西安电子科技大学

大 学 生 创 新 训 练 项 目

题　　目	基于无人机平台的校园违停车辆自动识别系统
学　　院	计算机科学与技术学院
专　　业	软件工程
学生姓名	黄小可 石虎 许文强
导师姓名	方厚章

目录

一、 项目背景

在小区中，大部分业主是比较自觉的，能够在指定区域内停车，但仍有部分业主不遵守规则，在指定区域外停车，阻塞了小区道路，让其他的车辆无法正常通行，给大家的生活带来了困扰，如果阻塞了消防通道，甚至会导致人员伤亡。

随着居民私家车拥有量急剧增加，小区内乱停车现象变得普遍，因此我们亟需一种方便快捷的方法检测车辆违停的情况。

检测违章停车的传统方法是人来进行：保安盯着监控视频，判断视频中是否出现违章车辆。这种做法有两个问题：

1. 这种方式是低效率的，需要人一直盯着。
2. 对于大型小区，监控摄像头可能覆盖面不够全面，摄像头外的区域无法检测到。
3. 对于老旧小区，几乎没有监控，这种方法无从谈起。

为了解决这几个问题，我们想到了使用无人机进行违章车辆的自动检测，使用价格低廉的无人机便可以全面检测到监控摄像头无法覆盖的区域，并且，配置深度学习算法使得违章车辆的检测自动化，节省大量人力且可长时间保证检测效率和准确率。

二、 项目简介

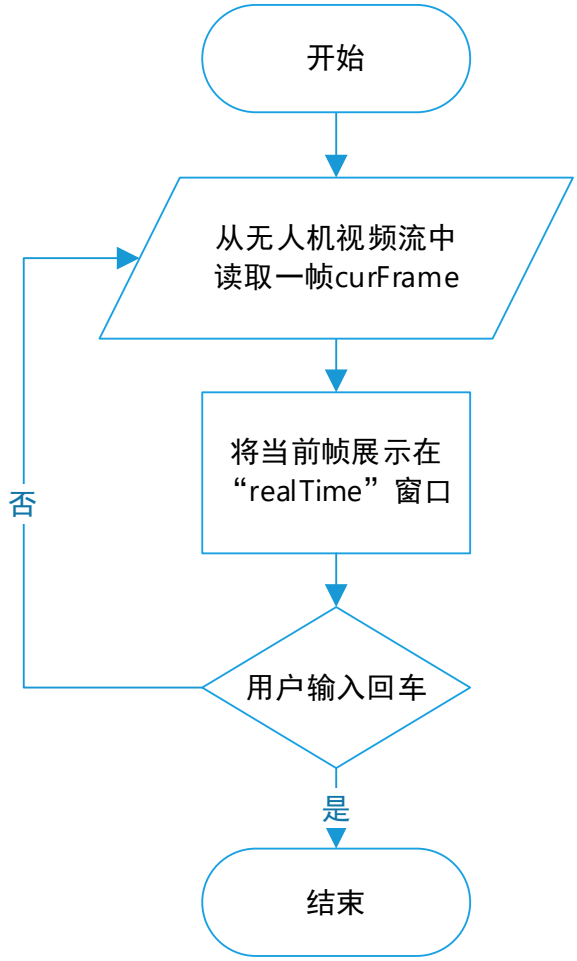
无人机定时巡逻，将视频流传回小区监控室，监视室的计算机通过 opencv 读取视频流，使用 yolov3 模型检测图像中的车辆，并通过 resnet18 网络训练出的二分类模型判断车辆是否在停车区之外，将违章的车辆使用 opencv 框出并展示出来，以达到监视和警示的作用。

三、 系统流程

1. 将无人机飞到指定高度，然后围绕小区移动
2. 无人机将实时采集到的视频流传回计算机
3. 计算机程序的主线程将无人机实时视频流展示在“realTime”窗口中
4. 线程 2 每隔一定间隔取一帧，进行下述操作
    a) 检测图片中的所有车辆
    b) 对检测到的每辆车判断是否在停车区内
        i. 如果在，用蓝框将车辆框出
        ii. 如果不在，用红框将车辆框出，作为警示
    c) 将处理结果展示在“output”窗口中
5. 用户按回车，结束程序

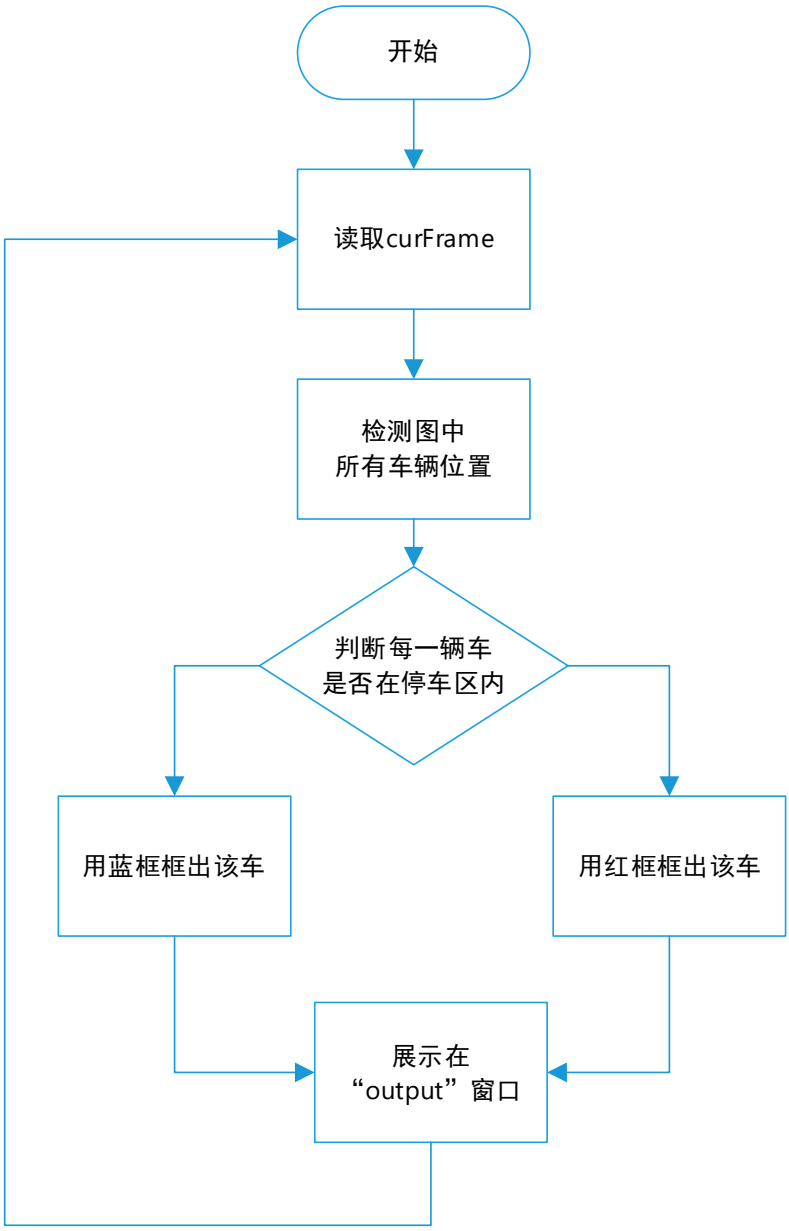
主线程流程图

主线程用于展示实时视频流



线程 2 流程图

线程 2 用于处理图片（车辆检测和违章判断），展示处理后的结果。



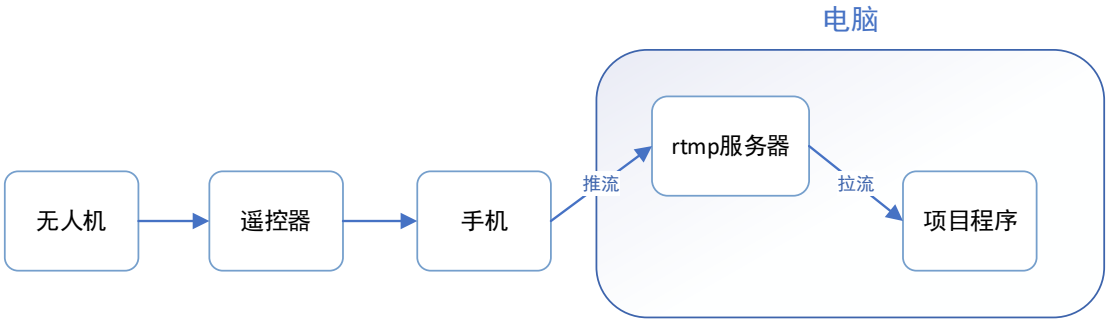
四、 系统功能模块

本系统包含视频回传、车辆检测、违章检测三大模块，下面一一介绍。

视频回传

将无人机拍摄的视频流传给电脑，具体流程如下

1. 搭建 rtmp 服务器
    1. 下载包含 rtmp 模块的 nginx
    2. 修改 nginx.conf，创建 rtmp 服务，使其监听 1935 端口
    3. 启动 nginx
2. 推流
    1. 下载“dji go 4”app
    2. 启动遥控器与无人机，等待他们连接
    3. 用数据线连接遥控器与手机
    4. 将手机与电脑置于同一无线局域网中（例如用手机为电脑开热点）
    5. 进入 dji go 4 的飞行界面，在“直播”设置中设置推流至电脑上的 nginx 服务器
3. 拉流
    1. 下载 OpenCV 源代码并编译
    2. 使用 OpenCV 读取 rtmp 服务器的视频流
    3. 成功从视频流中读取每一帧图像

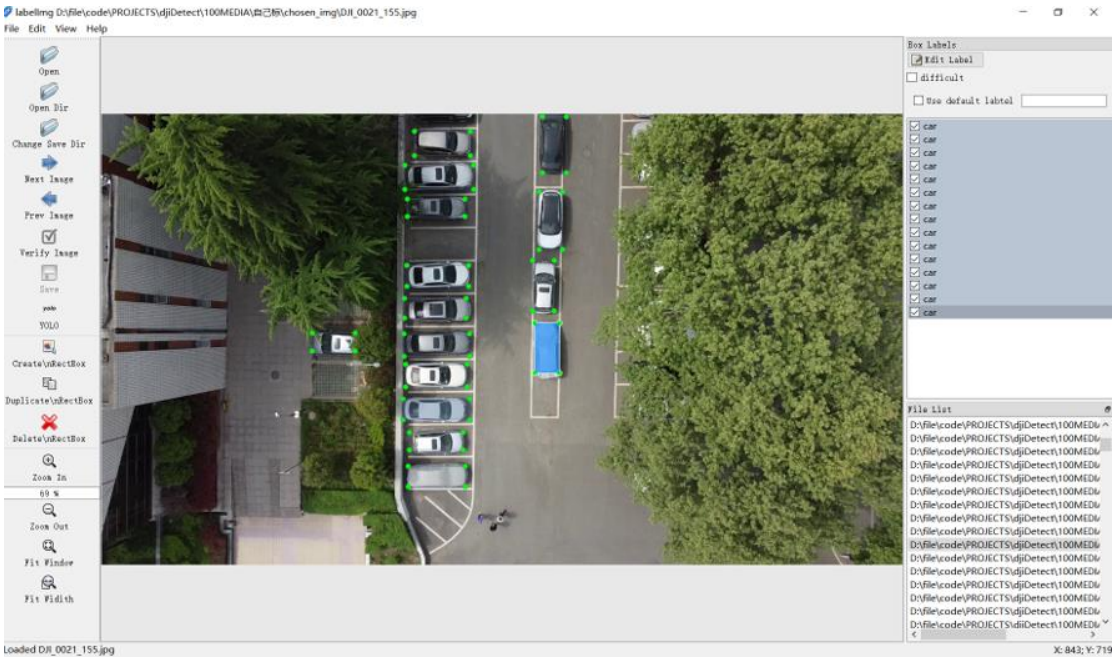


车辆检测

使用 yolov3 目标检测算法，训练车辆检测的模型

数据标记

1. 将无人机采集到的视频流转换为一帧帧图像，从中随机选取了 250 张图片（包含 4848 辆车）
2. 使用 labelImg 软件标记图片中的车辆（YOLO 格式）

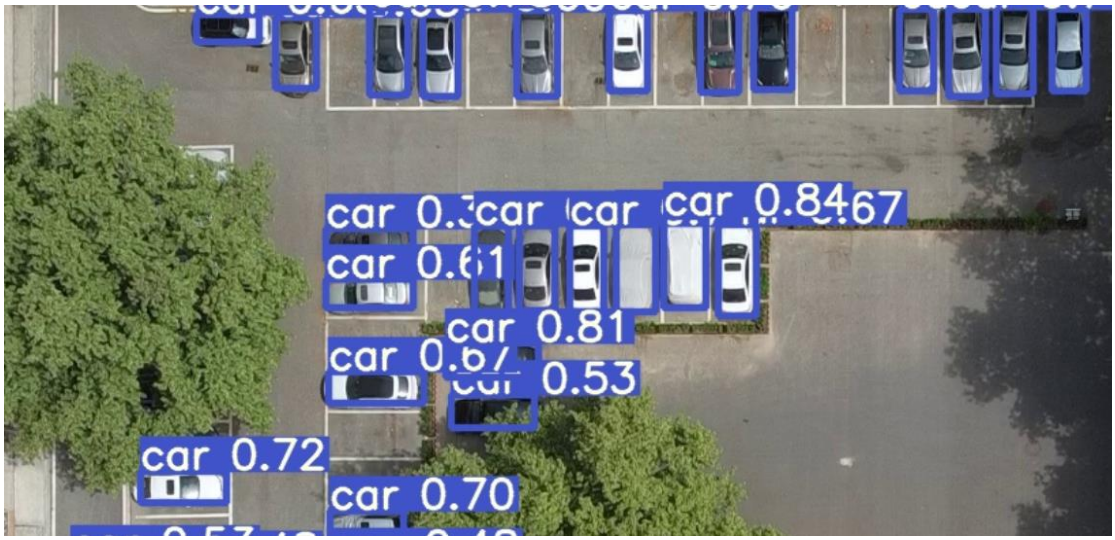


3. 编写 python 脚本，将数据集以及对应的标记文件分为训练集、验证集、测试集三部分，并放在指定文件夹中

模型训练

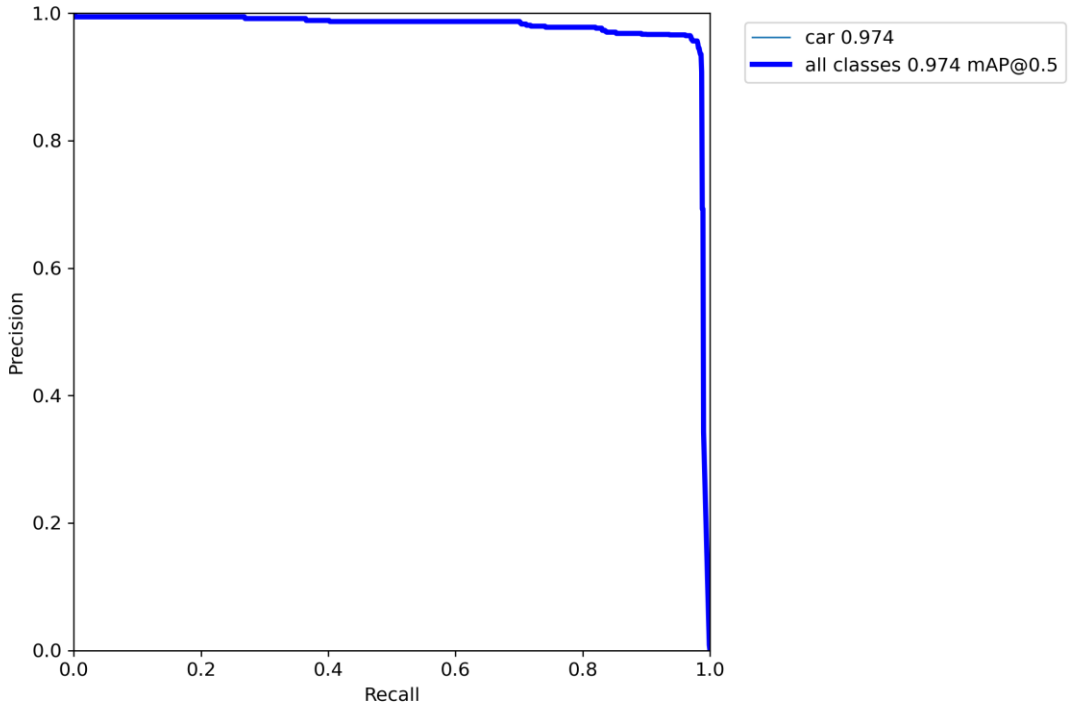
1. 在 kaggle 上新建一个 jupyter 笔记本，用于训练，并打开“网络”和“GPU”的按钮
2. 上传数据集
3. clone yolov3 项目
4. 设置超参数
   a) img_size：640
   b) batch-size：16
   c) epochs：150
5. 设置预训练权重为 yolov3.pt
6. 编写 car.yaml 配置文件用于读取数据集
7. 开始训练
8. 等待几个小时后，训练完成，将训练好的模型（weights/best.pt）下载到本地

模型测试



测试模型在测试集上的效果，评估指标如下表

Images	Labels	P	R	mAP@.5
25	650	0.956	0.98	0.974



在 batch size 为 32 的情况下，对一张 640x640 的图片的平均检测速度如下表

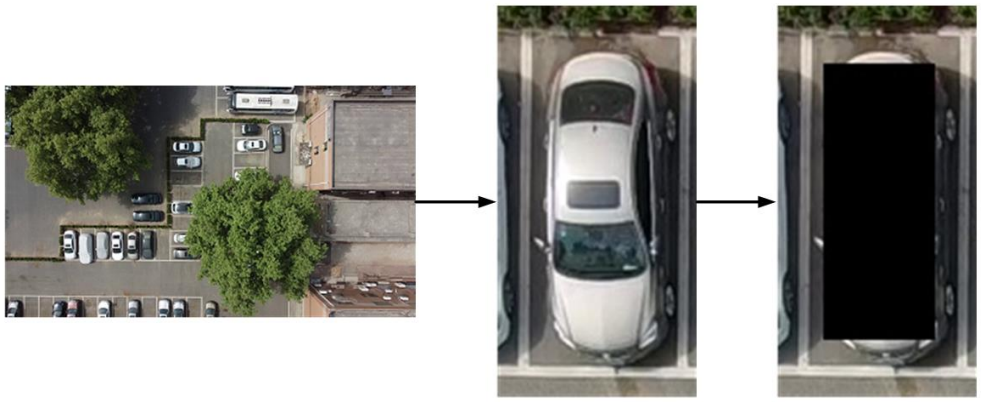
inference	NMS	total
659.1ms	1.3ms	660.4ms

违章检测

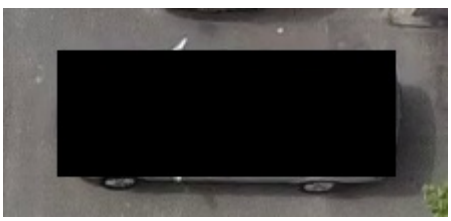
通过对车辆进行二分类（违章车辆和非违章车辆），实现违章检测

数据准备

1. 同样基于无人机采集的视频流，每隔 0.75s，取一帧图像
2. 使用上一节所训练的“车辆检测”模型，检测出图中所有车辆的坐标
3. 对于检测到的每一辆车，进行如下操作
    1. 使用 opencv 将车辆以及周围环境（目的是包含停车线）裁剪下来，作为一张新图片
    2. 将车辆所在的区域涂黑（避免车辆特征对停车线检测的干扰）
    3. 保存该图片



4. 从上述图片中选出 7500 张，进行二分类，分为违章车辆和非违章车辆



正样本

负样本

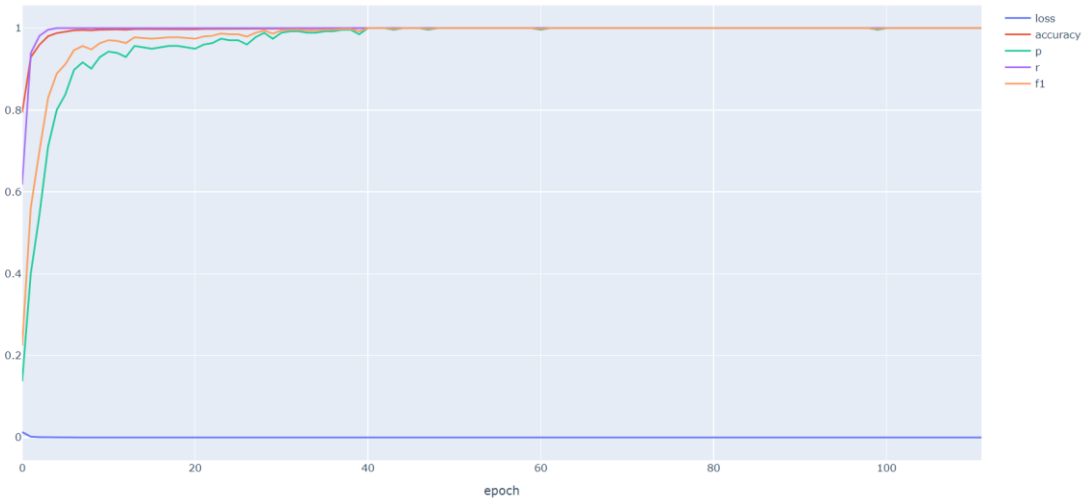
5. 编写 python 脚本，将数据切分为训练集和测试集两部分，图片路径信息存于 train.txt 和 test.txt 两个文件中。

模型训练

使用 pytorch 训练模型，网络结构为 ResNet18 网络，训练流程如下。

1. 定义超参数， lr=0.001、batch_size=128，epoch=111 等
2. 定义数据集（继承 Dateset），用于读取文件夹中的数据
3. 定义 transforms，即，对输入图像的处理，包含如下三步
    1. 将图片 resize 成128×128的大小
    2. 将图像转换为 tensor
    3. 对图像的均值和方差进行 Normalize（避免 loss 震荡，使模型更容易收敛）
4. 实现 dataLoader，从 dataset 中读取数据
5. 加载网络，这里我直接从 torch.hub 中加载了 resnet18 模型
6. 定义损失函数，这里使用了交叉熵函数，注意这里要对类别加权，因为样本集的类别不均衡，因此为正样本分配了 0.95 的权重，为负样本分配了 0.05 的权重
7. 定义优化函数，这里使用 SGD
8. 开始训练，在每个 iteration 中
    1. 得到 batch 的输入 tensor 和标签
    2. forward
    3. backward
    4. optimize
9. 每个 epoch 保存一次模型（last.pt）

一共训练了 111 个 epoch，评估指标伴随 epoch 的变化曲线如下图。

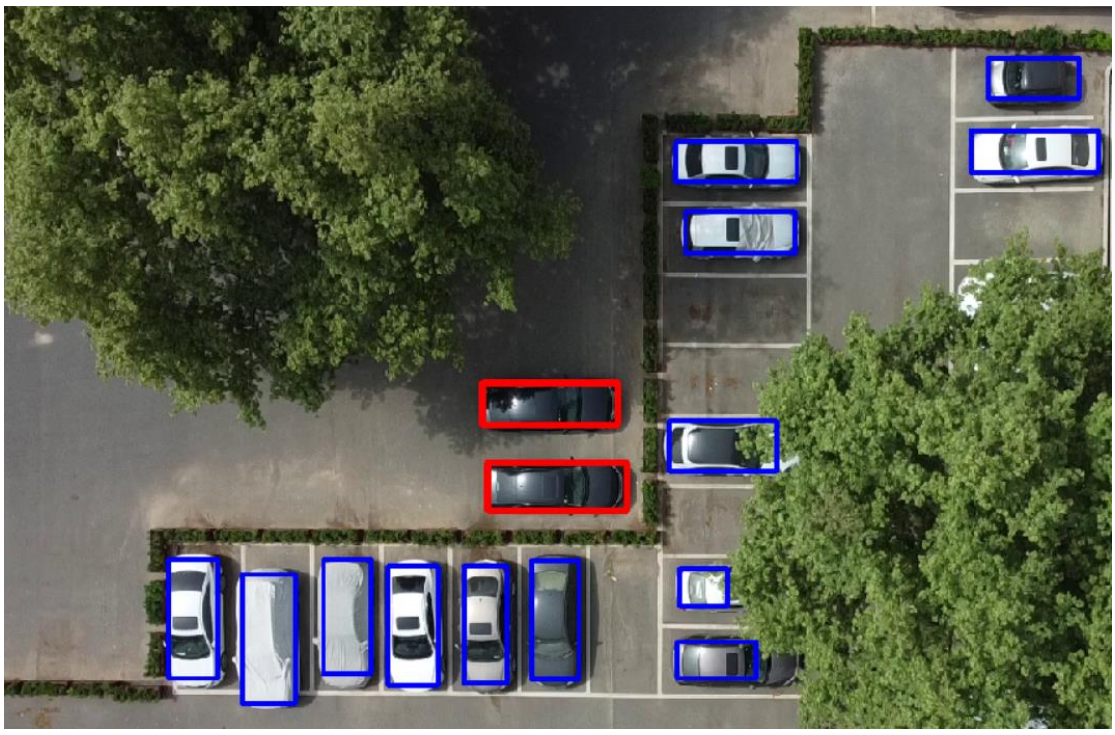


模型测试

测试集上的测试结果如下表。

loss	accuracy	p	r	f1	耗时
0.001	0.992	0.915	0.929	0.922	14.928s

违章检测效果如下图所示（粗红框代表违停车辆，细蓝框代表正常车辆）。



五、 总结

本项目实现了实时检测无人机视频流中的违章车辆，目前基本功能已经实现。接下来将从两个方面改进该项目。

1. 重新训练违章分类的模型

   目前是通过车辆周围是否有停车线判断是否违章，因此对停在停车线外的但是靠着停车线停靠的违章车辆不能很好的判断。

   我们下一步打算改进违章判断：对车辆上下左右四个方向的停车线均进行检测，至少在三个方向上存在停车线才认为没有违章。

2. 将模型部署在 C++上，加快检测速度

   目前主程序是用 C++编写，但车辆检测和违章判断均是用 python 实现，主程序中对每一帧都会调用 python 程序进行检测，每次调用 python 都会加载一次模型，加载模型的时间远大于实际检测的时间。

   如果将模型部署在 C++上，只需要在主程序启动时加载一次模型，之后不再需要加载模型，可以大大提高速度。