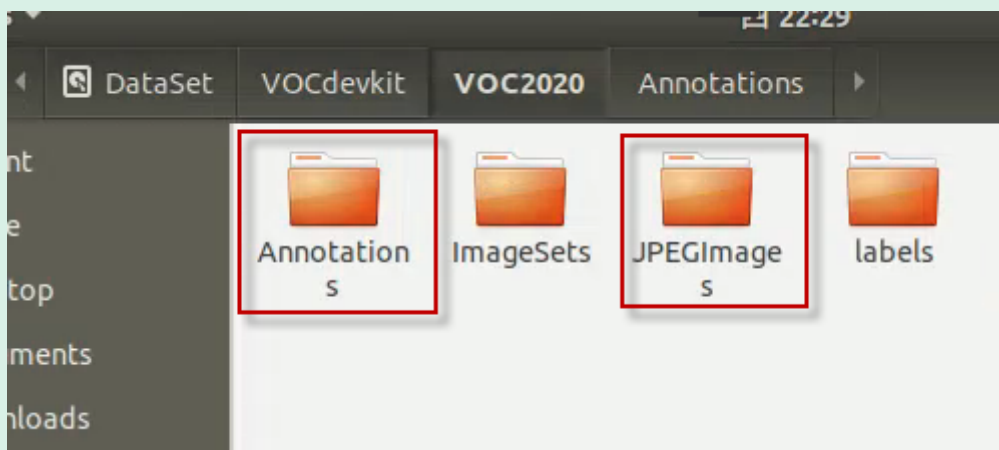


# Yolov4 Darknet版本说明

## 1.数据集及测试文件所在位置

Other Locations/DataSet/VOCdevkit

1.1. 咱们的数据集里面主要的是这两个部分Annotations（存放数据集图片的各种数据，像素，标注框，名字什么的xml文件），JPEGImages（图片文件）。需要在他们外面建立VOC2020文件夹，再建立一个VOCdevkit文件夹。



1.2. 和VOCdevkit文件夹同级目录下执行genfiles.py（该文件在附件中有），是用来做数据处理的，打开终端，执行。

```
python genfiles.py
```

执行完之后会生成两个文件2020\_test.txt和2020\_train.txt，分别是测试集和训练集的路径文件，后面需要用到（如果是在cpu上训练的话，不需此步，我已将这两个文件生成好了）。

注：如果需要修改标签类别需要修改genfile.py文件，只需要把需要修改的在划线部分显示即可。

```
import xml.etree.ElementTree as ET
import pickle
import os
from os import listdir, getcwd
from os.path import join
import random

classes=["groove","conveyer","coal_cutter","person","roller","front_board","big_coal"]

def clear_hidden_files(path):
    dir_list = os.listdir(path)
    for i in dir_list:
        abspath = os.path.join(os.path.abspath(path), i)
        if os.path.isfile(abspath):
            if i.startswith("."):
                os.remove(abspath)
            else:
                clear_hidden_files(abspath)
```

## 2. Yolov4训练过程

2.1. 这里需要安装opencv应用程序（gpu上面是安装了3.4.6版本），这个底层代码是c写的，不能像python环境下直接使用cv2这个库。

## 2.2.修改darknet-master（这个是darknet的源代码主目录）里面的makefile,将以下三项设置为1

```
GPU=1
CUDNN=1
OPENCV=1
```

然后执行

```
make clean
make
```

## 2.2. 测试GPU版本的darknet是否能正常运行

测试图片

```
./darknet detector test ./cfg/coco.data ./cfg/yolov4.cfg ./yolov4.weights data/dog.jpg
#yolov4.weights是预下载好的权重文件，老师按您给我的那个文件夹，应该是在那个yolov4的文件夹下
```

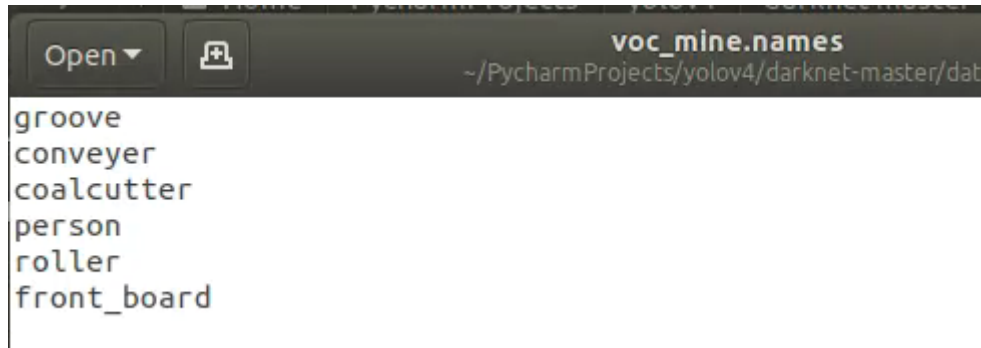
测试视频

```
./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights driving.mp4
```

## 2.3.修改配置文件

### 2.3.1 新建data/voc\_mine.names

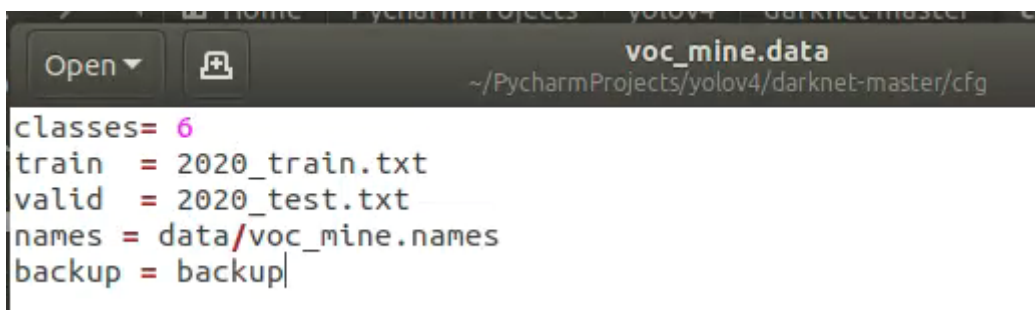
把咱们数据集的类别都写进去，我这边直接打包发给您，在这个文件夹目录下有个附件，里面修改过的配置文件我都放进去了。



```
groove
conveyer
coalcutter
person
roller
front_board
```

### 2.3.2 新建cfg/voc\_mine.data文件

将1.3.那一步生成的2020\_test.txt和2020\_train.txt的路径填入下面的train和valib的等号后；names是上一步的那个文件；backup是每一段时间生成的权重文件的存储路径，不需要修改。（附件有）



```
classes= 6
train  = 2020_train.txt
valid  = 2020_test.txt
names = data/voc_mine.names
backup = backup|
```

### 2.3.3新建cfg/yolov4\_mine.cfg文件

我修改的一些参数

**batch**一批训练样本的样本数量，每batch个样本更新一次参数

**subdivisions=8** batch/subdivisions作为一次性送入训练器的样本数量 如果内存不够大，将batch分割为subdivisions个子batch（subdivisions相当于分组个数，相除结果作为一次送入训练器的样本数量）上面这两个参数如果电脑内存小，则把batch改小一点，batch越大，训练效果越好 subdivisions越大，可以减轻显卡压力（分组数目越多，每组样本数量则会更少，显卡压力也会相应减少）

**width height**输入图片的尺寸

**burn\_in**多少轮存储一次权重

**max\_batches**循环多少轮

**steps**从此后开始降低学习率

```
batch=16
subdivisions=16
width=608
height=608
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 24000
policy=steps
steps=19200,21600
scales=.1,.1
```

最后有三个输出都需要相同操作，修改对应的filters和classes

**classes**类别数，我们这里是6

**filters**计算公式： $3 * (5 + \text{classes})$ ，这里classes是6，filters为33

```
[convolutional]
size=1
stride=1
pad=1
filters=33
activation=linear

[yolo]
mask = 6,7,8
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243,
459, 401
classes=6
num=9
```

(该文件附件有)

## 2.4. 训练数据集

```
./darknet detector train cfg/voc_mine.data cfg/yolov4_mine.cfg yolov4.conv.137 -map
#yolov4.conv.137是预下载好的权重文件，老师按您给我的那个文件夹，应该是在那个yolov4的文件夹下
#-map是为了显示训练过程中的map变化的。
./darknet detector train cfg/voc_mine.data cfg/yolov4_mine.cfg yolov4.conv.137 -map -
gups 0,1,2,3,4,5
#同时使用多块gpu训练，不设置使用0号gpu
./darknet detector train cfg/my_data.data cfg/my_yolov3.cfg darknet53.conv.74 -gups
0,1,2,3 myData/weights/my_yolov3.backup -gpus 0,1,2,3
#从停止处重新训练
```

## 3.测试训练出的网络模型

### 3.1. 创建训练cfg/yolov4\_mine\_test.cfg

其实这个文件和2.3.3那一步生成的yolov4\_mine.cfg是基本一样的，只是将batch和subdivision都设置为一（附件有）

### 3.2.测试

测试图片

```
./darknet detector test cfg/voc_mine.data cfg/yolov4_mine_test.cfg
backup/yolov4_mine_final.weights testfiles/test1.jpg
#testfiles/test1.jpg这个文件对应了1.1.中的测试文件，将相应的文件路径替换即可
```

测试视频

```
./darknet detector demo cfg/voc_mine.data cfg/yolov4_mine_test.cfg
backup/yolov4_mine_final.weights testfiles/test1.mp4
#testfiles/test1.mp4这个文件对应了1.1.中的测试文件，将相应的文件路径替换即可
```