学生实验报告

学号	1120192933	学院	计算机学院
姓名	李桐	专业	人工智能

基于 mmdetection 的神经网络构建

1 实验目的

（1）理解 mmdetection 在机器学习中的应用。
（2）根据 mmdetection 的基础模型搭建自定义的模型。
（3）学习使用框架进行机器学习研究的方法。

2 实验原理

（1）CascadeRCNN 模型
（2）resnet 模型
（3）FPN 模型
（4）RPNHead 的作用
（5）RoIExtractor 的作用

3 实验条件与环境

要求	名称	版本要求	备注
编程语言	python	3.6 以上	
开发环境	dsw	无要求	
第三方工具包/	opencv-python	4.5 以上	

库/插件			
第三方工具包/库/插件	tqdm	4.32	
第三方工具包/库/插件	mmdetection	1.2	
其他工具	无	无要求	
硬件环境	台式机、笔记本均可	无要求	

4 实验步骤及操作

（1）CascadeRCNN 模型
（2）resnet 模型
（3）FPN 模型
（4）RPNHead 的作用
（5）RoIExtractor 的作用

5 实验数据及其说明

属性（条目）	内容
数据集的名称	淘宝直播商品识别样例数据集
数据集的出处	淘宝直播
数据集的主要内容	训练数据主要由两部分构成，分为直播片段视频帧及对应的讲解文本、商品展示图及商品文本介绍，上述信息将作为算法的输入信息。数据的标注主要包括视频帧和商品图中的服饰信息。视频帧中服饰标注精确到检测框的粒度，主要包括服饰的检测框位置、对应的服饰类别、实例编号、是否为主播试穿、拍摄视角。商品的标注与视频帧标注类似，主要包括商品展示图中的服饰标注和商品的文本描述，其中每一个商品展示图中标注了服饰的检测框位置、对应的服饰类别、实例编号、是否为主播试穿、拍摄视角。视频帧中的商品和商品图中的商品通过实例编号关联起来，同一个实例编号对应同一个商品。
数据集的文件格式	Jpg,mp4,json
图片数	113
视频数	25
是否有缺失值	有
是否有异常值	无
下载 URL	http://tianchi-team.oss-cn-hangzhou.aliyuncs.com/%E6%AF%94%E8%B5%9B%E4%BB%A3%E7%A0%81%26%E6%95%B0%E6%8D%AE%E9%9B%86/%E6%95%B

	0%E6%8D%AE%E9%9B%86-%E6%B7%98%E5%AE%9D%E7%9B%B4%E6%92%AD%E5%95%86%E5%93%81%E8%AF%86%E5%88%AB.zip

6 实验步骤及其代码

步骤序号	1
步骤名称	选择 model 类型，并构建 backbone
步骤描述	backbone 的主要作用就是提取图像特征。这里选定 cascade rcnn 为主干网络，同时选取全部的 stage 特征。
代码及讲解	 选定 model 为 CascadeRCNN，cascade rcnn 的 backbone 选择的是 ResNet50，创建 backbone 的方式是将支持的模型注册到 registry 中，只后再通过 builder 进行实例化。backbone 在 forward 中 outs 取的是多 stage 的输出，先拼成一个 list 在转成 tuple，取哪些 stage 是根据 config 中的 out_indices。backbone 是 4stage,取了所有的 stage。

步骤序号	2
步骤名称	设置 neck 和 rpn_head 参数
步骤描述	Neck 是多特征融合的过程，rpn 和 roi align 是 two-stage detector 中比较关键的两个操作，这两个操作将 two-stage detector 中的两个 stage 连接起来，变成 end-to-end(端到端)的网络，同时也给整个检测方法的性能带来提升。rpn 为 roi align 提供高质量的候选框，即 proposal。
代码及讲解	FPN（feature pyramid networks）即多尺度的 object detection 算法，是 2017CVPR 提出来的。原来多数的 object detection 算法都是只采用顶层特征做预测，但我们知道低层的特征语义信息比较少，但是目标位置准确；高层的特征语义信息比较丰富，但是目标位置比较粗略。另外虽然也有些算法采用多尺度特征融合的方式，但是一般是采用融合后的特征做预测，而本文不一样的地方在于预测是在不同特征层独立进行的。 FPN 中的 in_channels 与之前 backbone 的输出相匹配，out_chan

	nels 为输出纬度。这部分也可以看成是在提取特征，到下面 RPN 部分就真正涉及到目标检测了。
	`neck=dict(`
	`  typr='FPN',`
	`  in_channels=[256,512,1024,2048],`
	`  out_channels=256,`
	`  num_outs=5`
	`),`
	cascade rcnn 的 rpn_head 乍一看感觉还挺简单的，因为这部分主要就两个网络。主要涉及到两个文件 mmdet/models/anchor_head/anchor_head.py 和 mmdet/models/anchor_head/rpn_head.py 后者是前者的子类。
	rpn_head 就只有两个网络，判断是否是前景(rpn_cls)，预测框的修改值(rpn_reg)。并且其中 self.num_anchors = len(self.anchor_ratios) * len(self.anchor_scales)。
	但是 RPN 的目标是得到候选框，所以这里就还要用到 anchor_head.py 中的另一个函数 get_bboxs()。
	`rpn_head=dict(`
	`  type='RPNHead',`
	`  in_channels=256,`
	`  feat_channels=256,`
	`  anchor_scales=[8],`
	`  anchor_ratios=[0.5,1.0,2.0],`
	`  anchor_strides=[4,8,16,32,64],`
	`  target_means=[.0,.0,.0,.0],`
	`  target_stds=[1.0,1.0,1.0,1.0],`
	`  loss_cls=dict(`
	`  type='CrossEntropyLoss',use_sigmoid=True,loss_weight=1.0 ),`
	`  loss_bbox=dict(type='SmoothL1Loss',beta=1.0/9.0,loss_weight=1.0)`
	`),`

步骤序号	3
步骤名称	设置 roiExtractor 和全连接层参数
步骤描述	这个阶段是在 rpn 提供的 proposal 的基础上，筛选出第二阶段的训练样本，并提取相应的特征，用于组建第二阶段的训练网络。

代码及讲解	`bbox_roi_extractor=dict(`
	`  type='SingleRoIExtractor',`
	`  roi_layer=dict(typr='RoIAlign',out_size=7,sample_num=2),`
	`  out_channels=256,`
	`  featmap_strides=[4,8,16,32]`
	`),`
	这里的 roi_layers 用的是 RoIAlign,RoI 的结果就可以送到 bbox head 了。bbox head 部分和之前的 rpn 部分的操作差不多，主要是针对每个框进行分类和坐标修正。之前 rpn 分为前景和背景两类，这里分为 N+1 类(实际类别 + 背景)。具体代码在 mmdet/models/bbox_head/convfc_bbox_head.py。
	mask 部分的流程和 bbox 部分相同，也是先对之前的候选框先做一次 RoI Pooling，这里的 RoI 与之前 bbox 网络都一样只是部分参数不同。

```
bbox_head=[
    dict(
        typr='SharedFCBBoxHead',
        num_fcs=2,
        in_channels=256,
        fc_out_channels=1024,
        roi_feat_size=7,
        num_calsses=2,
        target_means=[.0, .0, .0, .0],
        target_stds=[1.0, 1.0, 1.0, 1.0],
        reg_class_agnostic=True,
        loss_cls=dict(
        type='CrossEntropyLoss',use_sigmoid=True,loss_weight=1.0    ),
        loss_bbox=dict(type='SmoothL1Loss',beta=1.0,loss_weight=1.0)
    ),
```

```
    mask_roi_extractor=dict(
        type='SingleRoIExtractor',
        roi_layer=dict(type='RoIAlign', out_size=14, sample_num=2),
        out_channels=256,
        featmap_strides=[4, 8, 16, 32]),
    mask_head=dict(
        type='FCNMaskHead',
        num_convs=4,
        in_channels=256,
        conv_out_channels=256,
        num_classes=4,
        loss_mask=dict(
            type='CrossEntropyLoss', use_mask=True, loss_weight=1.0))
```

步骤序号	4
步骤名称	设置模型训练参数
步骤描述	设定训练以及测试的时候各个部分的参数
代码及讲解	 设定参数，如需要选择多少个框框、正负样本的阈值、比例、权重。

```
# model training and testing settings
train_cfg = dict(
    rpn=dict(
        assigner=dict(
            type='MaxIoUAssigner',
            pos_iou_thr=0.7,
            neg_iou_thr=0.3,
            min_pos_iou=0.3,
            ignore_iof_thr=-1),
        sampler=dict(
            type='RandomSampler',
            num=256,
            pos_fraction=0.5,
            neg_pos_ub=-1,
            add_gt_as_proposals=False),
        allowed_border=0,
        pos_weight=-1,
        debug=False),
```

步骤序号	5
步骤名称	设置 dataset 参数
步骤描述	设定数据集路径、与训练时与硬件相关的参数。
代码及讲解	分别设定 train、val、test 三个数据集的图片文件夹位置以及标注的位置。设定处理的 pipeline 为第 4 步中的 pipeline。

```
dataset_type = 'CocoDataset'
data_root = '/home/gp/dukto/Data0602/'
data = dict(
    imgs_per_gpu=2,  #每个计算机分配的图像数量
    workers_per_gpu=2,  #每个gpu分配的线程数
    train=dict(
        type=dataset_type,
        ann_file='./data/annotationstrain.json', #数据集annotation路径
        img_prefix='./data/train/image/',  #数据集的图片路径
        pipeline=train_pipeline),
    val=dict(
        type=dataset_type,
        ann_file='./data/annotationstval.json', #数据集annotation路径
        img_prefix='./data/val/image/',  #数据集的图片路径
        pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file='./data/annotationstest.json', #数据集annotation路径
        img_prefix='./data/test/image/',  #数据集的图片路径
        pipeline=test_pipeline))
```

步骤序号	6
步骤名称	数据增强
步骤描述	配置 pipeline 定义处理数据的所有步骤。
代码及讲解	mndetection 的数据读取方式分为两个部分，第一部分为数据集，第二部分为 data pipeline。 通常数据集定义如何处理标注信息，而 pipeline 定义处理数据字典的所有步骤，一个 pipeline 由一系列操作组成，每一个操作都采用一个 dict 作为输入，并且也输出一个 dict 作为下一个操作的输入。

```
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True, with_mask=True),
    dict(type='Resize', img_scale=[(2000, 400),(2000,600)], keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels']),
]
```

这里的操作依次是读入图片、载入图片标注，resize、随即翻转，归一化、padding。

```
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(2000, 500),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(type='Normalize', **img_norm_cfg),
            dict(type='Pad', size_divisor=32),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img']),
        ])
]
```

步骤序号	7
步骤名称	配置其他参数
步骤描述	设置数据和标注路径，学习率，优化器等参数
代码及讲解	学习率设定为 0.08，优化器为 SGD。

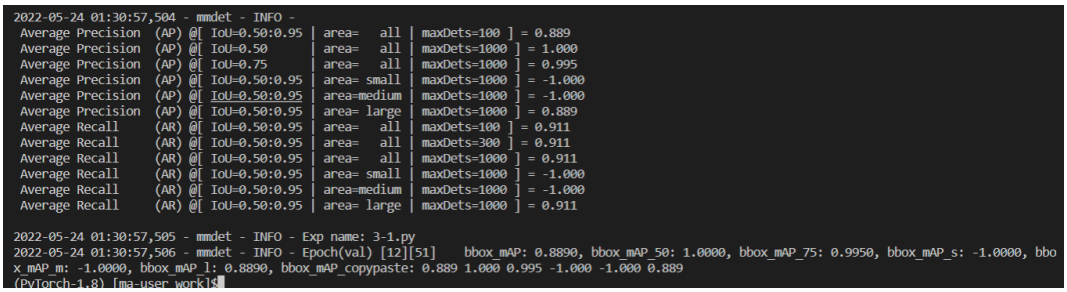
```
# optimizer
optimizer = dict(type='SGD', lr=0.08, momentum=0.9, weight_decay=0.0001)
optimizer_config = dict(grad_clip=dict(max_norm=35, norm_type=2))
# learning policy
lr_config = dict(
    policy='step',
    warmup='linear',
    warmup_iters=500,
    warmup_ratio=1.0 / 3,
    step=[8, 11])
checkpoint_config = dict(interval=1)
# yapf:disable
log_config = dict(
    interval=50,
    hooks=[
        dict(type='TextLoggerHook'),
        # dict(type='TensorboardLoggerHook')
    ])
```

7 实验结果及分析讨论

（1）最终结果的具体结果（文字说明）

能够成功训练，训练结果还不错。

（2）最终结果界面截图（界面截图）

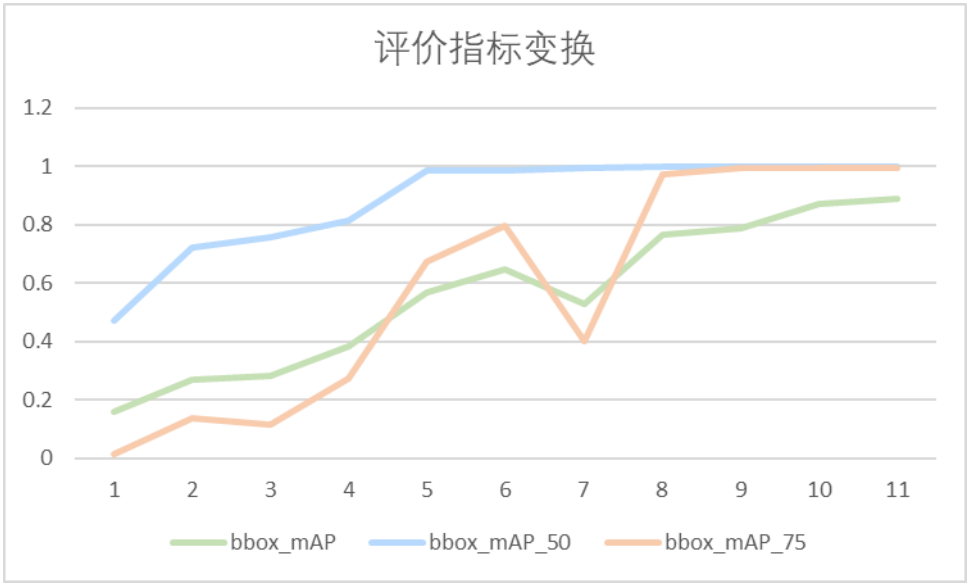


（3）最终结果的说明（注意事项或提醒）

最终结果:bbox_mAP: 0.8890, bbox_mAP_50: 1.0000, bbox_mAP_75: 0.9950, bbox_mAP_s: -1.0000, bbox_mAP_m: -1.0000, bbox_mAP_l: 0.8890, bbox_mAP_copypaste: 0.889 1.000 0.995 -1.000 -1.000 0.889

实测用 GPU 可以快很多，一会儿就出来了，就是比较贵。其他的注意事项可以看收获与体会中的 debug 过程。

（4）最终结果的解读与讨论



我其实在开始训练之前一直觉得可能效果会比较差呢。没想到啊没想到！居然肉眼可见的效果变好了。可以简单了解一下什么是 mAP：

对于目标检测而言，每一个类别都可计算出其 Precision 和 Recall，每个类别都可以得到一条 P-R 曲线，曲线下的面积就是 AP 的值。

假设存在 M 张图片，对于其中一张图片而言，其具有 N 个检测目标，其具有 K 个检测类别，使用检测器得到了 S 个 Bounding Box(BB)，每个 BB 里包含 BB 所在的位置以及 K 个类的得分 C。利用 BB 所在的位置可以得到与其对应的 GroundTruth 的 IOU 值。mAP 即 mean Average Precision，即各类 AP 的平均值。

简单理解就是评价检测好坏的一个指标。才运行了 12 个 epoch，mAP 效果逐渐上升！我认为是预训练模型的功劳！

8 收获与体会

在总体上看，了解了 mmdetection 在机器学习中的应用，学习了使用框架进行机器学习研究的方法。并且根据 mmdetection 的基础模型搭建自定义的模型。

具体地来说，认真地了解了 CascadeRCNN 模型的细节和实现，了解了 ResNet 模型、FPN 模型、RPNHead 的作用以及 RoIExtractor 的作用。

下面记录一下我遇到的问题，以及解决办法:

```
    test_cfg=cfg.get('test_cfg'))
  File "/home/ma-user/work/mmdetection/mmdet/models/builder.py", line 59, in build_detector
    cfg, default_args=dict(train_cfg=train_cfg, test_cfg=test_cfg))
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/registry.py", line 234, in build
    return self.build_func(*args, **kwargs, registry=self)
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/cnn/builder.py", line 27, in build_model_from_cfg
    return build_from_cfg(cfg, registry, default_args)
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/registry.py", line 69, in build_from_cfg
    raise type(e)(f'{obj_cls.__name__}: {e}')
TypeError: CascadeRCNN: __init__() got an unexpected keyword argument 'num_stages'
(PyTorch-1.8) [ma-user work]$
```

首先是不知道为什么，非得说 CascadeRCNN 没有 num_stages。

```
    test_cfg=cfg.get('test_cfg'))
  File "/home/ma-user/work/mmdetection/mmdet/models/builder.py", line 59, in build_detector
    cfg, default_args=dict(train_cfg=train_cfg, test_cfg=test_cfg))
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/registry.py", line 234, in build
    return self.build_func(*args, **kwargs, registry=self)
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/cnn/builder.py", line 27, in build_model_from_cfg
    return build_from_cfg(cfg, registry, default_args)
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/registry.py", line 69, in build_from_cfg
    raise type(e)(f'{obj_cls.__name__}: {e}')
TypeError: CascadeRCNN: __init__() got an unexpected keyword argument 'bbox_roi_extractor'
(PyTorch-1.8) [ma-user work]$
```

于是我把 num_stages 注释掉了，发现了又说没有 bbox_roi_extractor。这不可能呀，指定是哪里出问题了。

网上说原因是 mmdet 的版本不对，可是之前的实验明明弄得好好的……

```
    Installed /home/ma-user/work/mmdetection
Successfully installed mmdet-2.24.1 pycocotools-2.0.4
WARNING: You are using pip version 21.2.2; however, version 22.1 is available.
You should consider upgrading via the '/home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python -m pip install --upgrade pip' command.
Collecting package metadata (current_repodata.json): done
Solving environment: /
The environment is inconsistent, please check the package plan carefully
The following packages are causing the inconsistency:

  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/linux-64::certifi==2021.10.8=py37h06a4308_2
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/linux-64::libffi==3.3=he6710b0_2
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/linux-64::libgcc-ng==9.3.0=h5101ec6_17
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/linux-64::libstdcxx-ng==9.3.0=hd4cf53a_17
```

最后得到的解决方法是版本问题可以直接改 config 或者如果自己改了配置要使用 _delete_删除 base 里面的配置，否则会出现这个错误。

不过两个我都没太懂，后来按照 mmdetection 给的新版本的格式重新写了 config。

后来遇到了各种问题……一部分错误如下：

```
Traceback (most recent call last):
  File "mmdetection/tools/train.py", line 239, in <module>
    main()
  File "mmdetection/tools/train.py", line 132, in main
    setup_multi_processes(cfg)
  File "/home/ma-user/work/mmdetection/mmdet/utils/setup_env.py", line 30, in setup_multi_processes
    workers_per_gpu = cfg.data.get('workers_per_gpu', 1)
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/config.py", line 519, in __getattr__
    return getattr(self._cfg_dict, name)
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/config.py", line 50, in __getattr__
    raise ex
AttributeError: 'ConfigDict' object has no attribute 'data'
```

```
  , {'type': 'Normal', 'std': 0.001, 'override': {'name': 'fc_reg'}}, {'type': 'Xavier', 'distribution': 'uniform', 'override': [{'n
  }, {'name': 'cls_fcs'}, {'name': 'reg_fcs'}]}]
  Traceback (most recent call last):
    File "mmdetection/tools/train.py", line 239, in <module>
      main()
    File "mmdetection/tools/train.py", line 215, in main
      datasets = [build_dataset(cfg.data.train)]
    File "/home/ma-user/work/mmdetection/mmdet/datasets/builder.py", line 82, in build_dataset
      dataset = build_from_cfg(cfg, DATASETS, default_args)
    File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/registry.py", line 59, in build_from_cfg
      f'{obj_type} is not in the {registry.name} registry')
  KeyError: 'CoCoDataset is not in the dataset registry'
  (PyTorch-1.8) [ma-user work]$
```

```
  During handling of the above exception, another exception occurred:

  Traceback (most recent call last):
    File "mmdetection/tools/train.py", line 239, in <module>
      main()
    File "mmdetection/tools/train.py", line 215, in main
      datasets = [build_dataset(cfg.data.train)]
    File "/home/ma-user/work/mmdetection/mmdet/datasets/builder.py", line 82, in build_dataset
      dataset = build_from_cfg(cfg, DATASETS, default_args)
    File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/mmcv/utils/registry.py", line 69, in build_from_cfg
      raise type(e)(f'{obj_cls.__name__}: {e}')
  KeyError: "CocoDataset: 'id'"
  (PyTorch-1.8) [ma-user work]$
```

```
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/torch/utils/data/dataloader.py", line 971, in _reset
    self._try_put_index()
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/torch/utils/data/dataloader.py", line 1205, in _try_put_index
    index = self._next_index()
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/torch/utils/data/dataloader.py", line 508, in _next_index
    return next(self._sampler_iter)  # may raise StopIteration
  File "/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/torch/utils/data/sampler.py", line 227, in __iter__
    for idx in self.sampler:
  File "/home/ma-user/work/mmdetection/mmdet/datasets/samplers/group_sampler.py", line 36, in __iter__
    indices = np.concatenate(indices)
  File "<__array_function__ internals>", line 6, in concatenate
ValueError: need at least one array to concatenate
(PyTorch-1.8) [ma-user work]$
```

除了是我自己手误写错的，我感觉上面所有的问题中最后这个很有意义。因为这个视频里面好像没有说明，这个是需要在数据集下 CoCo 文件里面标注好自己的类别。

```
  File "/home/ma-user/work/mmdetection/mmdet/datasets/pipelines/loading.py", line 398, in __call__
    results = self._load_masks(results)
  File "/home/ma-user/work/mmdetection/mmdet/datasets/pipelines/loading.py", line 350, in _load_masks
    [self._poly2mask(mask, h, w) for mask in gt_masks], h, w)
  File "/home/ma-user/work/mmdetection/mmdet/datasets/pipelines/loading.py", line 350, in <listcomp>
    [self._poly2mask(mask, h, w) for mask in gt_masks], h, w)
  File "/home/ma-user/work/mmdetection/mmdet/datasets/pipelines/loading.py", line 306, in _poly2mask
    rles = maskUtils.frPyObjects(mask_ann, img_h, img_w)
  File "pycocotools/_mask.pyx", line 293, in pycocotools._mask.frPyObjects
IndexError: list index out of range
```

之后又遇到了这样的问题。发现是因为 segmentation 是空列表。随便改成了[1]，又报错 TypeError: Argument 'bb' has incorrect type (expected numpy.ndarray, got list)。报错原因是 json 文件里面的“segmentation”中的数据不对。“segmentation”:[[x,y,x,y,x,y.....x,y]] 是按顺序排列的点序列，点序列个数要求是偶数，同时点的个数至少要大于 2 个，因为要构成一个 polygon，也就是说 segmentation 列表的长度必须是偶数且大于 4。更改之后发现主要原因可能是我在代码里面还使用了 mask，但实际上没有，又去掉了代码里面 mask 的部分。之后就可以运行了。

9 备注及其他

无。