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

商品候选框预测

1 实验目的

（1）理解机器学习的预测环节。
（2）理解 mmdetection 的测试代码的使用
（3）理解模型预测结果的处理方式。

2 实验原理

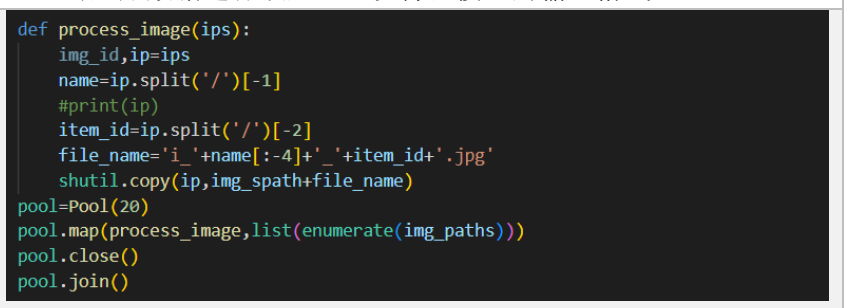
（1）mmdetection 测试脚本的使用
（2）pandas 使用
（3）mmcv 处理 json

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/库/插件	mmcv	和 mmdection 匹配	
第三方工具包/库/插件	opencv-python	4.5 以上	
第三方工具包/	pandas	1.0 以上	

库/插件			
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

步骤序号	1
步骤名称	数据预处理
步骤描述	对原始数据进行预处理，以符合模型的输入格式。
代码及讲解	  图片的处理跟之前第一节的处理是一样的，但是处理视频的时候进行了更改，每 40 帧取一张图片进行处理。

```
def process_image(ips):
    img_id,ip=ips
    name=ip.split('/')[-1]
    #print(ip)
    item_id=ip.split('/')[-2]
    file_name='i_'+name[:-4]+'_'+item_id+'.jpg'
    shutil.copy(ip,img_spath+file_name)
pool=Pool(20)
pool.map(process_image,list(enumerate(img_paths)))
pool.close()
pool.join()
```

```
def process_video(vps):
    img_id,vp=vps
    cap=cv2.VideoCapture(vp)
    video_id=vp.split('/')[-1][:-4]
    for frame in range(20,390,40):#40帧抽取一张
        frame_index=frame
        cap.set(cv2.CAP_PROP_POS_FRAMES,frame_index)
        flag,frame_img=cap.read()
        if not flag:
            cap.release()
            break

        img_id+=1
        vfile_name='v_'+video_id+'_'+str(frame_index)+'.jpg'
        #print(video_img_spath+vfile_name)
        cv2.imwrite(video_img_spath+vfile_name,frame_img)

        del frame_img
    cap.release()
```

步骤序号	2
步骤名称	模型预测
步骤描述	对视频、图片分别预测。
代码及讲解	利用第 3 节的模型，分别对上述的的两个数据进行预测。并且指定好结果保存的位置。所用的指令如下: 

```
#python mmdetection/tools/test.py 3-1.py ./work_dirs/3/i.pth --format-only --options='jsonfile_prefix=./result/i_fs_cascade'

#python mmdetection/tools/test.py 3-2v.py ./work_dirs/3-v/v.pth --format-only --options 'jsonfile_prefix=./result/v_fs_cascade'
```

步骤序号	3
步骤名称	将预测结果转换格式
步骤描述	将预测结果转为 COCO 标准格式。
代码及讲解	之前实验预测的保存的结果不满足 COCO 格式，主要是 annotations 有问题，而最开始的标注的 annotation（本节课的以及第

	一节课的标注都是 COCO 格式的），所以把这部分直接替换成之前的就好了。视频和图片都是这样处理。
	`#转换成coco`
	`#`
	`with open('./data/i_train.json', 'r') as f:`
	`  test_json_raw=json.load(f)`
	`#test_json_raw=json.load('i_train.json','r)`
	`test_json=json.load(open('{}'.format('./result/i_fs_cascade.bbox.json'),'r'))`
	`test_json_raw['annotations']=test_json`
	`with open('{}'.format('./result/i_fs_cascade.json'),'w') as fp:`
	`  json.dump(test_json_raw,fp)`
	`dt_df=pd.DataFrame(test_json_raw['annotations'])`
	`print('all_images: ',len(test_json_raw['images']))`
	`print('all unique image ann: ',len(dt_df['image_id'].unique()))`
	`with open('./data/v_train.json', 'r') as f:`
	`  test_json_raw=json.load(f)`
	`#test_json_raw=json.load('i_train.json','r)`
	`test_json=json.load(open('{}'.format('./result/v_fs_cascade.bbox.json'),'r'))`
	`test_json_raw['annotations']=test_json`
	`with open('{}'.format('./result/v_fs_cascade.json'),'w') as fp:`
	`  json.dump(test_json_raw,fp)`

步骤序号	4
步骤名称	结果过滤
步骤描述	对最后结果进行一次过滤，去除掉整张图里面所有 bbox 得分都很小的图像。

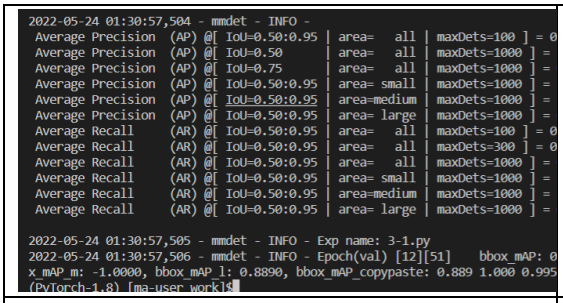
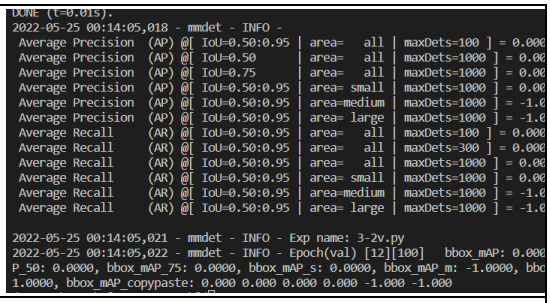
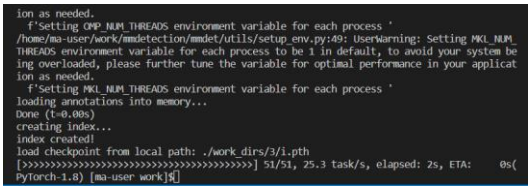
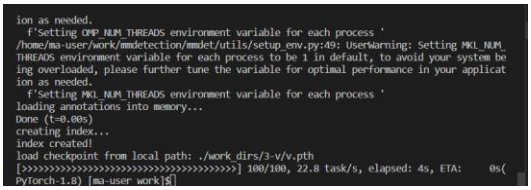
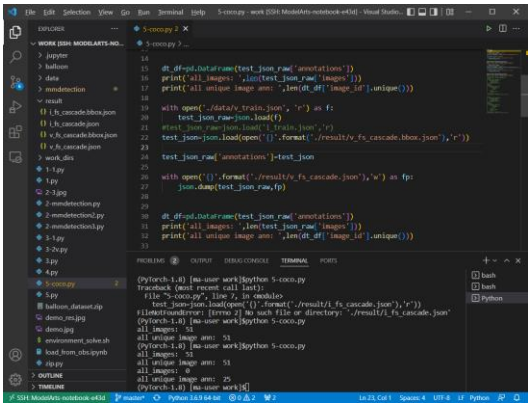
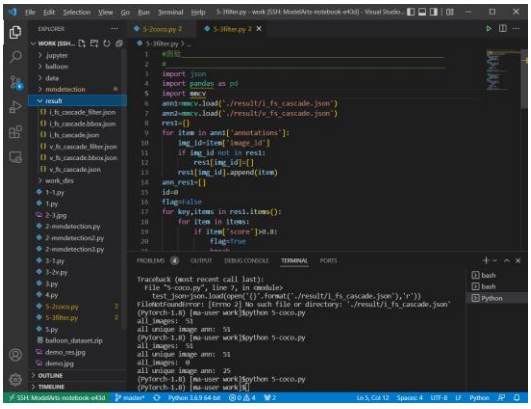
代码及讲解	`ann1=mmcv.load('./result/i_fs_cascade.json')`
	`ann2=mmcv.load('./result/v_fs_cascade.json')`
	`res1={}`
	`for item in ann1['annotations']:`
	`  img_id=item['image_id']`
	`  if img_id not in res1:`
	`  res1[img_id]=[]`
	`  res1[img_id].append(item)`
	`ann_res1=[]`
	`id=0`
	`flag=False`
	`for key,items in res1.items():`
	`  for item in items:`
	`  if item['score']>0.8:`
	`  flag=True`
	`  break`
	`  if flag:`
	`  ann_res1+=items`
	`  flag=False`
	`ann1['annotations']=ann_res1`
	`mmcv.dump(ann1,'./result/i_fs_cascade_filter.json')`
	`res2={}`
	`for item in ann2['annotations']:`
	`  img_id=item['image_id']`
	`  if img_id not in res2:`
	`  res2[img_id]=[]`
	`  res2[img_id].append(item)`
	`ann_res2=[]`
	`id=0`
	`flag=False`
	`for key,items in res2.items():`
	`  for item in items:`
	`  if item['score']>0.8:`
	`  flag=True`
	`  break`
	`  if flag:`
	`  ann_res2+=items`
	`  flag=False`
	`ann1['annotations']=ann_res2`
	`mmcv.dump(ann1,'./result/v_fs_cascade_filter.json')`
	遍历结果，如果结果的 bbox 得分大于 0.8，就保存下来。

5 实验结果及分析讨论

（1）最终结果的具体结果（文字说明）

对于图片以及视频截取下来的帧，进行预测，保留 bbox 得分大于 0.8 的候选框。

（2）最终结果界面截图（界面截图）

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>图片预测截图</td><td>视频帧预测截图</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>图片结果类型转换截图</td><td>视频帧结果类型转换截图</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>图片结果过滤截图</td><td>视频帧结果过滤截图</td></tr>
</table>

（3）最终结果的说明（注意事项或提醒）

前 4 个结果截图看得比较清晰，后两个过滤后的结果图主要可以关注一下屏幕左侧的 result 的文件夹的变化。

（4）最终结果的解读与讨论

视频帧和图片其实是一样的东西，处理上也都是一样的方法，这个实验里面加强了很多处理的时候的代码能力。这个实验其实是一个中间的阶段，没有什么太多需要解读的。但是我去查询了一下关于候选框的筛选的方法前，其实也有很多办法（比如接下来会叙述 NMS、SoftNMS、IoU-guided NMS），我们的这里使用的方法反而可能是最差的一种哈哈哈哈。比

如说可能很多框框超过了 0.8，但是他们都重叠了，可以用非极大值抑制的方法：

将 bbox 按照得分从大到小排列，取出 bbox 中得分最高的 bbox，并在剩下的 bboxes 中去掉与该 bbox IoU 大于一定阈值的（如 0.5、0.7 等）bbox，相当于去掉重复的 bbox。然后迭代选取，直到 bbox 被取完。

但是这种原始 NMS 也有缺陷：

1.得分最高的并不一定是最好的框。

2.难以 handle 两个 object 离得很近的情况，即相邻物体很容易被抑制。

3.并没有抑制 False Positive 值（IoU 过高容易产生误检，过低容易漏检）

传统 NMS 直接将所有大于阈值的框都抑制掉，会导致邻近物体被抑制。SoftNMS 不再是将这些框直接抑制掉，而是减少他们的置信分，实际上传统 NMS 是 SoftNMS 的一种特殊情况，即直接将大于 IOU 阈值的框的置信分降为 0。

传统的 NMS 只考虑了分类置信度，如 Faster-RCNN 用在 RPN 网络里，是将 anchors 按照前景分数从大到小排列后进行 NMS，这个过程中并没有用到定位信息。这种方法就是 IoU-guided NMS。IoU-Net 同时考虑定位置信度，实际上就是在 NMS 的过程中，将候选框按照定位置信度，即 IoU 的值从大到小排列，再进行 NMS，不过在进行 NMS 的过程中，会对分类置信度做一个调整：NMS 会剔除几个 IoU 非极大值，但是在这些 IoU 非极大值对应的候选框中，可能会有某些候选框的分类置信分比 IoU 极大值对应的候选框的分类置信分大，就将这个分类置信分赋值给该 IoU 极大值对应的候选框。相当于以定位置信度为标准，同时保留分类置信度的效果。

6 收获与体会

总体上，理解了机器学习的预测环节，理解 mmdetection 的测试代码的使用，理解模型预测结果的处理方式。代码层面上，学习了 mmdetection 测试脚本的使用、pandas 使用、mmcv 处理 json 的方法。

其实一直做到这里，会感觉有几个门槛，过了这几个门槛就好多了。基本就是数据集下载、数据集处理、模型的调用这几个方面。从实验 3 做过来，就开始变得比较顺了。在代码里面，我发现对列表处理的时候用的‘+’比较多，这样写很简单，不需要写 append，简单、方便。

7 备注及其他

无。