第三届 Sky Hackathon
参赛项目书

参赛学校：南方科技大学

参赛队名：Talented handsome cat and dogs

指导老师：赵佳华

团队成员：段心童、聂雨荷、庄湛

概要

本次参与 Sky Hackathon 关于交通感知的比赛，我们团队收获颇丰。在数据集的处理上，我们团队首先查看并筛选了导师提供的 BDD100k 数据集，并且挑选了清华-腾讯 100k 数据集中较好的图片以及通过校内取景、网络搜索的方式进一步丰富训练集。在神经网络训练和推理的任务中，我们小队采用并行训练的方式，一方面训练 NVIDIA 已给的 Transfer Learning Toolkit(TLT)中的 ssd-resnet18 和 yolov3-darknet53 两个模型进行训练，另一方面查阅并测试开源的 yolov5 的模型，两组数据同时对比，最终选取 mAP 更高的 yolov5 进行推理和导出，并将 engine 文件部署在 Jetson Nano 平台上。

剽窃澄清

本次比赛的全部内容由在队长的带领下由我们小队独立完成，除了参考官网论坛和互联网的信息以及在群内寻求导师的帮助以外，其它内容都由我们小组团队讨论得到。本报告中提出的所有内容都是我们自己努力的结果。此外，本报告没有任何部分是从其他来源复制的。本组明白任何抄袭及/或使用未经认可的第三者资料的证据将被视为严重事件处理。

鸣谢

在此非常感谢 NVIDIA 的导师和组织者能够给我们提供这样的平台和学习的机会，让我们能够利用 TLT 和 TensorRT 将自己的模型真正部署到 Jetson Nano 上，深入了解迁移学习和目标检测并实际运用到交通感知上。同时我们也非常感谢我们的带队老师赵佳华，他在平台安装和部署以及把控整体项目的进度上给予我们莫大的帮助，并且申请了学校的服务器、GPU 资源供我们测试调优，没有他的帮助，我们无法顺利完成任务。最后，感谢我们小队的所有同学的贡献，即使课程压力很大的情况下大家仍然能够抽出时间来做这个项目并努力攻克它。这确实是一次难忘的经历。

正文

1 数据集准备

1.1 BDD100k

BDD100K 是由组委会给我们提供的基础数据集[1]，里面包含了已经标注好的 7 万张国外街景图片。但是该数据集并不完备，我们对它进行了处理与清洗。首先我们手写脚本将对应的 kitty 标签标注在图片上并且展示。经过观察我们发现，数据集中单车数量非常少、有很多黑夜、汽车重叠的部分过多等问题。最终我们选取了图片大小前 2000 大的图片并手动挑选了一些包含自行车类别较多的图片，完成了数据集的清洗。同时在这个过程中我们将小尺寸标签和标注失误的图片进行了标签层次和图像层次的敲除。

1.1.1 数据展示

我们写了一个脚本将组委会提供的 BDD 数据集展示出来以供后续挑选，如**图 1**，代码脚本详见附录。其中**红色**为行人，**蓝色**为汽车，**青色**为路标，**绿色**为自行车。

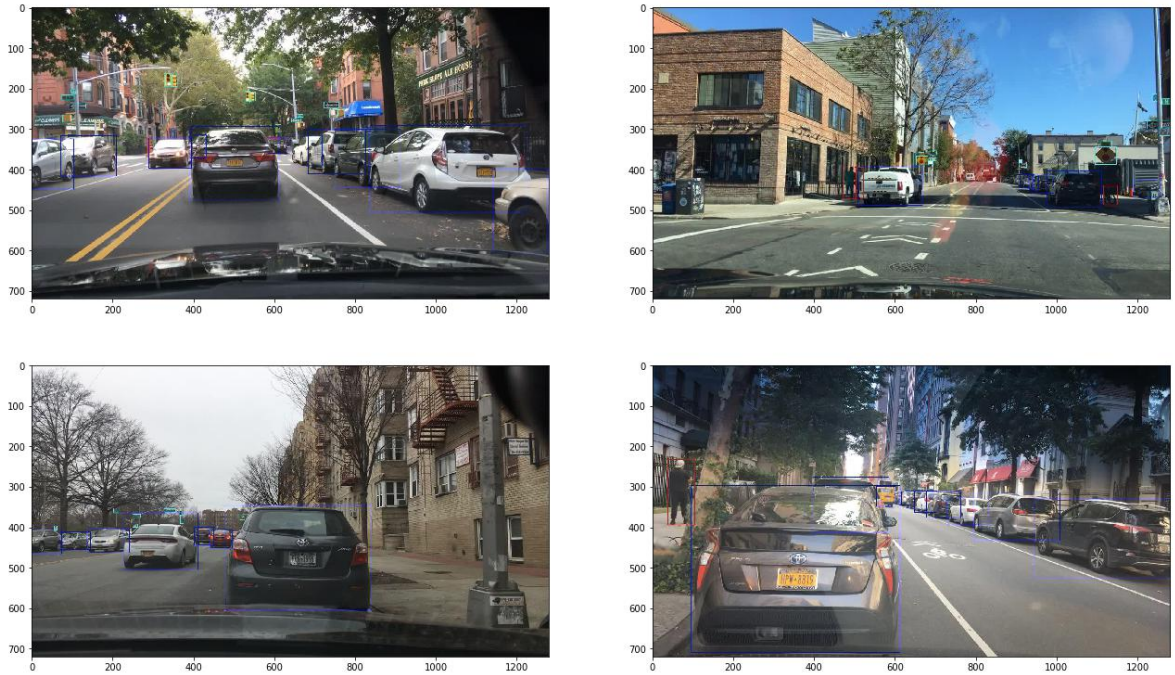

图 1

[1] https://bdd-data.berkeley.edu/

1.1.2 数据选取

刚开始观察的时候，认为 BDD100k 的数据集标注效果非常好，不仅没有错误标注，还有很多不易发现的小细节也被发现并标注。但经过一天的筛选和观察，我们发现了如下一些问题：

- 数据集的黑夜图片很多，此时人、车、自行车等过于昏暗，不易于训练
- 数据集中汽车的标签过多且重叠场景很多，自行车、行人、路标的标签较少
- 数据集中标签很小的情况很多
- 数据集都是国外街景，与国内街景存在差异（自行车形状（国内以电动车为主），路标形状等）

于是对于 BDD100k 的数据集清洗，我们做了如下操作：

- 将数据集按照图片大小进行排序，选择前 2000 张较大的图片（在图片尺寸相同的前提下，图片相对越大，像素点和颜色越多，导致图片所包含的信息和色彩丰富度更大，因此质量一般较好）。
- 其它图片进行人为筛选，选择白天、自行车，路标，行人较多且没有漏标错标的图片。

1.2 清华-腾讯 100k

由于 BDD100k 的上述缺点，我们寻找到了国内街景数据集丰富的清华-腾讯 100k 数据集[2]（Zhu et al. 2016），在其中找到图片清晰度高、画质好，且集中于白天的国内街景照片。

1.2.1 数据选取

于是对于清华-腾讯 100k 的数据集标注和清洗，我们做了如下操作：

- 进行人为筛选，选择白天、自行车，路标，行人较多的图片
- 使用推荐的 Labelme 软件进行手动标注

2 Zhu, Zhe, Liang, Dun, Zhang, Songhai, Huang, Xiaolei, Li, Baoli, and Hu, Shimin (2016).“Traffic-Sign Detection and Classification in the Wild”. In:The IEEE Conference on Com-puter Vision and Pattern Recognition (CVPR)

1.3 其它图片资源

即使选择了国内的数据集标注，仍然存在针对性不强的问题，比如自行车在总体占比仍然很少，与汽车相比仅占 5%，对此我们进行有针对的挑选图片。最终自行车标签在全部标签中占比达到了 14%，训练效果远比之前的测试好。

我们在校园内拍摄了大量的行人和自行车图片，如**图 2**。

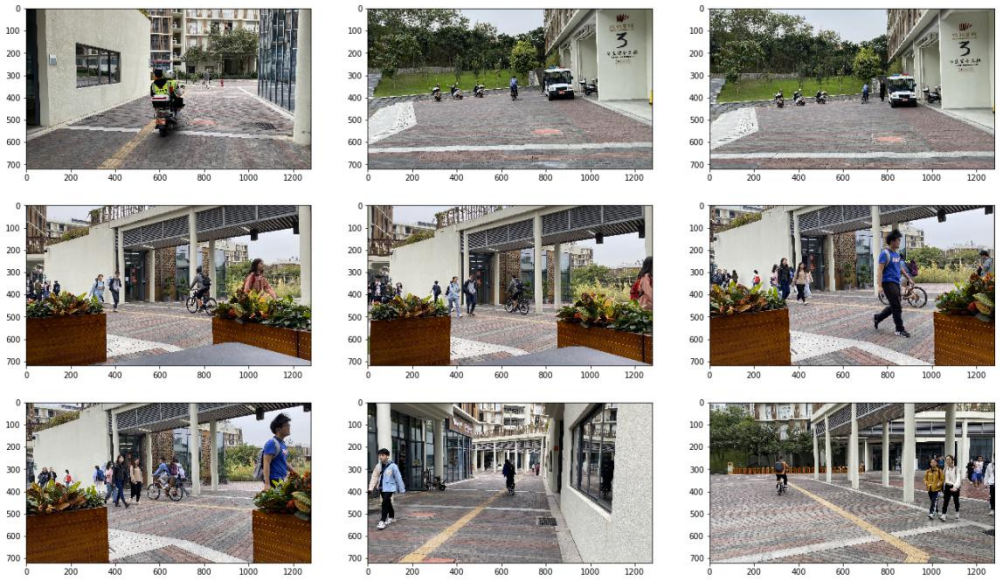

图 2

我们在网络上也针对性地选取了一些跟自行车有关的图片，如**图 3**。

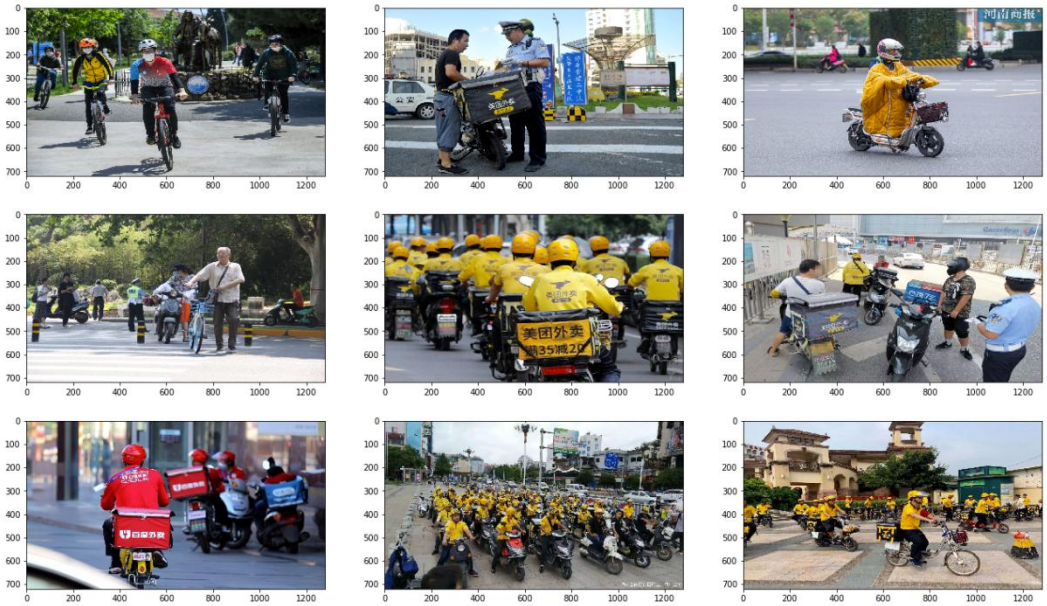

图 3

1.4 标注总结

采用组委会推荐的 labelme 对数据进行标注，标注效率很高，总共标注 1000 余张。最终训练用数据集共 2000 余张图片，经过尺寸统一后大小全部为 1280*720，其中行人标签总数为 7181 个，汽车标签总数为 17513 个，自行车标签总数为 3274 个，路标标签总数为 13588 个，位置和大小分布如**图 4**。

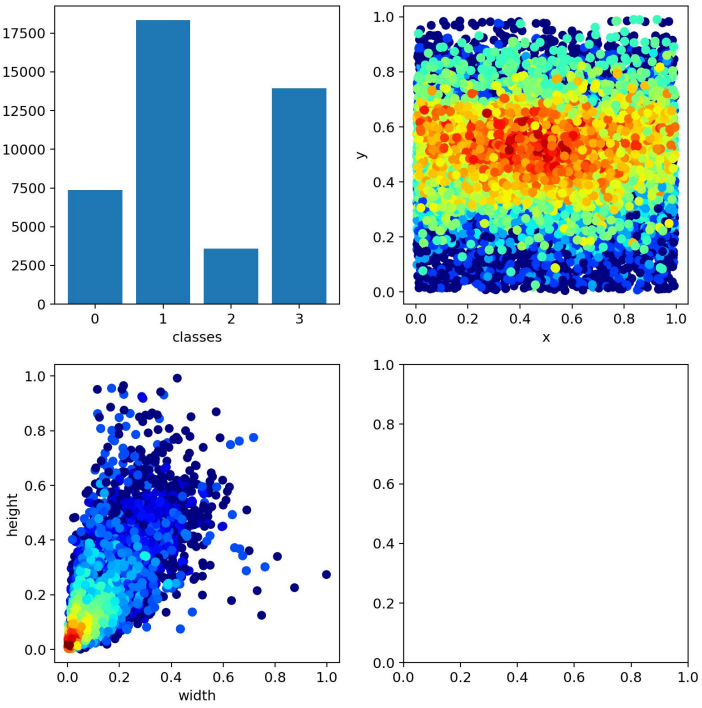

图 4

注意：标注时采用左上至右下的方式绘制边框，如果方向不对将导致 tlt 报错，可以通过脚本（change_label.py）见附录将 kitty 格式的数据修正。

1.5 数据增强

在组委会提供的 TLT 中，已经存在了对于色彩饱和度、灰度、翻转等进行调整的数据增强部分，但是我们仍然重写了这部分数据增强用于 yolov5 的训练中。除此之外，我们利用 python 上的 PIL 库和 opencv 模块，对部分标注图像进行了高斯模糊、椒盐模糊的噪声添加以及变亮、变暗的亮度调整等扩容操作。如**图 5**（分别为原图，标注，变暗，变亮，高斯模糊，椒盐模糊），最终我们将整个数据集扩容至 2944 余张。

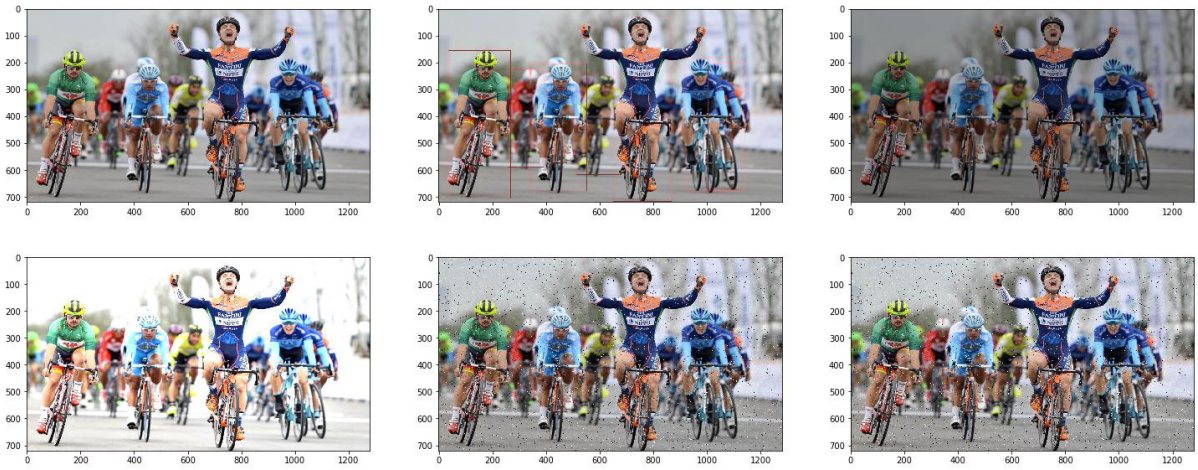

图 5

2. Transfer Learning Toolkit(TLT)训练模型

TLT 是有组委会提供的一套封装好的训练神经网络的集成环境，里面内置了丰富的神经网络模型和框架。首先我们跑通了组委会提供的 ssd-resnet18 的 baseline，然后使用我们训练好的数据集进行训练和剪枝，最终 map 在 0.592，效果欠佳。随后我们对 TLT 进行了重构，使用了神经网络模型中对于目标检测的常用搭配 yolov3-darknet53 进行训练，最终 map 为 0.636。

2.1 ssd-resnet18 模型

2.1.1 参数调整

参数	调整数值
batch_size_per_gpu	32（根据 GPU 的内存调整）
num_epochs(train)	300
pth(剪枝强度)	0.5
num_epochs(retrain)	300
val_split(数据集分离比)	14

2.1.2 训练效果

剪枝前在 mAP 在 0.59，剪枝后为 0.469，虽然模型最终大小比较小，但是目标检测效果不是很理想，如**图 6**，可以看出有很多车没有标注出来。

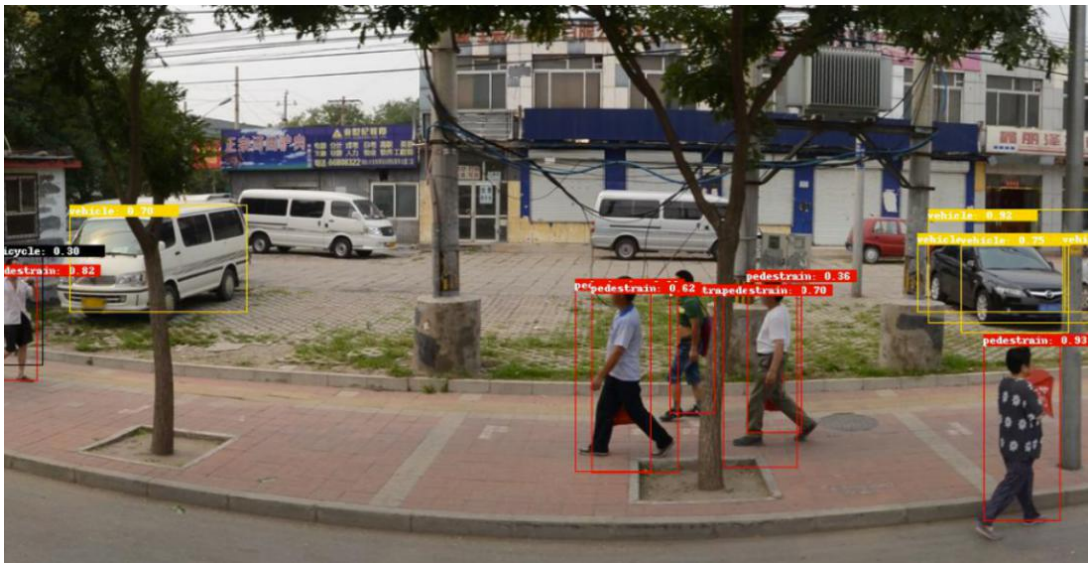

图 6

2.2 yolov3-darknet53 模型

2.2.1 参数调整

参数	调整数值
batch_size_per_gpu	32（根据 GPU 的内存调整）
num_epochs(train)	300
pth(剪枝强度)	0.5
num_epochs(retrain)	300
val_split(数据集分离比)	14

2.2.2 训练效果

如**图 7**，可以看出 yolov3-darknet53 对于目标检测的识别比率比 ssd-resnet18 的效果好，但是仍存在较少识别错误的部分。剪枝前 mAP 在 0.636，通过剪枝后训练 300 个 epoch 后降低至 0.501，第一次我们使用 pth=0.5 进行剪枝，mAP 较低，将剪枝系数调低后，仍不是很理想，且模型较大。

图 7

2.3 TLT 使用总结

在使用 TLT 的时候，我们也发现了一系列的优缺点。首先 TLT 作为一个集成环境，它的集成效果好，从数据集到训练出来的模型直接导出 engine 文件这一系列操作可以一次性完成。在 TLT 上支持多种模型和框架的选择，便于我们比较对比不同主干网络的性能。但是，可能是数据集准备不充分，抑或是 TLT 自身的局限性，我们最终跑出的很多个模型的 map 不理想，最高的仅有 0.66 左右。

2.3.1 模型对比

如**图 8**、**图 9** 所示，我们将我们使用的模型间的结果数据进行对比。

	Model	Size(M)	epoch	mAP	bicycle_AP	pedestrian_AP	road_sign_AP	vehicle_AP
0	ssd-resnet18	52.0	300	0.592	0.651	0.484	0.442	0.789
1	ssd-resnet18(pruned)	1.2	300	0.469	0.504	0.342	0.311	0.719
2	yolo-darknet53	236.0	300	0.636	0.699	0.553	0.505	0.786
3	yolo-darknet53(pruned)	23.5	300	0.501	0.497	0.405	0.398	0.705

图 8

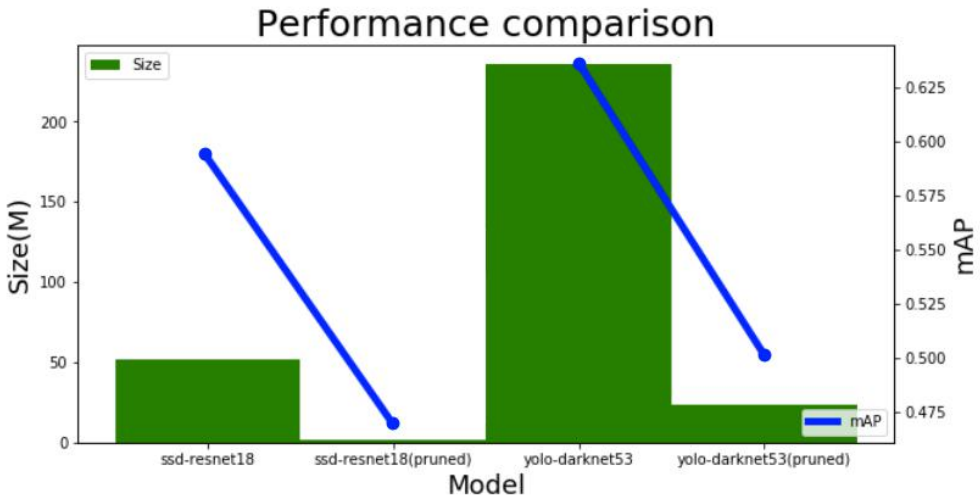

图 9

2.3.2 试错经验整理

- 数据集必须把图片(image)和标签(label)都 resize 到同一大小。
- 训练的时候需要注意自己 GPU 的显存大小,如果 batch_size 设置过大可能会出错且 loss 难以下降。
- 组委会提供的数据集中 pedestrian 拼成了 pedestrain。
- 注意训练集内的图片格式是.jpg 还是.png。
- yolov3 的数据集图片大小必须设置成 32 的倍数(416x416 或 640x640)。
- 对于 yolov3 模型，需要通过 kmeans 聚类得到 3 组 anchor 值。
- 在一个模型训练、推理、部署期间要保持 Key 值一致且有效。

3. Yolov5 训练模型

除了提供的 TLT 训练模型之外，为了追求更高的精确度，我们尝试采用了目前精度与速度都非常好的 yolov5 模型，我们尝试了 yolov5s，yolov5l 和 yolov5x，其中 yolov5l 的 map 最高，但是模型过大，部署上去时延过大，yolov5s 的精度和模型大小都较为理想，最终我们选择了 yolov5s 作为本次比赛的提交训练模型。

3.1 参数调整

在参数调整方面，我们发现在将训练图片大小调整到输入图片大小后整体 mAP 有着比较显著的提升（yolov5-l 模型 map 提升了接近 0.1）。我们认为主要可能是 resize 后的图片一些小目标会被模糊或者消失，失去细节。但是对于提升 mAP 最有效的方法还是数据集的处理。为了加强模型对小目标的识别我们在数据集中添加了一些高清含有较多小目标的图片，且选用较深的网络。经过前期的多次训练我们发现四个种类中自行车的 AP 远远低于其他三类，且 mAP 非常低，如**图 10**。在我们添加了自行车相关的数据后整体 mAP 又有了显著的提升。

3.2 训练效果

使用相同数据集的情况下，训练 100epoch，其 mAP 最后稳定在 0.763。除此之外，yolov5s 和 yolov5l 的结果分别如**图 11、图 12** 所示：

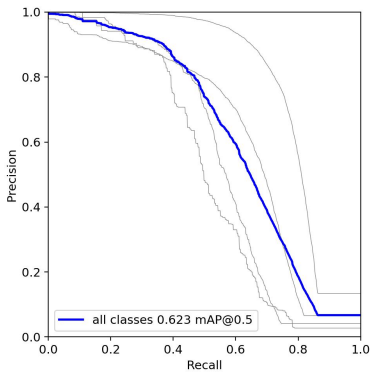

图 10

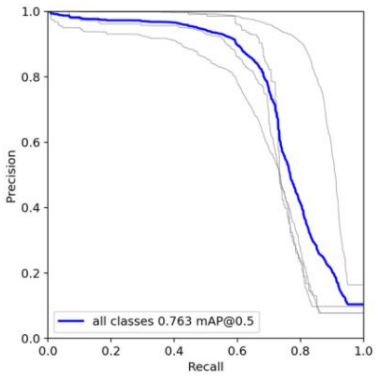

图 11

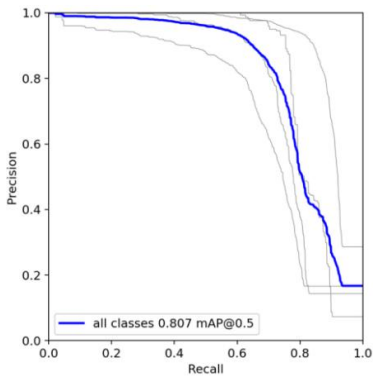

图 12

在部署到 Jetson Nano 上后，我们最终成功跑通测试视频，并得到了较好的效果如**图 13**。

图 13

3.3 Yolov5 使用总结

由于 yolov5 是一个较新的模型，目前开源剪枝资源比较少。在官方提供的剪枝不剪模型大小，但将其中接近 0 的参数调 0，将模型离散化。但是目前看来对模型的推理速度没有加速效果且会导致模型精度有一定程度的下降。由于课业过于繁忙也没有自己实现对模型的剪枝，希望在比赛后有机会再剪枝方面有更多的研究。

4. Jetson NANO 的部署和推理

NVIDIA 提供的 TensorRT 是一款高性能深度学习的推理优化器，利用它可以优化神经网络模型，利用 TensorRT Open Source Software (OSS)的接口插件和组委会提供的 ipynb 参考代码，我们可以很方便将模型部署到 Jetson Nano 上。如参考**图 13**，我们按照训练好的.tlt 文件到利用 tlt-export 将模型保存为.etlt 文件，最后通过在指导老师和组委会提供的 Jetson Nano 环境中利用 tlt-converter 将模型转出.engine 文件完成模型的推理和部署。

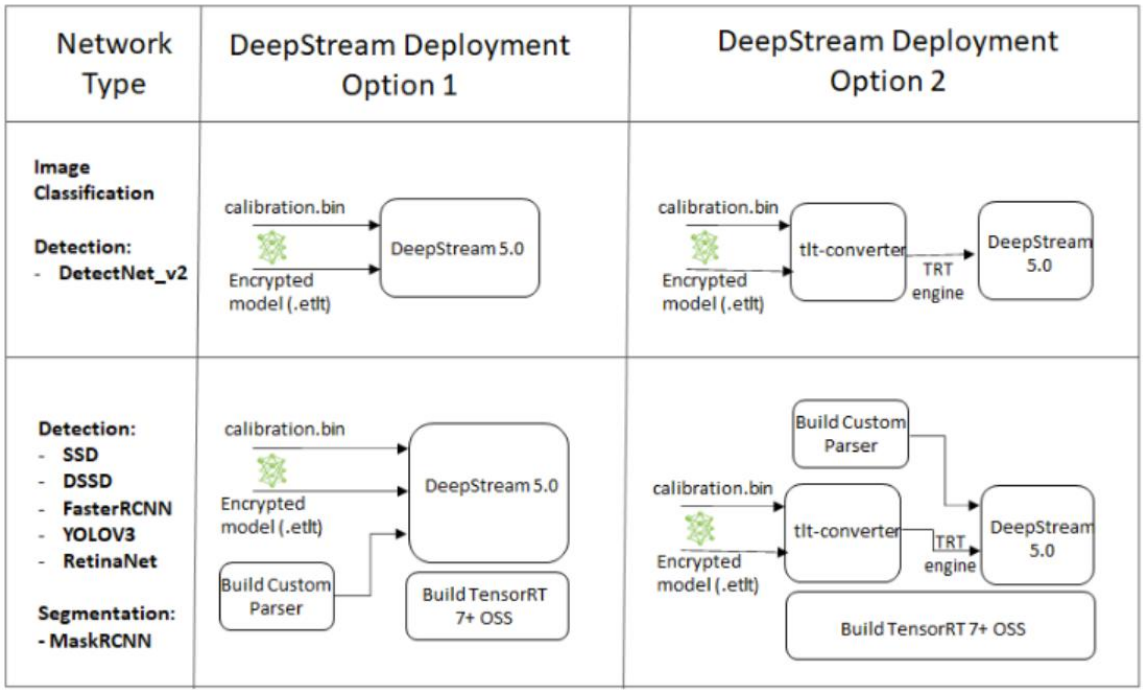

图 13

在模型推理的过程中，我们还选择了更小的精度 fp16 来加速我们的模型。

5. 总结（团队收获）

5.1 收获

- 收获了团队合作的快乐，小组共同完成任务，互相交流的宝贵经验
- 学习使用 Linux 操作系统的命令
- 了解了迁移学习在目标检测领域的详细流程
- 小组自己准备、标注数据集，了解了图像识别领域最基础的工作
- 了解了 GPU 用于训练神经网络的知识
- 学习了神经网络中的一些重要参数以及它们的优化和修改

5.2 遗憾

- 本次比赛由于时间紧迫，在数据集方面处理不全面，数据集本身还有很大的提升空间。
- 我们在查询 Yolov5 的剪枝时遇到了很大的困难，最终模型没有剪枝，在性能上存在一些欠缺。

5.3 希望

感谢组委会给我们提供的这次良好机会，祝 NVIDIA 类似的活动越办越多，越办越好，能与更多的高校和职场上的前辈们交流经验，谢谢！

附录

由于篇幅有限，此处大部分仅为核心代码部分。

show.py

```
count = 0
for picture_path in picture_path_list:
    image = Image.open(picture_path)
    draw = ImageDraw.Draw(image)
    label = labelList[count]
    for msg in label:
        if msg[0] == 'vehicle':
            color = (0,0,255)
        elif msg[0] == 'road_sign':
            color = (0,255,255)
        elif msg[0] == 'pedestrain' or 'pedestrian':
            color = (0,125,0)
        elif msg[0] == 'bicycle':
            color = (255,0,0)
        draw.rectangle([int(msg[1]),int(msg[2]),int(msg[3]),int(msg[4])], outline=color)
    image.show()
    image.save(new_picture_path)
    count += 1
```

change_label.py

```
    with open(label_path,"r") as f:
        for line in f.readlines():
            message = line.split()
            #修改pedestrain
            if(message[0] == 'pedestrain'):
                message[0] = 'pedestrian'
            x1 = message[4]
            y1 = message[5]
            x2 = message[6]
            y2 = message[7]
            #修改x1,x2,y1,t2
            if(float(x1) >= float(x2) and float(y1) >= float(y2)):
                tempX = x1
                message[4] = x2
                message[6] = tempX
                tempY = y1
                message[5] = y2
                message[7] = tempY
            elif(float(x1) < float(x2) and float(y1) >= float(y2)):
                tempY = y1
                message[5] = y2
                message[7] = tempY
            elif(float(x1) >= float(x2) and float(y1) < float(y2)):
                tempX = x1
                message[4] = x2
                message[6] = tempX
            newLabel.append(message)
```

rename.py

```
file_names = os.listdir(image_path)
name_list = dict()
i = 0
for name in file_names:
    s = "%05d" % i
    name_list[name.split('.')[0]] = s
    i+=1

i = 0
for name in file_names:
    nm = name.split(".")[0]
    hh = name.split(".")[1]
    if nm in name_list:
        os.rename(image_path + name, image_path + name_list[nm]  + '.' + hh)
        os.rename(label_path + name, label_path + name_list[nm]  + '.' + hh)
        i+=1
print(i)
```

remove.py

```
with open('bdd_wrong.txt', "r") as f:
    for line_ in f.readlines():
        target = line_.strip('\n')[:5]
        jpg = train02_path + target +'.jpg'
        txt = label02_path + target +'.txt'
        if os.path.exists(jpg) and os.path.exists(txt):
            os.remove(txt)
            os.remove(jpg)
        else:
            print(jpg)
            print(txt)
```

resize.py

```
def resizeImg(w, h, imgs_floder, save_folder, old_label_path, new_label_path):
    imgs = os.listdir(imgs_floder)
    for img in imgs:
        img_full_path = os.path.join(imgs_floder, img)
        ori_img = cv2.imread(img_full_path)
        img_r = cv2.resize(ori_img, (w, h))
        imgSize = Image.open(imgs_floder + img).size
        if imgSize[0]!= 1280 and imgSize[1]!=720:
            cv2.imwrite(os.path.join(save_folder, img), img_r)
            adjust_labels(img,w/imgSize[0],h/imgSize[1], old_label_path, new_label_path, img_r)
            print(imgSize,img)
    return 0


def adjust_labels(imgName, w, h, ori_labels_folder, new_labels_folder, img):
    with open(os.path.join(ori_labels_folder, imgName.split('.')[0] + '.txt'), 'r') as f:
        infos = f.readlines()
        with open(os.path.join(new_labels_folder, imgName.split('.')[0] + '.txt'), 'w') as w:
            for box in infos:
                info = box.strip('\n')
                label_info = info.split(' ')
                xmin = int(label_info[4]) * float(w)
                ymin = int(label_info[5]) * float(h)
                xmax = int(label_info[6]) * float(w)
                ymax = int(label_info[7]) * float(h)
                new_info = label_info[0] + ' 0.00 0 0.00 ' \
                           + str('{:.2f}'.format(xmin)) + ' ' + str('{:.2f}'.format(ymin)) + ' ' \
                           + str('{:.2f}'.format(xmax)) + ' ' + str('{:.2f}'.format(ymax)) \
                           + ' 0.00 0.00 0.00 0.00 0.00 0.00 0.00'
                w.write(new_info + '\n')
        w.close()
    f.close()
    return 0
```

bdd_wrong.txt (bdd100k 中挑选的较差图片序号)

```
['00002', '00008', '00020', '00033', '00050', '00061', '00063', '00070', '00074', '00092', '00102', '00121', '00165', '00180', '00193']
['00289', '00307', '00340', '00472', '00568', '00589', '00653', '00691', '00882', '00925', '00926', '00959', '00998', '00042', '00529']
['00761', '00781', '00811', '02104', '02107', '02112', '02115', '02117', '02118', '02119', '02133', '02136', '02139', '02140', '02141']
['02145', '02146', '02151', '02153', '02002', '02004', '02007', '02009', '02017', '02039', '02058', '02061', '02064', '02081', '02090']
['02100', '02157', '02161', '02171', '02245', '02256', '02321', '02348', '02358', '02362', '02440', '02444', '02447', '02499', '02509']
['02573', '02568', '02675', '02683', '02684', '02690', '02721', '02724', '02859', '02927', '02981', '06473', '06514', '06877', '07267']
['07662', '07696', '07814', '08056', '08068', '08084', '08097', '08125', '08139', '08273', '08378', '08529', '08653', '08687', '08722']
['08778', '08934', '08959', '08968', '09239', '09302', '09318', '09349', '09499', '10002', '10007', '10025', '10030', '10050', '10072']
['10222', '10904', '10924', '11297', '12251', '14504', '14515', '15001', '15007', '15008', '15010', '15011', '15015', '15018', '15034']
['15035', '15040', '15056', '15057', '15067', '15072', '15101', '15105', '15131', '15158', '15167', '15187', '15314', '15315', '15326']
['15333', '15335', '15343', '15346', '15349', '15350', '15375', '15404', '15412', '15420', '15426', '15430', '15440', '15443', '15484']
['15490', '15513', '15517', '15544', '18721', '18725', '18745', '18805', '18827', '18830', '18849', '18871', '19109', '19124', '15202']
['15225', '15259', '15301', '15303', '15875', '15905', '15952', '15983', '16003', '16010', '16045', '16057', '16093', '16100', '16128']
['16136', '15553', '15588', '15622', '15629', '15689', '15713', '15718', '15720', '15794', '15831', '15833', '15868', '16154', '16196']
['16252', '16260', '16412', '16518', '16540', '16582', '16609', '16760', '16810', '16820', '16829', '16837', '16846', '16941', '17008']
['17056', '17088', '17099', '17101', '17135', '17137', '17139', '17179', '17345', '17359', '17417', '17581', '17619', '17677', '17821']
['17997', '18437', '18460', '18526', '18559', '18589', '18596', '18632', '18676', '20110', '20129', '20194', '20230', '20249', '20261']
['20304', '20321', '20384', '20605', '20628', '20720', '20730', '20742', '20805', '20819', '20861', '20953', '21144', '21150', '21155']
['21163', '21218', '21310', '21384', '21394', '21396', '21421', '21435', '21571', '21575', '21613', '21651', '21659', '21715', '21763']
['21778', '21947', '21957', '22137', '22232', '22247', '22278', '22293', '22308', '22329', '21763', '21778', '21947', '21957', '22137']
['22232', '22247', '22278', '22293', '22308', '22329', '22508', '22530', '22540', '22601', '22639', '22775', '22846', '22905', '22977']
['23006', '23132', '23282', '23590', '23849', '24161', '25161', '25287', '26333', '26522', '27543', '25005', '25210', '25217', '25223']
['25274', '25297', '25292', '25545', '25574', '25581', '25722', '25783', '25787', '25803', '25814', '25856', '25858', '25860', '25928']
['25931', '26002', '26003', '26010', '26050', '26114', '26139', '26156', '26169', '26170', '26171', '26182', '26201', '26205', '26225']
['26227', '26242', '26253', '26255', '26259', '26262', '26272', '26279', '26286', '26299', '26320', '26322', '26333', '26335', '26350']
['26356', '26365', '26368', '26371', '26401', '26410', '26426', '26430', '26435', '26436', '26439', '26454', '26469', '26472', '26474']
['26477', '26479', '26480', '26483', '26500', '26519', '26522', '26526', '26529', '26535', '26545', '26550', '26554', '26565', '26571']
['26572', '26576', '26583', '26598', '26601', '26615', '26627', '26648', '26656', '26658', '26661', '26672', '26676', '26680', '26689']
['26711', '26716', '26741', '26751', '26758', '26764', '26777', '26785', '26788', '26794', '26796', '26802', '26804', '26805', '26813']
['26814', '26815', '26826', '26830', '26836', '26859', '26864', '26869', '26882', '26890', '26900', '26906', '26907', '26912', '26918']
['26920', '26924', '26925', '26926', '26933', '26934', '26937', '26945', '26953', '26959', '26960', '26965', '26967', '26972', '26990']
['26995', '27000', '27006', '27014', '27017', '27018', '27029', '27039', '27046', '27061', '27068', '27072', '27073', '27085', '27102']
['27109', '27121', '27159', '27161', '27187', '27199', '27227', '27260', '27275', '27279', '27286', '27306', '27322', '27335', '27365']
['27367', '27394', '27400', '27418', '27427', '27450', '27481', '27490', '27496', '27504', '27507', '27509', '27511', '27513', '27514']
['27535', '27537', '27542', '27543', '27555', '27559', '27563', '27604', '27621', '27635', '27647', '27683', '27684', '27686', '27689']
['27725', '27728', '27735', '27763', '27765', '27772', '27782', '27819', '27822', '27827', '27855', '27860', '27875', '28000', '28014']
['28023', '28057', '28114', '28149', '28174', '28177', '28208', '28210', '28211', '28220', '28232', '28308', '28314', '28342', '28358']
['28389', '28410', '28443', '28487', '28492', '28496', '28507', '28510', '28513', '28528', '28570', '28572', '28574', '28581', '28596']
['28601', '28609', '28619', '28624', '28668', '28696', '28706', '28755', '28794', '28809', '28861', '28916', '28917', '28938', '28981']
['28982', '29005', '29008', '29010', '29014', '29015', '29016', '29084', '29116', '29160', '29201', '29202', '29244', '29245', '29260']
['29270', '29342', '29345', '29346', '29348', '29377', '29469', '29522', '29537', '29542', '29561', '29607', '29612', '29640', '29649']
```