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

商品识别结果后处理

1 实验目的

（1）帮助学生理解后处理的作用。
（2）帮助学生理解在预测结果上优化的能力。
（3）帮助学生理解分析数据的能力。

2 实验原理

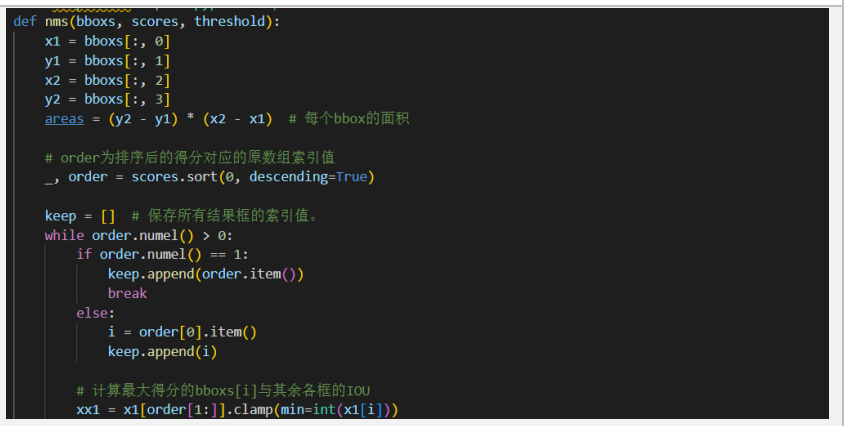
（1）faiss 库
（2）cosine 相似度
（3）F1 score

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	sklearn	无要求	
第三方工具包/库/插件	tqdm	4.32	

第三方工具包/库/插件	faiss	无要求	
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	商品识别结果后处理
步骤描述	商品识别结果后处理，比如非极大值抑制以及搜索和匹配。
代码及讲解	 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），之前第 5 节实验使用的方法可能是最差的。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

```python
def nms(bboxs, scores, threshold):
    x1 = bboxs[:, 0]
    y1 = bboxs[:, 1]
    x2 = bboxs[:, 2]
    y2 = bboxs[:, 3]
    areas = (y2 - y1) * (x2 - x1)  # 每个bbox的面积

    # order为排序后的得分对应的原数组索引值
    _, order = scores.sort(0, descending=True)

    keep = []  # 保存所有结果框的索引值。
    while order.numel() > 0:
        if order.numel() == 1:
            keep.append(order.item())
            break
        else:
            i = order[0].item()
            keep.append(i)

        # 计算最大得分的bboxs[i]与其余各框的IOU
        xx1 = x1[order[1:]].clamp(min=int(x1[i]))
```

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。

```
def Query(q,g,topK,nlist=24):
    #nlist=# 聚类质心有多少
    d=q.shape[1]#维度
    quantizer = faiss.IndexFlatL2(d)  # IndexFlatL2 & IndexFlatIP -> 欧式距离 & 内积搜索
    index = faiss.IndexIVFFlat(quantizer, d, nlist, faiss.METRIC_L2)

    assert not index.is_trained
    index.train(q)
    assert index.is_trained

    index.add(q)
    # METRIC_L2 & METRIC_INNER_PRODUCT -> 欧式距离 & 内积搜索


    k = 4                          # we want to see 4 nearest neighbors
    #[nq, k]  对query的每行找到k个相应的distance和index
    D, I = index.search(g, k)     # actual search (距离, ID)
    # print(I[:5])    # 输出前5行
    # print(D[-5:])   # 输出后5行
    return I[:topK]
```

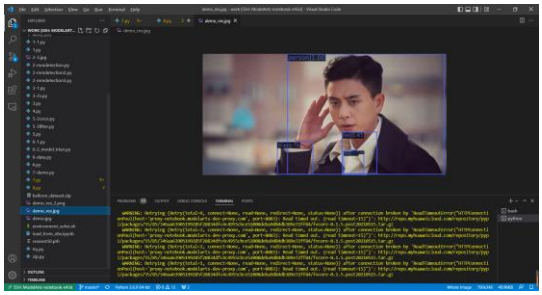
搜索和匹配的部分其实第 7 节实验的时候做过了，将传入进来的图片、视频帧提取特征，然后进行搜索和匹配。

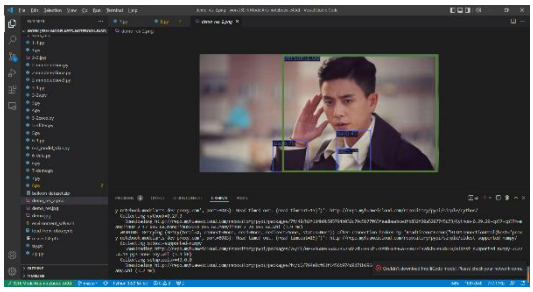
5 实验结果及分析讨论

（1）最终结果的具体结果（文字说明）

对图片和视频的候选框进行选择，之后进行了匹配。

（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）

左图是原图，右图是进行了非极大值抑制后的效果。

（4）最终结果的解读与讨论

非极大值抑制属于后处理一部分，之所以要进行这步操作，原因在于很多时候一个目标存在多个预测框，这时我们需要选出最好的那个作为预测结果。在目标检测(object detection)任务中，常会利用非极大值抑制算法(NMS)对生成的大量候选框进行后处理，去除冗余的候选框，得到最具代表性的结果，以加快目标检测的效率。流程如下:

1.首先，从生成的大量候选框中，选出置信度(score)最大的候选框出来。

2. 其次，计算剩余的候选框与刚才选出的候选框的 IoU 并且剔除 IoU 大于所指定

threshold 的候选框，保存小于 Threshold 的候选框。其意思是，IoU 越大，即说明重叠部分越大，所以可以选择剔除，那些重叠部分很小的，可能并不是同一物体的候选框，可选择保留下来。

3. 最后，重复以上 1，2 步骤，直到剔除所有的 IoU>Threshold 的候选框。

使用了非极大值抑制之后匹配和检索的速度确实有提高。

6 收获与体会

总体上，理解了后处理的作用、理解了在预测结果上优化的能力、理解了分析数据的能力。

后处理的这些内容其实我在实验 5、7 的时候就已经接触到了一些，没想到当时额外做的一些内容到后面的时候也可以用到，算是进一步巩固了实验 5、实验 7 的学习吧。

7 备注及其他

无。