**detectron2技术文档**

**github:** [**https://github.com/facebookresearch/detectron2**](https://github.com/facebookresearch/detectron2)

# 一、环境搭建

## 1.1基础镜像：

nvidia/cuda:10.1-cudnn7-devel

**备注：**

cat /usr/include/cudnn.h | grep CUDNN\_MAJOR -A 2 #7.6.3

cat /usr/local/cuda/version.txt #CUDA Version 10.1.243

## 1.2安装python3.6.8

apt-get update

apt-get install python-software-properties

apt-get install software-properties-common

add-apt-repository ppa:jonathonf/python-3.6 #python3

apt-get install python3.6 #安装python3.6

调整Python3的优先级，使得3.6优先级较高

update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.6 8

更改默认值，python默认为Python2，现在修改为Python3

update-alternatives --install /usr/bin/python python /usr/bin/python3 150

## 1.3安装pip3

apt install python3-pip –y

update-alternatives --install /usr/bin/pip pip /usr/bin/pip3 200

## 1.4安装opencv4.1.1

pip install opencv-python

import cv2

>>> cv2.\_\_version\_\_

'4.1.1'

## 1.5安装detectron2的依赖库

apt-get install –y libpng-dev libjpeg-dev build-essential pkg-config git curl wget automake libtool ca-certificates

rm -rf /var/lib/apt/lists/\*

## 1.6安装torch、torchvision、cython、cocoapi

pip install torch torchvision cython \

'git+https://github.com/facebookresearch/fvcore'

pip install 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'

## 1.7编译detectron2

python setup.py build develop

## 1.8测试

wget http://images.cocodataset.org/val2017/000000439715.jpg -O input.jpg

python3 demo/demo.py \

--config-file configs/COCO-InstanceSegmentation/mask\_rcnn\_R\_50\_FPN\_3x.yaml \

--input input.jpg \

--opts MODEL.WEIGHTS detectron2://COCO-InstanceSegmentation/mask\_rcnn\_R\_50\_FPN\_3x/137849600/model\_final\_f10217.pkl

--output input\_res.jpg

# 二、训练

## 2.1train.sh

#!/usr/bin/env bash

LOG="log/oil/res101\_1080\_4537.txt"

exec &> >(tee -a "$LOG")

echo Logging output to "$LOG"

#export CUDA\_VISIBLE\_DEVICES=0

python **tools/train\_net.py** \

--num-gpus 1 \

--config-file configs/Base-RetinaNet\_zj.yaml \

OUTPUT\_DIR output/oil/test/

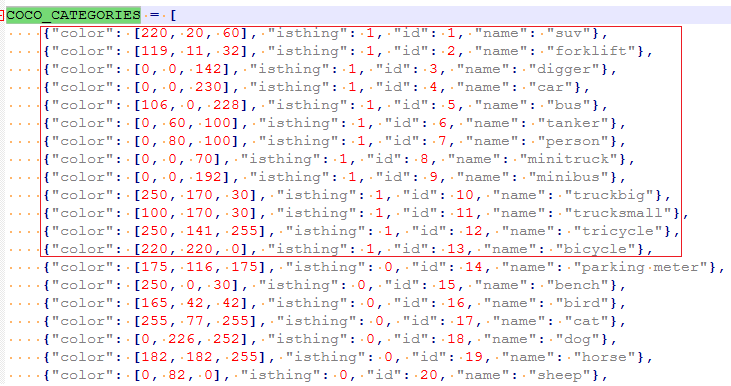
## 2.2 修改数据部分

**备注：**训练使用的json数据跟Detectron是一样的

### 2.2.1 修改类名

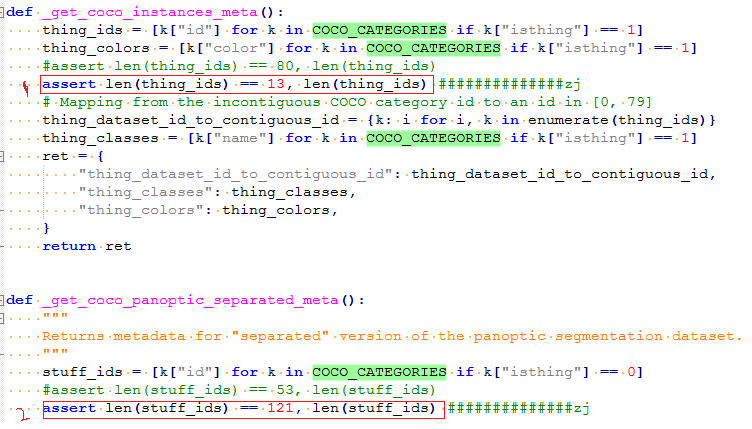
**代码位置**： detectron2/data/datasets/ builtin\_meta.py

#### 修改1：



**备注：**将要检测的类别的“isthing”设为1.

#### 修改2：



**备注：**

1. 1处，13是“isthing”=1的个数；
2. 2处，121是“isthing”=0的个数。

### 2.2.2 修改数据集路径

**代码位置**： detectron2/data/datasets/ builtin.py

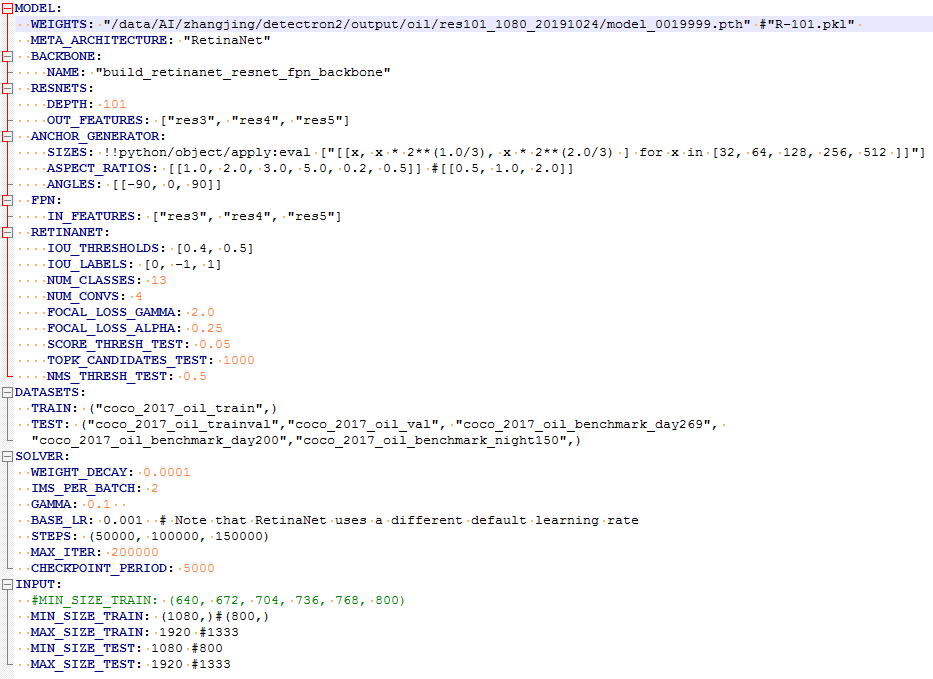


备注：

1. 1处，"coco\_2017\_oil\_train"是数据集的名称，配置文件中需要跟它保持一致；
2. 2处，"/data/AI/…/JPEGImages"是数据图片的存放路径；
3. 3处，"/data/AI/…/train.json"是根据xml制作的json数据。

## 2.3 配置文件修改部分

**代码位置**： /configs/COCO-Detection/ retinanet\_R\_101\_FPN\_3x.yaml



**备注：**

1. **WEIGHTS:** 模型参数，如果使用作者提供的预训练模型R-101.pkl, 训练迭代是从0开始；

如果是做**增量训练**，模型参数就需要指定到之前训练的模型具体位置，比如"/data/AI/…/model\_0019999.pth"

预训练模型的下载地址：

<https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md>

1. **ASPECT\_RATIOS**: 高宽比， 比如[[1.0, 2.0, 3.0, 5.0, 0.2, 0.5]]
2. **NUM\_CLASSES:** 类别个数，比如13
3. **TRAIN**: 训练集名称， ("coco\_2017\_oil\_train",)
4. **TEST**: 测试集名称，("coco\_2017\_oil\_trainval","coco\_2017\_oil\_val",)
5. **IMS\_PER\_BATCH**: 2
6. **CHECKPOINT\_PERIOD**: 保存模型间隔，比如5000
7. **MIN\_SIZE\_TRAIN**: (1080,)

**MAX\_SIZE\_TRAIN**: 1920

**MIN\_SIZE\_TEST**: 1080

**MAX\_SIZE\_TEST**: 1920

9. **BASE\_LR**: 0.001

**STEPS**: (50000, 100000, 150000)

**WARMUP\_FACTOR**:1.0 / 1000

**WARMUP\_ITERS**: 1000

**WARMUP\_METHOD**:"linear"

**备注：** 当Iter<1000, lr=（1.0 / 1000）\*Iter\* BASE\_LR；

当1000<Iter<50000, lr= BASE\_LR\*0.1;

当50000<Iter<100000, lr= BASE\_LR\*0.1\*0.1;

# 三、测试

## 3.1 test.sh

python tools/train\_net.py \

--eval-only \

--num-gpus 1 \

--config-file configs/Base-RetinaNet\_zj.yaml \

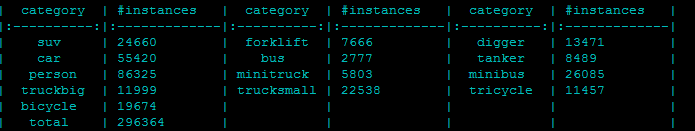
OUTPUT\_DIR output/oil/test/

## 3.2 实验结果

### 3.2.1 训练数据情况

All(99527)、train(98240)、trainval(1273)、val(1287)

SIZES: !!python/object/apply:eval ["[[x, x \* 2\*\*(1.0/3), x \* 2\*\*(2.0/3) ] for x in [32, 64, 128, 256, 512 ]]"]



### 3.2.2 测试指标

#### 实验1：

SIZES: !!python/object/apply:eval ["[[x, x \* 2\*\*(1.0/3), x \* 2\*\*(2.0/3) ] for x in [32, 64, 128, 256, 512 ]]"]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **迭代次数** | **Loss** | **trainval (1273)** | | | **val (1287)** | | | **Benchmark\_day (269)** | | | **Benchmark\_day (200)** | | | **Benchmark\_night(150)** | | |
| **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** |
| 5w | 0.314 | 0.49 | 0.781 | 0.644 | 0.49 | 0.767 | 0.650 | 0.34 | 0.67 | 0.550 | 0.45 | 0.645 | 0.712 | 0.37 | 0.594 | 0.594 |
| 6w | 0.263 | 0.561 | 0.853 | 0.686 | 0.565 | 0.848 | 0.688 | 0.439 | 0.778 | 0.591 | 0.495 | 0.709 | 0.722 | 0.441 | 0.680 | 0.622 |
| 9w | 0.194 | 0.581 | 0.877 | 0.693 | 0.583 | 0.869 | 0.691 | 0.445 | 0.787 | 0.592 | 0.570 | 0.809 | 0.732 | 0.455 | 0.712 | 0.627 |
| **备注**：IoU=0.50:0.95 recall | | | | | | | | | | | | | | | | |
| 10 | 0.161 | 0.586 | 0.876 | 0.981 | 0.582 | 0.867 | 0.971 | 0.444 | 0.789 | 0.970 | 0.587 | 0.829 | 0.992 | 0.468 | 0.710 | 0.974 |
| 15 | 0.197 | 0.596 | 0.886 | 0.982 | 0.593 | 0.876 | 0.977 | 0.452 | 0.794 | 0.974 | 0.591 | 0.836 | 0.995 | 0.469 | 0.716 | 0.961 |
| 19 | 0.172 | 0.597 | 0.887 | 0.983 | 0.593 | 0.876 | 0.976 | 0.452 | 0.797 | 0.983 | 0.598 | 0.840 | 0.995 | 0.470 | 0.714 | 0.961 |
| 20 | 0.22 | 0.596 | 0.887 | 0.983 | 0.593 | 0.876 | 0.976 | 0.452 | 0.796 | 0.980 | 0.594 | 0.840 | 0.995 | 0.470 | 0.713 | 0.961 |

#### 实验2：

SIZES: !!python/object/apply:eval ["[[x, x \* 2\*\*(1.0/4), x \* 2\*\*(2.0/4), x \* 2\*\*(3.0/4)] for x in [32, 64, 128, 256, 512 ]]"]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **迭代次数** | **Loss** | **trainval (1273)** | | | **val (1287)** | | | **Benchmark\_day (269)** | | | **Benchmark\_day (200)** | | | **Benchmark\_night(150)** | | |
| **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** |
| 5 | 0.236 | 0.477 | 0.760 | 0.944 | 0.471 | 0.747 | 0.942 | 0.354 | 0.652 | 0.937 | 0.427 | 0.627 | 0.949 | 0.363 | 0.597 | 0.936 |
| 6 | 0.192 | 0.564 | 0.862 | 0.976 | 0.563 | 0.845 | 0.969 | 0.428 | 0.765 | 0.966 | 0.523 | 0.748 | 0.992 | 0.466 | 0.704 | 0.975 |
| 10 | 0.182 | 0.589 | 0.885 | 0.976 | 0.590 | 0.872 | 0.974 | 0.447 | 0.786 | 0.958 | 0.554 | 0.800 | 0.992 | 0.484 | 0.733 | 0.974 |
| 11 | 0.254 | 0.593 | 0.890 | 0.977 | 0.592 | 0.873 | 0.972 | 0.456 | 0.791 | 0.962 | 0.564 | 0.791 | 0.993 | 0.485 | 0.736 | 0.990 |
| 18 | 0.194 | 0.595 | 0.892 | 0.978 | 0.595 | 0.876 | 0.975 | 0.459 | 0.798 | 0.959 | 0.565 | 0.794 | 0.993 | 0.493 | 0.742 | 0.955 |
| **在10w的基础上增量训练，(50000, 100000, 200000)** | | | | | | | | | | | | | | | | |
| 16 | 0.121 | 0.597 | 0.892 | 0.978 | 0.595 | 0.875 | 0.974 | 0.458 | 0.795 | 0.960 | 0.565 | 0.797 | 0.993 | 0.491 | 0.744 | 0.974 |

#### 实验3：

ANGLES: [[0]] #[[-90, 0, 90]]

SIZES: !!python/object/apply:eval ["[[x, x \* 2\*\*(1.0/4), x \* 2\*\*(2.0/4), x \* 2\*\*(3.0/4)] for x in [32, 64, 128, 256, 512 ]]"]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **迭代次数** | **Loss** | **trainval (1273)** | | | **val (1287)** | | | **Benchmark\_day (269)** | | | **Benchmark\_day (200)** | | | **Benchmark\_night(150)** | | |
| **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** |
| 8 | 0.252 | 0.577 | 0.870 | 0.971 | 0.574 | 0.858 | 0.970 | 0.440 | 0.798 | 0.969 | 0.547 | 0.772 | 0.990 | 0.440 | 0.692 | 0.978 |
| 20 |  | 0.598 | 0.891 | 0.976 | 0.597 | 0.876 | 0.972 | 0.462 | 0.820 | 0.970 | 0.580 | 0.811 | 0.989 | 0.460 | 0.708 | 0.988 |

#### 实验4：

ANGLES: [[-90, 0, 90]]

SIZES: !!python/object/apply:eval ["[[x, x \* 2\*\*(1.0/4), x \* 2\*\*(2.0/4), x \* 2\*\*(3.0/4)] for x in [32, 64, 128, 256, 512 ]]"]

训练集：benchmark269

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **迭代次数** | **Loss** | **trainval (1273)** | | | **val (1287)** | | | **Benchmark\_day (269)** | | | **Benchmark\_day (200)** | | | **Benchmark\_night(150)** | | |
| **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** | **AP** | **AP50** | **recall** |
| 1 | 0.022 |  |  |  |  |  |  | 0.919 | 0.999 | 1.000 |  |  |  |  |  |  |
| 6 | 0.001 |  |  |  |  |  |  | 0.997 | 1.000 | 1.000 |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **实验98：resnet101 1920\*1080 day200召回率：98.0 night150召回率：97.2** | | | | | | | | | | | | | | |
| **类别** | **AP** | **suv** | **forklift** | **digger** | **car** | **bus** | **tanker** | **person** | **mini**  **truck** | **mini**  **bus** | **truck**  **big** | **truck**  **small** | **tricycle** | **bicycle** |
| **验证** | 68.5 | 62.6 | 71.4 | 67.9 | 83.6 | 71 | 61.3 | 70.3 | 72.5 | 62.1 | 62.3 | 60.8 | 57.2 | 68.5 |
| **Benchmark269** | 45.5 | 42.7 | 46.3 | 48.3 | 67.2 | 60.2 | 22.3 | 47.8 | 58.5 | 53.8 | 54.5 | 42.6 | 30.8 | 45.5 |
| **Benchmark200** | 62.2 | 65.2 | 74.3 | 71.4 | 88.8 | 72.4 | 54.8 | 96.8 | 72.8 | 61.3 | 62.3 | 59.9 | 44.2 | 62.2 |
| **Benchmark150** | 67.6 | 52 | 94.2 | 57.9 | 90 | 19.8 | 43.6 | 65.3 | 61.4 | 40.4 | 62.5 | 16.7 | 55.9 | 67.6 |
|  | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

### 3.2.3 测试时间及GPU使用情况

**备注：** 输入图片大小1080\*1920，batchsize=2, 刚开始训练GPU占8.2G，训练一段时间GPU占9.98G，占10.6G.

|  |  |  |  |
| --- | --- | --- | --- |
| **输入参数** | **269张100次的平均耗时** | **GPU** | **耗时(ms)** |
| 1080\*1920、13cls、5anchor、p3-p7 |  | 1.7G |  |

# 四、错误

1. **Error:** RuntimeError: DataLoader worker (pid 1245) is killed by signal: Bus error. It is possible that

dataloader's workers are out of shared memory. Please try to raise your shared memory limit.

**解决方法及产生原因:** Pytorch在数据处理过程中使用的内存共享方法，而docker中默认

限制了shared memory，所以在运行多线程时会将超出限制的DataLoader kill掉。解决起来也很方便，在起Docker容器时，设置--ipc=host或--shm-size。

**参考文献**：http://www.luyixian.cn/news\_show\_185706.aspx

**2. 训练时，gpu内存一段增加**

**代码位置：detectron2/utils/** **events.py**

**C:\Users\ZHANGJ~2\AppData\Local\Temp\1571916204(1).png**

**原因：**显存释放问题

PyTorch 采用缓存分配器(caching memory allocator) 机制以加速内存分配(memory allocations)，使得在不需要设备同步(device synchronizations) 的情况下，实现快速的内存再分配(fast memory deallocation).

然而，采用 nvidia-smi命令，显存分配器所管理的无用内存仍然会显示被占用.

PyTorch 提供了 [memory\_allocated()](https://pytorch.org/docs/stable/cuda.html#torch.cuda.memory_allocated) 和 [max\_memory\_allocated()](https://pytorch.org/docs/stable/cuda.html#torch.cuda.max_memory_allocated) 用于监视 **tensors 占用的内存**； [memory\_cached()](https://pytorch.org/docs/stable/cuda.html#torch.cuda.memory_cached) 和 [max\_memory\_cached()](https://pytorch.org/docs/stable/cuda.html#torch.cuda.max_memory_cached) 用于监视**缓存**分配器所管理的内存.

PyTorch 提供了 [empty\_cache()](https://pytorch.org/docs/stable/cuda.html#torch.cuda.empty_cache) l来释放所有**未使用的缓存**的内存，以便其它 GPU 应用能够使用. 但是，并不能释放 tensors 所占用的 GPU 显存。

**作者的解释：**It will slowly increase at the beginning and stop increasing, and this is expected.

The reason is because images have different sizes and allocated memory are typically not returned to the driver. （它将在开始时缓慢增加，并停止增加，这是预料之中的。

原因是映像有不同的大小，并且分配的内存通常不会返回给驱动程序。）

# 五、代码部分

1. **tools/train\_net.py : launch**(

main,

args.num\_gpus,

num\_machines=args.num\_machines,

machine\_rank=args.machine\_rank,

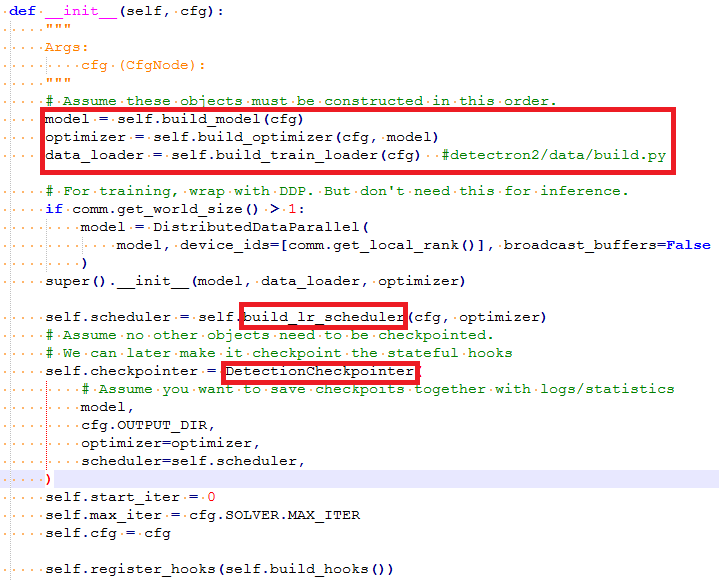
dist\_url=args.dist\_url,

args=(args,),

) # **detectron2/engine/ launch.py**

其中**main() :** trainer = Trainer(cfg) #**detectron2/engine/default.py**, trainer.train()

**1.1 trainer = Trainer(cfg)构造函数代码如下：**

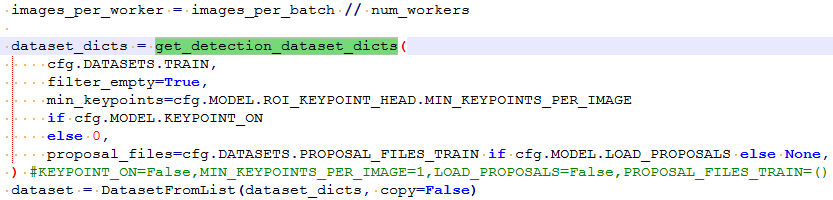


其中**build\_train\_loader()**

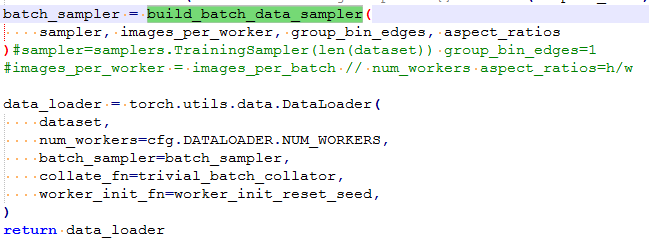
**->build\_detection\_train\_loader(cfg) {** #detectron2/data/build.py

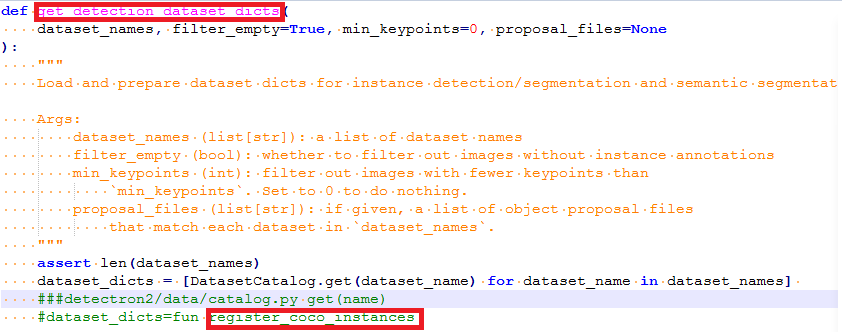
(images\_per\_batch = cfg.SOLVER.IMS\_PER\_BATCH ) }

**-> get\_detection\_dataset\_dicts()**

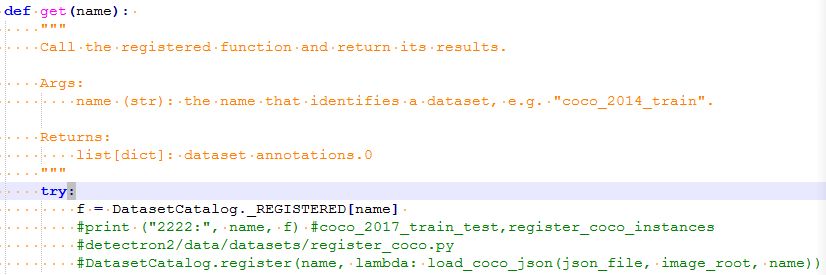


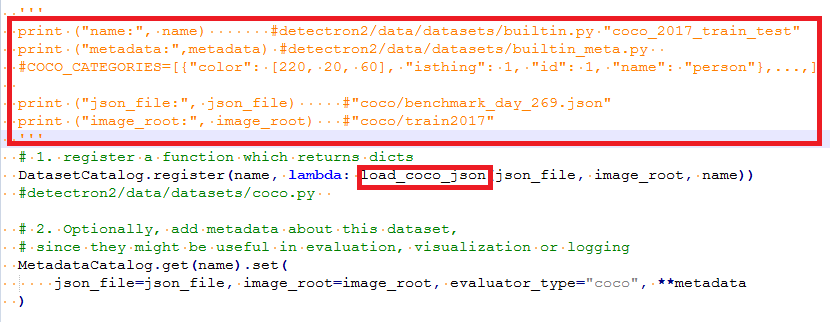
其中cfg.DATASETS.TRAIN就是配置文件中TRAIN: ("coco\_2017\_train\_test",)





其中





**#detectron2/data/datasets/coco.py load\_coco\_json()**

**meta = MetadataCatalog.get(dataset\_name) #**

修改**detectron2/data/datasets/builtin\_meta.py**

**assert len(stuff\_ids) == 121, len(stuff\_ids)**

**assert len(thing\_ids) == 13, len(thing\_ids)**

**COCO\_CATEGORIES 后，**meta中的thing\_classes、thing\_colors等就跟加载的数

据是一致的。

**#meta**: Metadata(evaluator\_type='coco',

image\_root='datasets/coco/train2017',

json\_file='datasets/coco/benchmark\_day\_269.json',

name='coco\_2017\_train\_test',

thing\_classes=['person', …, 'toothbrush'],

thing\_colors=[[220, 20, 60],… , [191, 162, 208]],

thing\_dataset\_id\_to\_contiguous\_id={1: 0, …, 90: 79})

**cat\_ids = sorted(coco\_api.getCatIds())**

#cat\_ids: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

**cats = coco\_api.loadCats(cat\_ids)**

#cats: [{'supercategory': 'none', 'id': 1, 'name': 'suv'}, …,

{'supercategory': 'none', 'id': 13, 'name': 'bicycle'}]

**Error:** RuntimeError: DataLoader worker (pid 1245) is killed by signal: Bus error. It is possible that dataloader's workers are out of shared memory. Please try to raise your shared memory limit.

**解决方法及产生原因:** http://www.luyixian.cn/news\_show\_185706.aspx

Pytorch在数据处理过程中使用的内存共享方法，而docker中默认限制了shared memory，所以在运行多线程时会将超出限制的DataLoader kill掉。  
解决起来也很方便，在起Docker容器时，设置--ipc=host或--shm-size

**detectron2/data/datasets/builtin.py "coco\_2017\_train\_test"**

**detectron2/data/datasets/builtin\_meta.py 改类别名称**

**detectron2/engine/train\_loop losses**

**detectron2/utils/** **events.py**

[10/21 23:27:22 d2.utils.events]: eta: 20:57:09 iter: 299 total\_loss: 1.618

loss\_cls: 1.164 loss\_box\_reg: 0.465 time: 0.8396 data\_time: 0.0024 lr:

0.000300 max\_mem: 6163

**detectron2/modeling/meta\_arch/retinanet.py**

**\_C.SOLVER.CHECKPOINT\_PERIOD = 500 #保存模型间隔**

**WEIGHTS: "output/oil/test/model\_0059999.pth" 测试模型，修改配置文件**

**test.sh**

python tools/train\_net.py \

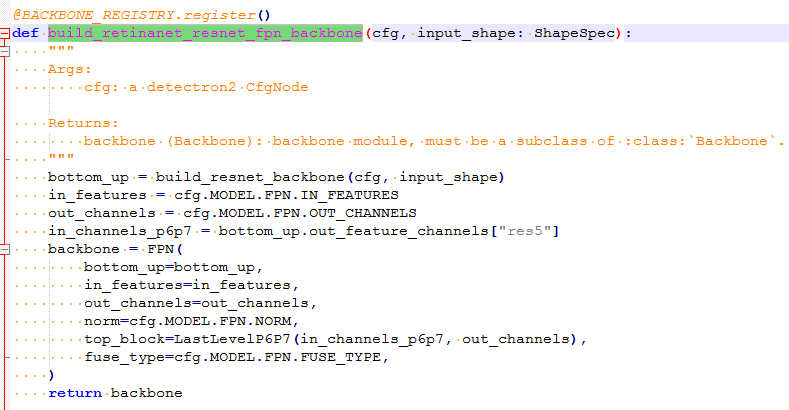
--eval-only \

--num-gpus 1 \

--config-file configs/Base-RetinaNet\_zj.yaml \

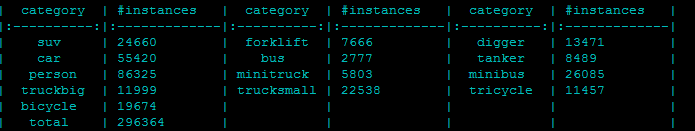
OUTPUT\_DIR output/oil/test/

**detectron2/modeling/backbone/ fpn.py**



\_\_setattr\_\_, \_\_getattr\_\_, \_\_delattr\_\_

<https://www.cnblogs.com/marsggbo/p/11678371.html>



batchsize=2, 8.2G（train）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 迭代次数 | trainval (1273) | | | val (1287) | | | Benchmark\_day (269) | | | Benchmark\_day (200) | | | Benchmark\_night(150) | | |
| AP | AP50 | recall | AP | AP50 | recall | AP | AP50 | recall | AP | AP50 | recall | AP | AP50 | recall |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

All(99527)、train(98240)、trainval(1273)、val(1287)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **实验98：resnet101 1920\*1080 day200召回率：98.0 night150召回率：97.2** | | | | | | | | | | | | | | |
| **类别** | **AP** | **suv** | **forklift** | **digger** | **car** | **bus** | **tanker** | **person** | **mini**  **truck** | **mini**  **bus** | **truck**  **big** | **truck**  **small** | **tricycle** | **bicycle** |
| **验证** | 68.5 | 62.6 | 71.4 | 67.9 | 83.6 | 71 | 61.3 | 70.3 | 72.5 | 62.1 | 62.3 | 60.8 | 57.2 | 68.5 |
| **Benchmark269** | 45.5 | 42.7 | 46.3 | 48.3 | 67.2 | 60.2 | 22.3 | 47.8 | 58.5 | 53.8 | 54.5 | 42.6 | 30.8 | 45.5 |
| **Benchmark200** | 62.2 | 65.2 | 74.3 | 71.4 | 88.8 | 72.4 | 54.8 | 96.8 | 72.8 | 61.3 | 62.3 | 59.9 | 44.2 | 62.2 |
| **Benchmark150** | 67.6 | 52 | 94.2 | 57.9 | 90 | 19.8 | 43.6 | 65.3 | 61.4 | 40.4 | 62.5 | 16.7 | 55.9 | 67.6 |
|  | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |