MMEdu_det教程

1. MMEdu_det简介

MMEdu_det为MMEdu团队在基于OpenMMlab项目的MMDection模块研发的教育版工具，在实现MMDection原始功能的基础上，代码更加精简，适用于教学场景、敏捷开发、迁移开发和新手入门。

主要功能：输出图片或视频中出现的多个对象名称，同时用方框框出对象所在方形区域。

2. MMEdu_det安装流程（以下内容为暂定，之后确定）

a. GPU环境下安装

i. 安装须知

- Ubuntu 16.04
- Python 3.8
- PyTorch 1.8.1
- CUDA 10.1
- GCC 5+
- mmcv-full 1.4.5

如果已经安装了 mmcv，首先需要使用 `pip uninstall mmcv` 卸载已安装的 mmcv，如果同时安装了 mmcv 和 mmcv-full，将会报 `ModuleNotFoundError` 错误。

ii. 安装环境

安装 PyTorch 和 torchvision（在第一讲中我们已经配置好了Python、PyTorch和mmcv-full）

安装 opencv（mmcv的依赖库，功能是图像读入、尺寸变换、图像展示等）

```Python
pip install opencv-python -i https://pypi.tuna.tsinghua.edu.cn/simple
```

iii. 安装MMDection

- 克隆mmdetection存储库

```Python
git clone https://github.com/open-mmlab/mmdetection.git
cd mmdetection
```

- 安装 MMDetection

```Python
pip install mmcv-full
python setup.py develop  # or "pip install -v -e ."
```

b. CPU环境下安装

i. 安装须知

- windows
- Python 3.8.12
- PyTorch 1.8.1

- Mmcv-full 1.4.5

如果已经安装了 mmcv，首先需要使用 `pip uninstall mmcv` 卸载已安装的 mmcv，如果同时安装了 mmcv 和 mmcv-full，将会报 `ModuleNotFoundError` 错误。

ii. 安装虚拟环境

安装 PyTorch 和 torchvision（ 在第一讲中我 们已经配置好了Python、PyTorch和mmcv-full ）

在anaconda promot中逐一执行以下代码即可：

- 安装虚拟环境（python环境使用3.8）,mm为自己命名的环境名

```
Python
1  conda create -n mm python=3.8
```

- 激活虚拟环境

```
Python
1  conda activate mm
```

- 确认python版本

```
Python
1  python -V
```

- 安装PyTorch 和 torchvision

```
Python
1  pip install torch==1.8.1 torchvision==0.9.1 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

- 安装mmcv-full

```
Python
1  pip install mmcv-full==1.4.5 -f https://download.openmmlab.com/mmcv/dist/cpu/torch1.8.0/index.html
```

- 安装opencv（mmcv的依赖库，功能是图像读入、尺寸变换、图像展示等）

```Python
pip install opencv-python -i https://pypi.tuna.tsinghua.edu.cn/simple
```

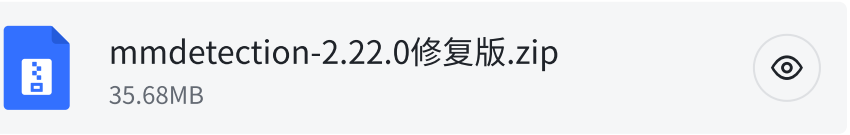
iii. 安装MMDection

- 克隆mmdetection存储库

在git里克隆mmdetection存储库

```Python
git clone https://github.com/open-mmlab/mmdetection.git
cd mmdetection
```

或者直接使用下面的文件夹：



- 打开anaconda prompt切换到mmdetection-2.22.0修复版文件夹解压路径内

```Python
cd C:\Users\luyanan\Downloads\MMEdu-main
```

- 激活虚拟环境（如已经处于虚拟环境可跳过）

```Python
conda activate mm
```

- 安装 MMDetection

```Python
pip install -v -e .
```

- 测试安装结果

```Python
pip list
```

注意：不要删除mmdetection-master文件夹。

3. MMEdu_det使用说明

a. 一张图像的直接推理

如果您想拿一张图片使用MMEdu_det体验一下图像检测，可以参考下面的代码。

Python

```python
from utils.my_utils_det import MMDetection# 导入detection模块
img = 'data/01.jpg' #图片路径
model = MMDetection(backbone="FasterRCNN", dataset_path="data/coco") #创建模型（使用FasterRCNN框架，数据集路径是data/coco）
model.inference(infer_data=img, rpn_threshold=0.5, rcnn_threshold=0.3) #推理检测我们指定的图片img，rpn_threshold：执行非极大值抑制时的阈值，rcnn_threshold：交并比，这两个参数用于筛选检测结果
```

我使用的程序（用了utils文件夹下的bird.JPEG)：

Python

```python
from utils.my_utils_det import MMDetection# 导入detection模块
img = 'utils/demo/bird.JPEG'
model = MMDetection(backbone="FasterRCNN")
result = model.inference(infer_data=img, rpn_threshold=0.5, rcnn_threshold=0.3
)
print(result)
```

接下来为您详细说明：

i. 准备图片

首先需要您准备好用来推理检测的图片，可以用自己喜欢的名字(为避免潜在的运行错误，请尽量使用英文命名)命名，如我这里命名为01.jpg。

一般情况下，建议在文件夹新建一个专门放数据的文件夹，可以放入数据集，如data。

路径如下：

├── data

│ ├──01.jpg

这边我们用img表示这张图片，方便后续使用。

代码参考：

Python

```python
img = 'data/01.jpg' #图片用img表示
```

使用utils文件夹下的demo文件夹中的bird.JPEG：

路径如下：

├——utils

│　├——demo

│　│　├——01.jpg

```python
img = 'utils/demo/bird.JPEG'
```

ii. 创建模型，配置图片路径

接下来您可以创建模型了，`MMDetection`是已经封装好的图像检测模型类，同时backbone参数我们为大家选择了Faster RCNN检测框架，速度更快（具体说明文档见下文），注意这里您需要配置数据集路径。

如何看图片路径?

如main文件下的图片路径是这样的：

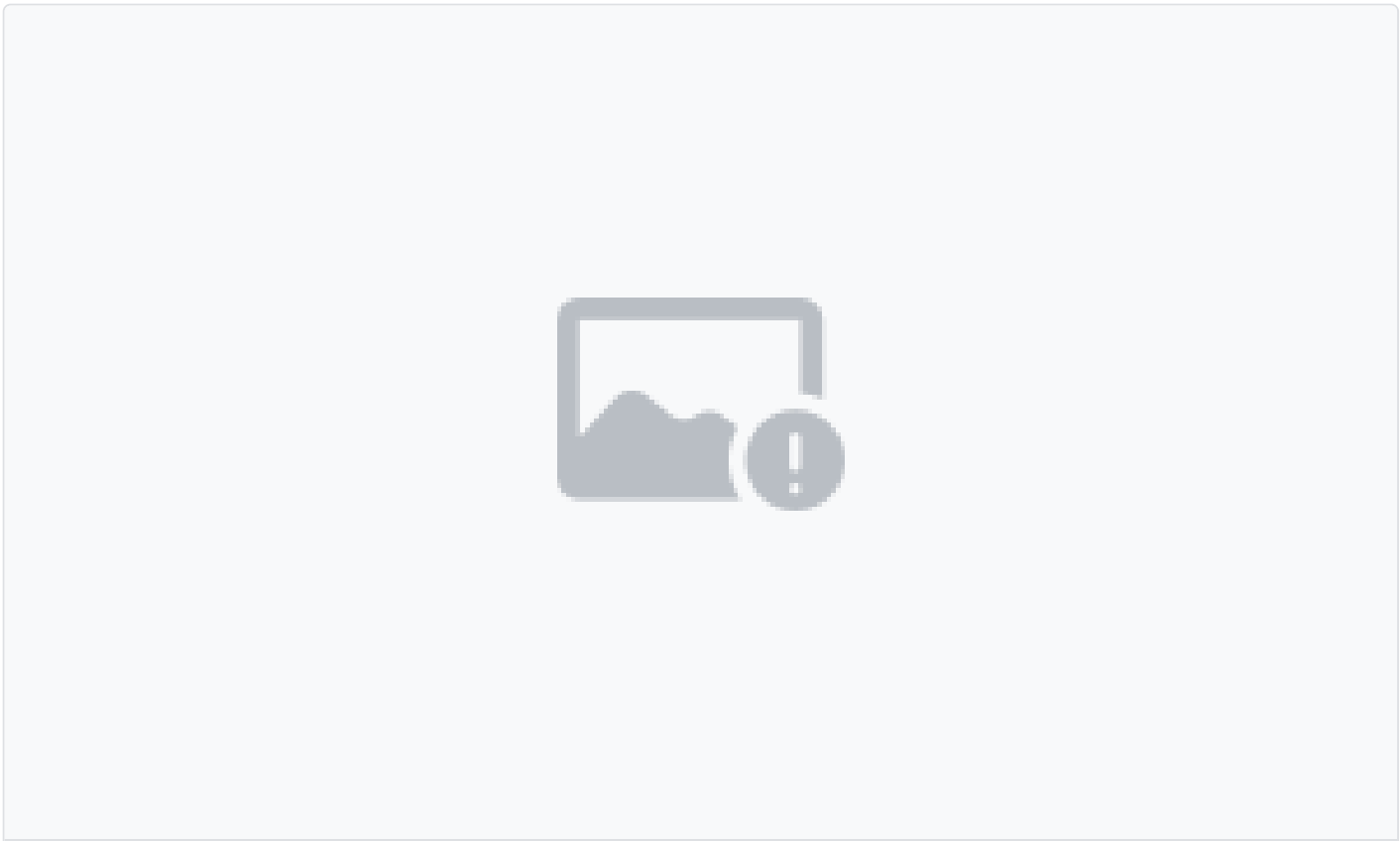
├—— data

│　├——01.jpg

那么这个图片的路径就是data/01.jpg

以此类推，utils文件夹下的demo文件夹中的bird.JPEG的路径便是utils/demo/bird.JPEG。

- backbone：主干网络，通常是一个用来提取特征图 (feature map) 的全卷积网络 (FCN network)，例如：Faster RCNN，ResNet, MobileNet。
- Faster RCNN检测框架的说明（内含预训练模型）：



Faster R-CNN说明文档.md

参考代码：

```python
model = MMDetection(backbone="FasterRCNN", dataset_path="data/coco") #创建模型（使用FasterRCNN框架，数据集路径是data/coco）
```

iii. 推理检测

现在是关键性的一步也是最后一步，我们可以根据图片检测结果了，我们要设置多个参数，首先是我们需要推理的数据，就是我们之前用来表示这张图片的img，为了准确筛选出检测结果，我们还需要设置2个参数，rpn_threshold：执行非极大值抑制时的阈值, rcnn_threshold：交并比。

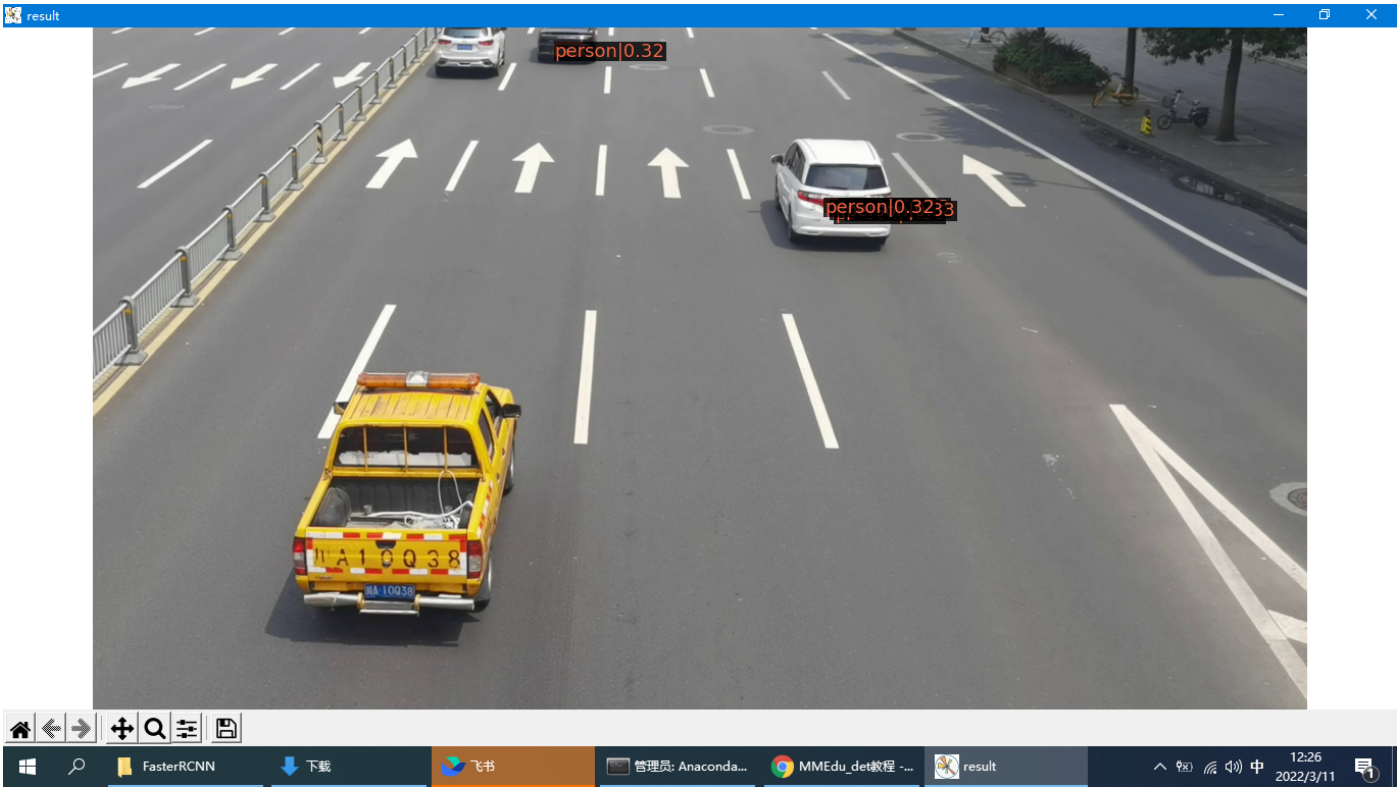
参考代码：

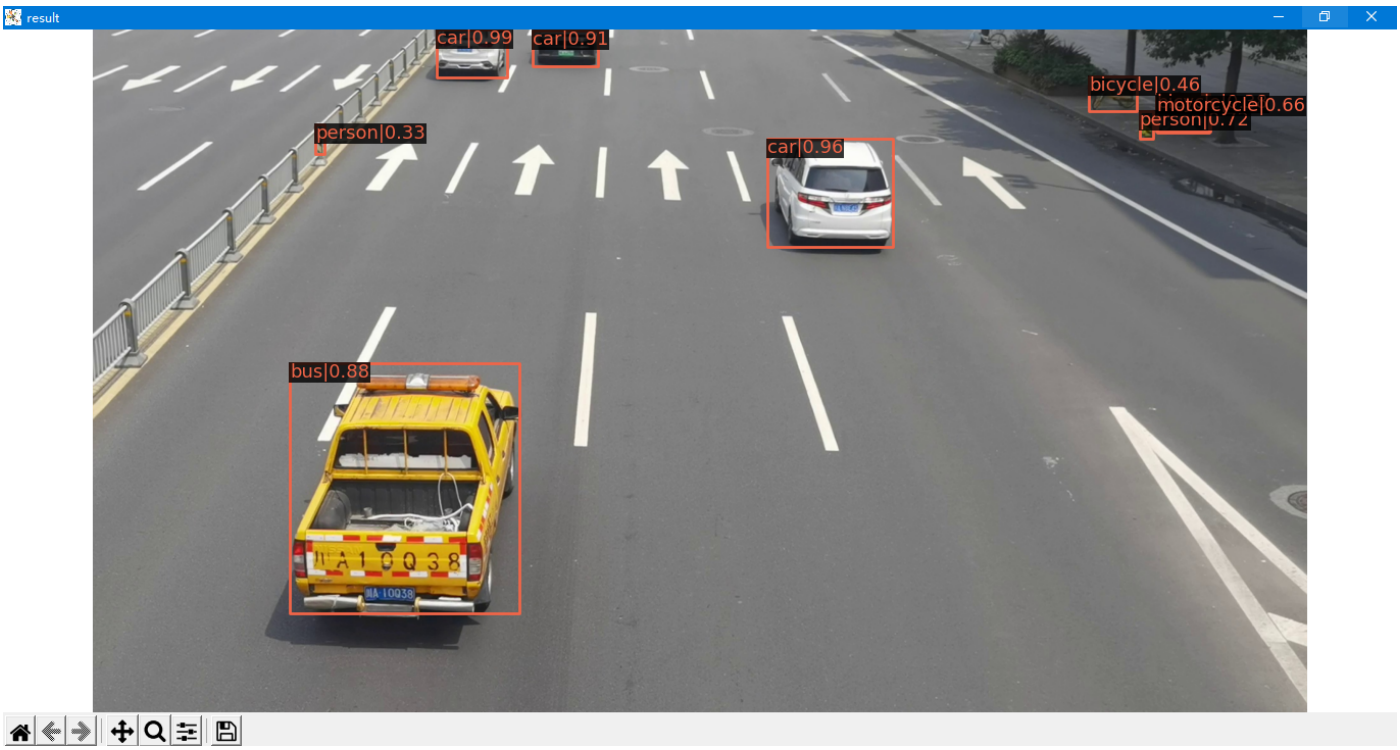
```python
model.inference(infer_data=img, rpn_threshold=0.5, rcnn_threshold=0.3) #推理检测，rpn_threshold：执行非极大值抑制时的阈值，rcnn_threshold：交并比，这两个参数用于筛选检测结果
```

推理结果图：


FasterRCNN
Faster R-CNN说明文档.md
FasterRCNN.pth
FasterRCNN.py


x=72. y=−0.5
[201, 192, 177]
bird|0.99





注：

报错：FileNotFoundError: ./utils/models/FasterRCNN/FasterRCNN.pth can not be found.

解决方案：提前检查该路径该文件是否存在，不存在则告诉用户去下载并重命名。

b. 最简训练

如果您想要简单体验一下使用MMEdu_det进行模型训练，可以参考最简训练的代码：

```python
from utils.my_utils_det import MMDetection# 导入detection模块
model = MMDetection() #创建模型
model.num_classes = 1 #指定类别数量
model.load_dataset(path='data/coco/') #加载数据集coco，路径是data/coco/
model.train(epochs=15) #训练数据集(迭代15轮)
```

接下来为您详细说明：

i. 准备数据集（需技术组同学设计自定义数据集）

官方提供的所有代码都默认使用的是coco格式的数据集，因此我们可以选择使用官方的数据集，可以放在让您新建的data文件夹下，按照以下的目录形式存储.

```
├——— data
|   ├——— coco
|   |   ├——— annotations
|   |   |   ├———train.json
|   |   ├——— train2017
|   |   ├——— val2017
|   |   ├——— test2017
```

ii. 创建模型

我们为您提供的是最简训练的代码，已经为您封装好图像检测模型类，您只需要写model = MMDetection()即可。

参考代码：

```python
model = MMDetection() #创建模型
```

iii. 指定类别数量

接下来的步骤必须在训练数据集前完成，那就是需要根据自己的数据集类别指定类别数量，MMEdu_cls我们也学过指定类别数量，而在MMEdu_det为什么也要进行这个操作呢？是因为检测其实是一个两步的任务，找到物体+对物体分类，所以也涉及了分类的步骤。

参考代码：

Python

```python
model.num_classes = 1 #指定类别数量
```

iv. 配置数据集路径

然后就是配置加载数据集的路径，根据数据集位置配置路径。

参考代码：

Python

```python
model.load_dataset(path='data/coco/') #加载数据集coco，路径是data/coco/
```

v. 训练模型

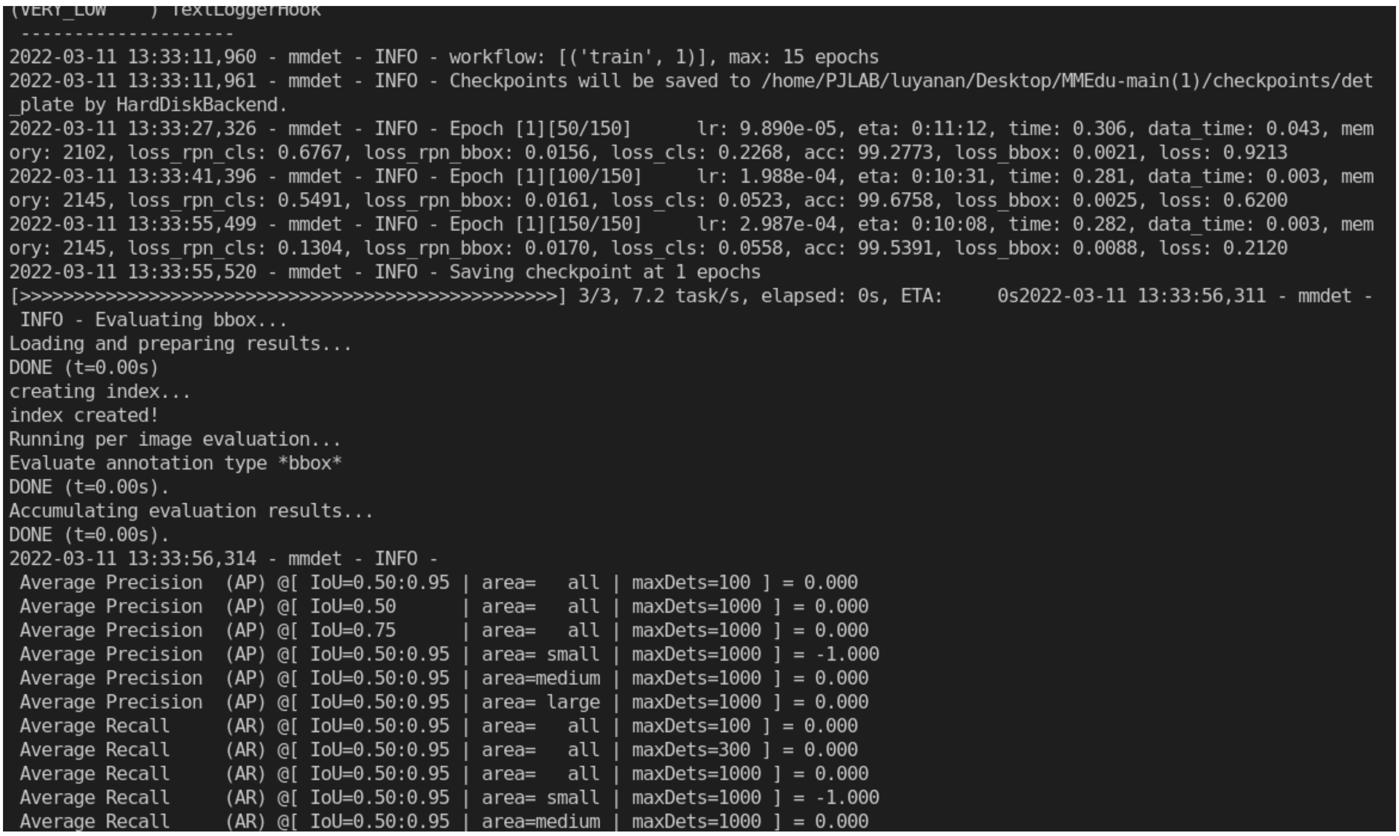
现在就可以训练属于自己的模型了，因为是最简训练，您只需要设置训练迭代轮次，也就是epochs值

参考代码：

Python

```python
model.train(epochs=15) #模型训练(迭代15轮)
```

训练结果：



```
(VERY_LOW    ) TextLoggerHook
 --------------------
2022-03-11 13:33:11,960 - mmdet - INFO - workflow: [('train', 1)], max: 15 epochs
2022-03-11 13:33:11,961 - mmdet - INFO - Checkpoints will be saved to /home/PJLAB/luyanan/Desktop/MMEdu-main(1)/checkpoints/det
_plate by HardDiskBackend.
2022-03-11 13:33:27,326 - mmdet - INFO - Epoch [1][50/150]      lr: 9.890e-05, eta: 0:11:12, time: 0.306, data_time: 0.043, mem
ory: 2102, loss_rpn_cls: 0.6767, loss_rpn_bbox: 0.0156, loss_cls: 0.2268, acc: 99.2773, loss_bbox: 0.0021, loss: 0.9213
2022-03-11 13:33:41,396 - mmdet - INFO - Epoch [1][100/150]     lr: 1.988e-04, eta: 0:10:31, time: 0.281, data_time: 0.003, mem
ory: 2145, loss_rpn_cls: 0.5491, loss_rpn_bbox: 0.0161, loss_cls: 0.0523, acc: 99.6758, loss_bbox: 0.0025, loss: 0.6200
2022-03-11 13:33:55,499 - mmdet - INFO - Epoch [1][150/150]     lr: 2.987e-04, eta: 0:10:08, time: 0.282, data_time: 0.003, mem
ory: 2145, loss_rpn_cls: 0.1304, loss_rpn_bbox: 0.0170, loss_cls: 0.0558, acc: 99.5391, loss_bbox: 0.0088, loss: 0.2120
2022-03-11 13:33:55,520 - mmdet - INFO - Saving checkpoint at 1 epochs
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 3/3, 7.2 task/s, elapsed: 0s, ETA:     0s2022-03-11 13:33:56,311 - mmdet -
 INFO - Evaluating bbox...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.00s).
Accumulating evaluation results...
DONE (t=0.00s).
2022-03-11 13:33:56,314 - mmdet - INFO -
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.000
```

开始训练后，系统会自动输出所训练的轮次情况以及loss值。当loss值的数值趋于稳定且尽可能小时，表明训练已经成熟，可以停止。

c. 典型训练

如果您不只是希望体验模型训练的过程，想要更加深入学习模型训练，可以参考典型训练的代码：

Python

```
from utils.my_utils_det import MMDetection# 导入detection模块
model = MMDetection(backbone='FasterRCNN') #创建模型(使用FasterRCNN框架)
model.num_classes = 1 #指定类别数量
model.load_dataset(path='data/coco/') #加载数据集coco，路径是data/coco/
model.save_fold = "checkpoints/det_plate" #训练文件储存在checkpoints下的det_plate，路径是checkpoints/det_plate
model.train(epochs=15, validate=True, Frozen_stages=1) #模型训练(迭代15轮，用验证数据集进行验证，冻结一行)
#推理检测使用
#model.inference(is_trained=True, pretrain_model = './checkpoints/det_plate/latest.pth', infer_data='./data/coco/images/test/0000.jpg', rpn_threshold=0.5, rcnn_threshold=0.3) #推理验证（用已经训练过的模型，权重文件的路径是./checkpoints/det_plate/latest.pth，要推理的图片路径是./data/coco/images/test/0000.jpg)
```

接下来为您详细说明：

i. 准备数据集（需技术组同学设计自定义数据集）

参考3.b.i

ii. 创建模型

拥有了自己的数据集，就可以开始创建模型了，关于backbone参数还是简易大家指定FasterRCNN框架。

参考代码：

Python

```
model = MMDetection(backbone='FasterRCNN') #创建模型(使用FasterRCNN框架)
```

iii. 指定类别数量

接下来的步骤必须在训练数据集前完成，那就是需要根据自己的数据集类别指定类别数量，这个步骤在MMEdu_cls我们也学过，而在MMEdu_det为什么也要进行这个操作是因为检测其实是一个两步的任务，找到物体+对物体分类，所以也涉及了分类的步骤。

参考代码：

```Python
model.num_classes = 1#指定类别数量
```

iv. 配置数据集路径

然后就是配置加载数据集的路径，根据数据集位置配置路径。

参考代码：

```Python
model.load_dataset(path='data/coco/') #加载数据集coco，路径是data/coco/
```

v. 训练文件储存

接下来这行代码是为了给训练过程中产生的log日志文件、每一轮次的pth文件寻找一个存储路径。

参考代码：

```Python
model.save_fold = "checkpoints/det_plate" #训练文件储存在checkpoints下的det_plate，路径是checkpoints/det_plate
```

vi. 训练模型

现在就可以训练属于自己的模型了，代码中为大家呈现了三个参数，首先是训练的迭代轮次，我们已经体验过很多次。

然后是`validate`参数，这个参数是一个布尔值，也就是只有true和false两种形态。其值为true则代表启用验证集。验证集通常是从训练集中单独划分出的一小块数据集，用来在模型进行训练时，每过固定的周期就进行一次验证，以反馈训练的准确率。模型会根据反馈结果继续调整。做图像检测的模型训练需要启用此值为true。

最后是Frozen_stages，这个参数是为了冻结某些stage的参数，可以使得训练速度更快。

参考代码：

```Python
model.train(epochs=15, validate=True, Frozen_stages=1) #模型训练(迭代15轮，用验证数据集进行验证，冻结一行
```

训练结果：

```
2022-03-10 19:09:37,567 - mmdet - INFO - workflow: [('train', 1)], max: 15 epochs
2022-03-10 19:09:37,567 - mmdet - INFO - Checkpoints will be saved to /home/PJLAB/luyanan/Desktop/MMEdu-main/checkpoints by HardDiskBackend.
/home/PJLAB/luyanan/anaconda3/envs/mmlab/lib/python3.9/site-packages/mmcv/runner/hooks/logger/text.py:112: DeprecationWarning: an integer is required (got type float).  Implicit conversion to integers using __int__
is deprecated, and may be removed in a future version of Python.
  mem_mb = torch.tensor([mem / (1024 * 1024)],
2022-03-10 19:09:52,818 - mmdet - INFO - Epoch [1][50/150]     lr: 9.890e-05, eta: 0:11:06, time: 0.303, data_time: 0.043, memory: 2144, loss_rpn_cls: 0.6872, loss_rpn_bbox: 0.0201, loss_cls: 2.3376, acc: 67.0685,
loss_bbox: 0.0006, loss: 3.0455
2022-03-10 19:10:06,378 - mmdet - INFO - Epoch [1][100/150]    lr: 1.988e-04, eta: 0:10:17, time: 0.272, data_time: 0.003, memory: 2145, loss_rpn_cls: 0.3175, loss_rpn_bbox: 0.0146, loss_cls: 0.1468, acc: 99.3853,
loss_bbox: 0.0038, loss: 0.4827
2022-03-10 19:10:19,431 - mmdet - INFO - Epoch [1][150/150]    lr: 2.987e-04, eta: 0:09:45, time: 0.261, data_time: 0.003, memory: 2145, loss_rpn_cls: 0.0694, loss_rpn_bbox: 0.0157, loss_cls: 0.0897, acc: 98.9587,
loss_bbox: 0.0119, loss: 0.1867
2022-03-10 19:10:19,457 - mmdet - INFO - Saving checkpoint at 1 epochs
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 3/3, 7.8 task/s, elapsed: 0s, ETA:     0s2022-03-10 19:10:20,323 - mmdet - INFO - Evaluating bbox...
Loading and preparing results...
2022-03-10 19:10:20,323 - mmdet - ERROR - The testing results of the whole dataset is empty.
2022-03-10 19:10:20,323 - mmdet - INFO - Epoch(val) [1][3]
2022-03-10 19:10:35,900 - mmdet - INFO - Epoch [2][50/150]     lr: 3.986e-04, eta: 0:09:47, time: 0.310, data_time: 0.043, memory: 2145, loss_rpn_cls: 0.0372, loss_rpn_bbox: 0.0115, loss_cls: 0.0428, acc: 99.0360,
loss_bbox: 0.0132, loss: 0.1048
2022-03-10 19:10:48,911 - mmdet - INFO - Epoch [2][100/150]    lr: 4.985e-04, eta: 0:09:22, time: 0.260, data_time: 0.003, memory: 2145, loss_rpn_cls: 0.0502, loss_rpn_bbox: 0.0159, loss_cls: 0.0435, acc: 98.9392,
loss_bbox: 0.0170, loss: 0.1266
2022-03-10 19:11:02,066 - mmdet - INFO - Epoch [2][150/150]    lr: 5.984e-04, eta: 0:09:02, time: 0.263, data_time: 0.003, memory: 2145, loss_rpn_cls: 0.0354, loss_rpn_bbox: 0.0123, loss_cls: 0.0256, acc: 99.2781,
loss_bbox: 0.0123, loss: 0.0856
2022-03-10 19:11:02,087 - mmdet - INFO - Saving checkpoint at 2 epochs
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 3/3, 7.7 task/s, elapsed: 0s, ETA:     0s2022-03-10 19:11:02,934 - mmdet - INFO - Evaluating bbox...
Loading and preparing results...
```

开始训练后，系统会自动输出所训练的轮次情况以及loss值。当loss值的数值趋于稳定且尽可能小时，表明训练已经成熟，可以停止。

```
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.00s).
Accumulating evaluation results...
DONE (t=0.00s).
2022-03-10 19:19:46,027 - mmdet - INFO -
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.483
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.168
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.501
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.500

2022-03-10 19:19:46,027 - mmdet - INFO - Epoch(val) [14][3]    bbox_mAP: 0.4830, bbox_mAP_50: 1.0000, bbox_mAP_75: 0.1680, bbox_mAP_s: -1.0000, bbox_mAP_m: 0.5010, bbox_mAP_l: 0.5000, bbox_mAP_copypaste: 0.483 1.00
0 0.168 -1.000 0.501 0.500
2022-03-10 19:20:01,992 - mmdet - INFO - Epoch [15][50/150]    lr: 1.000e-05, eta: 0:00:28, time: 0.318, data_time: 0.043, memory: 2145, loss_rpn_cls: 0.0039, loss_rpn_bbox: 0.0069, loss_cls: 0.0059, acc: 99.7748,
loss_bbox: 0.0081, loss: 0.0249
2022-03-10 19:20:15,445 - mmdet - INFO - Epoch [15][100/150]   lr: 1.000e-05, eta: 0:00:14, time: 0.269, data_time: 0.003, memory: 2145, loss_rpn_cls: 0.0023, loss_rpn_bbox: 0.0054, loss_cls: 0.0039, acc: 99.8336,
loss_bbox: 0.0073, loss: 0.0190
2022-03-10 19:20:28,960 - mmdet - INFO - Epoch [15][150/150]   lr: 1.000e-05, eta: 0:00:00, time: 0.270, data_time: 0.003, memory: 2145, loss_rpn_cls: 0.0034, loss_rpn_bbox: 0.0053, loss_cls: 0.0049, acc: 99.7970,
loss_bbox: 0.0096, loss: 0.0230
2022-03-10 19:20:28,981 - mmdet - INFO - Saving checkpoint at 15 epochs
```

d. 继续训练

如果您在学习过程中发现某个模型训练后准确率不够高想继续训练，或者是想继续训练上次来不及训练完的模型（特别是突然下课了，不能继续操作了），那么您可以学习一下下面的代码。

```Python
from utils.my_utils_det import MMDetection# 导入detection模块
model = MMDetection(backbone='FasterRCNN') #创建模型(使用FasterRCNN框架)
model.num_classes = 1#指定类别数量
model.load_dataset(path='data/coco/') #加载数据集coco，路径是data/coco/
model.save_fold = "checkpoints/det_plate" #训练文件储存在checkpoints下的det_plate，路径是checkpoints/det_plate
model.train(epochs=15, checkpoint='./checkpoints/det_plate/latest.pth', validate=True, Frozen_stages=1) #继续训练(迭代15轮，用已经训练过的模型，权重文件的路径是./checkpoints/det_plate/latest.pth，冻结一行)
```

接下来为您详细说明：

i. 准备数据集（需技术组同学设计自定义数据集）

参考3.b.i

ii. 创建模型

拥有了自己的数据集，就可以开始创建模型了，我们可以自主指定backbone参数，如baFaster RCNN，Mask RCNN，R-FCN，RetinaNet， Cascade R-CNN及ssd 等

参考代码：

```Python
model = MMDetection(backbone='FasterRCNN') #创建模型(使用FasterRCNN框架)
```

iii. 指定类别数量

同样的，我们需要根据自己的数据集类别指定类别数量

参考代码：

```Python
model.num_classes = 1 #指定类别数量
```

iv. 配置数据集路径

然后就是配置加载数据集的路径，根据数据集位置配置路径

参考代码：

```Python
model.load_dataset(path='data/coco/') #加载数据集coco，路径是data/coco/
```

v. 训练文件储存

接下来这行代码是为了给训练过程中产生的log日志文件、每一轮次的pth文件寻找一个存储路径

参考代码：

```Python
model.save_fold = "checkpoints/det_plate" #训练文件储存在checkpoints下的det_plate，路径是checkpoints/det_plate
```

vi. 继续训练

由于是继续训练，我们需要使用已经训练过的模型，那么就需要载入训练该模型时产生的最新的权重文件latest.pth，我们可以在我们设置的存储训练文件的文件夹下放入这个最新的权重文件，其他设置与典型训练一致。

参考代码：

```python
model.train(epochs=15, checkpoint='./checkpoints/det_plate/latest.pth', validate=True, Frozen_stages=1) #继续训练(迭代15轮，用已经训练过的模型，权重文件的路径是./checkpoints/det_plate/latest.pth，冻结一行)
```

训练结果：

算法同学还没上传pth文件

开始训练后，系统会自动输出所训练的轮次情况以及loss值。当loss值的数值趋于稳定且尽可能小时，表明训练已经成熟，可以停止。

4. MMEdu_det应用经典范例