AI Object Detection

Team members

朱昀玮 万唯循 吕品灏 汪欣瑞 吴天骋 郑欣雨 陆明奇

Speaker

汪欣瑞 吴天骋

Contents

- 模型调优
- 数据集制作
- 预处理与数据增广
- 评估指标与可视化
- 训练策略
- 超参数调节
- 结果展示

Distribution of Work

朱昀玮　组长　超参数调优

万唯循　超参数调优

吕品灏　超参数调优

汪欣瑞　特征提取网络结构修改　评估指标与可视化

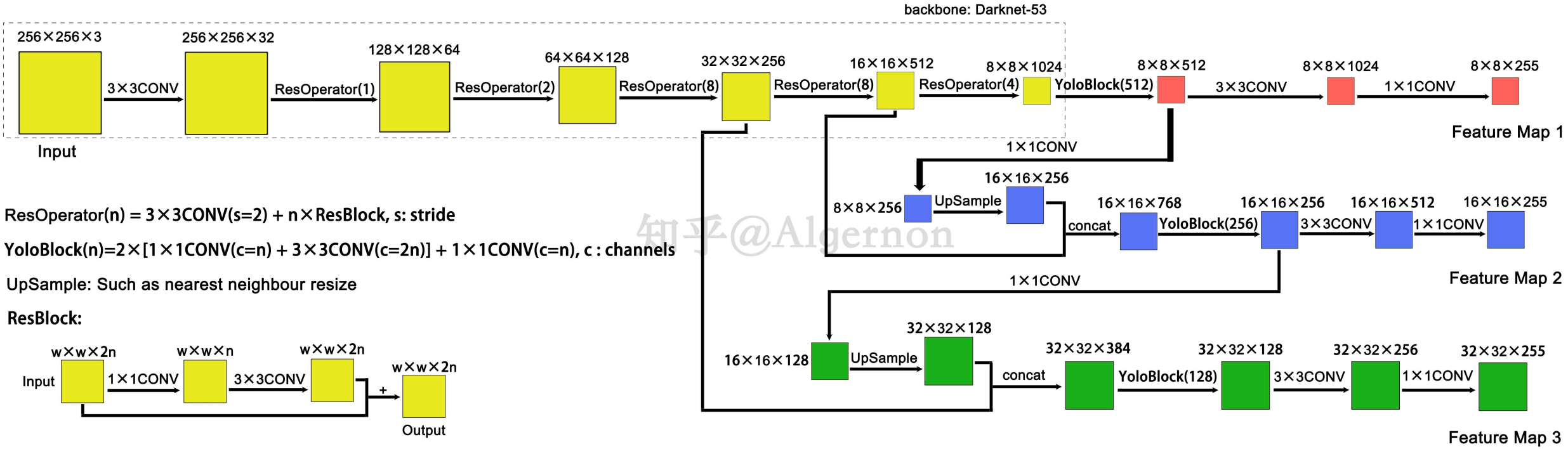
吴天骋　数据预处理与后处理　聚类分析与anchor计算

郑欣雨　数据集制作　超参数调优

陆明奇　超参数调优　数据集制作

Model optimization

YOLOv3



ResOperator(n) = 3×3CONV(s=2) + n×ResBlock, s: stride

YoloBlock(n)=2×[1×1CONV(c=n) + 3×3CONV(c=2n)] + 1×1CONV(c=n), c : channels

UpSample: Such as nearest neighbour resize

ResBlock:

图片来源：https://zhuanlan.zhihu.com/p/76802514

Darknet-53

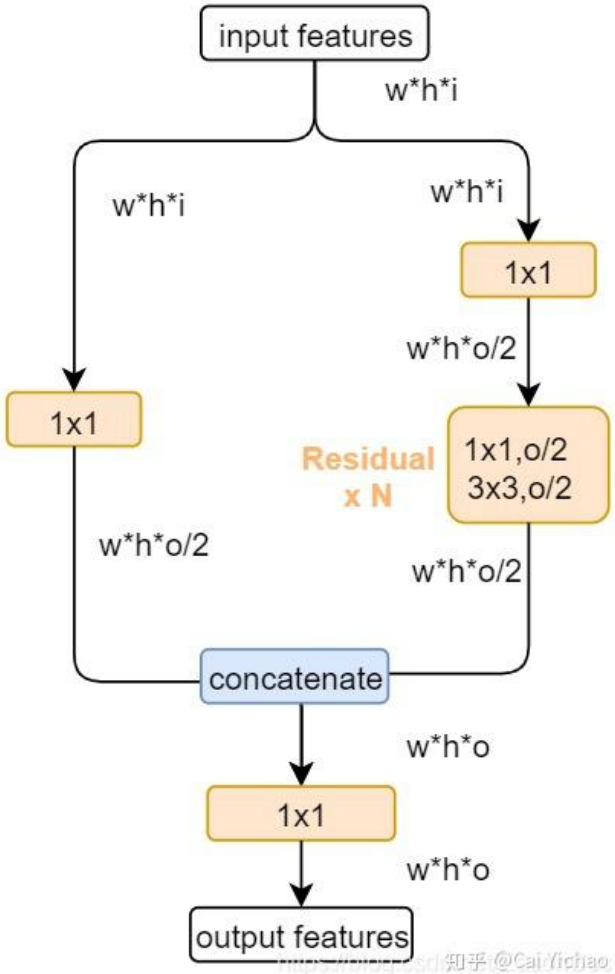
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			


```python
# 基本块：包含两个卷积/正则化层 --> 一个残差块 ResBlock
def basic_block(self, input, num_filters):
    conv1 = self.conv_bn(input, num_filters, filter_size=1, stride=1, padding=0)
    conv2 = self.conv_bn(conv1, num_filters * 2, filter_size=3, stride=1, padding=1)
    out = fluid.layers.elementwise_add(x=input, y=conv2, act=None)  # 计算H(x)=F(x)+x
    return out
```

```python
# 搭建网络模型 darknet-53
def net(self, img):
```

```python
# 循环创建basic_block组
for i, stage_count in enumerate(stages):
    block = self.layer_warp(downsample_,  # 输入数据
                            32 * (2 ** i),  # 卷积核数量
                            stage_count)  # 基本块数量
    blocks.append(block)
    if i < len(stages) - 1:  # 如果不是最后一组，做降采样
        downsample_ = self.down_sample(block, block.shape[1] * 2)
blocks = blocks[-1:-4:-1]  # 取倒数三层，并且逆序，后面跨层级联需要
```

CSPDarknet



- CSPNet (Cross Stage Partial Network)
  - ✓ 增加梯度路径，避免不同的层学习到重复的梯度信息
  - ✓ 平衡各层的计算量，减少内存流量
- Applying CSPNet to Darknet
  - ✓ YOLOv4与YOLOv5采用的特征提取网络结构
  - ✓ 在初始代码残差块的连接中引入CSP结构，构建CSPDarknet

图片来源：https://zhuanlan.zhihu.com/p/263555912

CSPDarknet

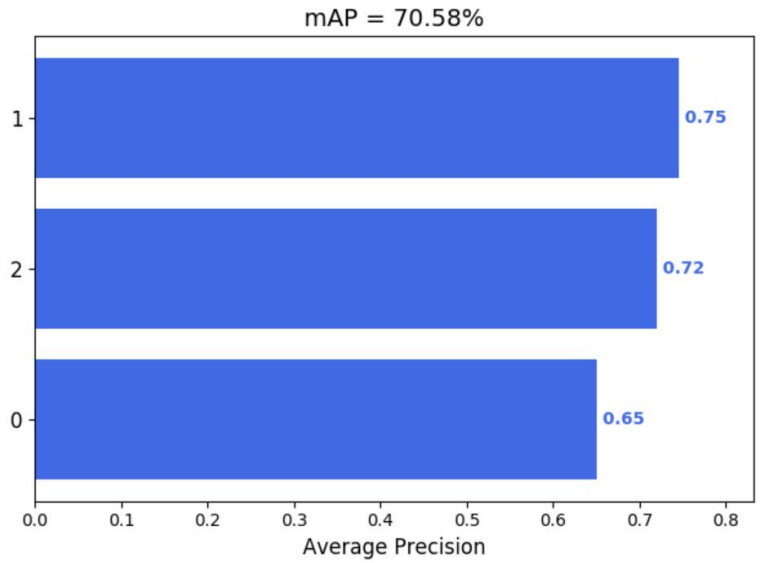
```
# 循环创建basic_block组
for i, stage_count in enumerate(stages):
    CSP_retain = self.conv_bn(downsample_, num_filters=downsample_.shape[1]//2, filter_size=1, stride=1, padding=0)# 第二个参数为卷积核数量
    CSP_conv = self.conv_bn(downsample_, num_filters=downsample_.shape[1]//2, filter_size=1, stride=1, padding=0)  # 第二个参数为卷积核数量
    block = self.layer_warp(CSP_conv,  # 输入数据
                            16 * (2 ** i),  # 卷积核数量 32 * (2 ** i) -> 16 * (2 ** i)
                            stage_count)  # 基本块数量
    block = fluid.layers.concat(input=[CSP_retain, block], axis=1)
    block = self.conv_bn(block, num_filters=block.shape[1], filter_size=1, stride=1, padding=0)  # 第二个参数为卷积核数量
    blocks.append(block)
    # 下一组ResOperator的3*3卷积部分
    if i < len(stages) - 1:  # 如果不是最后一组，做降采样
        downsample_ = self.down_sample(block, block.shape[1] * 2)
blocks = blocks[-1:-4:-1]  # 取倒数三层，并且逆序，后面跨层级联需要
```

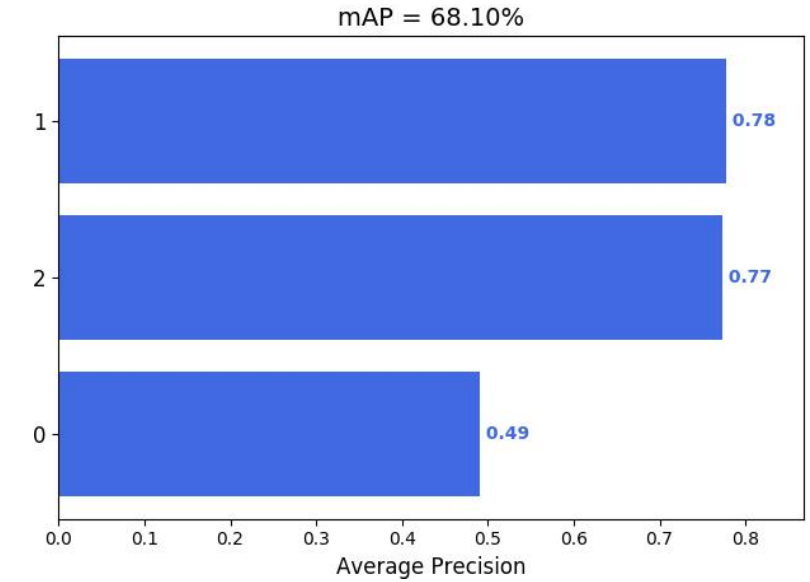
CSPDarknet

num_epochs=100, ignore_thresh=0.5, nms_thresh=0.45

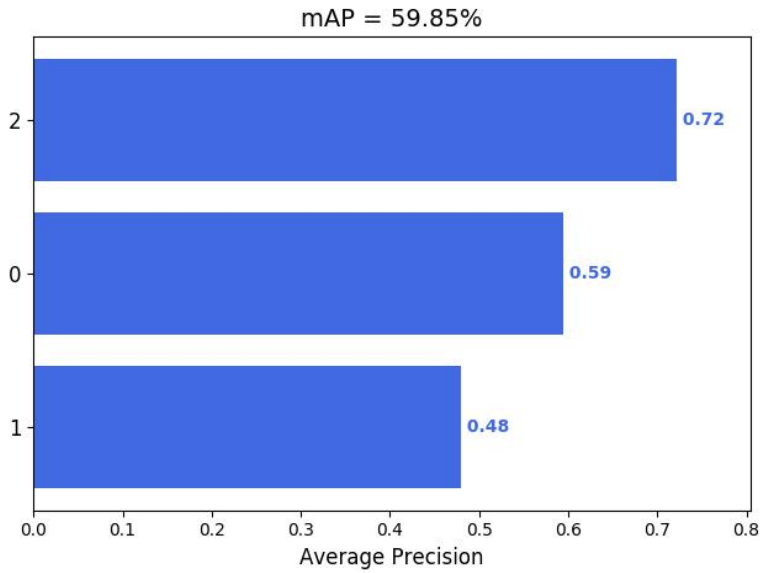
num_epochs=120, ignore_thresh=0.6, nms_thresh=0.407

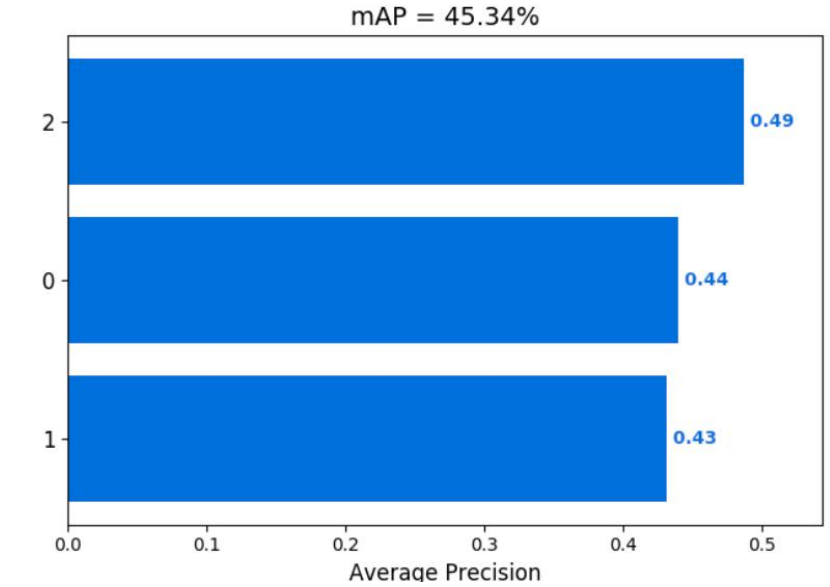
CSPDarknet-53





Darknet-53





Dataset

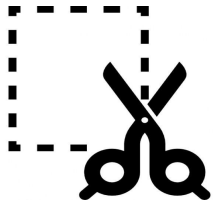
Dataset

- 校园采拍

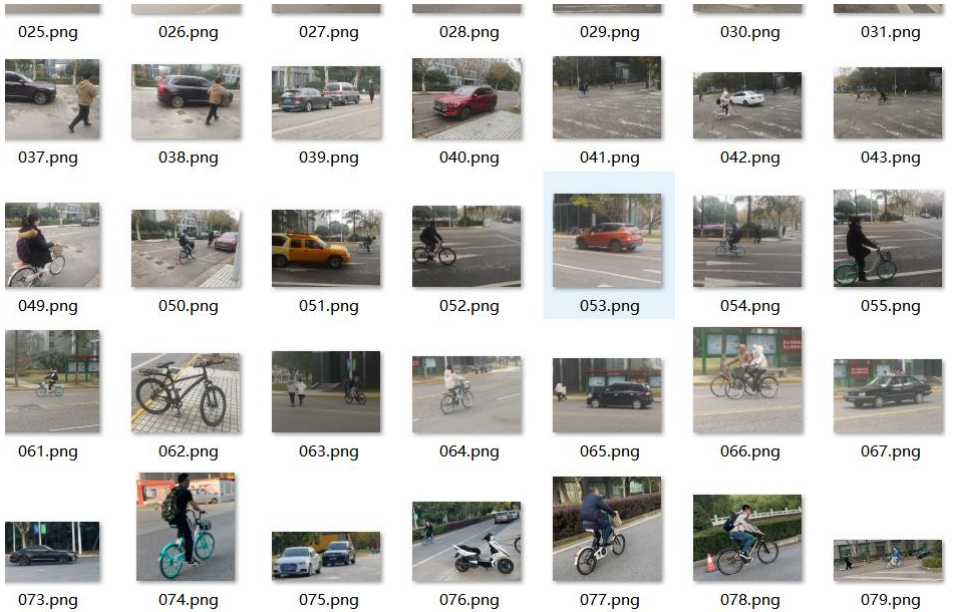
- 原始示例图片剪裁
- 权威数据集下载

Dataset

- 校园采拍
- 原始示例图片剪裁
- 权威数据集下载



101张原始图片，训练1轮耗时5min



剪裁后加入其他图片，训练1轮耗时2.5min

Dataset

- 校园采拍
- 原始示例图片剪裁
- 权威数据集下载

- VOC
- COCO
- ImageNet

400张校园道路图片-->800张

Augmentation & Preprocessing

原始代码实现了：

- 色调、对比度、饱和度、亮度的随机改变
- 1:5的扩增与随机剪裁
- 图像各像素的中心化

增加：图像的水平翻转

```python
# 预处理：图像样本增强，维度转换
def preprocess(img, bbox_labels, input_size, mode):
    img_width, img_height = img.size
    sample_labels = np.array(bbox_labels)

    if mode == 'train':
        if train_params['apply_distort']:  # 是否扭曲增强
            img = distort_image(img)

        img, gtboxes = random_expand(img, sample_labels[:, 1:5])  # 扩展增强
        img, gtboxes, gtlabels = random_crop(img, gtboxes, sample_labels[:, 0])  # 随机
        sample_labels[:, 0] = gtlabels
        sample_labels[:, 1:5] = gtboxes

    img = resize_img(img, sample_labels, input_size)
    img = np.array(img).astype('float32')
    img -= train_params['mean_rgb']
    img = img.transpose((2, 0, 1))  # HWC to CHW
    img *= 0.007843
    return img, sample_labels
```

Training strategy

Learning rate

Warm-up

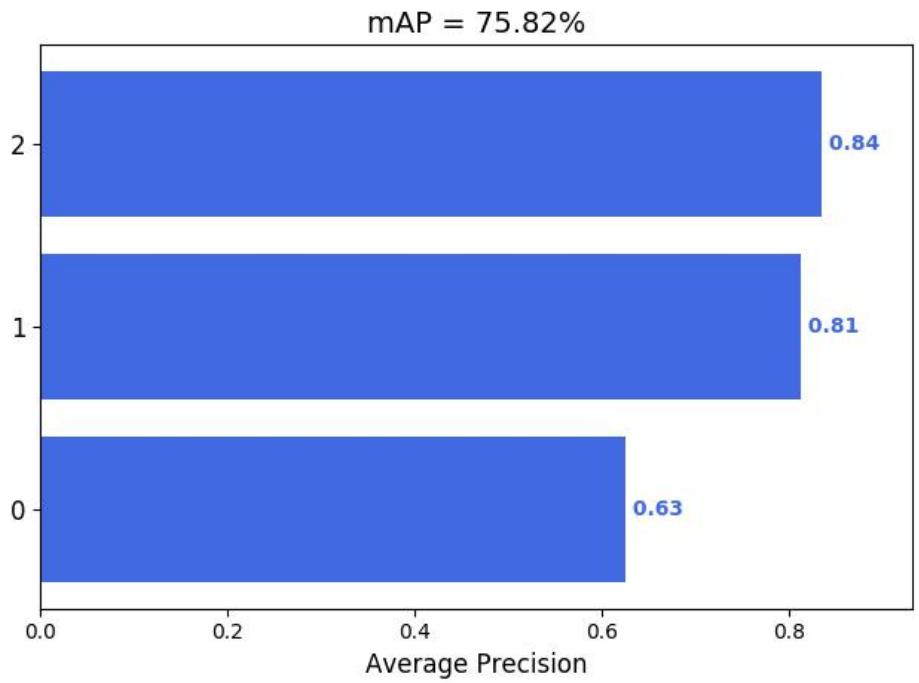
神经网络在刚开始训练的时候是非常不稳定的，因此刚开始的学习率应当设置得很低很低，这样可以保证网络能够具有良好的收敛性

Decay

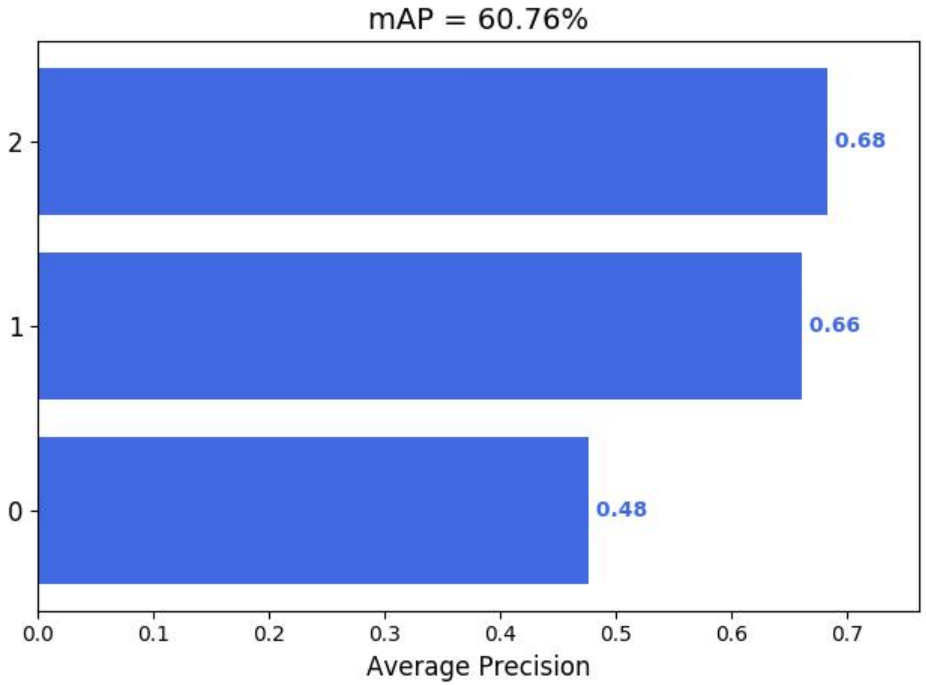
保证收敛到最优点附近时，以较小的学习率进行参数更新，以避免来回震荡

learning rate times: 0.5 --> 0.8 --> 1 --> 0.5 --> 0.25 --> 0.1

Optimizer



SGD, ignore_thresh=0.5,
num_epochs=100, nms_thresh=0.45
(num_epochs和nms_thresh均为条件最优)



Adam, ignore_thresh=0.5,
num_epochs=90, nms_thresh=0.45
(num_epochs和nms_thresh均为条件最优)

Hyperparameter adjustment

Evaluation index

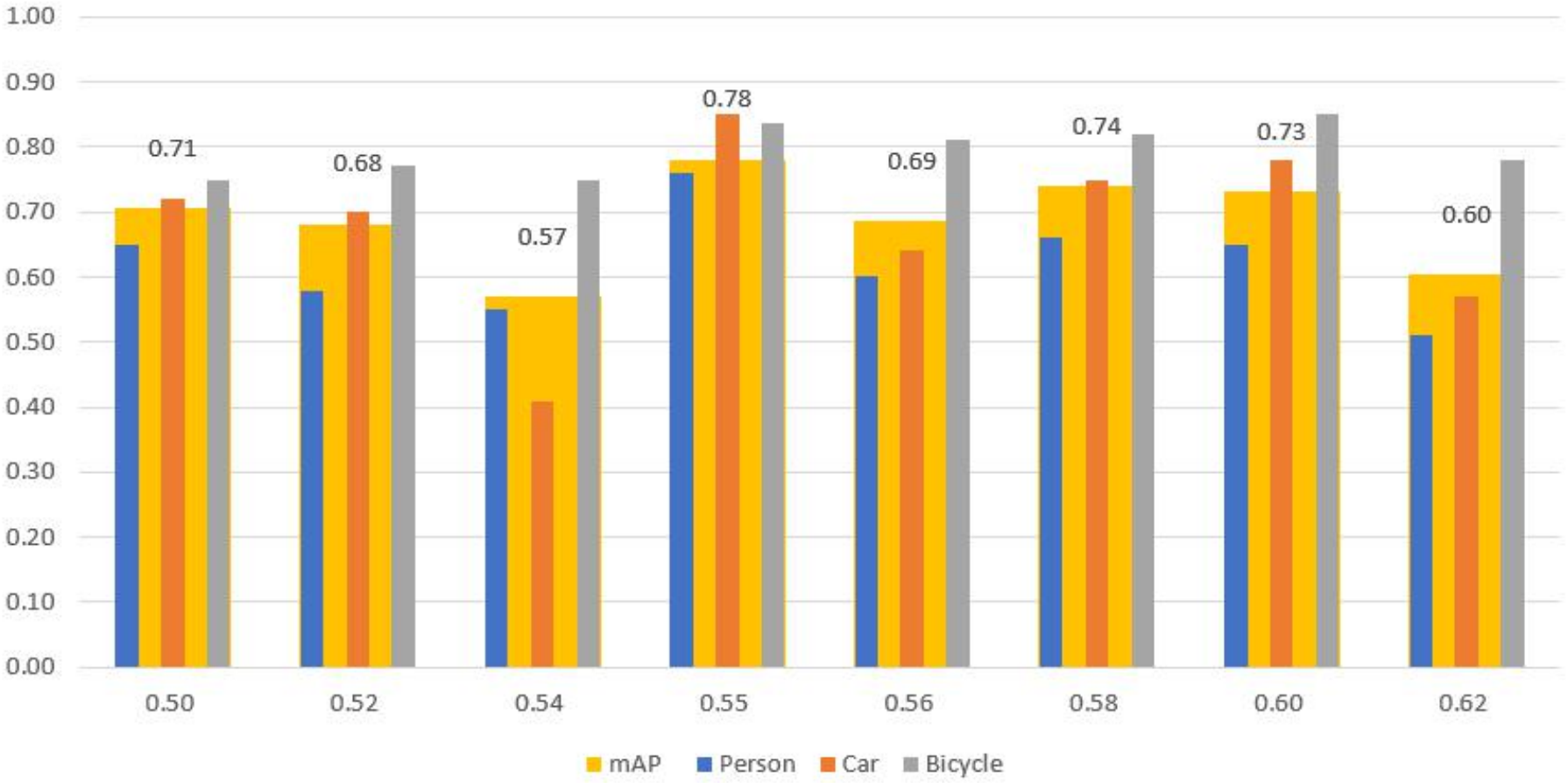
- 迫切需要评价指标指引方向。。。
- mAP
  - mAP: mean Average Precision, 即各类别AP的平均值
  - AP: Precision-Recall曲线下的面积
  - Precision: TP / (TP + FP)
  - Recall: TP / (TP + FN)
  - TP: IoU>0.5的检测框数量（同一Ground Truth只计算一次）
  - FP: IoU<=0.5的检测框，或者是检测到同一个Ground Truth的多余检测框的数量
  - FN: 没有检测到的Ground Truth的数量
  - IoU: “预测的边框” 和 “真实的边框” 的交集和并集的面积比值
- 修改GitHub开源代码，配适本项目，实现相关指标的计算与可视化

GitHub项目地址：https://github.com/Cartucho/mAP
参考资料：https://www.zhihu.com/question/53405779/answer/419532990

Parameters

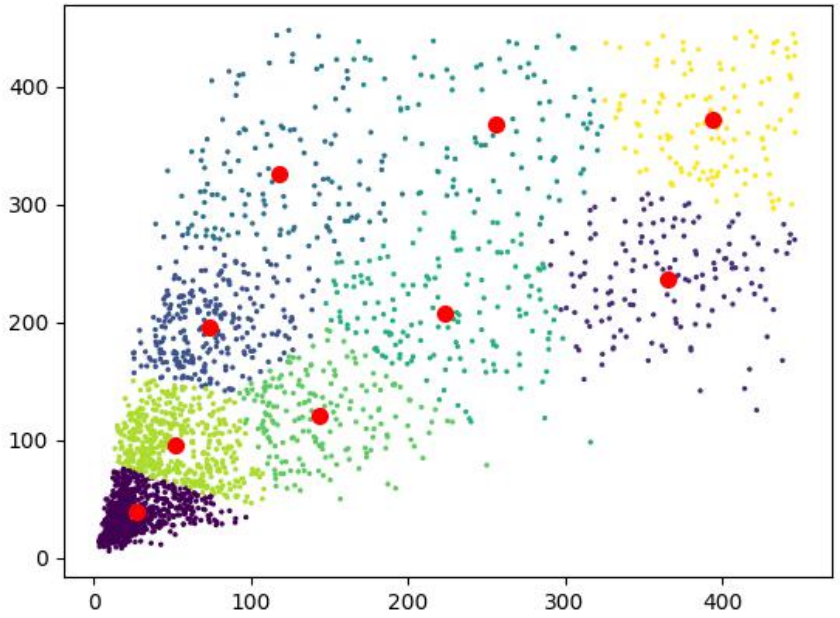
参数	含义
anchors	预设一组不同尺度不同、位置的固定参考框，将目标检测问题转换为"这个固定参考框中有没有认识的目标，目标框偏离参考框多远"
ignore_thresh	与训练过程中loss的计算有关，IoU超过该阈值的负样本不参与loss计算
nms_thresh	nms算法中使用的阈值，可通俗理解为”多大程度上允许同类型框的重叠”
valid_thresh	过滤掉置信度低的框

ignore_thresh

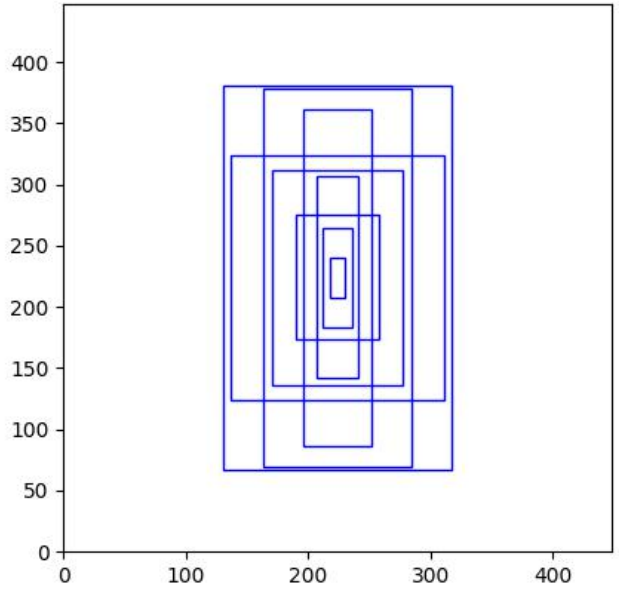


各ignore_thresh组，调节不同epoches、nms_thresh、valid_thresh取得的最大mAP

Anchors



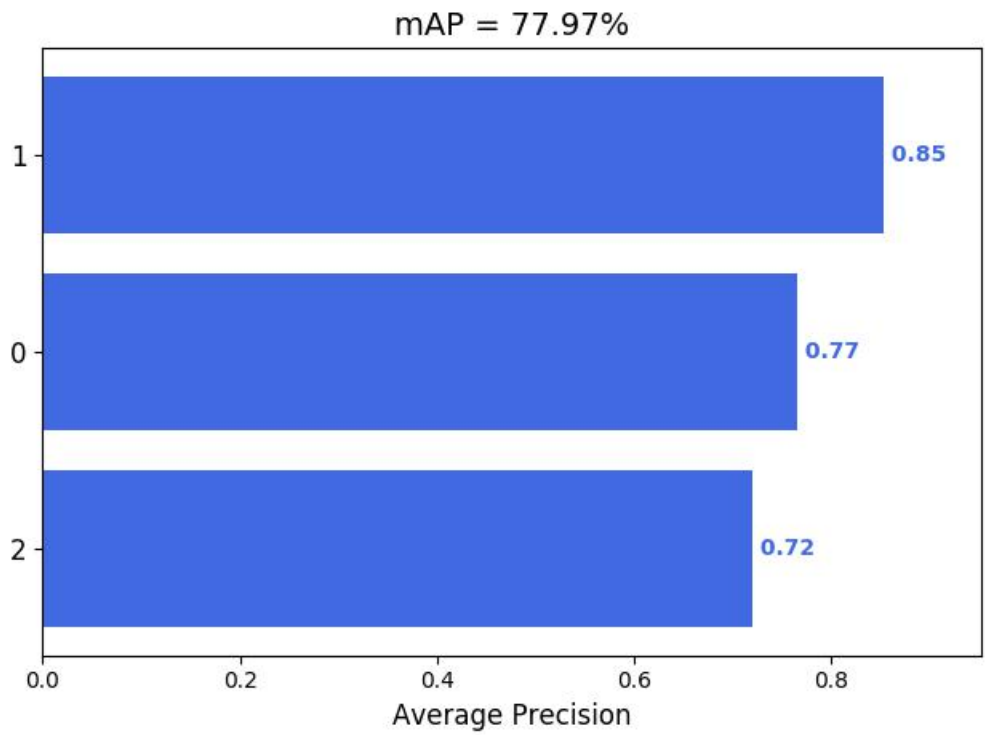
KMeans聚类结果中心点分布

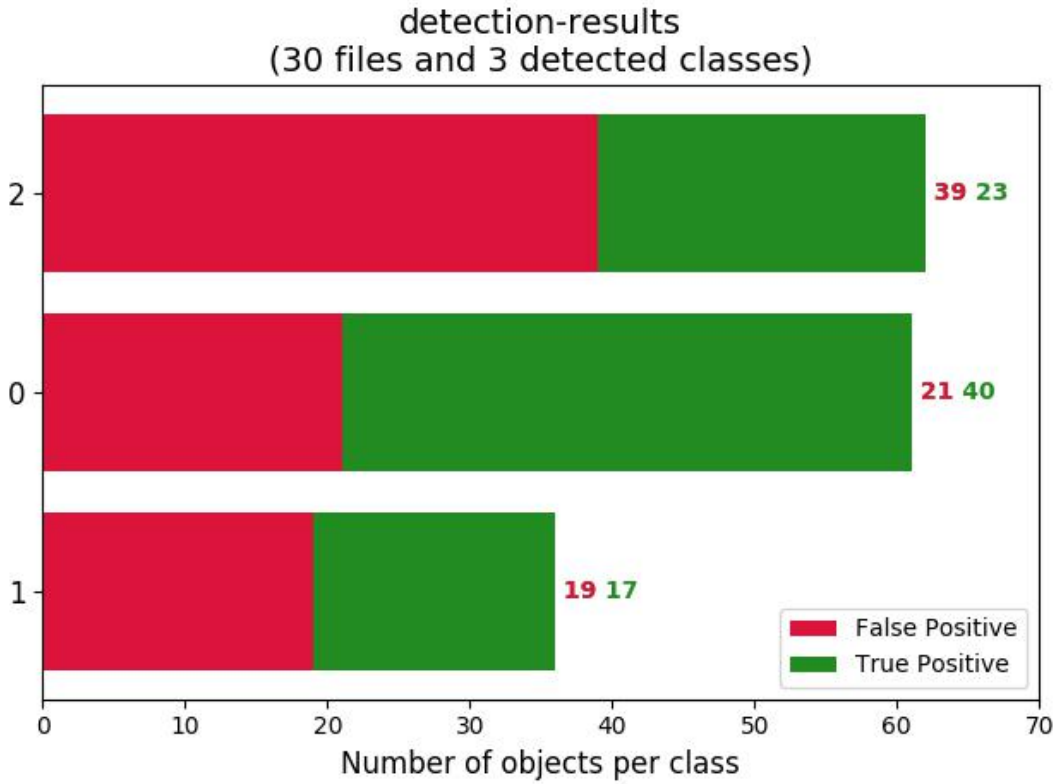


anchor预测框大小可视化

Github参考： https://github.com/decanbay/YOLOv3-Calculate-Anchor-Boxes

Anchors





Our Best Result

800pics dataset, cspnet, sgd, custom_anchor, ignore0.55, 85 epochs

valid_thresh

valid_thresh = 0.01
FP占比极高

valid_thresh = **0.07**
精准干净的预测

Strategy

姓名缩写	模型(原始/CSP)	num_epochs	ignore_thresh	nms_thresh	valid_thresh	数据集（400/800	优化器	anchor(原始/重新	class 0	class 1	class 2	map
wxr	CSP	140	0.60	0.407	0.010	400	SGD	原始	65.00%	69.00%	85.00%	73.04%
wxr	CSP	180	0.60	0.407	0.010	400	SGD	原始	58.00%	77.00%	80.00%	71.34%
wxr	CSP	120	0.60	0.407	0.010	400	SGD	原始	49.00%	78.00%	77.00%	68.10%
wxr	CSP	120	0.60	0.389	0.010	400	SGD	原始	48.00%	78.00%	77.00%	67.73%
lmq	CSP	80	0.58	0.450	0.010	400	SGD	原始	66.00%	74.00%	82.00%	74.00%
zyw	CSP	120	0.54	0.450	0.010	400	SGD	原始	55.00%	41.00%	75.00%	57.00%
wtc	CSP	85	0.55	0.450	0.070	800	SGD	重新计算	76.00%	85.00%	72.00%	78.00%
lmq	CSP	85	0.58	0.400	0.010	400	SGD	原始	61.00%	75.00%	81.00%	72.00%
wtc	CSP	90	0.55	0.450	0.010	400	Adam	原始	48.00%	66.00%	68.00%	61.00%
lmq	CSP	85	0.58	0.350	0.010	400	SGD	原始	61.00%	74.00%	81.00%	72.00%
zxy	CSP	100	0.50	0.450	0.010	400	SGD	原始	65.00%	72.00%	75.00%	70.58%
zxy	CSP	100	0.52	0.450	0.010	400	SGD	原始	58.00%	70.00%	77.00%	68.14%
wwx	CSP	145	0.62	0.407	0.010	400	SGD	原始	51.00%	52.00%	78.00%	60.42%
wwx	CSP	135	0.62	0.407	0.010	400	SGD	原始	50.00%	57.00%	74.00%	59.95%
wwx	原始	120	0.60	0.389	0.010	400	SGD	原始	42.00%	49.00%	69.00%	53.60%
wwx	原始	170	0.60	0.400	0.010	400	SGD	原始	59.00%	61.00%	73.00%	64.40%
lmq	原始	100	0.50	0.450	0.010	400	SGD	原始	59.00%	48.00%	72.00%	59.90%
lph	CSP	80	0.56	0.450	0.010	400	SGD	原始	60.00%	64.00%	81.00%	68.50%
lph	CSP	110	0.55	0.438	0.010	400	SGD	原始	62.59%	81.32%	83.55%	75.80%

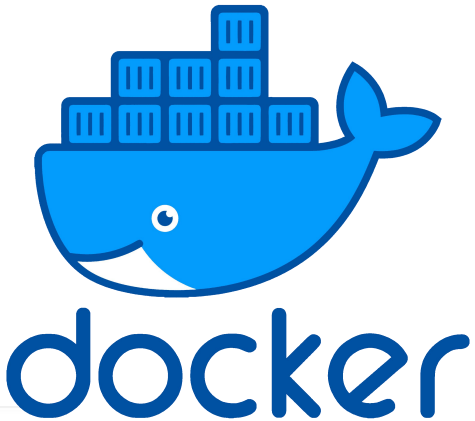
部分训练成果概览

OutOfTimeErr

本周剩余0.0小时GPU环境使用时间

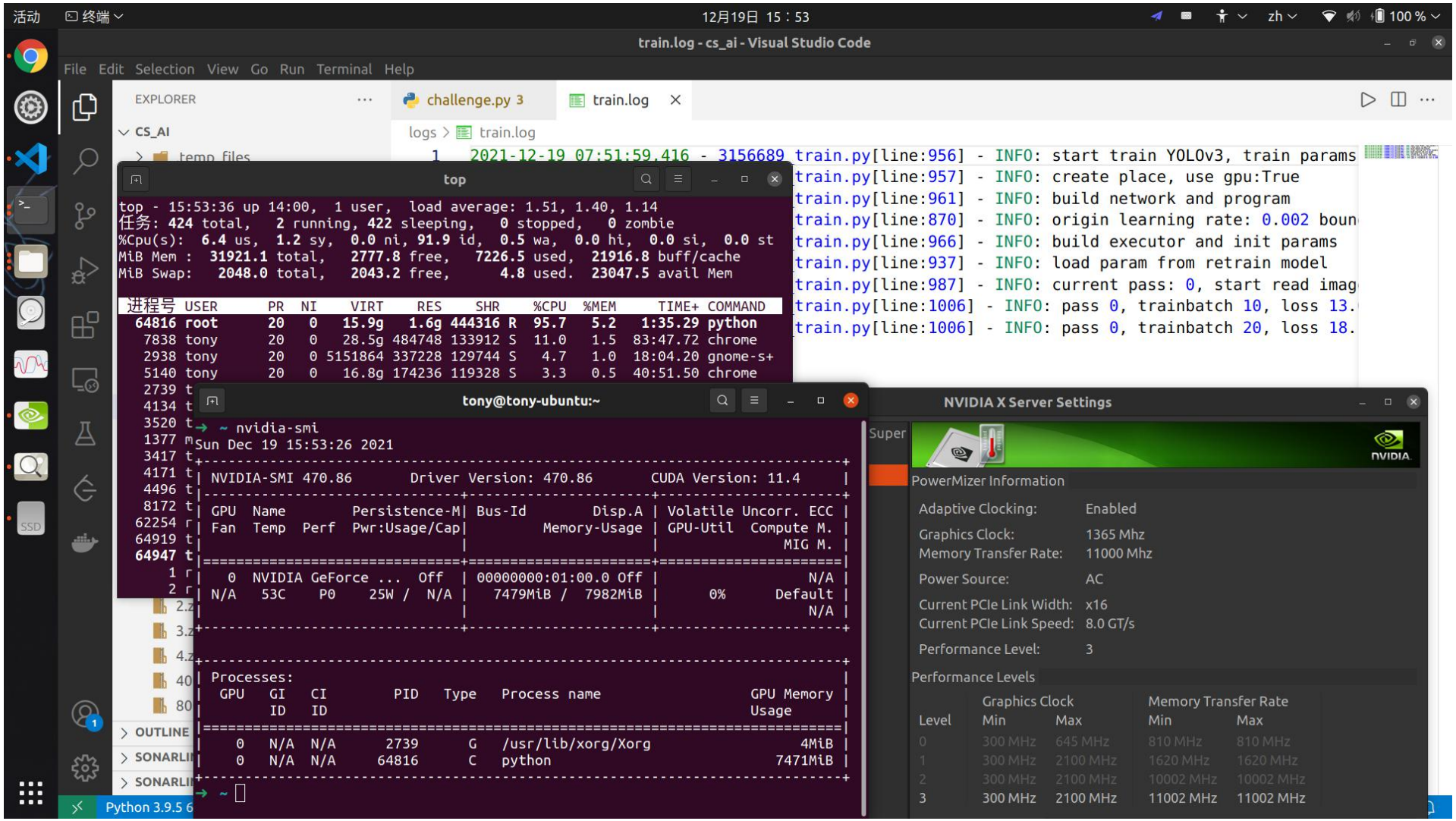
OutOfTimeErr

Docker Env

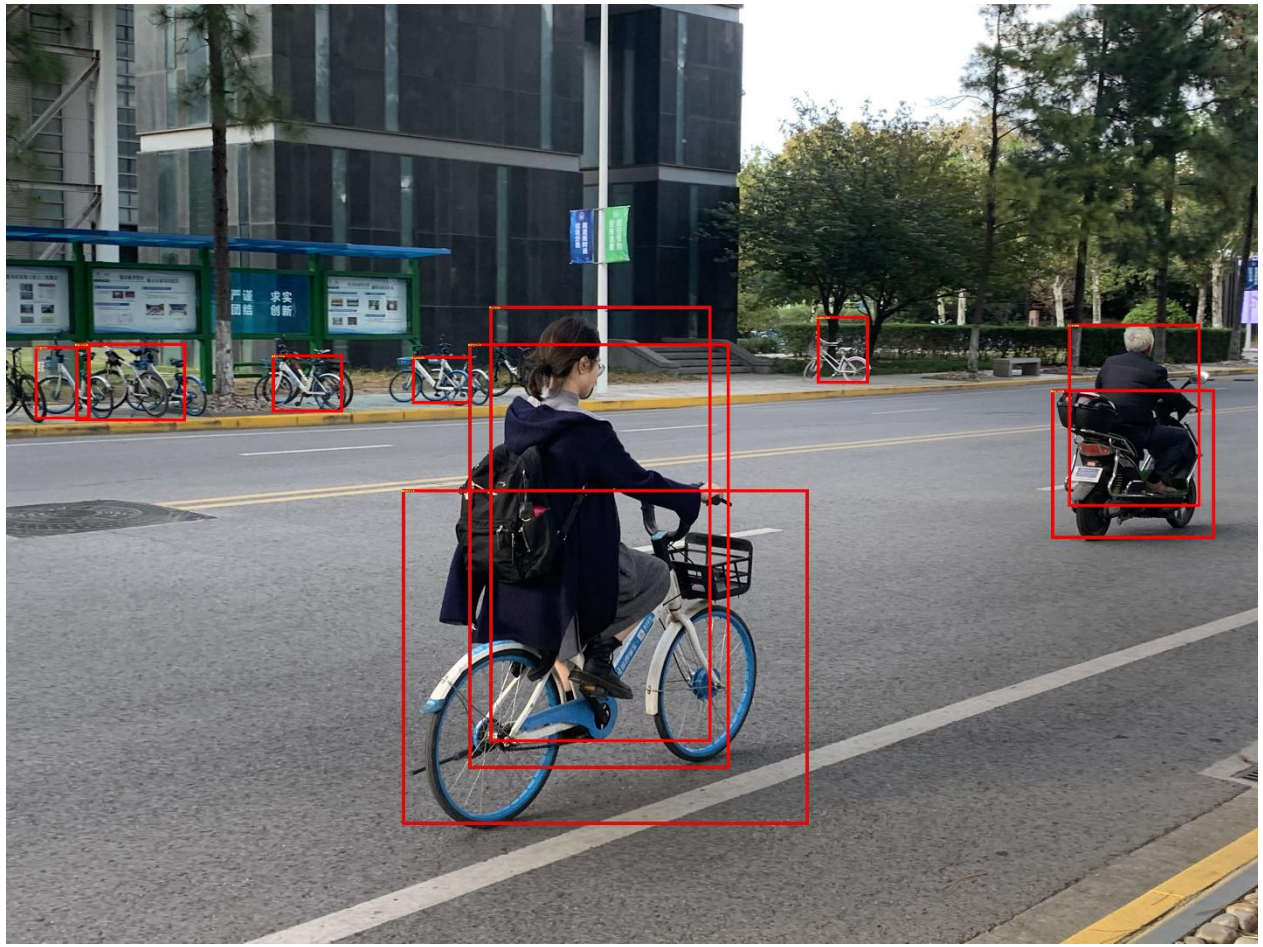

docker

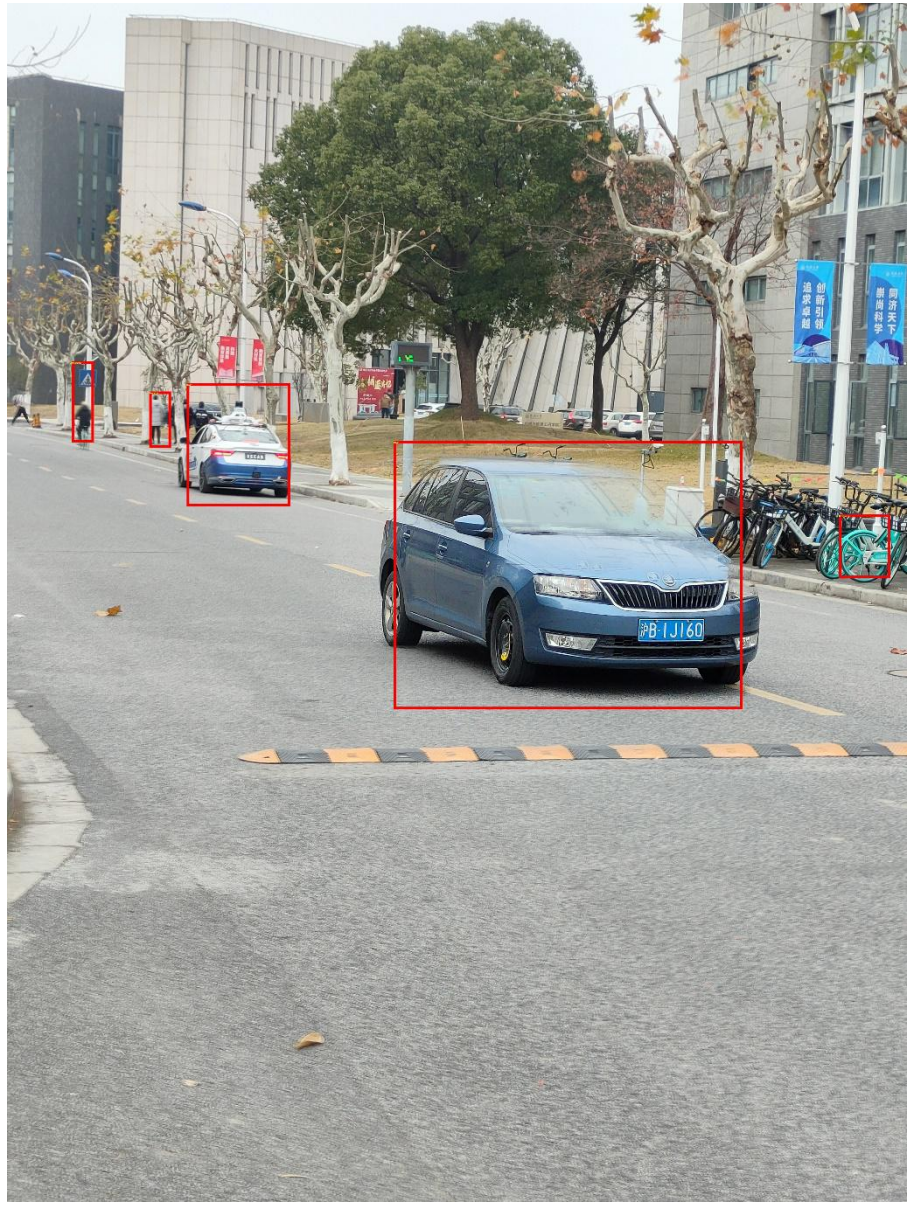

1.5
新手入门
使用指南
进阶使用
API
环境变量FLAGS
FAQ
快速开始
安装说明
Ubuntu下安装
CentOS下安装
MacOS下安装
Windows下安装
使用conda安装
使用Docker安装
从源码编译
附录
深度学习基础教程
Fluid编程指南
文档 > 新手入门 > 安装说明 > 使用Docker安装
使用Docker安装
Docker是一个开源的应用容器引擎。使用Docker，既可以将PaddlePaddle的安装&使用与系统环境隔离，也可以与主机共享GPU、网络等资源
环境准备
目前支持的系统类型，请见安装说明，请注意目前暂不支持在CentOS 6使用Docker
在本地主机上安装Docker
如需在Linux开启GPU支持，请安装nvidia-docker
安装步骤
1. 拉取PaddlePaddle镜像
CPU版的PaddlePaddle: docker pull hub.baidubce.com/paddlepaddle/paddle:[版本号]
GPU版的PaddlePaddle: docker pull hub.baidubce.com/paddlepaddle/paddle:[版本号]-gpu-cuda9.0-cudnn7
如果您的机器不在中国大陆地区，可以直接从DockerHub拉取镜像:
CPU版的PaddlePaddle: docker pull paddlepaddle/paddle:[版本号]
GPU版的PaddlePaddle: docker pull paddlepaddle/paddle:[版本号]-gpu-cuda9.0-cudnn7

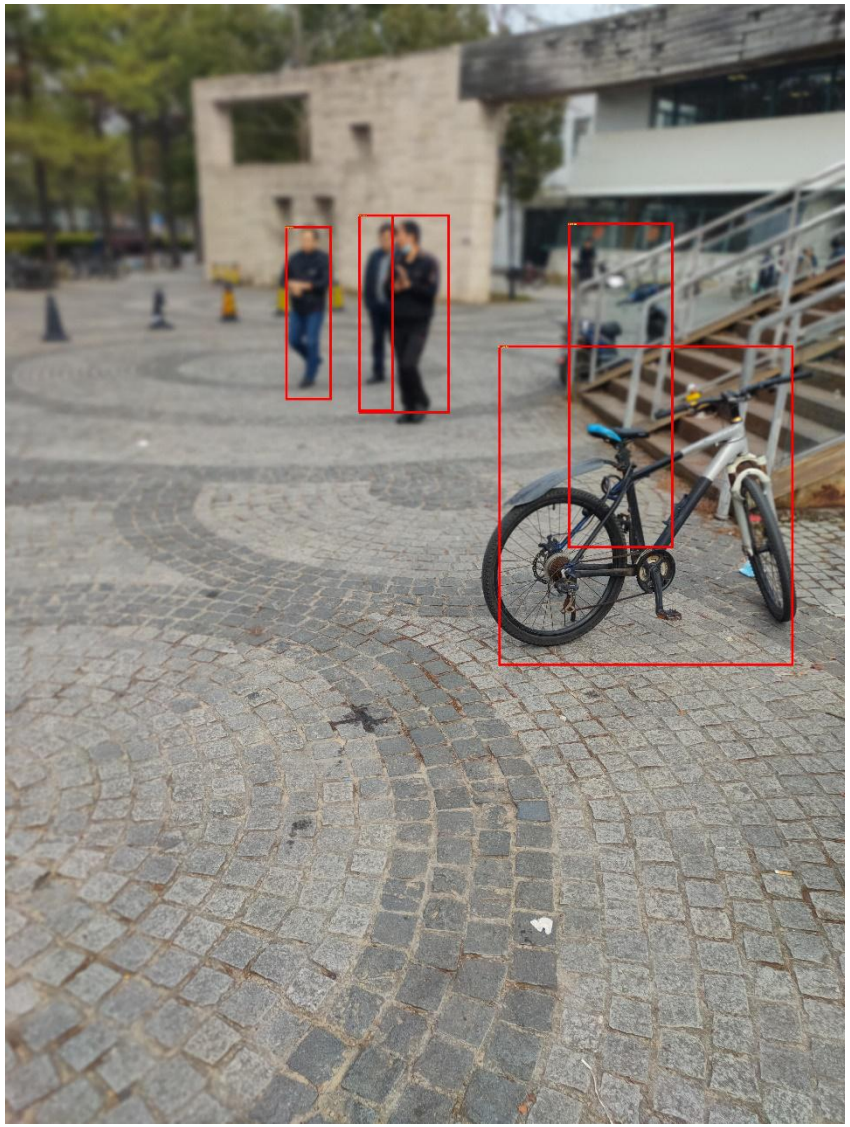
Docker Env

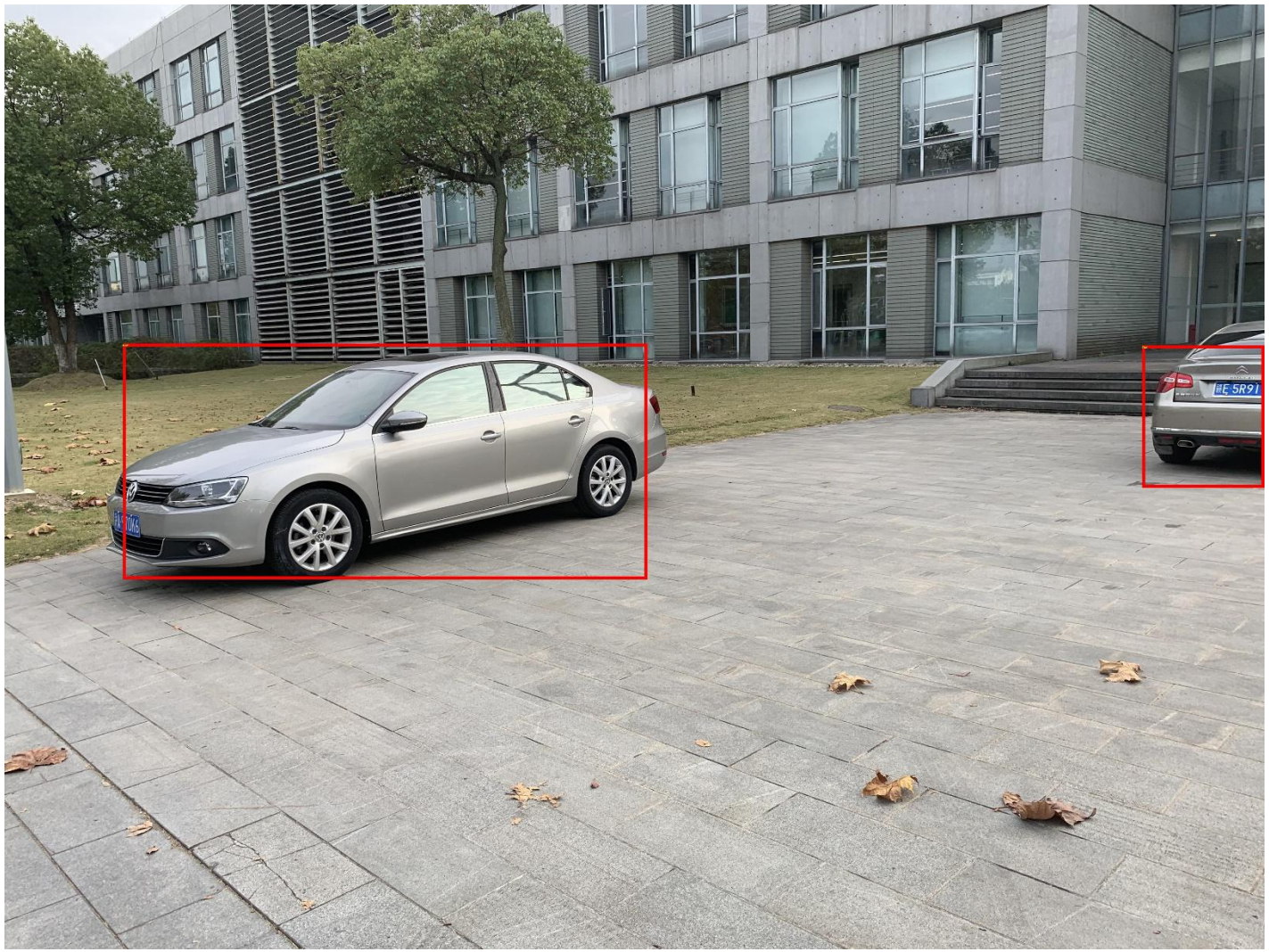

活动
终端
12月19日 15：53
zh
100 %
train.log - cs_ai - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
CS_AI
challenge.py 3
train.log
logs > train.log
1 2021-12-19 07:51:59,416 - 3156689 train.py[line:956] - INFO: start train YOLOv3, train params
train.py[line:957] - INFO: create place, use gpu:True
train.py[line:961] - INFO: build network and program
train.py[line:870] - INFO: origin learning rate: 0.002 boun
train.py[line:966] - INFO: build executor and init params
train.py[line:937] - INFO: load param from retrain model
train.py[line:987] - INFO: current pass: 0, start read imag
train.py[line:1006] - INFO: pass 0, trainbatch 10, loss 13.
train.py[line:1006] - INFO: pass 0, trainbatch 20, loss 18.
top
top - 15:53:36 up 14:00, 1 user, load average: 1.51, 1.40, 1.14
任务: 424 total, 2 running, 422 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.4 us, 1.2 sy, 0.0 ni, 91.9 id, 0.5 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 31921.1 total, 2777.8 free, 7226.5 used, 21916.8 buff/cache
MiB Swap: 2048.0 total, 2043.2 free, 4.8 used. 23047.5 avail Mem
进程号 USER PR NI VIRT RES SHR %CPU %MEM TIME+ COMMAND
64816 root 20 0 15.9g 1.6g 444316 R 95.7 5.2 1:35.29 python
7838 tony 20 0 28.5g 484748 133912 S 11.0 1.5 83:47.72 chrome
2938 tony 20 0 5151864 337228 129744 S 4.7 1.0 18:04.20 gnome-s+
5140 tony 20 0 16.8g 174236 119328 S 3.3 0.5 40:51.50 chrome
tony@tony-ubuntu:~
~ nvidia-smi
Sun Dec 19 15:53:26 2021
NVIDIA-SMI 470.86 Driver Version: 470.86 CUDA Version: 11.4
GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC
Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M.
MIG M.
0 NVIDIA GeForce ... Off 00000000:01:00.0 Off N/A
N/A 53C P0 25W / N/A 7479MiB / 7982MiB 0% Default
N/A
Processes:
GPU GI ID CI ID PID Type Process name GPU Memory Usage
0 N/A N/A 2739 G /usr/lib/xorg/Xorg 4MiB
0 N/A N/A 64816 C python 7471MiB
NVIDIA X Server Settings
Super
NVIDIA
PowerMizer Information
Adaptive Clocking: Enabled
Graphics Clock: 1365 Mhz
Memory Transfer Rate: 11000 Mhz
Power Source: AC
Current PCIe Link Width: x16
Current PCIe Link Speed: 8.0 GT/s
Performance Level: 3
Performance Levels
Level
Graphics Clock Min Max
Memory Transfer Rate Min Max
0 300 MHz 645 MHz 810 MHz 810 MHz
1 300 MHz 2100 MHz 1620 MHz 1620 MHz
2 300 MHz 2100 MHz 10002 MHz 10002 MHz
3 300 MHz 2100 MHz 11002 MHz 11002 MHz
OUTLINE
SONARLI
SONARLI
Python 3.9.5 6

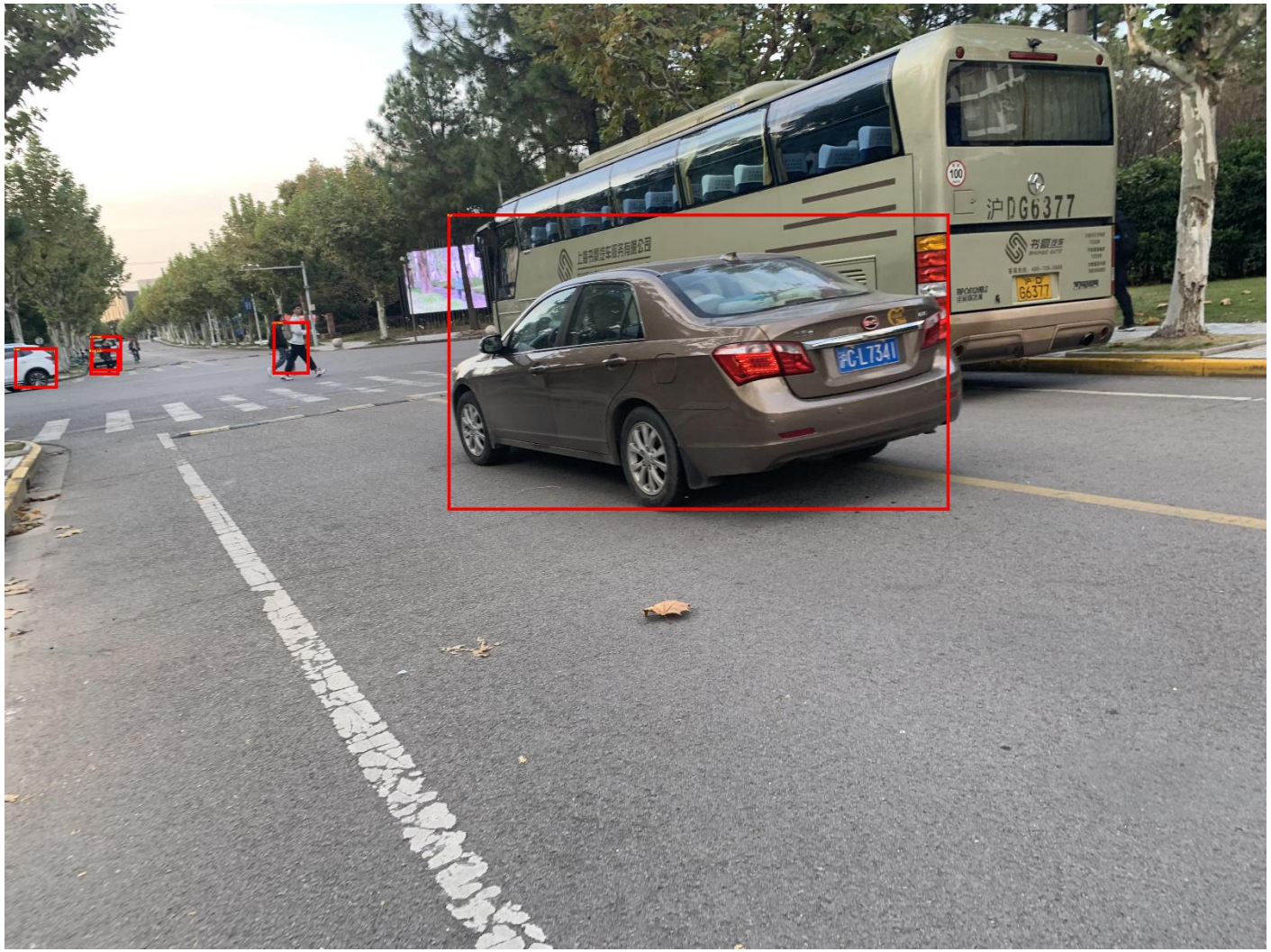
Show Time

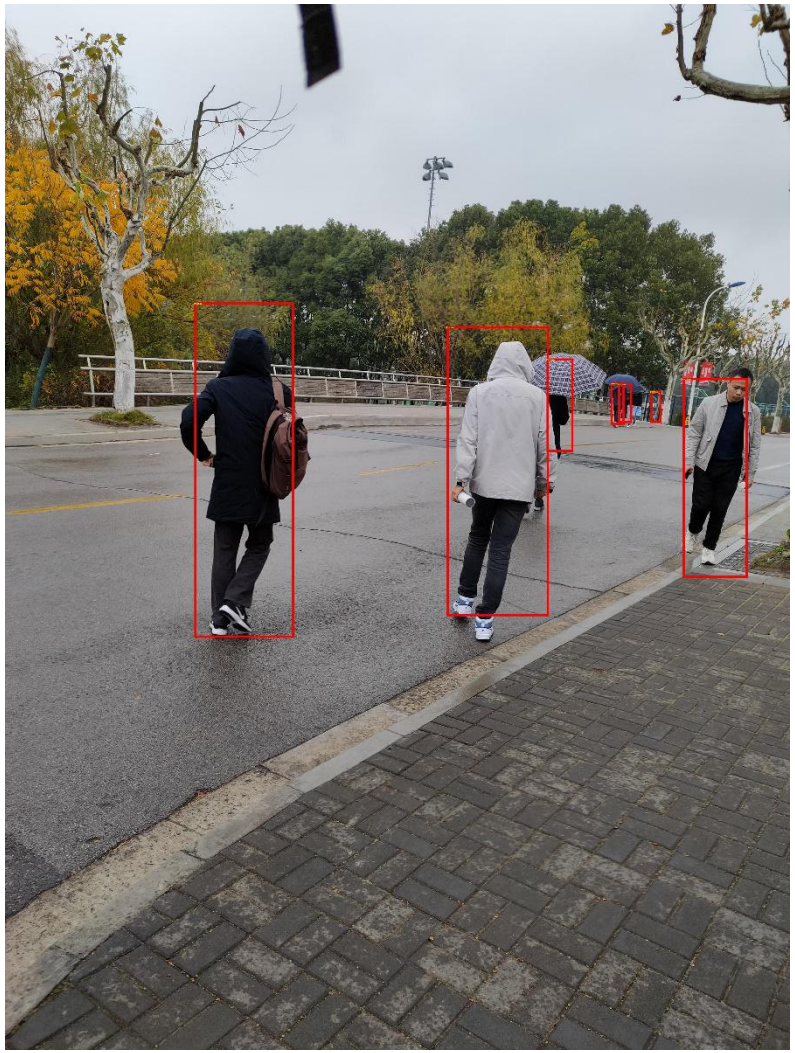


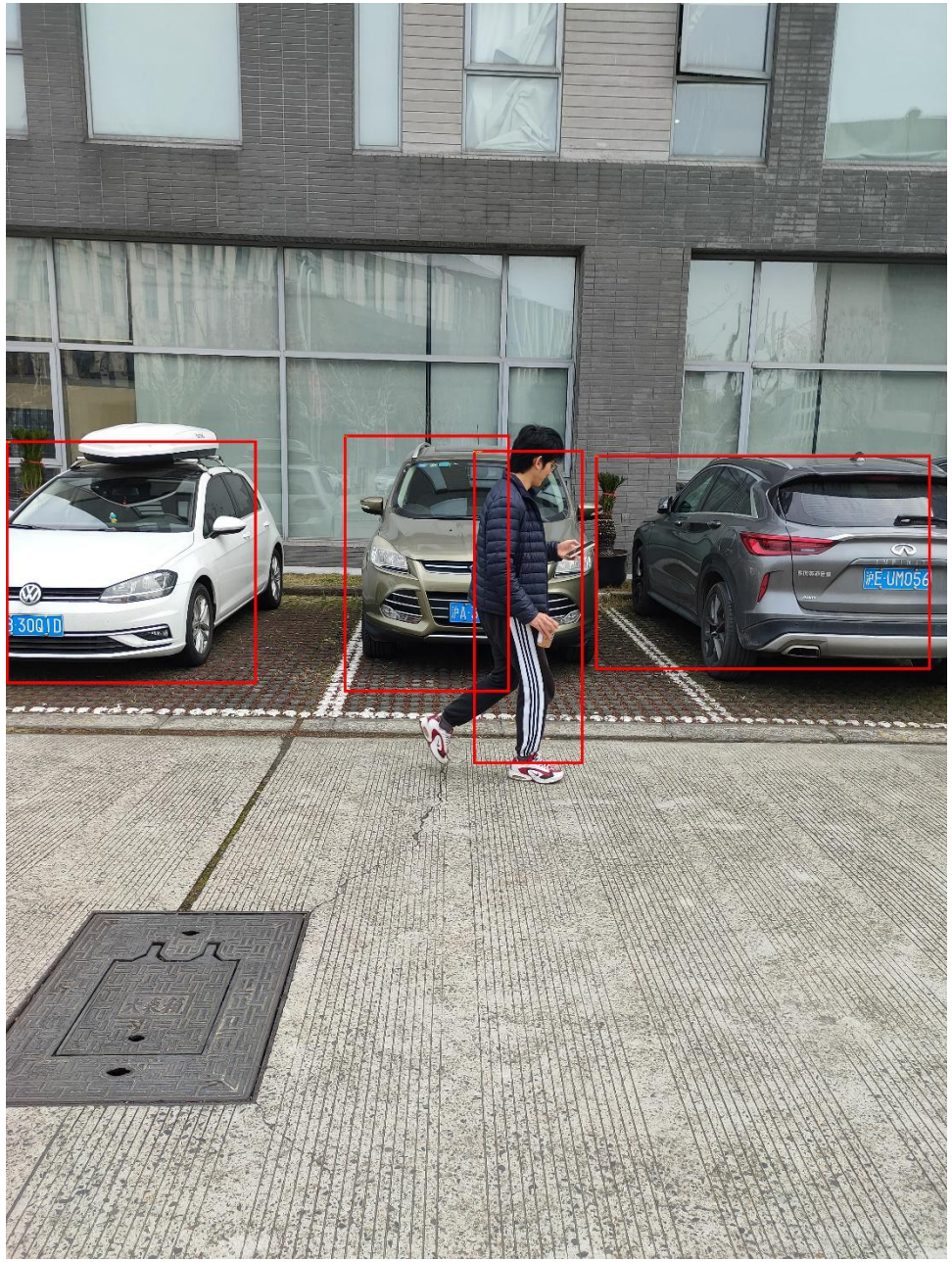

BIJ160

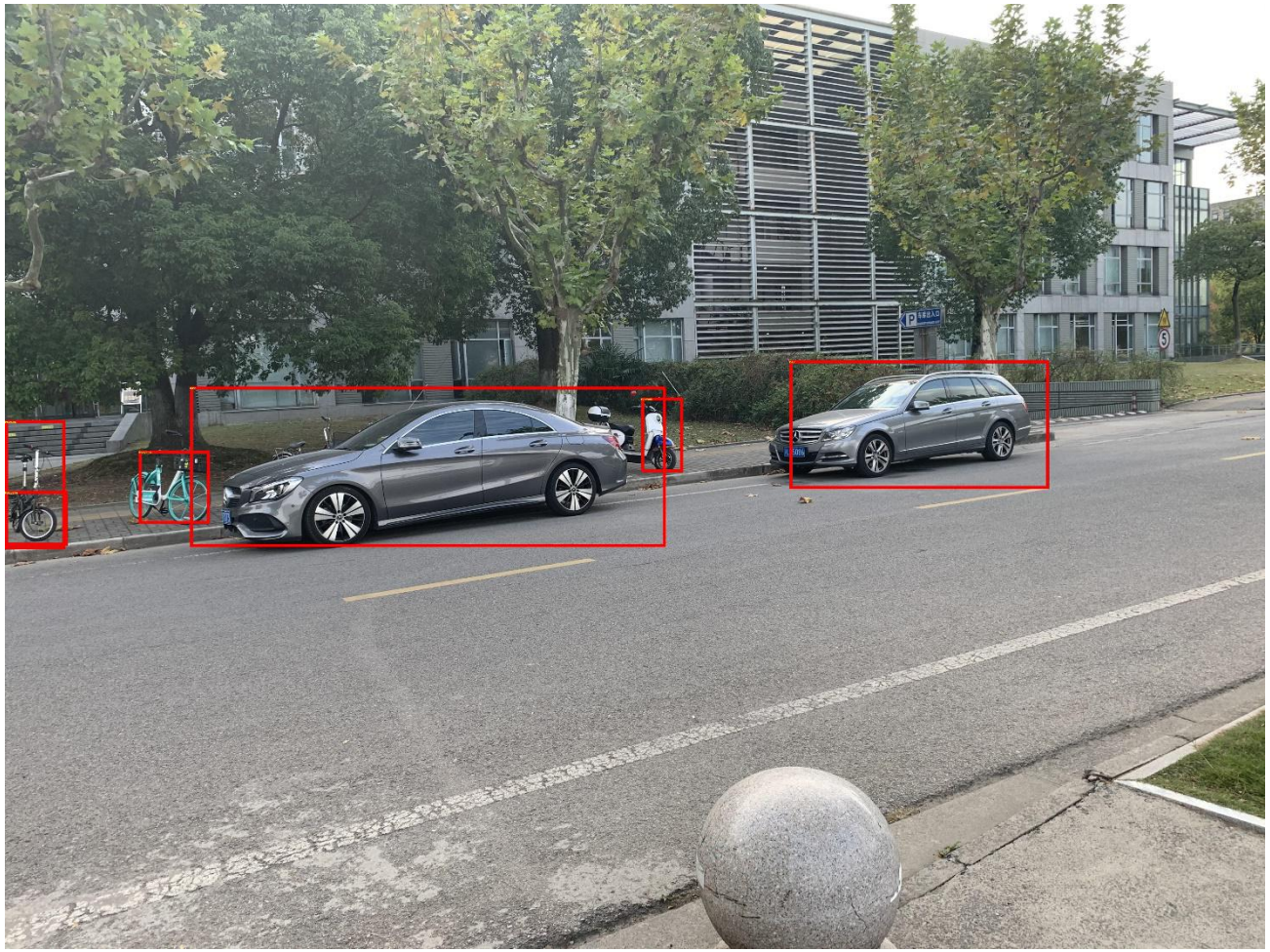





沪DG6377
G6377
FC-L7341







Thanks For Listening
&
Thanks to my teammates