哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2023 年秋季学期

《视听觉信息理解》课程

视觉部分实验：

目标检测/实例分割

姓名	刘天瑞
学号	**7203610121**
班号	**20W0362**
专业	人工智能（视听觉信息处理）
指导老师	任冬伟
时间	**2023.11.14**

一、 实验内容

NMS 对于取得检测、分割结果很重要：

论文[1]提出了 Cluster-NMS，速度快，且能够嵌入各种因素提高性能，如下图 1 所示：

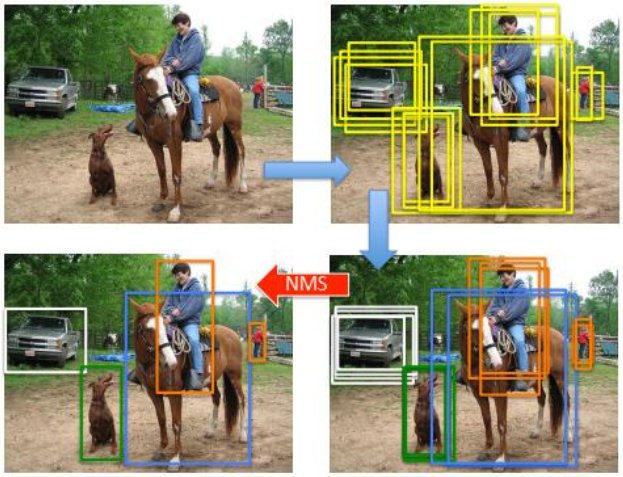

图 1

目标检测&实例分割：

以 YOLACT[2]为基础框架实现基础的 Cluster-NMS，在 PASCAL VOC / COCO 数据集比较其与原始 NMS 的结果。将 DIoU-NMS、Weighted-NMS、Score Penalty Mechanism NMS 应用到 Cluster-NMS。

实验源码、实验环境和安装步骤参考 github 项目：https://github.com/Zzh-tju/CIoU；https://github.com/dbolya/yolact。

二、 选择的实验平台

Windows 11 Powershell 终端；Anaconda3 Python=3.7 虚拟环境。

三、 算法描述

1. Cluster-NMS：如下图 2 所示，额外的 for 循环之前的部分与 Fast-NMS 算法完全相同，其次便是 Cluster-NMS 算法所特有的部分，目的是为了避免 Fast-NMS 算法中所存在的过滤掉边界框时删除过多的问题。通过该 for 循环，从而修改了判断其中一个边界框是否应该被删除过滤掉的阈值。

```
def cc_cluster_nms(self, boxes, masks, scores, iou_threshold: float = 0.5, top_k: int = 200):
    # Collapse all the classes into 1
    scores, classes = scores.max(dim = 0)
    _, idx = scores.sort(0, descending = True)
    idx = idx[:top_k]
    boxes_idx = boxes[idx]
    iou = jaccard(boxes_idx, boxes_idx).triu_(diagonal = 1)
    B = iou
    # 如下所示的for循环修改了之前fast-nms算法中较为简单的直接取maxA方式，从而可以防止fast-nms算法中删除过滤掉的边界框box过多
    for i in range(200):
        A = B
        maxA, _ = torch.max(A, dim = 0)
        E = (maxA <= iou_threshold).float().unsqueeze(1).expand_as(A)
        B = iou.mul(E)
        if A.equal(B) == True:
            break
    idx_out = idx[maxA <= iou_threshold]
    return boxes[idx_out], masks[idx_out], classes[idx_out], scores[idx_out]
```

图 2

2. Cluster-NMS 加上 DIoU-NMS：如下图 3 所示，该算法仅在 Cluster-NMS 算法的基础上进行了一个修改：即在计算两个边界框 box 的重叠部分（IoU）时，不再只单纯地考虑计算两个边界框 box 面积的重叠，而是改为将边界框中心点的位置考虑加进来，增添了 DIoU loss 的损失惩罚项。

```python
def cc_cluster_diounms(self, boxes, masks, scores, iou_threshold: float = 0.5, top_k: int = 200):
    # Collapse all the classes into 1
    scores, classes = scores.max(dim = 0)
    _, idx = scores.sort(0, descending = True)
    idx = idx[:top_k]
    boxes_idx = boxes[idx]
    # 在计算交并比iou时需要考虑到边界框中心点之间的距离
    iou = diou(boxes_idx, boxes_idx).triu_(diagonal = 1)
    B = iou
    for i in range(200):
        A = B
        maxA, _ = torch.max(A, dim = 0)
        E = (maxA <= iou_threshold).float().unsqueeze(1).expand_as(A)
        B = iou.mul(E)
        if A.equal(B) == True:
            break
    idx_out = idx[maxA <= iou_threshold]
    return boxes[idx_out], masks[idx_out], classes[idx_out], scores[idx_out]
```

图 3

3. Cluster-NMS 加上 Weighted-NMS：如下图 4 所示，该算法首先利用了 score 来更新交并比 IoU 矩阵，然后再根据更新后的交并比 IoU 矩阵来重新计算边界框各个点的二维平面坐标。

```python
def cc_cluster_SPM_dist_weighted_nms(self, boxes, masks, scores, iou_threshold: float = 0.5, top_k: int = 200):
    # Collapse all the classes into 1
    scores, classes = scores.max(dim = 0)
    _, idx = scores.sort(0, descending = True)
    idx = idx[:top_k]
    boxes_idx = boxes[idx]
    scores = scores[idx]
    boxes = boxes_idx
    masks = masks[idx]
    classes = classes[idx]
    n = len(scores)
    iou = jaccard(boxes_idx, boxes_idx).triu_(diagonal = 1)
    B = iou
    for i in range(200):
        A = B
        maxA, _ = torch.max(A, dim = 0)
        E = (maxA <= iou_threshold).float().unsqueeze(1).expand_as(A)
        B = iou.mul(E)
        if A.equal(B) == True:
            break
    D = distance(boxes, boxes)
    X = (B >= 0).float()
    scores = torch.prod(torch.min(torch.exp(-B ** 2 / 0.2) + D * ((B > 0).float()), X), 0) * scores
    idx_out = scores > 0.01
    weights = (B * (B > 0.8).float() + torch.eye(n).cuda()) * (scores.reshape((1, n)))
    xx1 = boxes[:, 0].expand(n, n)
    yy1 = boxes[:, 1].expand(n, n)
    xx2 = boxes[:, 2].expand(n, n)
    yy2 = boxes[:, 3].expand(n, n)

    weightsum = weights.sum(dim = 1)
    xx1 = (xx1 * weights).sum(dim = 1) / (weightsum)
    yy1 = (yy1 * weights).sum(dim = 1) / (weightsum)
    xx2 = (xx2 * weights).sum(dim = 1) / (weightsum)
    yy2 = (yy2 * weights).sum(dim = 1) / (weightsum)
    boxes = torch.stack([xx1, yy1, xx2, yy2], 1)
    return boxes[idx_out], masks[idx_out], classes[idx_out], scores[idx_out]
```

图 4

4. Cluster-NMS 加上 Score Penalty：如下图 5 所示，该算法将交并比 IoU 矩阵的每列元素连乘作为惩罚项来进行加权平均聚合。

```python
def cc_cluster_SPM_nms(self, boxes, masks, scores, iou_threshold: float = 0.5, top_k: int = 200):
    # Collapse all the classes into 1
    scores, classes = scores.max(dim = 0)
    _, idx = scores.sort(0, descending = True)
    idx = idx[:top_k]
    boxes_idx = boxes[idx]
    scores = scores[idx]
    boxes = boxes_idx
    masks = masks[idx]
    classes = classes[idx]
    iou = jaccard(boxes_idx, boxes_idx).triu_(diagonal = 1)
    B = iou
    for i in range(200):
        A = B
        maxA, _ = torch.max(A, dim = 0)
        E = (maxA <= iou_threshold).float().unsqueeze(1).expand_as(A)
        B = iou.mul(E)
        if A.equal(B) == True:
            break
    # 将得分进行加权平均聚合
    scores = torch.prod(torch.exp(-B ** 2 / 0.2), 0) * scores
    idx_out = scores > 0.01
    return boxes[idx_out], masks[idx_out], classes[idx_out], scores[idx_out]
```

图 5

四、 算法复现过程

下载相应的源代码到本地：`(base) PS D:\> git clone https://github.com/Zzh-tju/CIoU.git`

使用 Anaconda 配置实验虚拟环境：conda env create -f environment.yml

提前准备好 COCO 2017 训练与测试数据集，按照 README.md 中的具体指导来运行指令完成批处理脚本，需要花费较长时间：

`(yolact) PS D:\CIoU> sh data/scripts/COCO.sh` `(yolact) PS D:\CIoU> sh data/scripts/COCO_test.sh`

完成下载解压后得到标注文件夹里的 json 文件以及 images 文件夹里的图片数据：

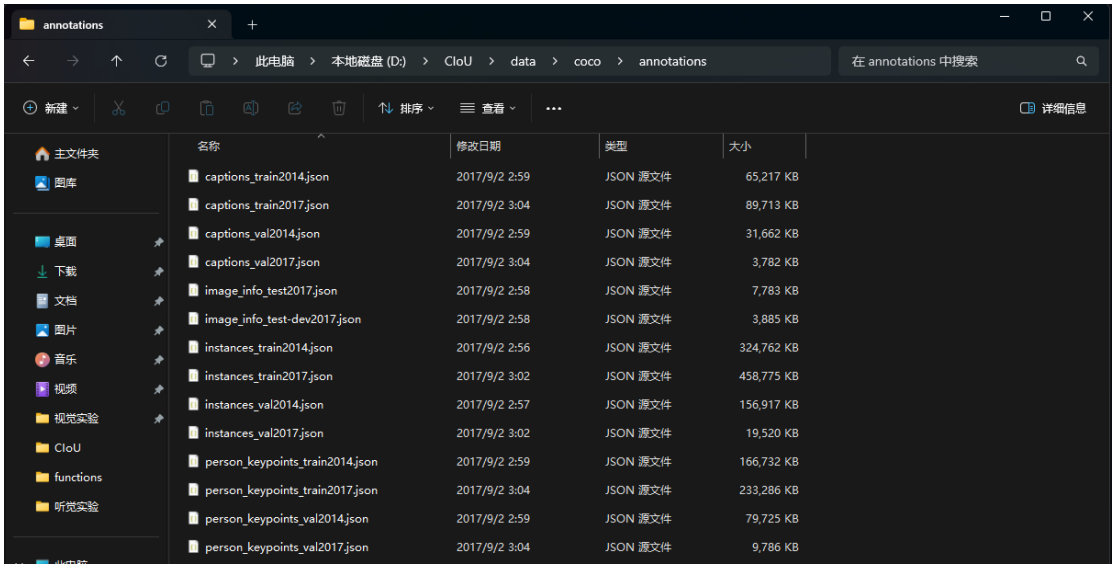

图 6

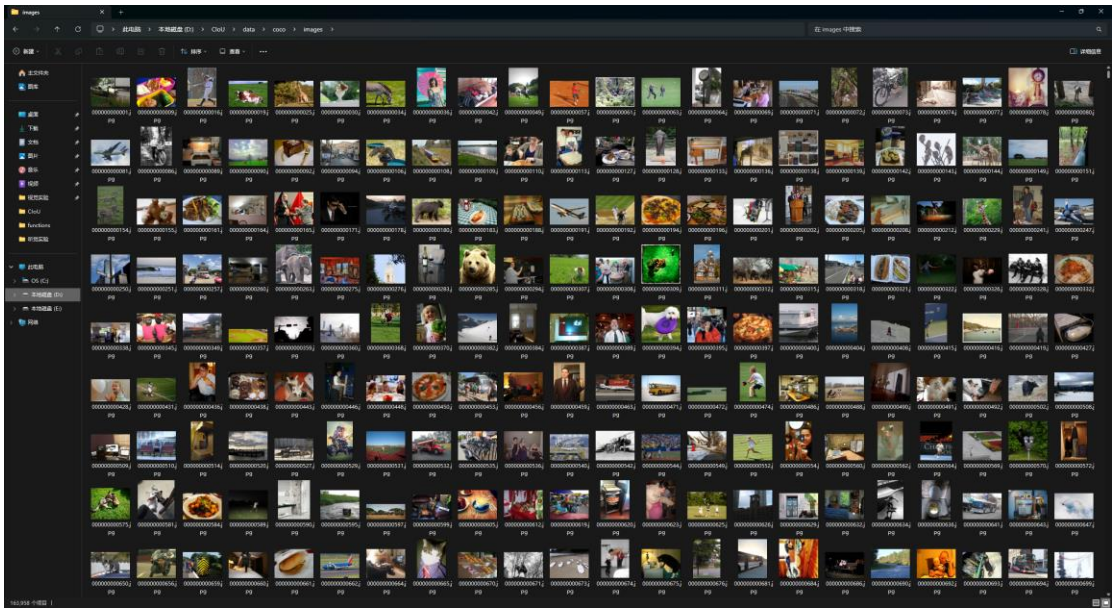

图 7

提前准备下载好相应的预训练模型权重文件 yolact_base_54_800000.pth，然后将其放在目录./weights 中。

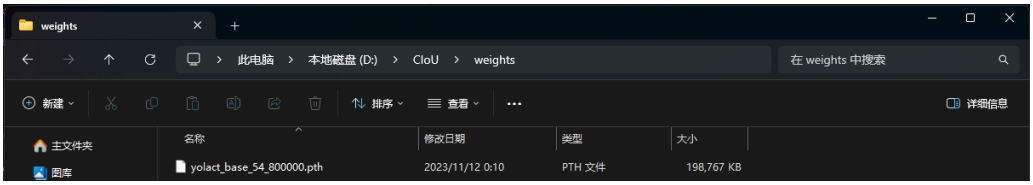

图 8

最后执行根目录的 eval.py 文件：

测试 COCO 基准（仅在验证集的前 1000 个图像上运行原始模型）：

```
(yolact) PS D:\CIoU> python eval.py --trained_model=weights/yolact_base_54_800000.pth --benchmark --max_images=1000 --cluste
r_diounms=False --spm=False --spm_dist=False --spm_dist_weighted=False
Config not specified. Parsed yolact_base_config from the file name.

D:\Anaconda3\envs\yolact\lib\site-packages\torch\__init__.py:614: UserWarning: torch.set_default_tensor_type() is deprecated
 as of PyTorch 2.1, please use torch.set_default_dtype() and torch.set_default_device() as alternatives. (Triggered internal
ly at ..\torch\csrc\tensor\python_tensor.cpp:453.)
  _C._set_default_tensor_type(t)
loading annotations into memory...
Done (t=0.39s)
creating index...
index created!
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'lat_layers' was found in ScriptModule
constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'downsample_layers' was found in Script
Module constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'pred_layers' was found in ScriptModule
 constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
Loading model... Done.

Processing Images                          1000 /   1000 (100.00%)    49.39 fps

Stats for the last frame:

      Name      | Time (ms)
----------------+-----------
  Network Extra |     0.1292
       backbone |    10.2731
            fpn |     0.4140
          proto |     0.3868
     pred_heads |     1.3577
     makepriors |     0.0082
         Detect |     4.0160
    Postprocess |     0.4848
           Copy |     0.6684
           Sync |     0.0375
----------------+-----------
          Total |    17.7757

Average: 49.39 fps, 20.25 ms
```

图 9

显示指定图像的处理测试结果：

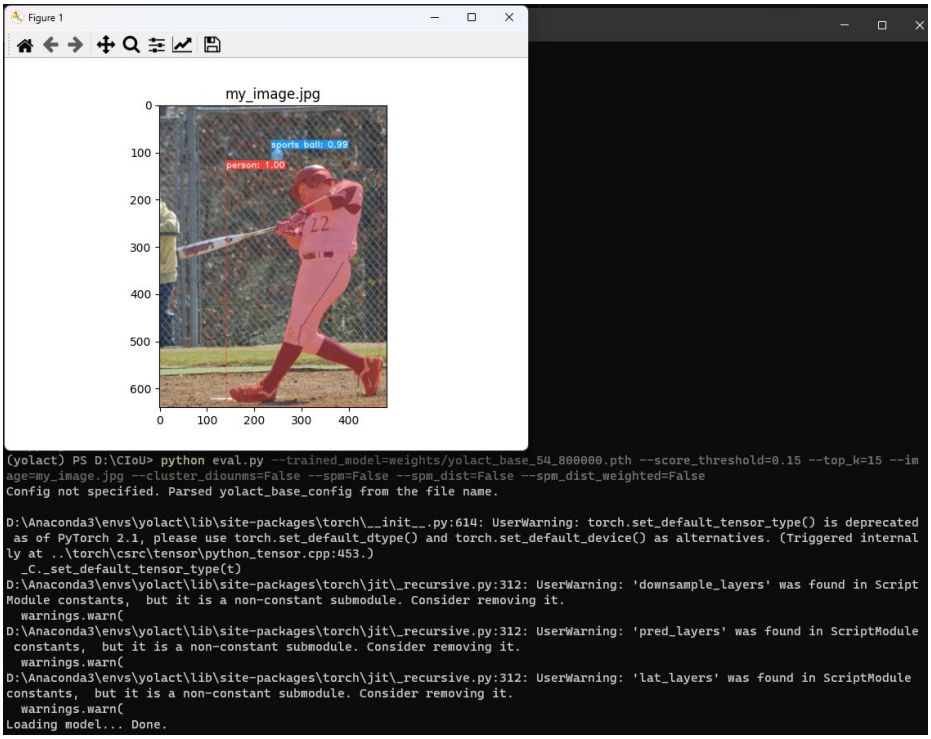

图 10

选择相应的非极大值抑制 NMS 算法：

具体指令：python eval.py --trained_model=weights/yolact_base_54_800000.pth --fast_nms=True/False --cross_class_nms=True/False --cluster_nms=True/False --cluster_diounms=Ture/False --spm=True/False --spm_dist=True/False --spm_dist_weighted=True/False

五、 实验结果

最终得到的实验结果如下图所示：

1. FAST_NMS with CROSS_CLASS_NMS：

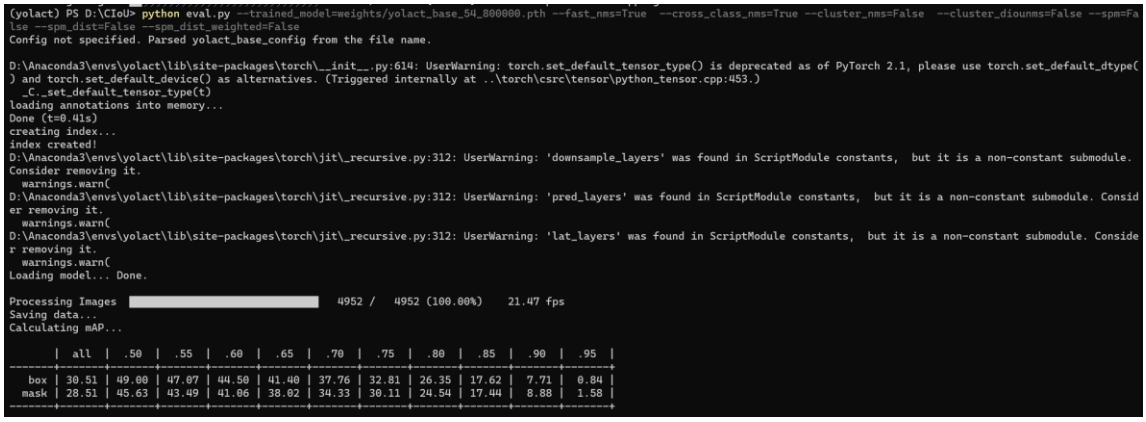

图 11

2. FAST_NMS without CROSS_CLASS_NMS：

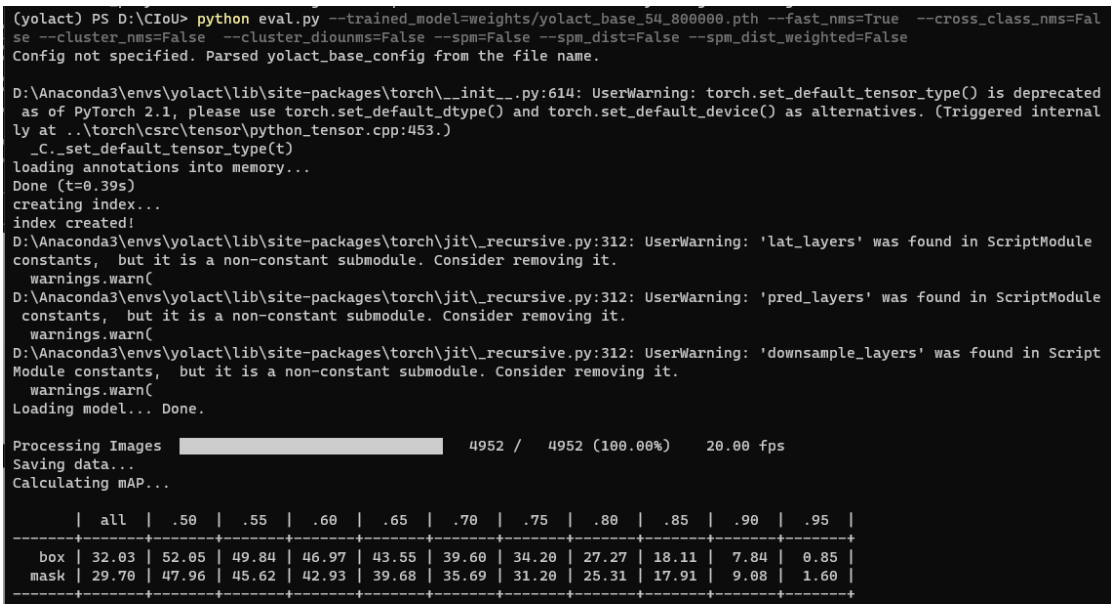

图 12

3. Cluster_NMS with CROSS_CLASS_NMS：

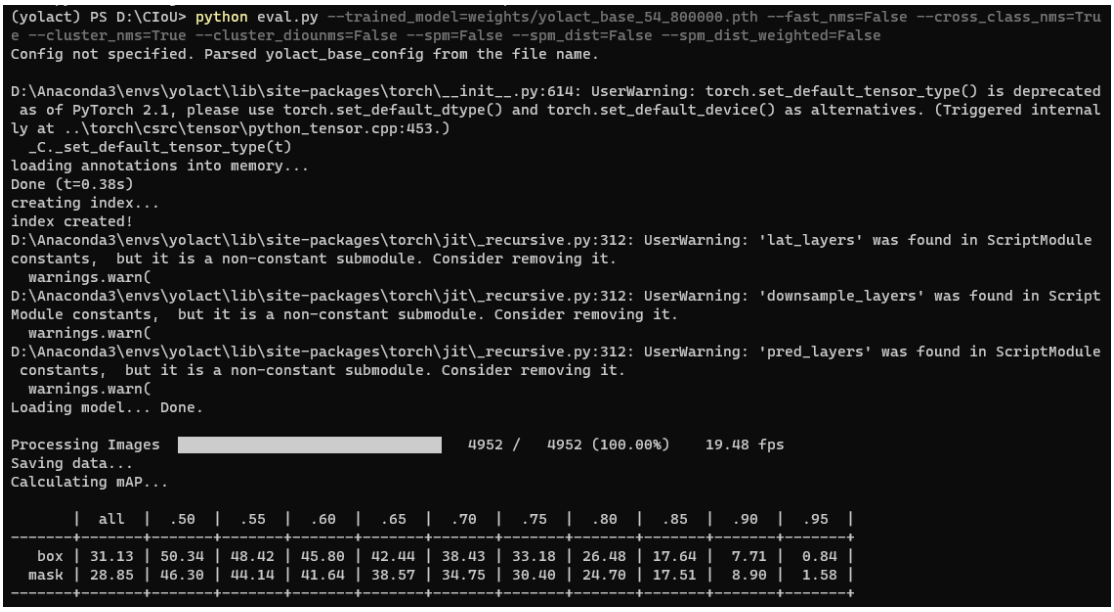

图 13

4. Cluster_NMS without CROSS_CLASS_NMS：

```
(yolact) PS D:\CIoU> python eval.py --trained_model=weights/yolact_base_54_800000.pth --fast_nms=False --cross_class_nms=Fal
se --cluster_nms=True --cluster_diounms=False --spm=False --spm_dist=False --spm_dist_weighted=False
Config not specified. Parsed yolact_base_config from the file name.

D:\Anaconda3\envs\yolact\lib\site-packages\torch\__init__.py:614: UserWarning: torch.set_default_tensor_type() is deprecated
 as of PyTorch 2.1, please use torch.set_default_dtype() and torch.set_default_device() as alternatives. (Triggered internal
ly at ..\torch\csrc\tensor\python_tensor.cpp:453.)
  _C._set_default_tensor_type(t)
loading annotations into memory...
Done (t=0.40s)
creating index...
index created!
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'downsample_layers' was found in Script
Module constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'pred_layers' was found in ScriptModule
 constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'lat_layers' was found in ScriptModule
constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
Loading model... Done.

Processing Images  [                              ]   4952 /   4952 (100.00%)    10.53 fps
Saving data...
Calculating mAP...

       |  all  |  .50  |  .55  |  .60  |  .65  |  .70  |  .75  |  .80  |  .85  |  .90  |  .95  |
-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
   box | 32.37 | 52.63 | 50.61 | 47.85 | 44.24 | 39.97 | 34.33 | 27.27 | 18.09 |  7.84 |  0.85 |
  mask | 29.79 | 48.05 | 45.78 | 43.11 | 39.90 | 35.84 | 31.30 | 25.32 | 17.90 |  9.06 |  1.60 |
-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
```

图 14

5. SPM with CROSS_CLASS_NMS：

```
(yolact) PS D:\CIoU> python eval.py --trained_model=weights/yolact_base_54_800000.pth --fast_nms=False --cross_class_nms=Tru
e --cluster_nms=False --cluster_diounms=False --spm=True --spm_dist=False --spm_dist_weighted=False
Config not specified. Parsed yolact_base_config from the file name.

D:\Anaconda3\envs\yolact\lib\site-packages\torch\__init__.py:614: UserWarning: torch.set_default_tensor_type() is deprecated
 as of PyTorch 2.1, please use torch.set_default_dtype() and torch.set_default_device() as alternatives. (Triggered internal
ly at ..\torch\csrc\tensor\python_tensor.cpp:453.)
  _C._set_default_tensor_type(t)
loading annotations into memory...
Done (t=0.39s)
creating index...
index created!
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'lat_layers' was found in ScriptModule
constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'pred_layers' was found in ScriptModule
 constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
D:\Anaconda3\envs\yolact\lib\site-packages\torch\jit\_recursive.py:312: UserWarning: 'downsample_layers' was found in Script
Module constants,  but it is a non-constant submodule. Consider removing it.
  warnings.warn(
Loading model... Done.

Processing Images  [                              ]   4952 /   4952 (100.00%)    19.38 fps
Saving data...
Calculating mAP...

       |  all  |  .50  |  .55  |  .60  |  .65  |  .70  |  .75  |  .80  |  .85  |  .90  |  .95  |
-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
   box | 31.92 | 50.78 | 48.91 | 46.44 | 43.39 | 39.67 | 34.48 | 27.72 | 18.59 |  8.27 |  0.93 |
  mask | 29.41 | 47.03 | 44.84 | 42.37 | 39.28 | 35.47 | 31.08 | 25.31 | 17.96 |  9.09 |  1.64 |
-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
```

图 15

6. SPM without CROSS_CLASS_NMS：

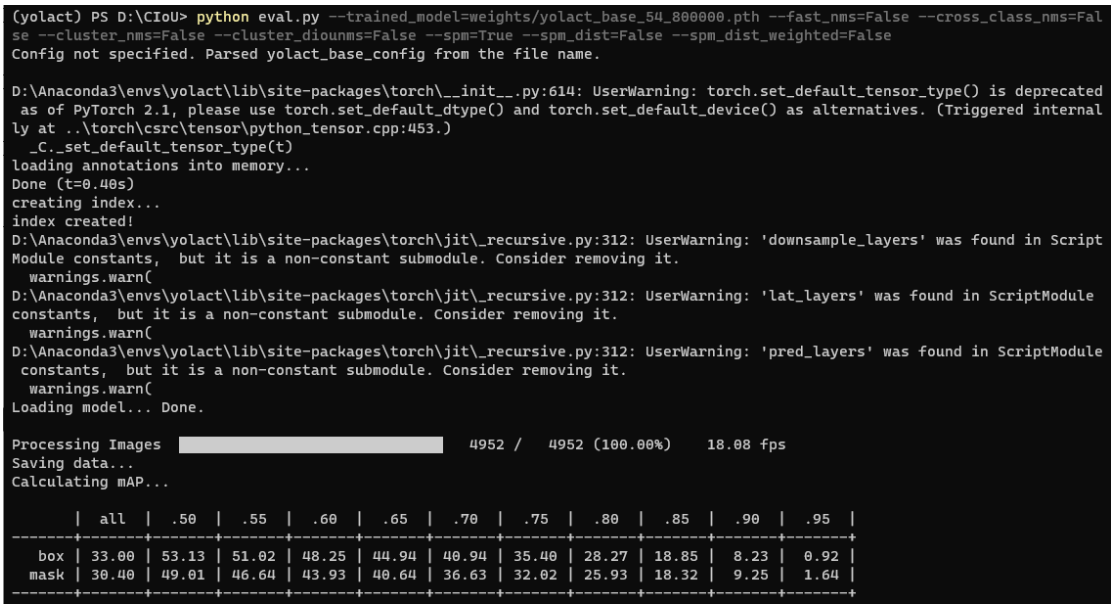

图 16

7. SPM_DIST with CROSS_CLASS_NMS：

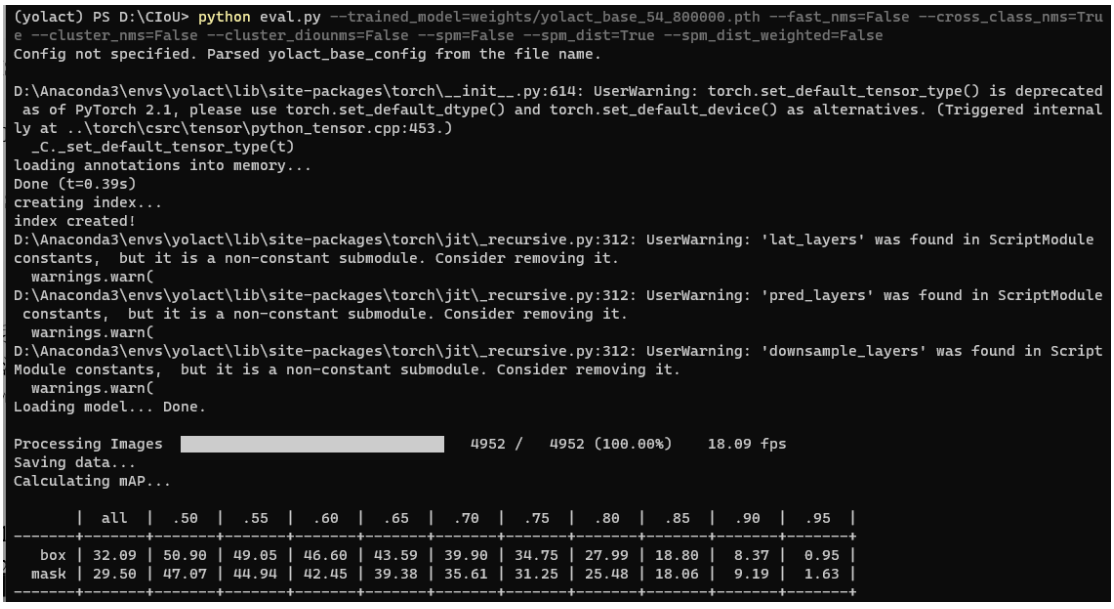

图 17

8. SPM_DIST without CROSS_CLASS_NMS：

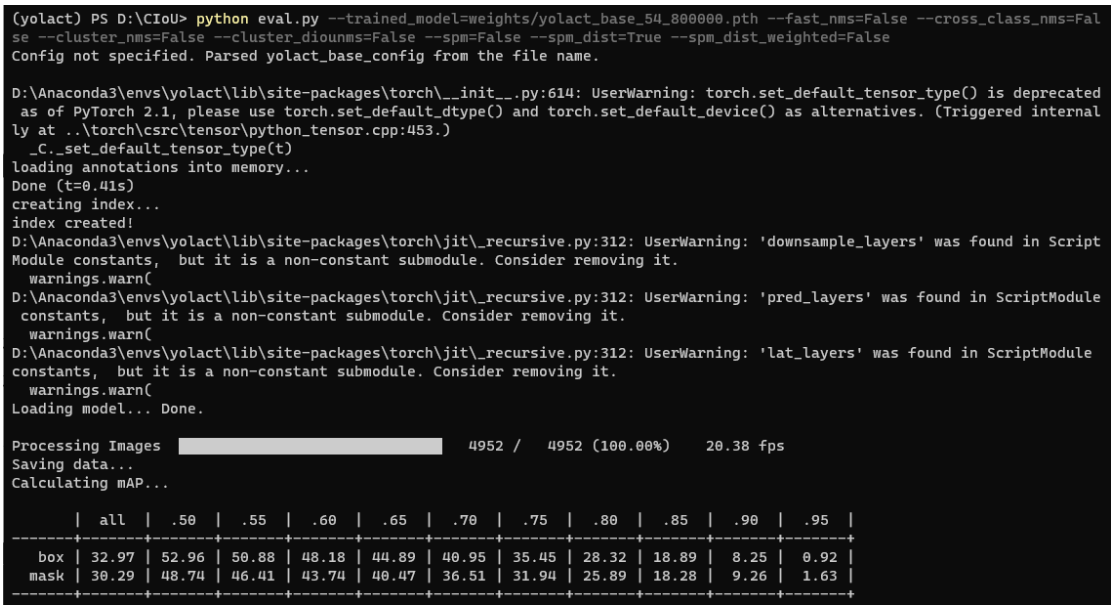

图 18

9. SPM_DIST_WEIGHTED with CROSS_CLASS_NMS：

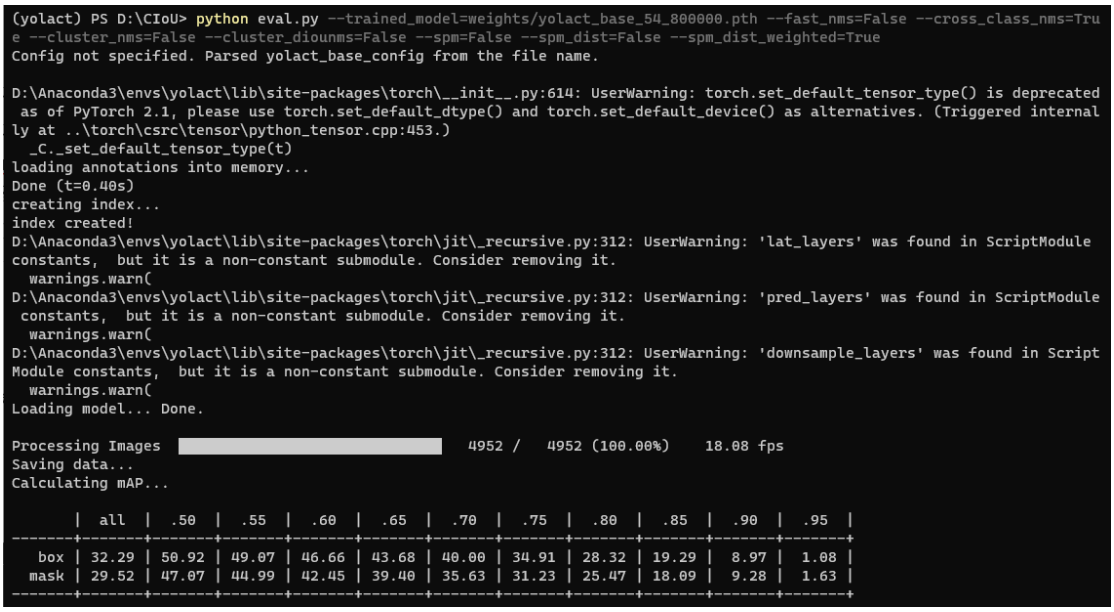

图 19

10. SPM_DIST_WEIGHTED without CROSS_CLASS_NMS：

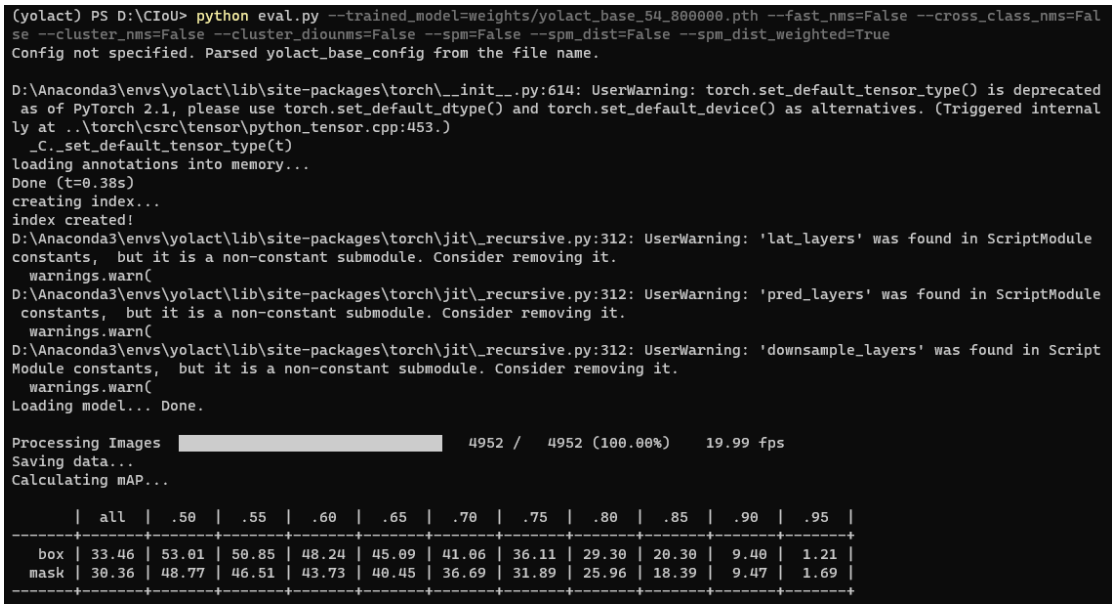

图 20

六、 实验心得体会

本次实验我认为总体来说较为简单，但是在进行虚拟环境的必要依赖包安装与使用 eval.py 文件进行非极大值抑制算法测试时遇到了挺多的坑，大多数时间也都花费在解决这些随着实验深入出现的问题上面。

其中首先由于初始 environment.yml 配置文件里的 pytorch 与我自己电脑的 cuda12.1 版本不匹配，且 conda 安装无法正确找到 unzip 与 conda-forge 包，所以我改为手动安装，解决不兼容问题。

其次完成本次实验仅需要下载 COCO 的测试数据集，即 test-dev。但是在测试时反而还需要下载完整 COCO 训练数据集才能够获得标注文件夹里的必要文件 instances_val2017.json 等等，在下载数据集期间有时下载速度会被限制到很小导致多次重新下载解压。

后续在测试时发现 yolact 的源码有语法缩进的报错，并且还需要将 detection.py 文件更换为 CIoU 项目里完整的版本，才能够避免运行程序时缺少必要算法。同时还应该及时查看 eval 里的命令格式默认设置，防止摸索过久。

最后可能在编译过程中，我安装扩展包期间中发生了报错，提示需要 Microsoft Visual C++14.0 或更高版本来获取 Microsoft C++ Build Tools。报错原因是 pip 所安装的包需要使用 C++编译后才能够正常安装，但是当前安装环境中缺少完整的 C++编译环境，因此安装失败。（Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/）

因为我刚开始对整体目标检测框架不够熟练，所以经过了很多次尝试，过程比较漫长，要有耐心。尤其我认为在配置前置所需库时遇到阻遏，需要自己想方设法绕过去，同时避免依赖冲突。而如果绕不过去，就应该找到可替代的解决方案。

以上便是我本次实验的收获与体会。

七、 参考资料

[1] Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation, IEEE TCYB 2022.

[2] YOLACT Real-time Instance Segmentation.