboflow.com/
可以看到这个平台上的一些模型，搜索关键词“drones”，可以看到一些和无人机相关的数据集。并可以将他们下载到本地。



修改 data.yaml 中的训练集、验证集、路径，让它符合我们本地的文件路径。

labels 中的 txt 文件包含的信息对应：类别 id、锚框 X 轴中心坐标、锚框 Y 轴中心坐标、锚框长度、锚框宽度。

data.yaml - 记事本

```
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
train: D:/code/drone_dataset/dataset_all/train/images
val: D:/code/drone_dataset/dataset_all/valid/images
test: D:/code/drone_dataset/dataset_all/test/images

nc: 1
names: ['Drones']

roboflow:
  workspace: cranfieldyolo1
  project: droneyolo5
  version: 3
  license: CC BY 4.0
  url: https://universe.roboflow.com/cranfieldyolo1/droneyolo5/dataset/3
```

0001_jpg.rf.7208ffef35d568dc0584fda37ae79ebe.txt - 记事本

```
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
0 0.5140625 0.29765625 0.8921875 0.5953125
0 0.08515625 0.50390625 0.1703125 0.22578125
0 0.30390625 0.80234375 0.43203125 0.3515625
0 0.73671875 0.80546875 0.1453125 0.1796875
0 0.9046875 0.43515625 0.190625 0.1546875
```

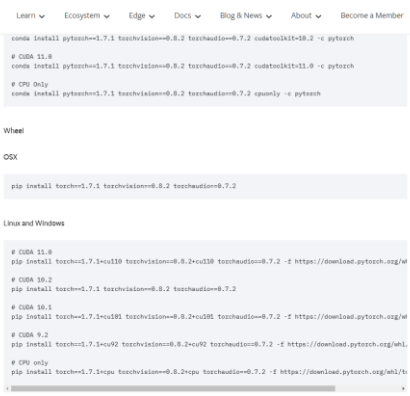
Roboflow平台数据集的文件格式：

```
--dataset
{----data.yaml
|----README.txt
|----README.roboflow.txt
|----test
|----{----images
|----|----{----1_21_jpeg_jpg.rf.90532eb47f1b01ad4b3d31822afdcfa0.jpg
|----|----|----.....
|----|----}
|----|----labels
|----|----{----1_21_jpeg_jpg.rf.90532eb47f1b01ad4b3d31822afdcfa0.txt
|----|----|----.....
|----|----}
|----}
|----train
|----{----images
|----|----{----1_0_jpeg_jpg.rf.383172111dac9ecb00b81cea26512a91.jpg
|----|----|----.....
|----|----}
|----|----labels
|----|----{----1_0_jpeg_jpg.rf.383172111dac9ecb00b81cea26512a91.txt
|----|----|----.....
|----|----}
|----}
|----valid
|----{----images
|----|----{----1_21_jpeg_jpg.rf.90532eb47f1b01ad4b3d31822afdcfa0.jpg
|----|----|----.....
|----|----}
|----|----labels
|----|----{----1_21_jpeg_jpg.rf.90532eb47f1b01ad4b3d31822afdcfa0.txt
|----|----|----.....
|----|----}
|----}
}
```

在服务器上安装 CUDA，使得 python 编译环境能够调用显卡资源进行训练，当前我的 CUDA 版本为 10.1

原本的 torch 是 CPU 版本，需要将其替换为 GPU 版本。下载地址：https://download.pytorch.org/whl/torch_stable.html

对应版本 torch1.7.1+cu101（文件较大，1.2GB）、torchvision0.8.2+cu101、torchaudio0.7.2



训练指令：

python train.py --img 640 --batch 24 --epochs 15 --data D:\code\drone_dataset\dataset_all\data.yaml --cfg yolov5s.yaml --weights yolov5s.pt --name custom_model05

img：图片统一为 640*640

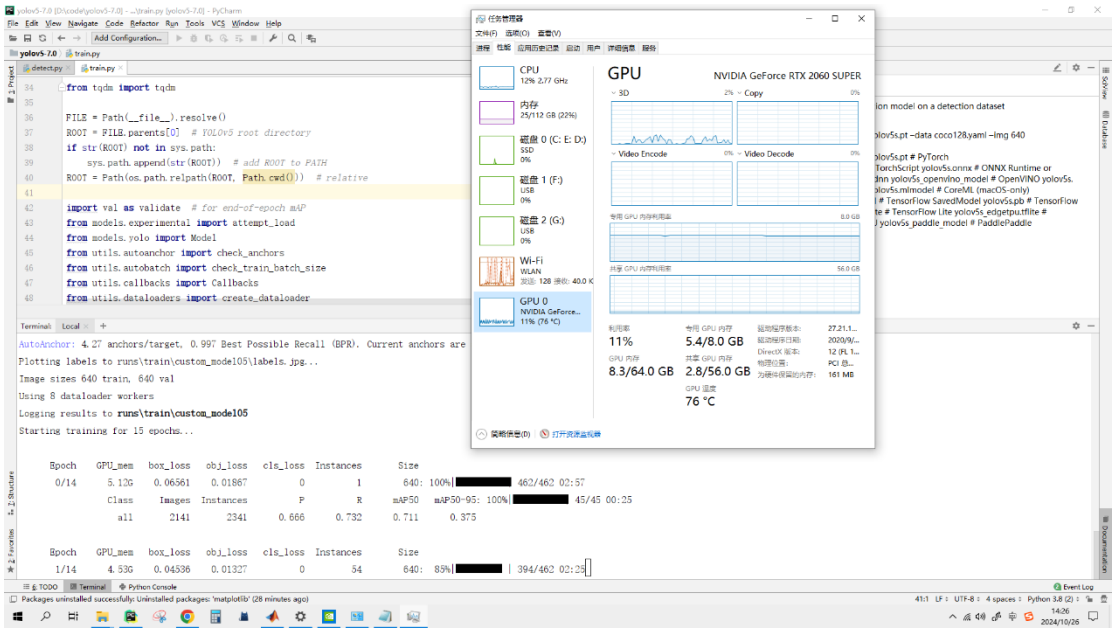
batch：每个批次采用训练集的 24 张图片

epochs：训练将完整地遍历训练集 15 次

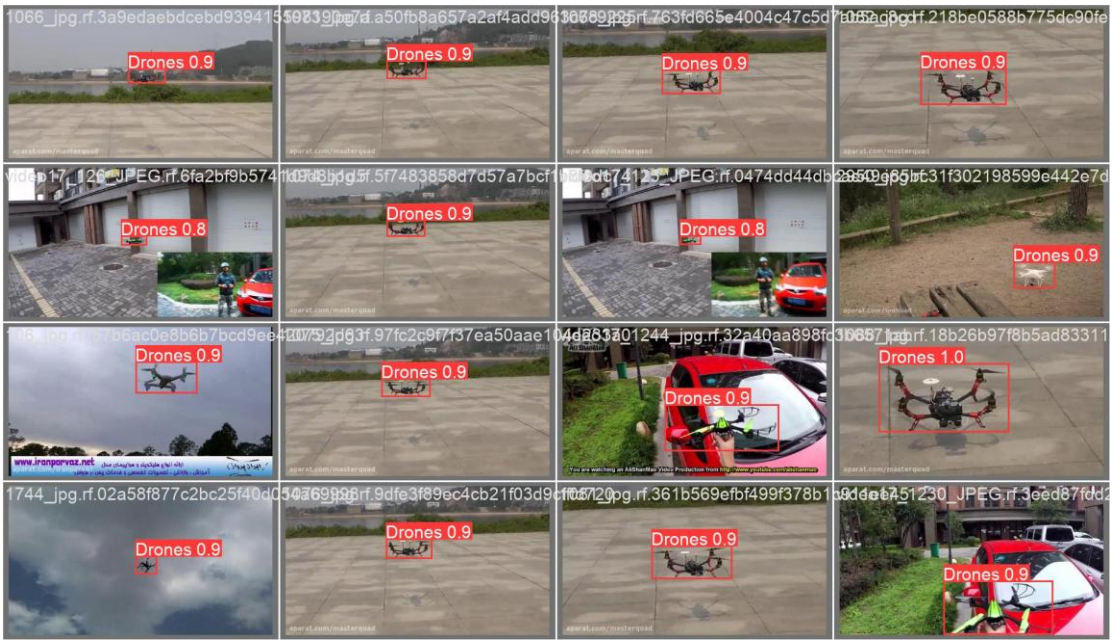
data：指定数据集的 yaml 文件

weights：初始权重文件，采用 COCO 数据集训练的 yolov5s.pt

name：模型名称



训练完毕后，通过 custom_model05.pt 替换掉原 yolov5 默认的模型，就可以针对性的识别无人机了。



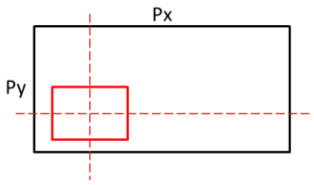
将 Deep-SORT 与 Yolov5 结合

Deep-SORT 是一个追踪算法，在画面中包含多个目标时，根据上一帧锚框的位置运动趋势，使用卡尔曼滤波器预测本帧位置，并将预测与本帧锚框根据 IoU 进行匹配，最终实现的效果是对每一帧的每个锚框打上 id，实现对其追踪。开源地址：https://github.com/nwojke/deep_sort

该项目的缺点是，其识别的功能是缺失的。该程序只是读取一个 txt 文件里边的数字表示锚框的横坐标、纵坐标、长、宽、置信度等参数，这些参数对应着固定的图像，并没有实现对视频源的实时检测。

为了实现对视频源的实时检测，我将 Deep-SORT 项目与 yolov5 相结合，改写了 Deep-SORT 中的 def create_detections()函数，在其中调用 yolov5 模块对图像进行检测，替换掉原来直接读取本地 txt 文件的形式。

由于 Deep-SORT 的坐标采用 MOT 格式，与 yolo 含义不相同，因此需要人为手动进行转化：



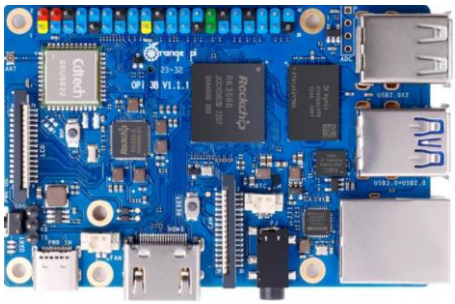
YOLO 格式：(class_id, Center_X, Center_Y, Width, Height)
MOT 格式：(frame_idx, track_id, x, y, width, height)

x=Px*Center_X-Px*Width/2
y=Py*Center_Y-Py*Height/2
width=Px*Width
height=Py*Height

实际运行效果：


在开发板上的部署

开发板 OrangePi_3B, Rockchip RK3566



通过工具将 ubuntu 镜像烧入 TF 卡，插在开发板 TF 卡槽上得到显示输出：

配置 pip 使用清华镜像网站：

```
# 建立配置文件
mkdir -p ~/.pip
cat > ~/.pip/pip.conf <<'EOF'
[global]
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
[install]
trusted-host = pypi.tuna.tsinghua.edu.cn
EOF
```

将 pip 版本更新到最新：sudo pip3 install --upgrade pip

随后指定更新后的 pip 安装依赖包，注意这里会自动替换为 arm 版本的包：

```
/usr/local/bin/pip3 install numpy
/usr/local/bin/pip3 install opencv-python
/usr/local/bin/pip3 install tensorflow
/usr/local/bin/pip3 install pandas
/usr/local/bin/pip3 install torchvision
……
```

cd 到 yolo 工程文件夹下，执行（模型的权重文件已经被替换，如果没有需要手动指定自定义模型）：

```
python3 detect.py –source .data/images/screenshot_6.png
```

这样就可以在开发板上实现 yolov5 对图像的识别：