## MMDetection是一个开源的目标检测工具箱，用于训练和部署目标检测模型。它基于PyTorch深度学习框架，并提供了丰富的目标检测算法和模型实现，包括Faster R-CNN、Mask R-CNN、Cascade R-CNN、RetinaNet等。MMDetection提供了易于使用的API和丰富的功能，使得用户可以方便地进行目标检测任务的开发和研究。同时，MMDetection还支持多种数据增强方法、模型融合、分布式训练等功能，以提升目标检测模型的性能和效果。

## Installation

Please refer to this website for installation instructions.

https://mmdetection.readthedocs.io/en/latest/get\_started.html

python demo/image\_demo.py demo/demo.jpg rtmdet\_tiny\_8xb32-300e\_coco.py --weights rtmdet\_tiny\_8xb32-300e\_coco\_20220902\_112414-78e30dcc.pth --device cuda

python demo/image\_demo.py demo/test2.jpg rtmdet\_tiny\_8xb32-300e\_coco.py --weights rtmdet\_tiny\_8xb32-300e\_coco\_20220902\_112414-78e30dcc.pth --device cuda

python demo/image\_demo.py demo/demo.jpg retinanet\_r50\_fpn\_1x\_coco.py --weights retinanet\_r50\_fpn\_1x\_coco\_20200130-c2398f9e.pth --device cuda

python demo/image\_demo.py demo/demo.jpg retinanet\_r50\_fpn\_2x\_coco.py --weights retinanet\_r50\_fpn\_2x\_coco\_20200131-fdb43119.pth --device cuda

python demo/image\_demo.py demo/demo.jpg work\_dirs/retinanet\_r50\_fpn\_1x\_coco/retinanet\_r50\_fpn\_1x\_coco.py --weights work\_dirs/retinanet\_r50\_fpn\_1x\_coco/epoch\_12.pth --device cuda

python demo/image\_demo.py demo/test.jpg work\_dirs/retinanet\_r50\_fpn\_1x\_coco/retinanet\_r50\_fpn\_1x\_coco.py --weights work\_dirs/retinanet\_r50\_fpn\_1x\_coco/epoch\_12.pth --device cuda

python demo/image\_demo.py demo/test.jpg work\_dirs/retinanet\_r50\_fpn\_1x\_coco/retinanet\_r50\_fpn\_1x\_coco.py --weights work\_dirs/retinanet\_r50\_fpn\_1x\_coco/epoch\_12.pth --device cuda

python demo/image\_demo.py demo/black\_ducks.jpg work\_dirs/retinanet\_r50\_fpn\_1x\_coco/retinanet\_r50\_fpn\_1x\_coco.py --weights work\_dirs/retinanet\_r50\_fpn\_1x\_coco/epoch\_12.pth --device cuda

python demo/image\_demo.py demo/black\_ducks.jpg work\_dirsMFL1.75/retinanet\_r50\_fpn\_1x\_coco.py --weights work\_dirsMFL1.75/epoch\_12.pth --device cuda

python demo/image\_demo.py demo/white\_ducks.jpg work\_dirs\_cityscapes\_lr/retinanet\_r50\_fpn\_1x\_cityscapes.py --weights work\_dirs\_cityscapes\_lr/epoch\_12.pth --device cuda

python demo/image\_demo.py demo/white\_ducks.jpg work\_dirs\_cityscapes\_MFL1.75\_lr/retinanet\_r50\_fpn\_1x\_cityscapes.py --weights work\_dirs\_cityscapes\_MFL1.75\_lr/epoch\_12.pth --device cuda

该命令是在Python环境下运行一个名为image\_demo.py的脚本文件，并传入一些参数。具体解释如下： - `python demo/image\_demo.py`：运行名为image\_demo.py的Python脚本文件。

- `demo/demo.jpg`：作为输入的图片路径，即要进行目标检测的图片。

- `rtmdet\_tiny\_8xb32-300e\_coco.py`：模型配置文件的路径，该文件定义了模型的结构和参数。

- `--weights rtmdet\_tiny\_8xb32-300e\_coco\_20220902\_112414-78e30dcc.pth`：指定了模型的权重文件路径，即已经训练好的模型参数。

- `--device cuda`：指定使用的设备为cuda，即使用GPU进行计算加速

为了验证 MMDetection 是否安装正确，我们提供了一些示例代码来执行模型推理。

****步骤 1.**** 我们需要下载配置文件和模型权重文件。

mim download mmdet --config rtmdet\_tiny\_8xb32-300e\_coco --dest .

mim download mmdet --config retinanet\_r50\_fpn\_1x\_coco --dest .

mim download mmdet --config retinanet\_r50\_fpn\_2x\_coco --dest .

下载将需要几秒钟或更长时间，这取决于你的网络环境。完成后，你会在当前文件夹中发现两个文件 rtmdet\_tiny\_8xb32-300e\_coco.py 和 rtmdet\_tiny\_8xb32-300e\_coco\_20220902\_112414-78e30dcc.pth。

****步骤 2.**** 推理验证。

方案 a：如果你通过源码安装的 MMDetection，那么直接运行以下命令进行验证：

python demo/image\_demo.py demo/demo.jpg rtmdet\_tiny\_8xb32-300e\_coco.py --weights rtmdet\_tiny\_8xb32-300e\_coco\_20220902\_112414-78e30dcc.pth --device cpu

你会在当前文件夹中的 outputs/vis 文件夹中看到一个新的图像 demo.jpg，图像中包含有网络预测的检测框。

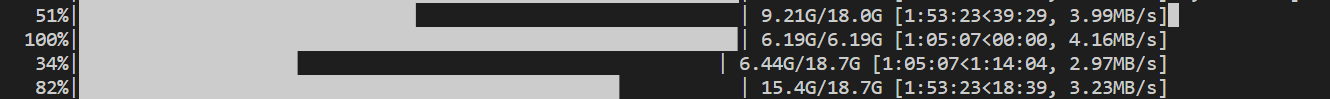
方案 b：如果你通过 MIM 安装的 MMDetection，那么可以打开你的 Python 解析器，复制并粘贴以下代码：

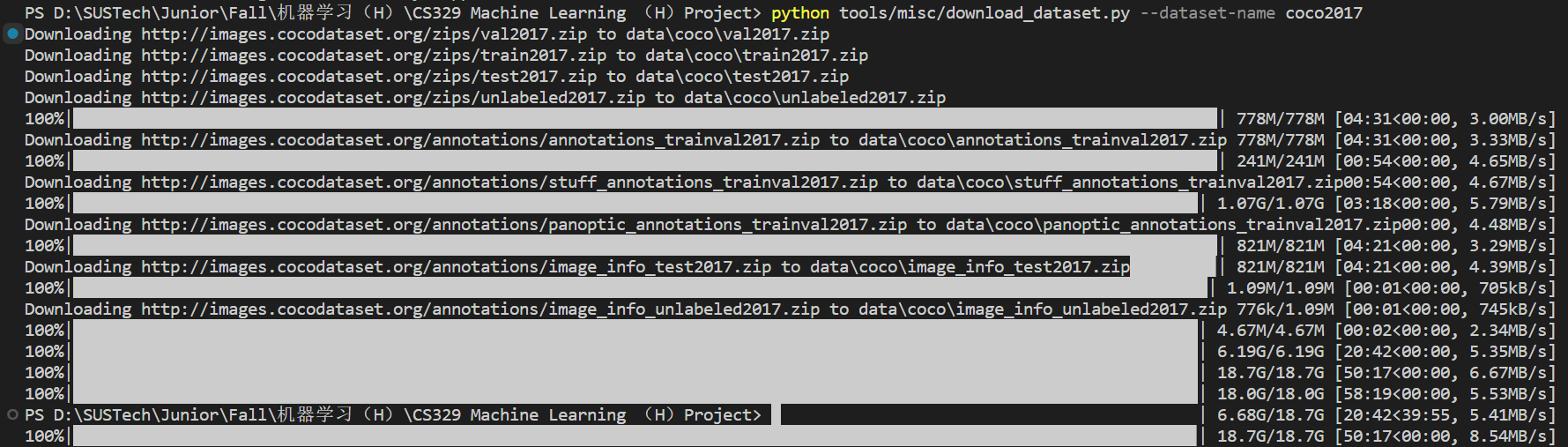
**from** **mmdet.apis** **import** **init\_detector,** **inference\_detector**

**config\_file** **=** 'rtmdet\_tiny\_8xb32-300e\_coco.py'**checkpoint\_file** **=** 'rtmdet\_tiny\_8xb32-300e\_coco\_20220902\_112414-78e30dcc.pth'**model** **=** **init\_detector(config\_file,** **checkpoint\_file,** **device=**'cpu'**)** *# or device='cuda:0'***inference\_detector(model,** 'demo/demo.jpg'**)**

你将会看到一个包含 DetDataSample 的列表，预测结果在 pred\_instance 里，包含有检测框，类别和得分。

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_2x\_coco.py





salloc -p gpulab02 -N1 -c6 --gres=gpu:1 -q gpulab02

salloc -p gpulab02 -N1 -c24 --gres=gpu:4 -q gpulab02

salloc -p gpulab02 -N1 -c18 --gres=gpu:3 -q gpulab02

BwReSdcT(8p6q^Z113

#!/bin/bash

#SBATCH -o job.%12112328.out # 脚本执行的输出将被保存在当job.%j.out文件下，%j表示作业号;

#SBATCH --partition=gpulab02 # 作业提交的指定分区队列为titan

#SBATCH --qos=gpulab02 # 指定作业的QOS

#SBATCH -J ZHENGZubinGPUJob # 作业在调度系统中的作业名为myFirstJob;

#SBATCH --nodes=1 # 申请节点数为1,如果作业不能跨节点(MPI)运行, 申请的节点数应不超过1

#SBATCH --ntasks-per-node=6 # 每个节点上运行一个任务，默认一情况下也可理解为每个节点使用一个核心；

#SBATCH --gres=gpu:4 # 指定作业的需要的GPU卡数量，集群不一样，注意最大限制;

nvidia-smi

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_coco.py

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_coco.py --work-dir work\_dirsMFL

python tools/train.py configs/retinanet/retinanet\_r18\_fpn\_1x\_coco.py --work-dir work\_dir1

python tools/train.py configs/rtmdet/rtmdet\_tiny\_8xb32-300e\_coco.py

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_cityscapes.py --work-dir work\_dirs\_cityscapes\_lr

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_cityscapes.py --work-dir work\_dirs\_cityscapes\_MFL1.75\_lr

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_cityscapes.py --work-dir work\_dirs\_cityscapes\_MFL2.0\_lr

python tools/test.py \

work\_dirs/retinanet\_r50\_fpn\_1x\_coco/retinanet\_r50\_fpn\_1x\_coco.py \

work\_dirs/retinanet\_r50\_fpn\_1x\_coco/epoch\_12.pth \

--out work\_dirs/retinanet\_r50\_fpn\_1x\_coco/result.pkl \

--show-dir work\_dirs/retinanet\_r50\_fpn\_1x\_coco/infered\_images

python tools/test.py \

work\_dirs\_cityscapes/retinanet\_r50\_fpn\_1x\_cityscapes.py \

work\_dirs\_cityscapes/epoch\_12.pth \

--out work\_dirs\_cityscapes/result.pkl \

--show-dir work\_dirs\_cityscapes/infered\_images

python tools/test.py \

work\_dirs\_cityscapes\_MFL1.75/retinanet\_r50\_fpn\_1x\_cityscapes.py \

work\_dirs\_cityscapes\_MFL1.75/epoch\_12.pth \

--out work\_dirs\_cityscapes\_MFL1.75/result.pkl \

--show-dir work\_dirs\_cityscapes\_MFL1.75/infered\_images

Abstract: advertisement, novelty

8 sentences:

1. challenges/problems
2. This paper presents...
3. The novelty of this works is the follows
4. (1)
5. (2)
6. (3)
7. Evaluation platform
8. The proposed algorithm outperforms

Introduction: motivation, contribution

Conclusion: finished... can improve.. results, future work

python tools/analysis\_tools/analyze\_logs.py plot\_curve work\_dirs/retinanet\_r50\_fpn\_1x\_coco/20231204\_222508/vis\_data/20231204\_222508.json --keys loss\_cls --legend loss\_cls

python tools/analysis\_tools/analyze\_logs.py plot\_curve work\_dirs/retinanet\_r50\_fpn\_1x\_coco/20231204\_222508/vis\_data/20231204\_222508.json --keys loss\_cls loss\_bbox --out losses.pdf

用python的seaborn绘制FL\_1(p\_t) = −((1 − p\_t) ^2 )log(p\_t) 与FL\_2(p\_t) = −log(p\_t)/(p\_t)^{\alpha},的超参数{\alpha}分别为0.125, 0.25, 0.5, 1, 2的图像(六条曲线画在同一张)， 其中自变量p\_t 在0到1之间。横轴标签为probability of ground truth class，纵轴标签为loss，要有6条曲线的图例

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_coco.py --work-dir work\_dirsMFL0.125

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_coco.py --work-dir work\_dirsMFL0.25

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_coco.py --work-dir work\_dirsMFL0.5

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_coco.py --work-dir work\_dirsMFL2.0

python tools/train.py configs/retinanet/retinanet\_r50\_fpn\_1x\_coco.py --work-dir work\_dirsMFL1.0

2.0

1.75

tmux 0, 2

