《计算机视觉》实验单元二

目标检测

|  |  |
| --- | --- |
| https://www.scut.edu.cn/_upload/article/images/93/f1/da8bef494e929b2303b75fcae24a/395da264-74a6-4841-ae9f-3530ed01cf84.jpg | C:\Users\jwx341670\AppData\Local\Microsoft\Windows\INetCache\Content.Word\HW_POS_RBG_Vertical-150ppi.png |
| 华南理工大学 | 华为技术有限公司 |

版本说明：2022年春季华南理工版，

修订依据：华为智能基座《计算机视觉》课件包v2.1

使用者： 软件工程专业 2019级 数字媒体方向 第六学期《计算机视觉》课

修订人： 华南理工大学软件学院 彭绍武，张熠鹏，李泽淇，赵鹏程，黄锟城，毛玺羽

教师：彭绍武

助教：赵鹏程，李泽淇

目录

[1 实验1介绍 3](#_Toc106796435)

[1.2 实验目的 3](#_Toc106796436)

[1.3 实验清单 4](#_Toc106796437)

[1.4 实验开发环境 4](#_Toc106796438)

[1.5 开发平台介绍 4](#_Toc106796439)

[2 搭建SSD网络实现目标检测 6](#_Toc106796440)

[2.1 集成开发环境和AI框架准备 6](#_Toc106796441)

[2.2 SSD模型简介及代码、数据下载 6](#_Toc106796442)

[2.2.1 SSD模型简介 6](#_Toc106796443)

[2.2.2 获取代码文件 7](#_Toc106796444)

[2.2.3 COCO2017数据集 8](#_Toc106796445)

[2.3 脚本说明 10](#_Toc106796446)

[2.3.1 脚本及样例代码 10](#_Toc106796447)

[2.3.2 脚本参数 11](#_Toc106796448)

[2.4 运行工程 12](#_Toc106796449)

[2.4.1 打开项目文件 12](#_Toc106796450)

[2.4.2 模型训练 13](#_Toc106796451)

[2.4.3 模型评估 16](#_Toc106796452)

[2.5 实验小结 18](#_Toc106796453)

[3 实验2介绍 19](#_Toc106796454)

[3.2 实验目的 19](#_Toc106796455)

[3.3 实验清单 19](#_Toc106796456)

[3.4 实验开发环境 20](#_Toc106796457)

[3.5 开发平台介绍 20](#_Toc106796458)

[4 搭建Faster-RCNN网络实现目标检测 21](#_Toc106796459)

[4.1 数据集准备 21](#_Toc106796460)

[4.2 Faster-RCNN模型介绍 22](#_Toc106796461)

[4.2.1 模型架构 22](#_Toc106796462)

[4.2.2 获取代码文件与NoteBook创建 23](#_Toc106796463)

[4.3 Faster-RCNN模型训练 25](#_Toc106796464)

[4.3.1 依赖安装 25](#_Toc106796465)

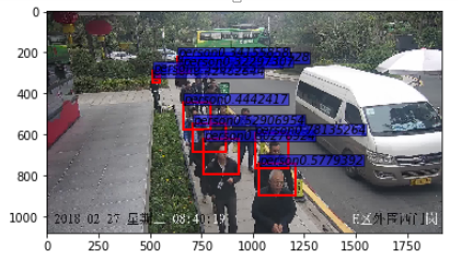
[4.3.2 训练 25](#_Toc106796466)

[4.3.3 评估 27](#_Toc106796467)

[5 开放性问题 29](#_Toc106796468)

# 实验1介绍

目标检测任务是找出图像或视频中人们感兴趣的物体，并同时检测它们的位置和大小。不同于图像分类任务，目标检测不仅要解决分类问题，还要解决定位问题，是属于Multi-Task的问题。如图1-1所示人脸检测就是目标检测任务应用之一。



人脸检测

作为计算机视觉的基本问题之一，目标检测构成了许多其他视觉任务的基础，例如实例分割、图像标注和目标跟踪等等；在基于区域的卷积神经网络提出后，深度卷积网络开始在目标检测领域普及，更快的基于区域的卷积神经网络将整个目标检测过程合成在一个统一的深度网络框架上．随后 YOLO 和 SSD 等目标检测框架的提出进一步提升目标检测的效率。

## 实验目的

本实验基于MindSpore深度学习开源框架，使用昇腾全流程开发工具—MindStudio进行SSD目标检测模型的搭建，并使用COCO2017开源数据集进行模型训练与评估。

通过模型训练，我们对SSD模型、常见参数（如epoch、batch-size等）的含义及设置会有进一步的了解；通过模型评估，我们对目标检测常用评估指标(AP、mAP)会有深入的了解。

## 实验清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| 基于MindStudio搭建SSD网络实现目标检测任务 | 本实验基于MindSpore深度学习开源框架，使用昇腾全流程开发工具—MindStudio进行SSD目标检测框架的搭建，并使用COCO2017开源数据集进行模型训练与评估。 | 中级 | Python3.7.5、MindSpore1.5、MindStudio3.0.4 | PC机配置要求：  Windows操作系统；CPU不小于2核2.0GHz；内存不小于8G |

## 实验开发环境

实验平台：windows PC机、MindStudio3.0.4

框架：MindSpore1.5

硬件：CPU

## 开发平台介绍

MindSpore是一种适用于端边云场景的新型开源深度学习训练/推理框架。 MindSpore提供了友好的设计和高效的执行，旨在提升数据科学家和算法工程师的开发体验，并为Ascend AI处理器提供原生支持，以及软硬件协同优化。

同时，MindSpore作为全球AI开源社区，致力于进一步开发和丰富AI软硬件应用生态。

更多关于MindSpore框架的知识信息，请参考如下链接：

<https://www.mindspore.cn/>

MindStudio提供您在AI开发所需的一站式开发环境，支持模型开发、算子开发以及应用开发三个主流程中的开发任务，依靠模型可视化、算力测试、IDE本地仿真调试等功能，MindStudio能够帮助您在一个工具上就能高效便捷地完成AI应用开发。

MindStudio采用了插件化扩展机制，开发者可以通过开发插件来扩展已有功能。

更多MindStudio工具使用，请参考如下链接： <https://support.huaweicloud.com/mindstudio3036/index.html>

# 搭建SSD网络实现目标检测

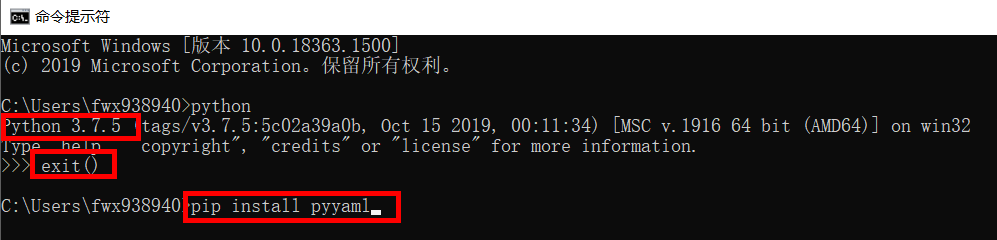
## 集成开发环境和AI框架准备

该环节操作均在本地windows下进行。

请参考《MindStudio环境搭建指南》手册，完成集成开发环境MindStudio和AI框架MindSpore、相关python依赖库的安装。



本实验的进行需要一些第三方库，请安装numpy，pycocotools，opencv-python，xml-python，Pillow，pyyaml等程序库到python3.7.5环境下，下图是安装示意图。



安装命令如下：

pip install numpy

pip install pycocotools-windows

pip install opencv-python –i <https://pypi.tuna.tsinghua.edu.cn/simple> #添加镜像，安装速度较快

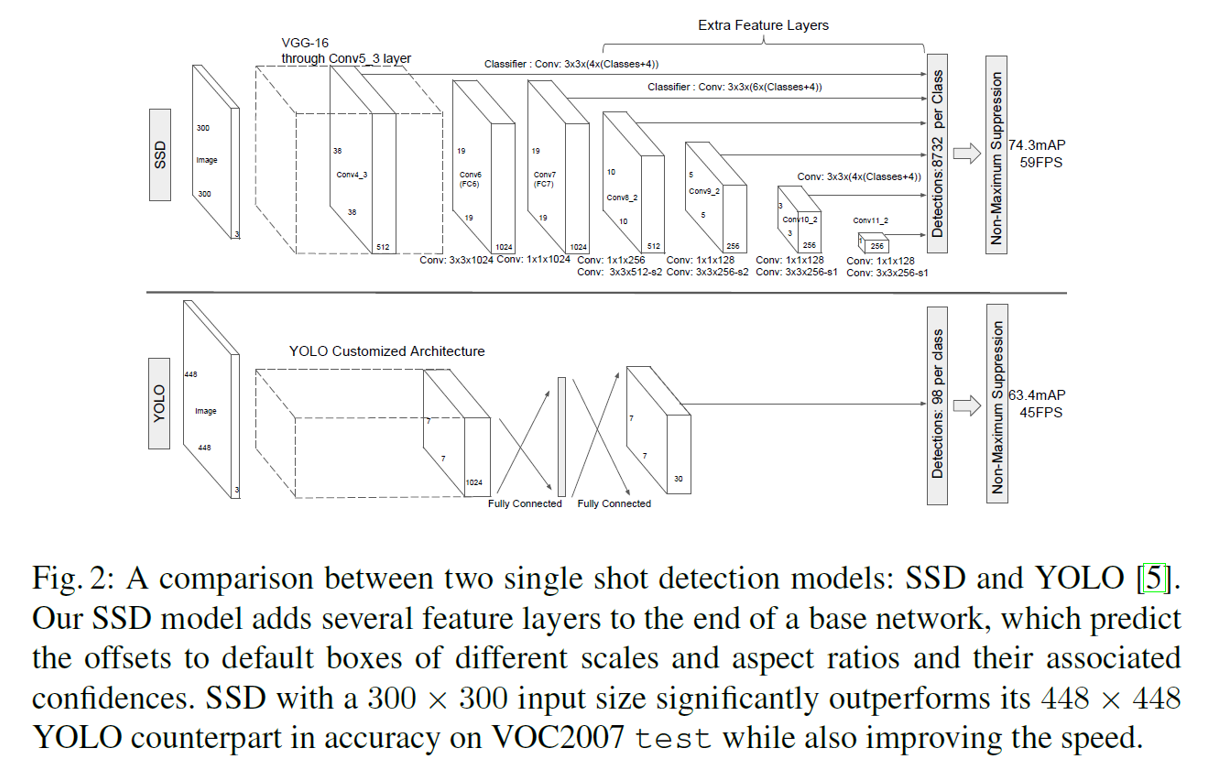
pip install xml-python

pip install Pillow

pip install pyyaml

## SSD模型简介及代码、数据下载

### SSD模型简介



SSD模型框架图

总体来讲，SSD将边界框的输出空间离散成一组默认框，每个特征映射位置具有不同的纵横比和尺度。在预测时，网络对每个默认框中存在的对象类别进行评分，并对框进行调整以更好地匹配对象形状。此外，网络将多个不同分辨率的特征映射的预测组合在一起，自然处理各种大小的对象。

论文参考如下：<https://arxiv.org/abs/1512.02325>

从模型架构角度讲，SSD方法基于前向卷积网络，该网络产生固定大小的边界框集合，并针对这些框内存在的对象类实例进行评分，然后通过非极大值抑制步骤进行最终检测。早期的网络层基于高质量图像分类的标准体系结构，被称为基础网络。后来通过向网络添加辅助结构进行检测。

本案例提供了4种不同的基础架构：

* ssd300，参考论文实现。使用mobilenet-v2作为骨干网络，并使用和论文相同的bbox预测器。
* ssd-mobilenet-v1-fpn，使用mobilenet-v1和FPN作为特征提取器，并使用权重共享box预测器。
* ssd-resnet50-fpn，使用resnet50和FPN作为特征提取器，并使用权重共享box预测器。
* ssd-vgg16，参考论文实现。使用vgg16作为骨干网络，并使用和论文相同的bbox预测器。

### 获取代码文件

在本机D盘新建文件夹code

此时目录路径为D:\code（目录路径根据自己的情况而定）。

进入gitee项目网站，下载项目压缩包。

网站地址：https://gitee.com/mindspore/models/tree/r1.5/official/cv/ssd，直接下载zip，将下载的项目文件放于步骤1创建的目录下，解压缩即可。

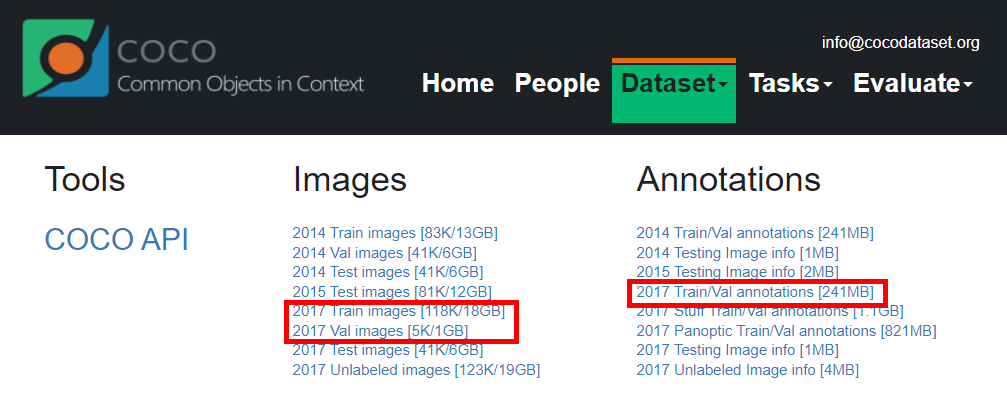


### COCO2017数据集

COCO数据集是一个大型的、丰富的物体检测，分割和字幕数据集。这个数据集以scene understanding为目标，主要从复杂的日常场景中截取，图像中的目标通过精确的segmentation进行位置的标定。图像包括91类目标，328,000影像和2,500,000个label。目前为止有语义分割的最大数据集，提供的类别有80 类，有超过33 万张图片，其中20 万张有标注，整个数据集中个体的数目超过150 万个。

数据集地址（暂不需下载）：[https://cocodataset.org/#download](https://cocodataset.org/" \l "download)

本实验使用的数据集文件如下图所示：



实验所用数据

数据集详细如下：

* 训练数据集：2017 Train images[118K/18GB]，**18GB**，118000张图像；
* 评估数据集：2017 Val images[5K/1GB]，**1GB**，5000张图像；
* 标注信息文件：2017 Train/Val annotations [241MB]，**241MB**， json文件；

从上述信息可知，训练数据集和评估数据集都较大，尤其是训练数据集，18G，下载不太方便；**所以，本实验中我们也提供了COCO数据集的刷选版本 — cocodataset**。

cocodataset文件夹目录结构及所含文件详细信息如下：

.

└─cocodataset

├─annotations #标注文件夹

├─mini\_instance\_train2017.json #训练数据标注信息

└─mini\_instance\_val2017.json #验证数据标注信息

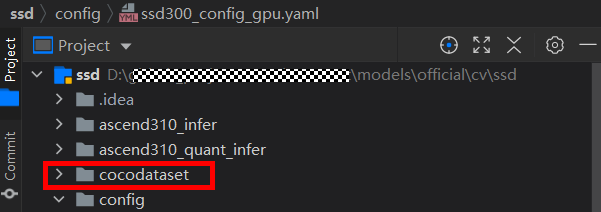
├─val2017 #验证数据集，1000张图片，154MB

└─train2017 #训练数据集，5000张图片，789MB

数据下载路径如下：

<https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/MindStudio-pc/cocodataset.zip>

数据下载解压后得到cocodataset文件夹，并将cocodataset文件夹置于ssd项目文件夹下，如下图所示：



数据文件夹cocodataset

注：实验可以尝试使用自己的数据集，将数据集信息整理成TXT文件，每行如下：

train2017/0000001.jpg 0,259,401,459,7 35,28,324,201,2 0,30,59,80,2

每行是按空间分割的图像标注，第一列是图像的相对路径，其余为[xmin,ymin,xmax,ymax,class]格式的框和类信息。我们从IMAGE\_DIR（数据集目录）和ANNO\_PATH（TXT文件路径）的相对路径连接起来的图像路径中读取图像。在\*yaml中设置IMAGE\_DIR和ANNO\_PATH。

在脚本执行时需要修改相应的数据集路径。

## 脚本说明

### 脚本及样例代码

项目文件中的脚本如下所示，本实验所用的脚本字体已加粗。

.

└─ cv

└─ ssd

├─ README.md ## SSD相关说明

├─ ascend310\_infer ## 实现310推理源代码

├─ scripts

├─ docker start.sh ## 容器启动脚本

├─ run\_distribute\_train.sh ## Ascend分布式shell脚本

├─ run\_distribute\_train\_gpu.sh ## GPU分布式shell脚本

├─ run\_eval.sh ## Ascend评估shell脚本

├─ run\_eval\_gpu.sh ## GPU评估shell脚本

└─ run\_infer\_310.sh ## 310推理脚本

├─ src

├─ \_\_init\_\_.py ## 初始化文件

├─ anchor\_generator.py ## 锚点生成器

├─ **box\_util.py** ## bbox工具

├─ config.py ## 总配置

├─ config\_ssd\_mobilenet\_v1\_fpn.py ## 特征提取网络使用mobilenet-v1 with fpn

├─ config\_ssd\_resnet50\_fpn.py ## 特征提取网络使用resnet50 with fpn

├─ config\_ssd\_vgg16.py ## 特征提取网络使用vgg16

├─ config\_ssd300.py ## 特征提取网络使用mobilenet-v2

├─ **dataset.py** ## 创建并处理数据集

├─ **eval\_callback.py** ## eval回调方法定义

├─ **eval\_utils.py** ## eval工具

├─ fpn.py ## 特征金字塔网络

├─ **init\_params.py** ## 参数工具

├─ **lr\_schedule.py** ## 学习率生成器

├─ mobilenet\_v1.py ## mobilenet-v1网络定义

├─ resnet.py ## resnet网络定义

├─ **ssd.py**  ## SSD架构

└─ vgg16.py ## vgg16网络定义

├── **model\_utils**

│ ├──**config.py**  ## 参数配置

│ ├──device\_adapter.py ## 设备配置

│ ├──local\_adapter.p**y** ## 本地设备配置

│ ├──**moxing\_adapter.py**  ## modelarts设备配置

├─ config

├─ ssd\_mobilenet\_v1\_fpn\_config.yaml ## 参数配置

├─ ssd\_resnet50\_fpn\_config.yaml ## 参数配置

├─ ssd\_vgg16\_config.yaml ## 参数配置

├─ **ssd300\_config.yaml** ## 参数配置

├─ ssd\_mobilenet\_v1\_fpn\_config\_gpu.yaml ## GPU参数配置

├─ ssd\_resnet50\_fpn\_config\_gpu.yaml ## GPU参数配置

├─ ssd\_vgg16\_config\_gpu.yaml ## GPU参数配置

├─ ssd300\_config\_gpu.yaml ## GPU参数配置

├─ Dockerfile ## docker文件

├─ **eval.py** ## 评估脚本

├─ export.py ## 导出 AIR,MINDIR模型的脚本

├─ postprocess.py ## 310推理后处理脚本

├─ **train.py** ## 训练脚本

└─ mindspore\_hub\_conf.py ## MindSpore Hub接口

### 脚本参数

train.py和config.py中主要参数如下：

"device\_num": 1 # 使用设备数量

"lr": 0.05 # 学习率初始值

"dataset": coco # 数据集名称

"epoch\_size": 500 # 轮次大小

"batch\_size": 32 # 输入张量的批次大小

"pre\_trained": None # 预训练检查点文件路径

"pre\_trained\_epoch\_size": 0 # 预训练轮次大小

"save\_checkpoint\_epochs": 10 # 默认情况下，每10个轮次都会保存检查点。

"loss\_scale": 1024 # 损失放大

"class\_num": 81 # 数据集类数

"image\_shape": [300, 300] # 作为模型输入的图像高和宽

"mindrecord\_dir": "/data/MindRecord\_COCO" # MindRecord路径

"coco\_root": "/data/coco2017" # COCO2017数据集路径

"voc\_root": "" # VOC原始数据集路径

"image\_dir": "" # 其他数据集图片路径，如果使用coco或voc，此参数无效。

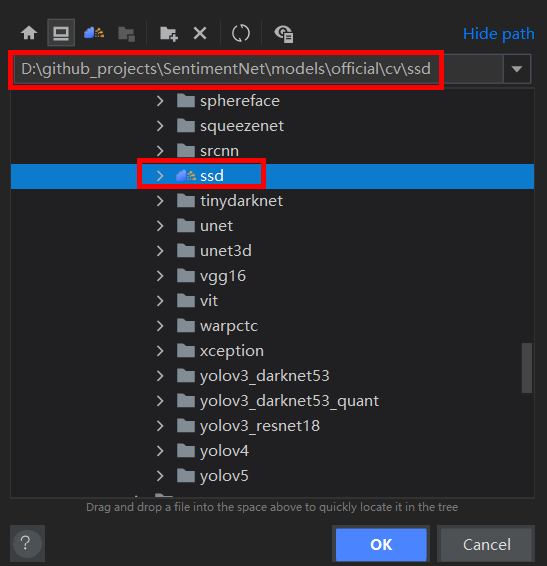
"anno\_path": "" # 其他数据集标注路径，如果使用coco或voc，此参数无效。

如果尝试使用coco原数据集进行训练，因为训练数据集比较大，可将batch\_size设置为16，避免在训练过程中出现“out of memory(内存溢出)”的报错。

## 运行工程

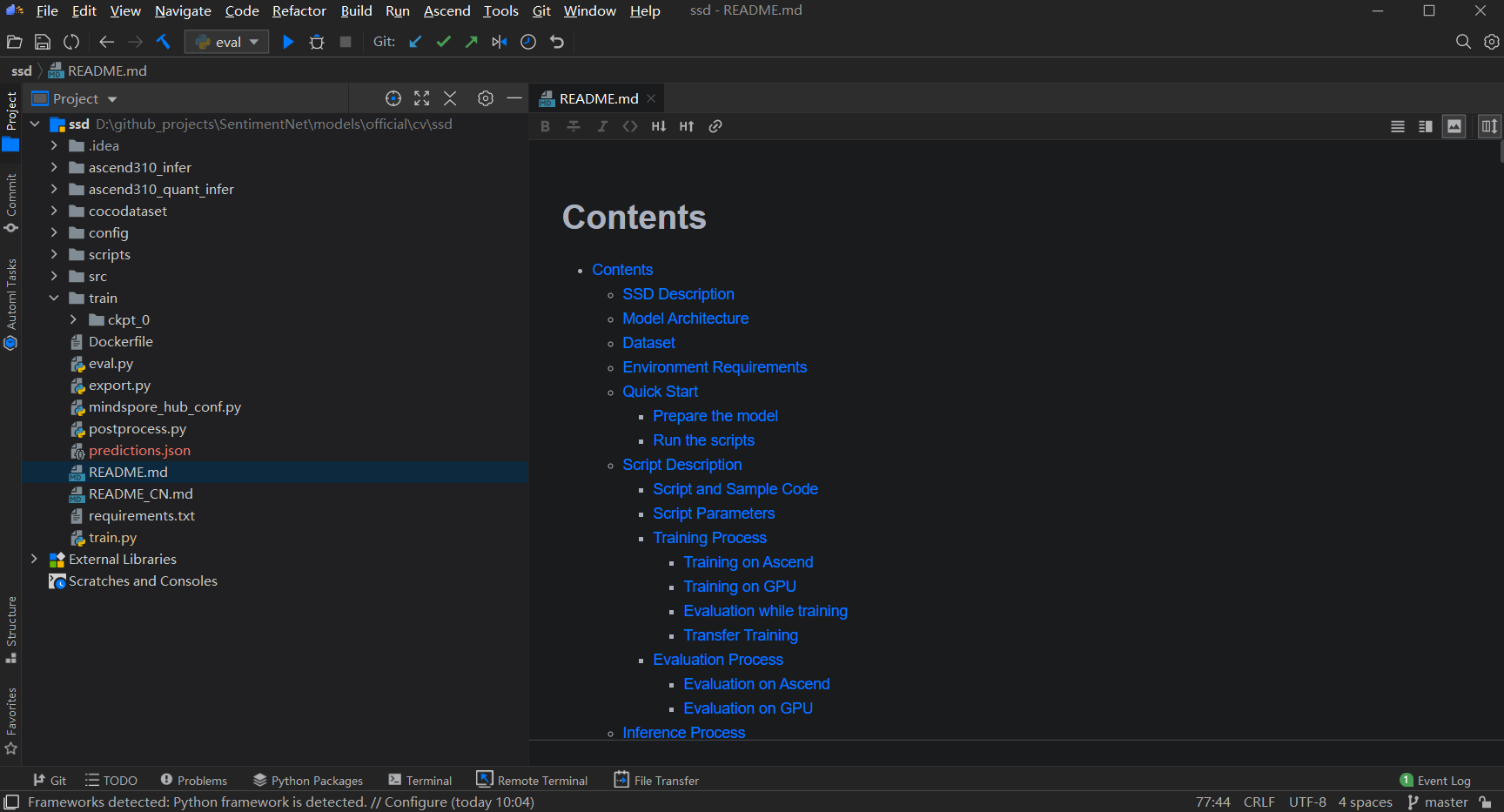
### 打开项目文件

用MindStudio打开项目文件夹（根据自己ssd项目文件的路径）：



打开MindStudio

显示加载情况如下：



文件情况

Python依赖环境的设置请参考本章环境准备部分附件--《MindStudio环境搭建指南》中章节2.4，设置python编译环境的依赖；

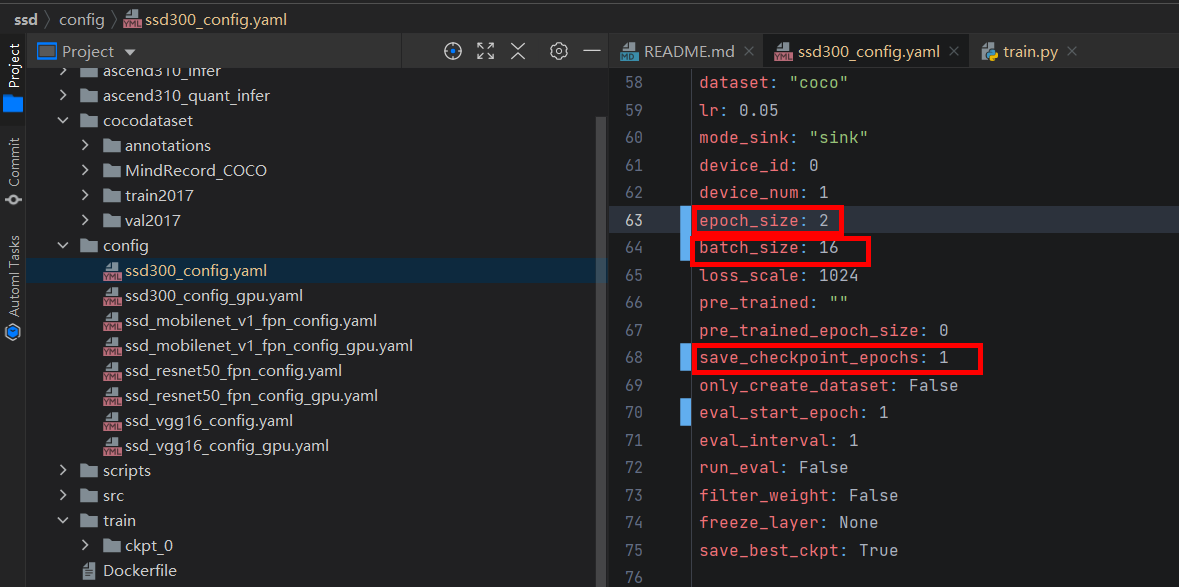
### 模型训练

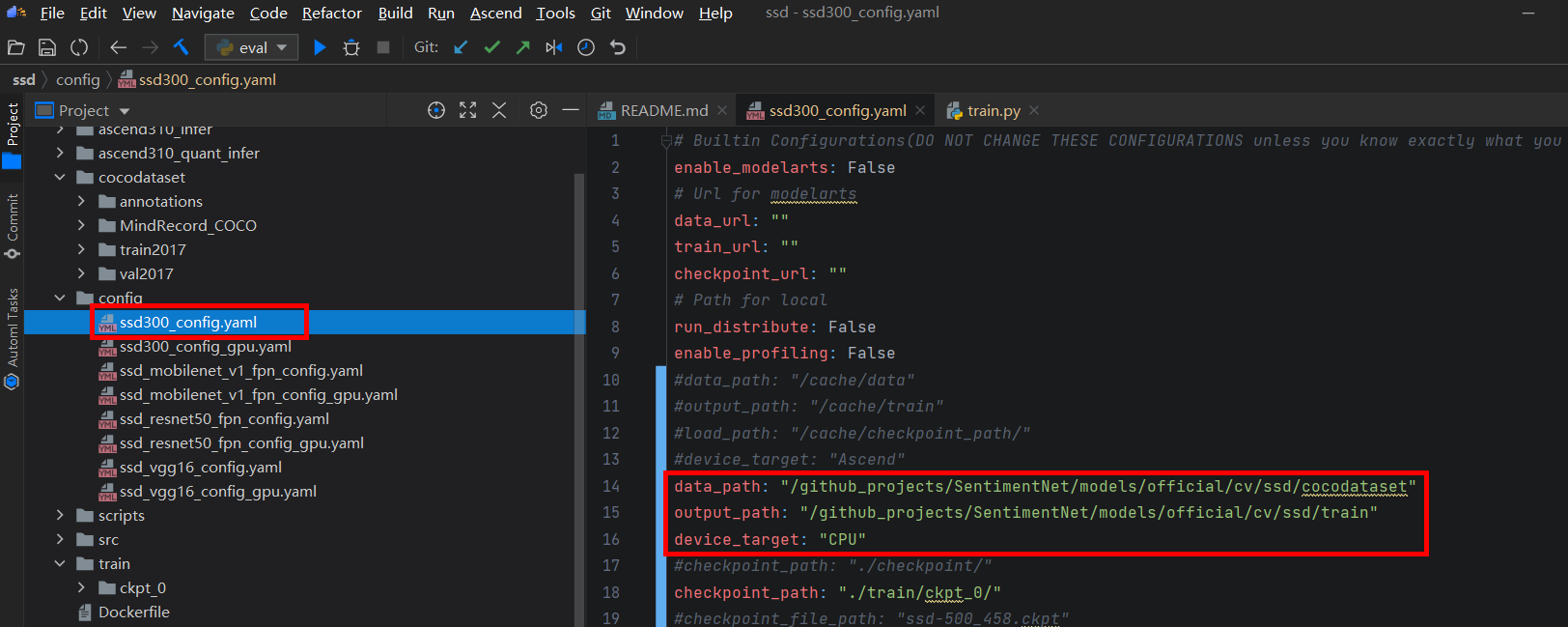
运行设备、数据路径等参数修改

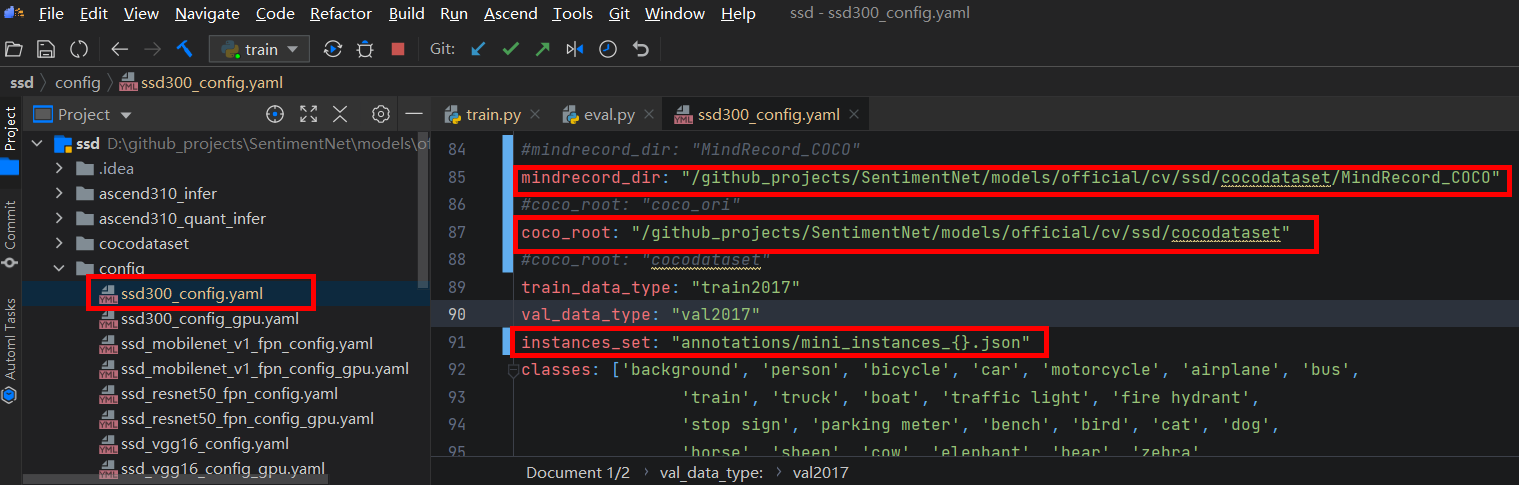
打开“ssd300\_config.yaml”文件，修改其中字段：

* “device\_target”为“CPU”；
* 由于本地训练时间耗时较长，可根据时间和内存大小等条件，调节“epoch\_size”等参数。此处“epoch\_size”（训练轮数）修改为“2”，“batch\_size”修改为16，并修改“save\_checkpoint\_epochs”参数（每x轮保存ckpt）为1；
* “data\_path”，“output\_path”，“mindrecord\_dir”，“coco\_root”等参数更改为自己的相对应文件的绝对路径；
* “instances\_set”参数更改：由"annotations/instances\_{}.json"改为"annotations/mini\_instances\_{}.json"

|  |  |  |
| --- | --- | --- |
| 配置 | 示意值 | 说明 |
| epoch\_size | 2 | 数据训练轮数 |
| batch\_size | 16 | 每次训练的样本数 |
| save\_checkpoint\_epochs | 1 | 每训练epochs轮数保存一次模型 |
| data\_path | "/github\_projects/SentimentNet/models/official/cv/ssd/cocodataset" | 数据所在的绝对路径（根据自己的数据集路径填写） |
| output\_path | "/github\_projects/SentimentNet/models/official/cv/ssd/train" | 训练后文件的保存路径 |
| device\_target | CPU | 实验所用的算力硬件 |
| mindrecord\_dir | "/github\_projects/SentimentNet/models/official/cv/ssd/cocodataset/MindRecord\_COCO" | MindRecord数据默认输出路径 |
| coco\_root | "/github\_projects/SentimentNet/models/official/cv/ssd/cocodataset" | coco数据集的根目录 |
| instances\_set | "annotations/mini\_instances\_{}.json" | 数据集标注信息json文件相对路径 |

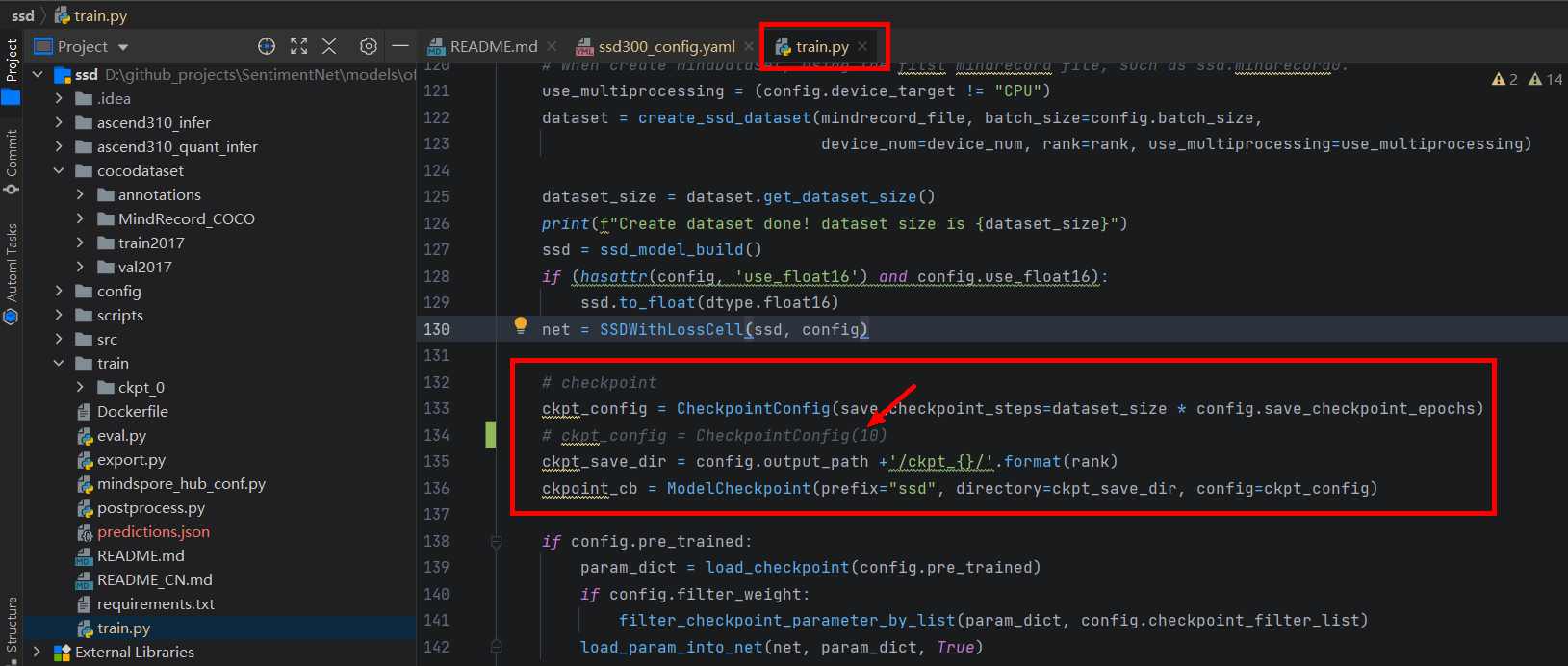






修改yaml配置文件

注：为进一步快速训练得到ckpt文件进行推理，可以修改“train.py”文件中模型保存相关代码，如下图所示：



修改train.py

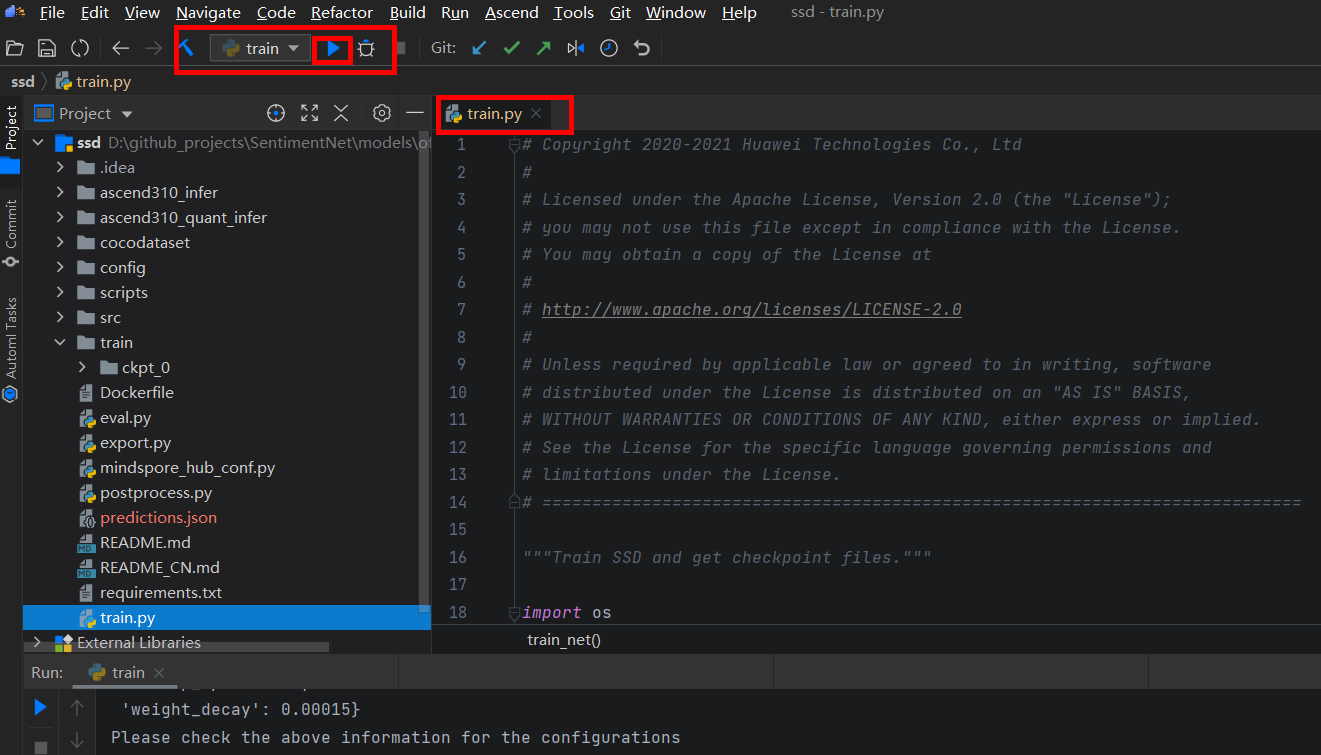
例如此处直接指定为每10steps保存一次ckpt文件：

ckpt\_config = CheckpointConfig(10)

模型训练

点击运行按钮开始训练，或是右键，选择“Run ‘train’”选项；

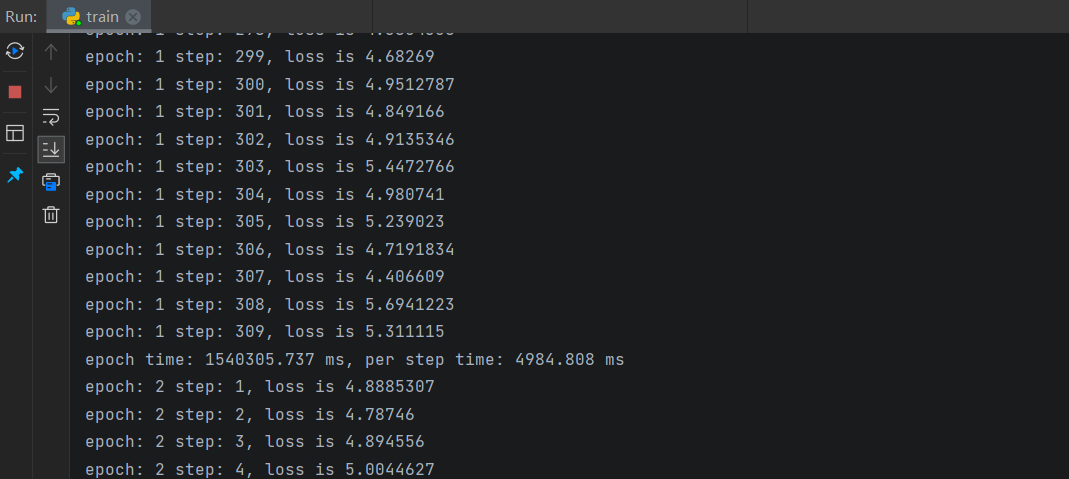
若无法执行训练，考虑关闭浏览器等内存占用大的程序。



运行按钮

查看过程和结果

在输出窗口查看训练过程和训练结果，等待结束。从训练过程可见，训练一个epoch的时间大概是26分钟。



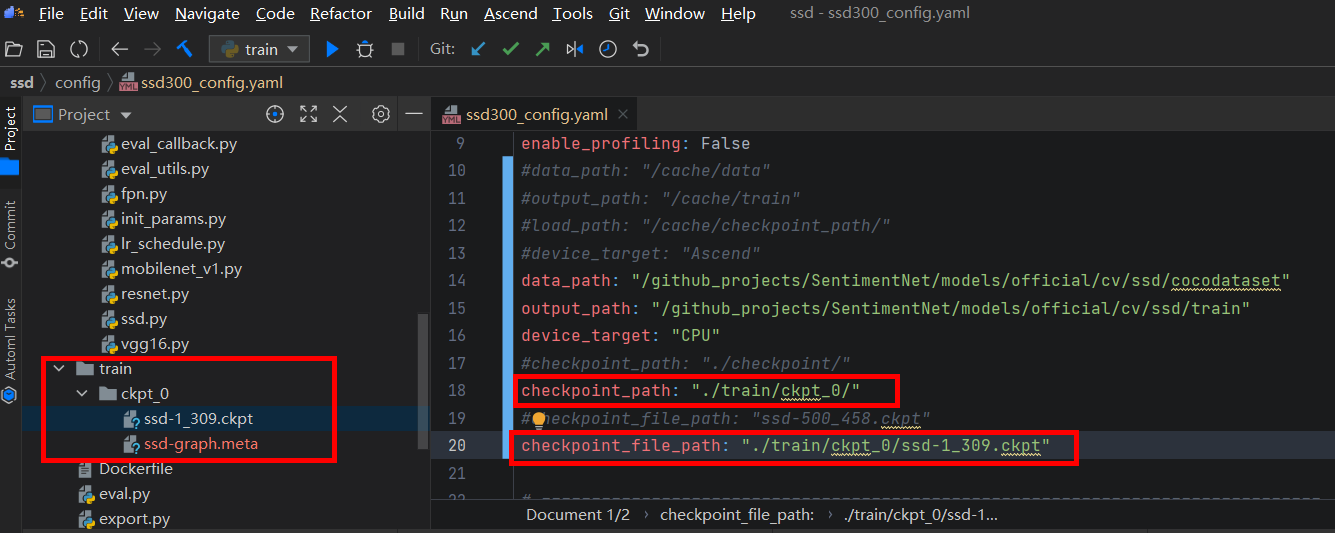
运行中界面

### 模型评估

Checkpoint等文件路径参数修改。

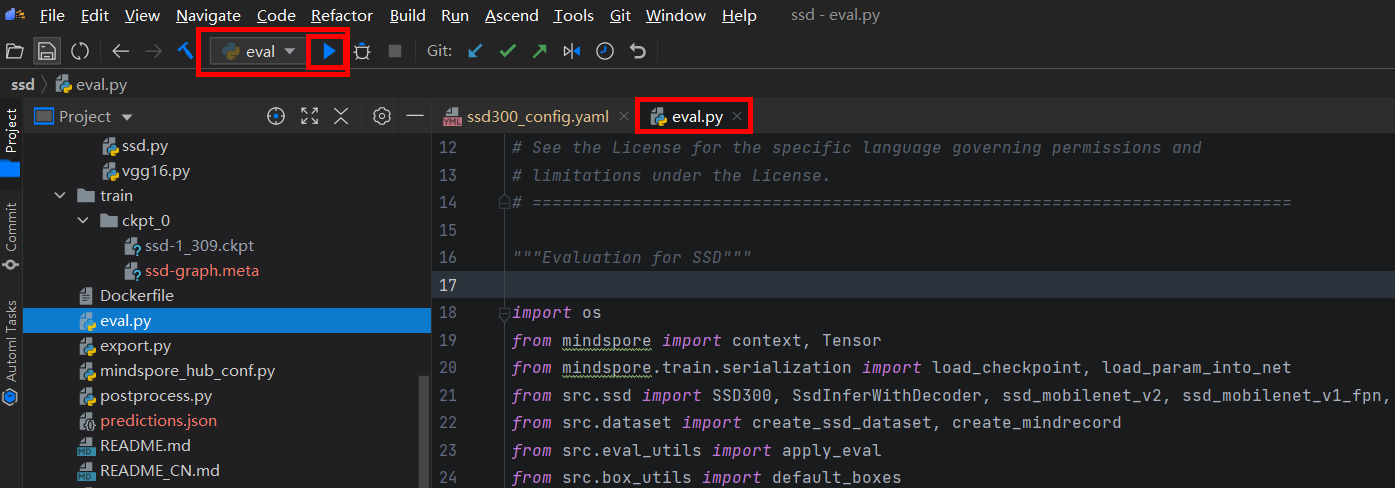
2.4.2模型训练后会在train文件夹下生成存有模型文件的checkpoint文件夹，如图2-10中的ckpt\_0，该文件夹下存有模型文件ssd-1\_309.ckpt。在模型评估环节，checkpoint\_path和checkpoint\_file\_path文件路径需要改为实际的相对路径。

|  |  |  |
| --- | --- | --- |
| 配置 | 示意值 | 说明 |
| checkpoint\_path | “./train/ckpt\_0/” | 模型文件所在的文件夹路径 |
| checkpoint\_file\_path | “./train/ckpt\_0/ssd-1\_309.ckpt” | 模型文件路径 |



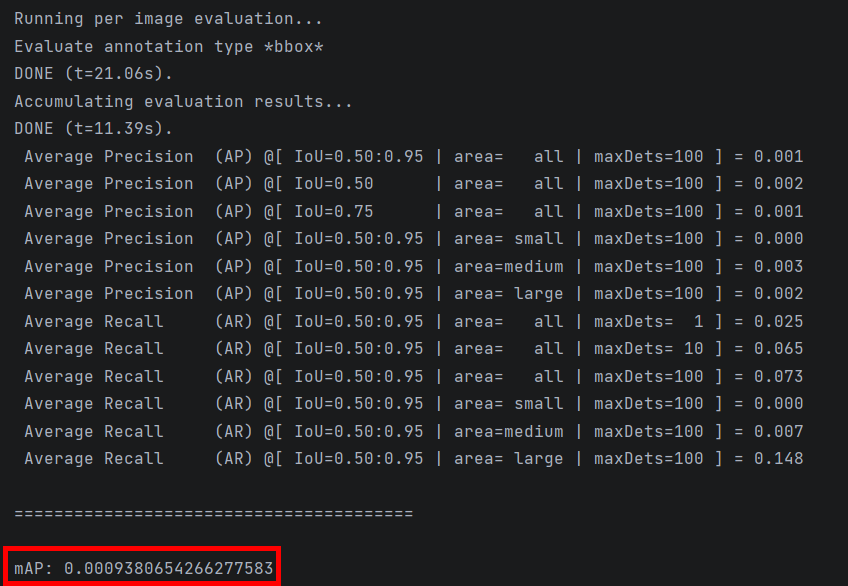
修改yaml配置文件

点击运行按钮开始训练，在输出窗口查看评估结果。



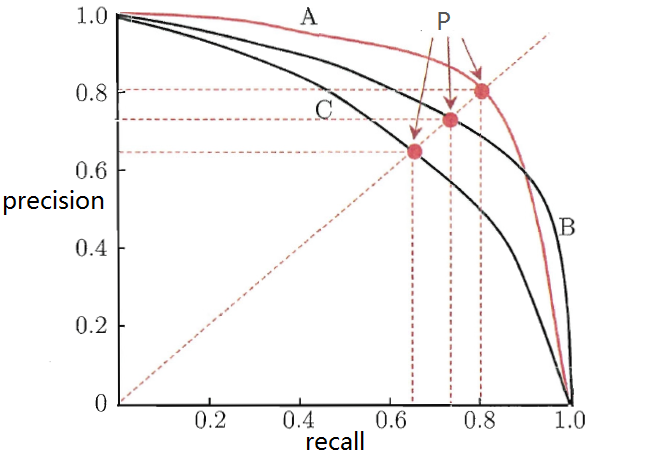
点击run

等待运行完成



模型评估

目标检测在识别类别部分的性能度量指标和分类一致，也包括有精度和召回率等。平均精度（AP）和平均精度均值（mAP）。通常在目标检测中作为评估标准使用。



Precision-recall曲线

AP就是Precision-recall 曲线下面的面积，通常来说一个越好的分类器，AP值越高。

mAP是多个类别AP的平均值。这个mean的意思是对每个类的AP再求平均，得到的就是mAP的值，mAP的大小一定在[0,1]区间，越大越好。该指标是目标检测算法中最重要的一个。mAP公式计算如下：



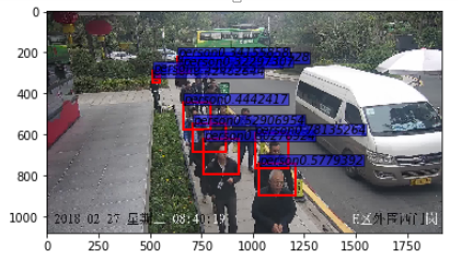
总的来说，AP概括了precision/recall曲线的形状，mAP需要计算整个数据集。

## 实验小结

本案例在PC机上安装python3.7.5及昇腾全栈全流程开发工具-MindStudio3.0.4，并使用COCO2017开源数据集对基于MindSpore框架的SSD网络进行训练和评估。通过模型训练，我们对SSD模型、常见参数（如epoch、batch-size等）的含义及设置有进一步的了解；通过模型评估，我们对目标检测常用评估指标(AP、mAP)有深入的了解。

# 实验2介绍

目标检测任务是找出图像或视频中人们感兴趣的物体，并同时检测它们的位置和大小。不同于图像分类任务，目标检测不仅要解决分类问题，还要解决定位问题，是属于Multi-Task的问题。如图1-1所示人脸检测就是目标检测任务应用之一。



人脸检测

作为计算机视觉的基本问题之一，目标检测构成了许多其他视觉任务的基础，例如实例分割、图像标注和目标跟踪等等；在基于区域的卷积神经网络提出后，深度卷积网络开始在目标检测领域普及，更快的基于区域的卷积神经网络将整个目标检测过程合成在一个统一的深度网络框架上．

## 实验目的

本实验基于MindSpore深度学习开源框架，使用COCO2017开源数据集进行模型训练与评估。

通过模型训练，我们对Faster-RCNN模型、常见参数（如epoch、batch-size等）的含义及设置会有进一步的了解。

## 实验清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| 基于MindSpore框架搭建Faster-RCNN网络实现目标检测任务 | 本实验基于MindSpore深度学习开源框架， 完成fasterrcnn目标检测框架的搭建，并使用COCO2017开源数据集的筛选子集进行模型训练与评估。 | 中级 | Python3.7.5、MindSpore1.5 | 云端或本地CPU/GPU机器 |

## 实验开发环境

实验平台：windows PC机

框架：MindSpore1.7

硬件：CPU,GPU

## 开发平台介绍

MindSpore是一种适用于端边云场景的新型开源深度学习训练/推理框架。 MindSpore提供了友好的设计和高效的执行，旨在提升数据科学家和算法工程师的开发体验，并为Ascend AI处理器提供原生支持，以及软硬件协同优化。

同时，MindSpore作为全球AI开源社区，致力于进一步开发和丰富AI软硬件应用生态。

更多关于MindSpore框架的知识信息，请参考如下链接：

<https://www.mindspore.cn/>

# 搭建Faster-RCNN网络实现目标检测

## 数据集准备

COCO数据集是一个大型的、丰富的物体检测，分割和字幕数据集。这个数据集以scene understanding为目标，主要从复杂的日常场景中截取，图像中的目标通过精确的segmentation进行位置的标定。图像包括91类目标，328,000影像和2,500,000个label。目前为止有语义分割的最大数据集，提供的类别有80 类，有超过33 万张图片，其中20 万张有标注，整个数据集中个体的数目超过150 万个。

数据集地址（暂不需下载）：[https://cocodataset.org/#download](https://cocodataset.org/" \l "download)

本实验使用的数据集文件如下图所示：

图形用户界面, 文本, 应用程序

描述已自动生成

实验所用数据

数据集详细如下：

* 训练数据集：2017 Train images[118K/18GB]，**18GB**，118000张图像；
* 评估数据集：2017 Val images[5K/1GB]，**1GB**，5000张图像；
* 标注信息文件：2017 Train/Val annotations [241MB]，**241MB**， json文件；

从上述信息可知，训练数据集和评估数据集都较大，尤其是训练数据集，18G，下载不太方便；**所以，本实验中我们也提供了COCO数据集的刷选版本 — cocodataset**。

cocodataset文件夹目录结构及所含文件详细信息如下：

.

└─cocodataset

├─annotations #标注文件夹

├─mini\_instance\_train2017.json #训练数据标注信息

└─mini\_instance\_val2017.json #验证数据标注信息

├─val2017 #验证数据集，1000张图片，154MB

└─train2017 #训练数据集，5000张图片，789MB

数据下载路径如下：

<https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/MindStudio-pc/cocodataset.zip、>

下载数据集后，将数据集上传至OBS仓库当中。例如，在my-coco数据集当中的dataset路径：

图形用户界面, 应用程序

描述已自动生成

## Faster-RCNN模型介绍

### 模型架构

图示

描述已自动生成

在Faster R-CNN之前，目标检测网络依靠区域候选算法来假设目标的位置，如SPPNet、Fast R-CNN等。研究结果表明，这些检测网络的运行时间缩短了，但区域方案的计算仍是瓶颈。

Faster R-CNN提出，基于区域检测器（如Fast R-CNN）的卷积特征映射也可以用于生成区域候选。在这些卷积特征的顶部构建区域候选网络（RPN）需要添加一些额外的卷积层（与检测网络共享整个图像的卷积特征，可以几乎无代价地进行区域候选），同时输出每个位置的区域边界和客观性得分。因此，RPN是一个全卷积网络，可以端到端训练，生成高质量的区域候选，然后送入Fast R-CNN检测。

论文： Ren S , He K , Girshick R , et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 39(6).

### 获取代码文件与NoteBook创建

代码在课程目录中提供，其结构目录为：

|  |
| --- |
| .  └─faster\_rcnn  ├─README.md // Faster R-CNN相关说明  ├─ascend310\_infer // 实现310推理源代码  ├─scripts  ├─run\_standalone\_train\_ascend.sh // Ascend单机shell脚本  ├─run\_standalone\_train\_gpu.sh // GPU单机shell脚本  ├─run\_distribute\_train\_ascend.sh // Ascend分布式shell脚本  ├─run\_distribute\_train\_gpu.sh // GPU分布式shell脚本  ├─run\_infer\_310.sh // Ascend推理shell脚本  └─run\_eval\_ascend.sh // Ascend评估shell脚本  └─run\_eval\_gpu.sh // GPU评估shell脚本  ├─src  ├─FasterRcnn  ├─\_\_init\_\_.py // init文件  ├─anchor\_generator.py // 锚点生成器  ├─bbox\_assign\_sample.py // 第一阶段采样器  ├─bbox\_assign\_sample\_stage2.py // 第二阶段采样器  ├─faster\_rcnn.py // Faster R-CNN网络  ├─fpn\_neck.py // 特征金字塔网络  ├─proposal\_generator.py // 候选生成器  ├─rcnn.py // R-CNN网络  ├─resnet.py // 骨干网络  ├─resnet50v1.py // Resnet50v1.0骨干网络  ├─inceptionresnetv2.py // inception resnet v2骨干网络  ├─roi\_align.py // ROI对齐网络  └─rpn.py // 区域候选网络  ├─dataset.py // 创建并处理数据集  ├─lr\_schedule.py // 学习率生成器  ├─network\_define.py // Faster R-CNN网络定义  ├─util.py // 例行操作  └─model\_utils  ├─config.py // 获取.yaml配置参数  ├─device\_adapter.py // 获取云上id  ├─local\_adapter.py // 获取本地id  └─moxing\_adapter.py // 云上数据准备  ├─default\_config.yaml // Resnet50相关配置  ├─default\_config\_101.yaml // Resnet101相关配置  ├─default\_config\_152.yaml // Resnet152相关配置  ├─default\_config\_InceptionResnetV2.yaml // inception resnet v2相关配置  ├─export.py // 导出 AIR,MINDIR模型的脚本  ├─eval.py // 评估脚本  ├─postprogress.py // 310推理后处理脚本  └─train.py // 训练脚本 |

进入华为Modelarts界面，选择开发环境->创建Notebook，然后点击创建：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

自动停止时间设为24小时，选择mindsopre1.2.0-cuda10.1-cudnn7-ubuntu18.04镜像，选择GPU类型中的【GPU: 1\*V100(32GB)|CPU: 8核 64GB】设备类型。存储配置当中选择云硬盘EVS来实现持久化存储功能。磁盘规格可自行调整，本指南使用100GB。最后点击创建。

然后在NoteBook中挂载我们之前创建的数据集：

|  |
| --- |
| import moxing as mox  mox.file.copy\_parallel('obs://my-datasets/cocodataset.zip', '/home/ma-user/work/dataset/cocodataset.zip') |

挂载之后需要用unzip命令解压。

## Faster-RCNN模型训练

### 依赖安装

环境使用为MindSpore 1.7.0，注意默认环境为1.2.0！

安装如下依赖：

pip install Cython

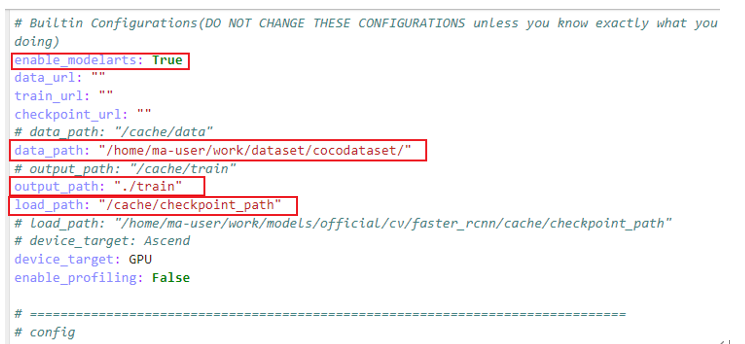
pip install pycocotools

pip install mmcv==0.2.14

pip install seaborn

### 训练

打开default.config.yaml文件，修改对应的数据路径



修改 enbale\_modelarts为True

修改data\_path为刚才上传数据集解压的自定义路径，本教程在/dataset/目录下。

修改output\_path路径，为模型输出路径。

修改load\_path为””（空字符）

修改数据集相关配置，如下图所示：

修改coco\_root为cocodataset，修改instance\_set为mini\_instances\_{}.json

文本, 信件

描述已自动生成

设置record文件的path：

文本

描述已自动生成

其他配置，读者可自行更改，教程演示使用的配置如下，因为演示所以epoch为1，实际可以设置更大一点。

文本

描述已自动生成

然后运行：python train.py 开始训练，第一次时间比较久，因为需要生成record文件。

文本, 信件

描述已自动生成

文本

描述已自动生成

一个epoch大约8~10分钟。

### 评估

修改image\_dir和anno\_path的位置：

图形用户界面, 应用程序

描述已自动生成

然后运行python eval.py：evaluation结果如下所示

表格

描述已自动生成

会显示每个类别的Precison和Recall：

散点图

描述已自动生成

相关评估结果在eval\_result中显示：

图形用户界面, 文本, 应用程序

描述已自动生成

# 开放性问题

* rcnn、Fast-RCNN、Faster-RCNN是一个系列的文章，阅读这三篇文章，你觉得他们各自之间有什么关系，又有什么改进？
* 调整config.py中的各项参数，例如bbox采样的参数，nms的阈值等等，考虑一下这些参数会对我们的模型造成什么样的影响，调整之后动手训练一下，然后思考在不同场景的数据集应该如何调整这些超参数。
* Faster-RCNN是二阶段目标检测算法，你还知道哪些二阶段算法？什么是二阶段算法？他和一阶段算法之间的区别是什么？利用上述流程，阅读华为MindSpore和ModelZoo中的代码以及模型，你能尝试自己复现一个一阶段的目标检测算法吗？(拓展问题，不作硬性要求)