上海大学

SHANGHAI UNIVERSITY

2021-2022 学年夏季学期

《智能系统联合大作业(0869A006)》课程报告

学　　号：20123101

学　　生：李昀哲

指导教师：沈俊、王欣芝

平时成绩 30%	项目成绩 20%	比赛成绩 20%	报告成绩 30%	总成绩

计算机工程与科学学院

2022 年 5 月 26 日

《智能系统联合大作业(0869A006)》课程报告

一、模型设计

本报告将主要就“安全帽检测”任务进行分析。模型选择上，由于该任务的核心关键在于目标检测，因此考虑 R-CNN 和 YOLO 系列模型。经过测试发现：R-CNN 在训练时是一个多阶段的过程，各个 pipline 是隔离的，很耗费时间和空间；同时，R-CNN 由于需要先选取候选框，再对候选框内的进行分类和回归这样的两步法（two-stage），就会导致物体检测速度很慢。因此选择了 YOLO 系列模型完成该任务。

YOLO 系列模型相比于上述提到的 R-CNN 的两步法（two-stage）的局限性，YOLO 系列模型的主要思路则是均匀地在图片的不同位置进行密集采样，采样时可以采用不同的尺度和长宽比，利用 CNN 提取特征后进行分类和回归。整个过程只需要一步，因此也称为一步法（one-stage），速度也会相较于 R-CNN 更快。

本任务的基线模型选择了 YOLOv3，YOLOv3 在 YOLOv2 的基础上，改良了网络的主干利用多尺度特征图进行检测，改进了多个独立的 Logistic regression 分类器来取代 softmax 来预测类别分类。主干网络 YOLOv3 选择了 Darknet-53，从第 0 层到第 74 层，一共有 53 层卷积层，其余均为 Resnet 层。

和 Darknet-19 相比，Darknet-53 去除了所有的 maxpooling 层，增加了更多的 1X1 和 3X3 的卷积层，但因为加深网络层数很容易导致梯度消失或爆炸，所以 Darknet-53 加入了 ResNet 中的 Residual 块来解决梯度的问题。

由图 1 的 Darknet-53 架构可以看到共加入了 23 个 Residual block。

	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			

图 1 Darknet-53 架构

由于加深了网络层数,Darknet-53 相比 Darknet-19 慢了许多，但是 Darknet-53 处理速度 78fps，还是比同精度的 ResNet 快很多，如表 1 所示，YOLOv3 依然保持了高性能，。

表 1 不同网络性能对比

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19	74.1	91.8	7.29	1246	**171**
ResNet-101	77.1.	93.7	19.7	1039	53
ResNet-152	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

在多尺度预测层面，借鉴了 FPN 的方法，采用多尺度的特征图对不同大小的物体进行检测，以提升小物体的预测能力。通过下采样 32 倍、16 倍得到 2 个不同尺度的特征图,例如输入 416X416 的图像,则会得到 13X13(416/32), 26X26(416/16), 这 2 个尺度的特征图。如图 2 所示:

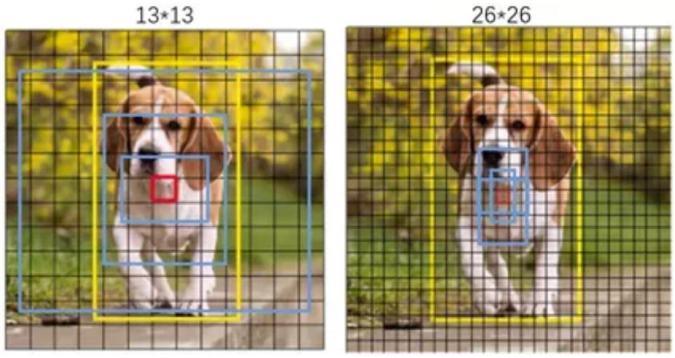


图 2 不同尺度下的特征图

每个尺度的特征图会预测出 3 个 Anchor prior, 而 Anchor prior 的大小则采用 K-means 进行聚类分析。在 COCO 数据集上，按照输入图像的尺寸为 416X416，得到 9 种聚类结果(Anchor prior 的 wxh): (10X13), (16X30), (33X23), (30X61), (62X45), (59X119), (116X90), (156X198), (373X326)。

13X13 的特征图(有较大的感受野)用于预测大物体,所以用较大的 Anchor prior 即(116X90), (156X198), (373X326)。26X26 的特征图(中等的感受野)用于预测中等大小物体,所以用中等的 Anchor prior 即(30X61), (62X45), (59X119)。

模型结构如图 3 所示，图中可以清晰的看到三个预测层分别来自的什么地方，以及 Concatenate 层与哪个层进行拼接。

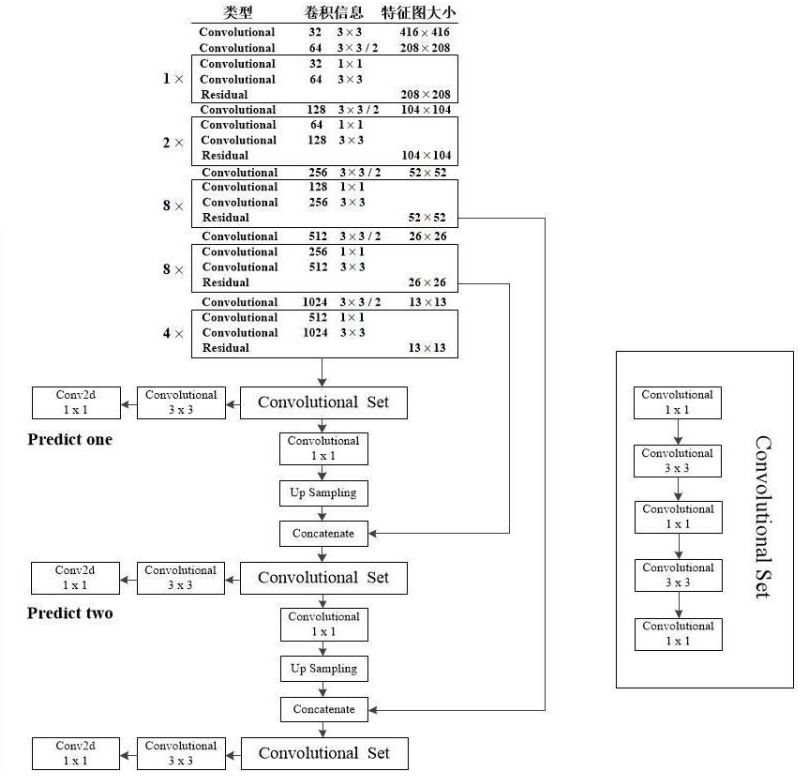


图 3 模型结构

框的预测上，使用 Logistic regression 来预测每个 bounding box 的 confidence，以 bounding box 与 gt 的 IOU 为判定标准，对每个 gt 只分配一个最好的 bounding box。通过利用这种方式,在做 Detect 之前可以减少不必要的 Anchor 进而减少计算量。

损失函数的计算，主要分为三个部分：目标定位偏移量损失 $L_{loc}(t,g)$，目标置信度 $L_{conf}(o,c)$以及目标分类损失 $L_{cla}(O,C)$, $\lambda_1, \lambda_2, \lambda_3$ 是平衡系数。

$$L(O,o,C,c,t,g) = \lambda_1 L_{conf}(o,c) + \lambda_2 L_{cla}(O,C) + \lambda_3 L_{loc}(t,g)$$

二、实验结果

训练过程

- 数据切分：将训练集和验证集按照 8.5：1.5 的比例划分；
- 定义数据路径；
- 模型初始化；
- 模型训练；
- 以 COCO 指标作为评价指标。

参数设计

批次、轮次、迭代等参数的设计如下所示：

```
model.train(
        num_epochs=200,  # 训练轮次
        train_dataset=train_dataset,  # 训练数据
        eval_dataset=eval_dataset,  # 验证数据
        train_batch_size=16,  # 批大小
        pretrain_weights='COCO',
        learning_rate=0.005 / 12,  # 学习率
        warmup_steps=500,  # 预热步数
        warmup_start_lr=0.0,  # 预热起始学习率
        save_interval_epochs=5,  # 每 5 个轮次保存一次，有验证数据时，自动评估
        lr_decay_epochs=[85, 135],  # step 学习率衰减
        save_dir='output/yolov3_darknet53',  # 保存路径
```

模型改进方案

第一次训练后预测的结果并不优秀，因此考虑分析可能的原因进行优化。

1. 基线模型的选择

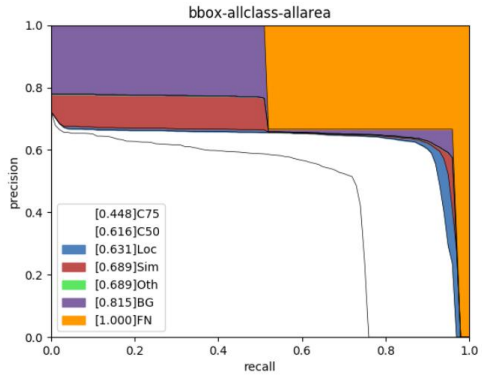
根据第一节“模型设计”所述，由于两步法相较于一步法速度受限，故选择精度略低但速度更快的一步法检测模型，即：以 Darknet53 为骨干网络的 YOLOv3。

表 2 基线模型性能

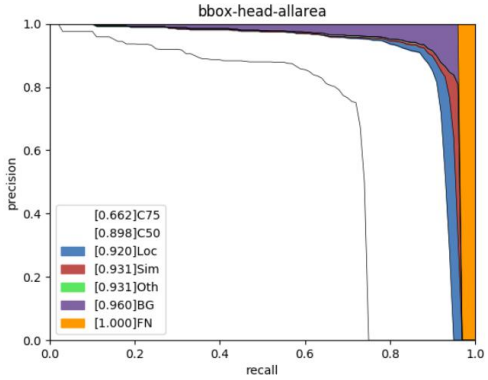
模型	推理时间	Map(Iou-0.5)	(coco)mmap	安全帽
YOLOv3+Darknet-53+cluster_anchor+image	50.34	61.6	39.2	94.58

2. 基线模型效果分析与优化

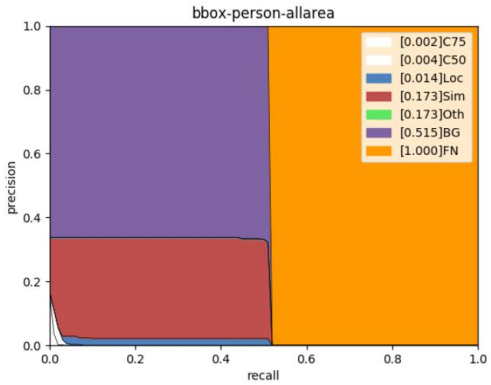
使用 PaddleX 提供的 paddlex.det.coco_error_analysis 接口对模型在验证集上预测错误的原因进行分析，分析结果以图表的形式展示如图 4 所示：



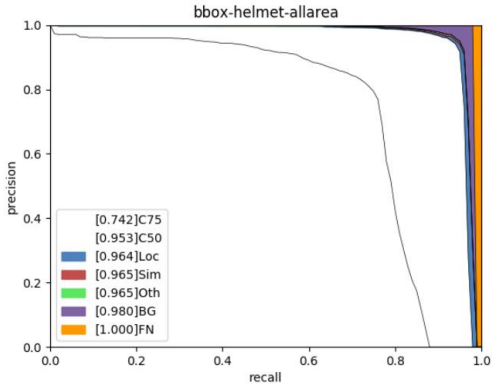
(a) all class



(b) head



(b) person



(d) helmet

图 4 基线模型效果分析

从分析图表中可以看出，head、helmet 两类检测效果较好，但仍然存在漏检的情况，特别是 person 存在很大的漏检问题；此外，通过 helmet 中 C75 指标可以看出，其相对于 C50 的 0.946 而言有些差了，因此定位性能有待进一步提高。为进一步理解造成这些问题的原因，将验证集上的预测结果进行了可视化，然后发现数据集标注存在以下问题：

- 本数据集主要考虑到头部和安全帽的检测，因此在人检测时，有个图片中标注了，而有的图片中没有标注，从而导致学习失效，引发 person 漏检。
- head 与 helmet 大多数情况标注较好，但由于部分拍摄角度导致有的图片中的 head 和 helmet 发生重叠以及太小导致学习有困难。

对于漏检问题，查询相关资料发现是特征学习不够，无法识别出物体，因此基于这个方向，尝试替换 backbone: DarkNet53 --> ResNet50_vd_dcn，在指标上的提升如表 3 所示:

表 3 漏检问题优化后性能指标

模型	推理时间	Map(Iou-0.5)	(coco)mmap	安全帽
YOLOv3+ResNet50_vd_dcn+cluster_anchor +image（480）	53.81	**61.7**	39.1	**95.35**

对于定位问题，通过尝试放大图片，不同的网络结构以及定位的优化策略：利用`cluster_yolo_anchor`生成聚类的 anchor 或开启 iou_aware。最终得到上线模型 PPYOLOV2 的精度如表 4 所示:

表 4 定位问题优化后性能指标

模型	推理时间	Map(Iou-0.5)	(coco)mmap	安全帽
YOLOv3+ResNet50_vd_dcn+cluster_anchor +image（608）	81.52	61.6	**41.3**	95.32

优化总结：

- 通过选用适当更优的骨干网络可以改善漏检的情况，因此漏检方面的优化可以考虑先从骨干网络替换上开始——当然必要的数据清洗也是不可缺少的，要是数据集本身漏标，则会从根本上影响模型的学习。

- 通过放大图像，可以对一些中小目标的物体检测起到一定的优化作用。

- 通过聚类 anchor 以及 iou_aware 等操作可以提高模型的定位能力，直接体现是再高 Iou 上也能有更好的表现。因此，定位不准可以从模型的 anchor 以及模型的结构上入手进行优化。

结果示例

选取一张典型的检测图，可以发现其中检测准确度较高和偏低的特征。对于检测目标较少的情况，整体精度较高，而目标较多时，就会出现漏检、错检的情况。同时，由于应用场景的拍摄需要较大的广度和较长的距离，因此取图时往往目标较小、目标较多，上述两种局限无法避免。

如图 5(a)中从左至右第三、第四个目标所示，由于环境光和各种因素的影响，导致精度下降，但我认为这样的影响可能是无法很好优化的，可能只有增加数据集或者对原有的数据集进行数据增强才能得到较好的效果。图 5(b)这种清晰度较高，目标单一的检测，精度就很客观。

(a) (b)

图 5 安全帽检测示例

三、收获与体会

先就本报告的任务而言，安全帽检测是施工现场重要的安全保障监督设备，利用智能化的检测系统，可以帮助监管者管控安全风险。对于技术的使用，也从 R-CNN 和 YOLO 系列模型进行了对比，虽然可能最后的结果并不是非常好，但在不断的尝试中，提升精度，学到各种技术上、应用中可能出现的局限点，进行优化。我认为这不仅是技术上的实现，更是科研、工程素养的提升。本任务所学也不仅仅是技术，更是处理问题的能力和思考问题的方式，这都是在未来学习、科研、工程中，非常重要的。同时，在不断的迭代和性能分析中，了解到了很多前沿的模型以及创新技术。

同时我认为，模型固然重要，但数据集的获取同样不能忽视。谷歌公司在一次员工会议上提到：“现在太多的人只想要做模型，只有很少的人去做数据，这是非常可怕的。”数据是一切模型的根本，没有数据，再好的模型也是空谈。更多的数据意味着更多的情境，也能更好的帮助模型提升精度。安全帽这个任务的数据集的获取并不容易，因此对于一些目标较多、目标较小、光线不佳的情况，误识别、漏识别就会频繁出现。

计算机视觉是人工智能领域重要的一部分，此项技术也广泛应用于生活、工业、医疗等多种领域。我也在课余时间参加了机器人视觉的研发，但在此次课程之前，我始终觉得自己在网络所学不够系统化，只

是支离破碎的填鸭式学习。但此次课程，不仅让我系统地学习了深度学习和机器学习，也给了我很多实践的机会，同时也提升了查看手册、官方 document 的能力。将所学技术投入应用，虽然现在只是处于起步阶段，但相信有了这一次的学习，会给我在以后的学习中打下坚实的基础。

对于课程的建议，我认为课程中的同学间的交流是非常有益的，但交流的机会和深度是否可以再提升一些，比如现在只是同学单方面的讲，若能加入听众的讨论和提问，这样的学习和交流会变得更有价值，也能激发大家对于问题、创新技术的思考。

四、项目部分源代码

流程代码仅为训练、推理的调用，在此不过多赘述。此处仅列出训练和推理部分代码（列出内容均为修改、调整后的代码）。

训练：训练参数以于“实验结果-参数设计”章节列出。train.py

```
# 定义安全帽检测训练和验证所用的数据集
train_dataset = pdx.datasets.VOCDetection(
        data_dir='/home/aistudio/work/dataset',
        file_list='/home/aistudio/work/dataset/train_list.txt',
        label_list='/home/aistudio/work/dataset/labels.txt',
        transforms=train_transforms,
        shuffle=True)

eval_dataset = pdx.datasets.VOCDetection(
        data_dir='/home/aistudio/work/dataset',
        file_list='/home/aistudio/work/dataset/val_list.txt',
        label_list='/home/aistudio/work/dataset/labels.txt',
        transforms=eval_transforms,
        shuffle=False)

# 初始化模型，并进行训练
num_classes = len(train_dataset.labels)
model = pdx.det.PPYOLOv2(
        num_classes=num_classes,
        backbone='ResNet50_vd_dcn',
        label_smooth=True,
        use_iou_aware=True,
        ignore_threshold=0.6)
```

推理：infer.py

```
...
# 以上部分为读取数据集路径，使用 glob.glob.在此省略
for image_path in images_paths:
        image_name = image_path
        save_name    = image_path[27:-4]


        # 读取图片与获取预测结果
        print(image_name)
        img = cv2.imread(image_name)
        result = model.predict(img)

        # 解析预测结果，并保存到 txt 中
        keep_results = []
        areas = []
        f = open('result.txt', 'a')
        # count = 0
```

```
        for dt in np.array(result):
            cname, bbox, score = dt['category'], dt['bbox'], dt['score']
            if score < 0.5:
                continue
            keep_results.append(dt)
            # count += 1
            f.write(save_name + ' ')
            f.write(str(dt['score'])[:5] + ' ' + str(int(dt['bbox'][0])) + ' ' + str(int(dt['bbox'][1])) + ' ' +
str(int(dt['bbox'][0]+dt['bbox'][2])) + ' ' + str(int(dt['bbox'][1]+dt['bbox'][3])) + ' ' + str(dt['category_id']) + '\n' )
            # f.write(str(dt) + '\n')
            # f.write('\n')
            areas.append(bbox[2] * bbox[3])
        areas = np.asarray(areas)
        sorted_idxs = np.argsort(-areas).tolist()
        keep_results = [keep_results[k]
                        for k in sorted_idxs] if len(keep_results) > 0 else []
```