**Python**

1. ->

例子def deleteHead(self) -> int:

->常常出现在python函数定义的函数名后面，为函数添加元数据,描述函数的返回类型，从而方便开发人员使用。

好处：使用预期的类型来注释参数，然后在函数返回值验证时检验参数的类型或者将其强制转换成预期的类型

如果产生语法错误（只有python3才能用->）的bug，将->与后面的删除就好

1. list的pop()方法

pop() 函数用于移除列表中的一个元素（默认最后一个元素），并返回所删除对象

list.pop([index=-1])

1. SyntaxError:invalid syntax 语法错误

错因：①版本问题 python2和python3②路径问题③粗心问题④安装第三方模块时

需要检查一些是否是在cmd窗口下安装，同时，要到python的安装目录里面，找到pip所在的目录里面进行安装

1. dict() 函数用于创建一个字典

5．from \_\_future\_\_ import division导入精确除法

6.argparse用法（就是把终端敲的参数传进到脚本代码中）

a.创建解释器

parser=argparse.ArgumentParser(description=”your script description”) 创建一个ArgumentParser对象,它包含将命令行解析成Python数据类型所需的全部信息

b.添加参数

parser.add\_argument() 向该对象中添加要关注的命令行参数和和选项

位置参数：用法是不用带 - ，例如parser.add\_argument(“e”)

可选参数: - 指定的短参数 -h -- 指定的长参数，如 --help说白了就是大小名

当位置参数和可选参数都存在时，敲的时候不写–v直接写值，则默认给位置参数赋值

例如parser.add\_argument(“-v”,”--verbosity”,help=”添加输出verbosity”)

-v必须指定参数值，不然会报错，如果想让它像-h那样不需要指定参数值也不报错，可以利用action=”store\_true”

parser.parse\_args() 进行解析

可选值choices=[ ] 用于限定某个值的取值范围

7.字符串格式化方法

f-strings 是指以 f 或 F 开头的字符串，其中以 {} 包含的表达式会进行值替换。

8.代码一行写不下用\ 换行

9.conda list(即安装的一些包)的路径

/data/lihongxi/anaconda3/envs/fsaf/lib/python3.7/site-packages

10. site-packages在约定中，是用来存放第三方库的，也就是你pip install安装的module.

11.打印 python的版本号

12.卸载anaconda

rm -rf ~/anaconda3

13.下载miniconda

wget<https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh>

14.setup()函数

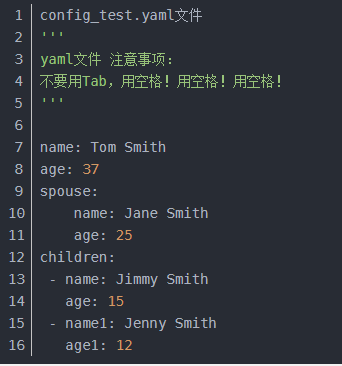
15. platform模块获取操作系统的信息

16.subprocess模块

subprocess 模块允许我们启动一个新进程，并连接到它们的输入/输出/错误管道，从而获取返回值

17.config.yaml文件的使用

  config.yaml文件读入后是一个字典，可用来配置程序中的相关参数；

  
18. \*args 表示参数列表()里面剩余的值

\*\*kargs表示剩余的参数列表

19.try except

异常报错提示可能会影响到输出结果的显示，此时就可以使用try:...except:...进行异常处理②，将错误直接输出打印，而不是以报错的形式显示

20.读取图片，显示图片

a.matplotlib

显示图片

import matplotlib.pyplot as plt # plt 用于显示图片

import matplotlib.image as mpimg # mpimg 用于读取图片

import numpy as np

lena = mpimg.imread('lena.png') # 读取和代码处于同一目录下的 lena.png

# 此时 lena 就已经是一个 np.array 了，可以对它进行任意处理

lena.shape #(512, 512, 3)

plt.imshow(lena) # 显示图片

plt.axis('off') # 不显示坐标轴

plt.show()

b．用PIL

显示图片

from PIL import Image

im = Image.open('lena.png')

im.show()

21.conda新建虚拟环境中包的路径（以于红服务器为例）

/home/lihongxi/anaconda3/envs/mmd2.16.0/lib/python3.6/site-packages

**网络**

1. 利用校园webvpn连接实验室服务器

vpn连接后显示哪个ip地址？

1. 查看ip地址

命令ipconfig/all

1. 公用网络和专用网络

公用网络没有加密，专用网络有加密

4.下载大的文件，使用断点续传

Wget-c {url}

5.403为权限问题

6.连接局域网上的打印机

此电脑->网络->找到直接连接打印机的电脑->打开

如果出现输入该电脑的用户名密码，则在那个电脑上不设置密码就好

7.连不上校园网

Ip has been log in please log out

在自服务里把已经上线的终端注销

8.xshell连不上服务器

用笔记本可以连上服务器

工位电脑不能

解决方法：关掉蓝灯，把**网线拔**了，插上无线网卡，连上校园网 ping一下服务器地址

然后把网线插上，连上校园网，ping一下服务器地址，解决

9.连上校园网上的ip地址

工位电脑10.105.219.43

笔记本 172.19.192.88

笔记本的DHCP服务器 172.19.0.1

（动态主机配置协议是一个局域网的[网络协议](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/328636)。指的是由[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8/100571)控制一段IP地址范围，客户机登录服务器时就可以自动获得服务器分配的IP地址和[子网掩码](https://baike.baidu.com/item/%E5%AD%90%E7%BD%91%E6%8E%A9%E7%A0%81/100207)）

笔记本的DNS服务器 202.118.66.6

**实验室服务器以及环境列表**

1. 泰坦IP210.30.97.95 端口号 22

系统及版本号'Linux-4.15.0-153-generic-x86\_64-with-debian-buster-sid'

用户 lihongxi密码 123456789

1. 大数据中心

http://ise.dlut.edu.cn/

**username:software\_lihongxi\_yuhong**

**password:123456a?**

第一个使用周期为：2020年9月6日-2020年11月5日（截止日期遇节假日顺延）

**每2个月为一个使用周期，需要提交一次使用报告。**

硕士可用GPU核数16，GPU卡数 1  （实际给你分配了CPU核数为32核GPU1卡）

3.连接打印机的电脑DESKTOP-LEVLKJH

3.于红老师课题组服务器

IP 210.30.97.224

用户lihongxi密码 lihongxi

Root 密码dlut

环境列表

环境名mmcv124项目名mmdetection-2.7.0

**mmdetection**

**mmdet已经废掉，需要重新按下面的步骤配新的环境**

**mmcv124,mmcv1.2.4已废**

1. 简介：是基于pytorch的目标检测的工具箱,linux，注意mmdetection版本要与mmcv版本相对应
2. **安装方法**(这里的环境是pytorch1.6.0+cuda10.1+python3.7)
3. 创建一个conda环境并激活

conda create -n mmcv1.2.4 python=3.7 -y

conda activate mmcv1.2.4

b.安装pytorch和torchvision

pip install torch==1.6.0+cu101 torchvision==0.7.0+cu101 -f <https://download.pytorch.org/whl/torch_stable.html>

c.安装mmcv(打开github链接，点击与版本对应的，花括号{}里填上版本号，这里填的是1.2.4)

pip install mmcv-full==1.2.4 -f <https://download.openmmlab.com/mmcv/dist/cu101/torch1.6.0/index.html>

d.下载mmdetection项目文件（这里用的是FSAF而不是2.7.0，为了和mmcv-full匹配，当然也可以直接克隆github，但直接克隆一般是最新版本，很容易出现问题）

进入该目录cd mmdetection项目名

e.安装依赖库，然后安装mmdetection(通过github安装pycocotools)

pip install -r requirements/build.txt

pip install -v -e . # or "python setup.py develop"

按照以上方法可以成功安装mmdetection

# 安装mmd2.16.0

python=3.6

pytorch=1.9

cudatoolkit=11.1

3.配数据集

在项目目录下，新建一个data文件夹（级别与configs平行）

下载各种数据集

4.训练demo

a.单GPU训练

python tools/train.py ${CONFIG\_FILE}

CUDA\_VISIBLE\_DEVICES=3 python tools/train.py configs/faster\_rcnn/faster\_rcnn\_r50\_fpn\_1x\_coco.py

b.多GPU训练

./tools/dist\_train.sh ${CONFIG\_FILE} ${GPU\_NUM} [optional arguments]

chmod +x tools/dist\_train.sh

CUDA\_VISIBLE\_DEVICES=1,2./tools/dist\_train.shconfigs/faster\_rcnn/faster\_rcnn\_r50\_fpn\_1x\_coco.py2

5.测试demo

# single-gpu testing

python tools/test.py ${CONFIG\_FILE} ${CHECKPOINT\_FILE} [--out ${RESULT\_FILE}] [--eval ${EVAL\_METRICS}] [--show]

# multi-gpu testing

./tools/dist\_test.sh ${CONFIG\_FILE} ${CHECKPOINT\_FILE} ${GPU\_NUM} [--out ${RESULT\_FILE}] [--eval ${EVAL\_METRICS}]

chmod +x tools/dist\_test.sh

CUDA\_VISIBLE\_DEVICES=1,2./tools/dist\_test.sh configs/mask\_rcnn/mask\_rcnn\_r50\_fpn\_1x.py checkpoints/mask\_rcnn\_r50\_fpn\_1x\_coco\_20200205-d4b0c5d6.pth2 --out results.pkl --eval bbox segm

5.报错总结

ModuleNotFoundError: No module named 'mmcv.\_ext'

报错原因：mmcv版本不匹配，另一个说法是安装的是mmcv而不是mmcv-full，需要重新安装

Mmcv安装方法：

法1：打开github：<https://github.com/open-mmlab/mmcv>

点击想要安装的版本

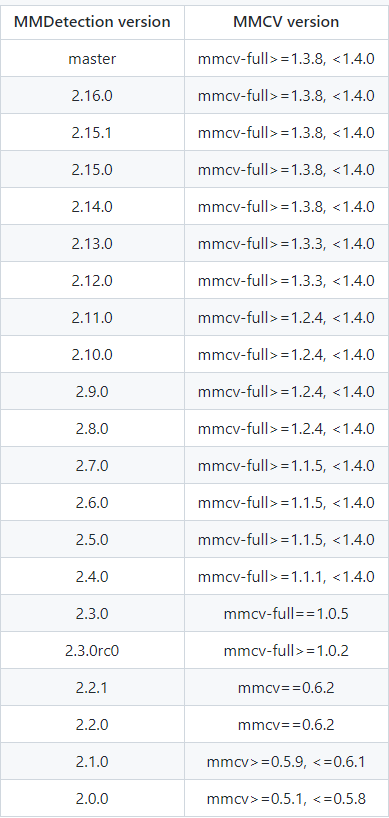
法2：到官网把whl下载下来，离线安装(建议此方法)

官网<https://download.openmmlab.com/mmcv/dist/index.html>

我所安装的whl的链接[https://download.openmmlab.com/mmcv/dist/cu101/torch1.6.0/](https://download.openmmlab.com/mmcv/dist/cu101/torch1.6.0/mmcv_full-1.2.4-cp37-cp37m-manylinux1_x86_64.whl)

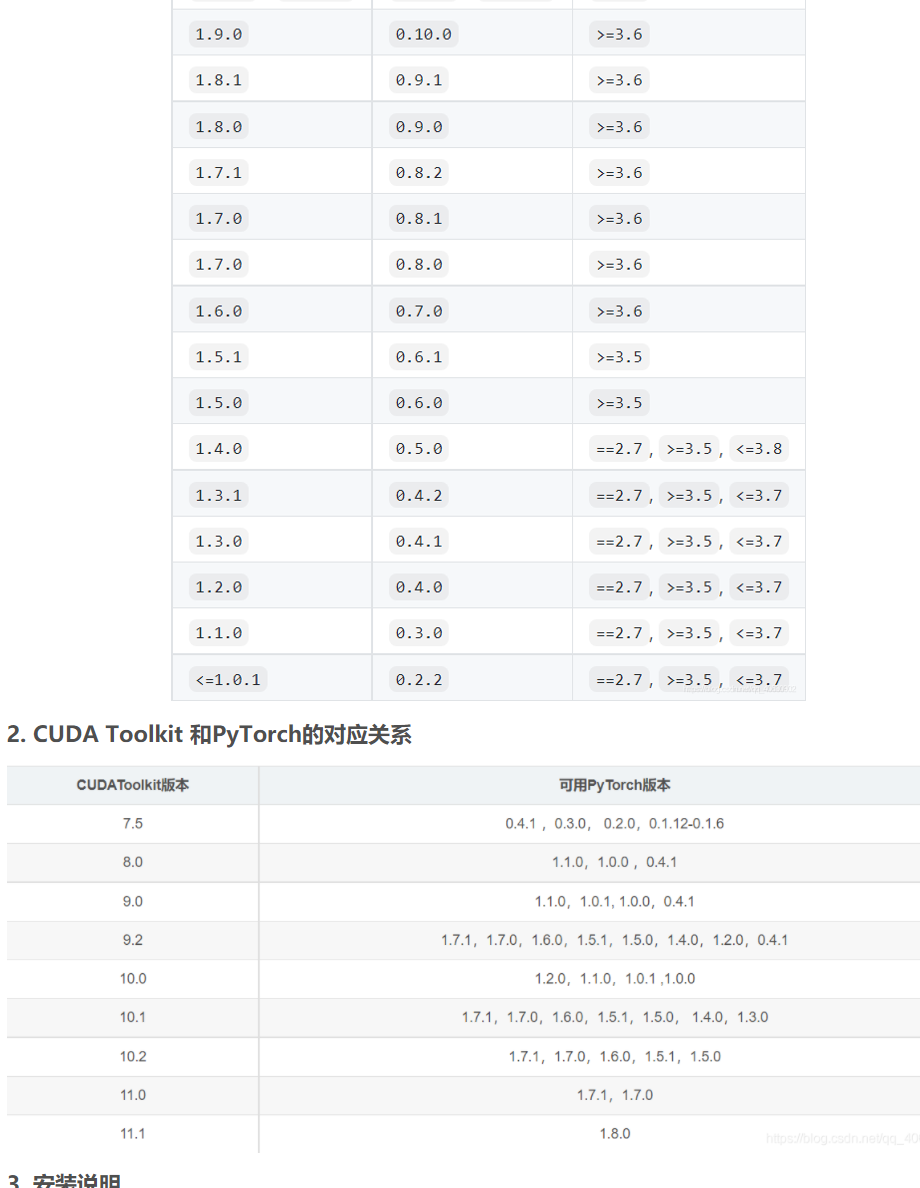
然后pip install \*.whl即可

Mmcv与mmdetection的版本对应关系（官网有）



**Pytorch与python对应关系**

**CUDA与cudatoolkit对应关系**

****

**github**

1.github克隆项目

git clone url

2.克隆mmdetection库时报错

fatal: unable to access 'https://github.com/ros-planning/navigation.git/': Failed to connect to 127.0.0.1 port 40491: Connection refused

解决方法：git clone https://github.com/”改为git clone git://github.com/

3.安装cocoAPI出错ERROR: Command errored out with exit status 128

原因：网络问题，资源在国外

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

解决方法：使用国内的资源

pip install "git+https://gitee.com/wsyin/cocoapi.git#subdirectory=PythonAPI"

**Linux**

1. 创建目录mkdir

2. #!用来声明脚本由什么shell解释，否则使用默认shell

脚本的第一行，指出这个文件中的代码用什么可执行程序去运行

#!/usr/bin/python

告诉操作系统执行这个脚本时，调用/usr/bin下的python解释器

#!/usr/bin/env python这种用法是为了防止操作系统用户没有将python装在默认的/usr/bin路径里。当系统看到这一行的时候，首先会到env设置里查找python的安装路径，再调用对应路径下的解释器程序完成操作。这种写法会去环境设置寻找python目录,推荐这种写法

3.bash是一种shell，即命令语言解释器

.sh文件是脚本文件，一般都是bash脚本

.sh文件运行方法：直接./加上文件名.sh

4.Linux的$

a.显示脚本参数（本质属于变量替换）

$0就是该bash文件名

1. ${varname:-word}

如果varname存在且非null，则返回其值；否则，返回word。用途：如果变量未定义，则返回默认值

5.dirname命令可以返回文件所在的目录

$(dirname "$0")返回当前文件所在目录

6.linux的换行符\(很多时候都是这个)

7.linux复制命令 cp

cp dir1/a.doc dir2 表示将dir1下的a.doc文件复制到dir2目录下

cp -r dir1 dir2 表示将dir1及其dir1下所包含的文件复制到dir2下

cp -r dir1/. dir2 表示将dir1下的文件复制到dir2,不包括dir1目录

说明：cp参数 -i：询问，如果目标文件已经存在，则会询问是否覆盖；

8.文件移动 mv(以后补充)

文件移动不同于文件拷贝，文件移动相当于我们word中的术语剪切和粘贴

命令：mv AAA BBB 表示将AAA改名成BBB

9.停止跑代码

ctrl + z 停止跑代码

10.杀进程

kill SIGNAL PID

（kill -9 PID）

11.”<”表示的是输入重定向的意思，就是把<后面跟的文件取代键盘作为新的输入设备。  
”| ”则表示一个管道的意思，可以理解为东西从管道的一边流向另外一边。

12.退出命令行如一直显示>，让你输入东西却无法退出

13.linux解压zip文件

在.zip文件所在目录下unzip \*.zip

解决方法 ctrl+c

14.shell脚本：一些命令的集合

a.简介

以.sh为后缀

#! /bin/bash意思为它使用bash语法，如果不设置该行，那么你的shell脚本就不能被执行

b.shell脚本的执行方法

sh filename

或者chmod +x test.sh

./test.sh

sh命令加-x选项可以调试

sh -x test.sh

15.date命令

a.打印日期

date "+%Y-%m/%d %H:%M:%S"打印年月日时间

-d选项打印n天前或n天后的日期

date -d "+1 month" "+%Y-%m/%d"

b.星期几

date+%w

16.压缩文件

gzip：是公认的压缩这速度最快，压缩大文件的时候与其他的压缩方式相比更加明显，历史最久，应用最广泛的压缩方式



打包并压缩文件夹压缩后的压缩包指定名称为FileName.tar.gz  
tar zcvf FileName.tar.gz DirName

17.建立软链接

ln -s为某一个文件在另外一个位置建立一个同步的链接

当我们需要在不同的目录，用到相同的文件时，我们不需要在每一个需要的目录下都放一个必须相同的文件，我们只要在某个固定的目录，放上该文件，然后在其它的目录下用ln命令链接（link）它就可以，不必重复的占用磁盘空间

ln [参数][源文件或目录][目标文件或目录]

硬链接的意思是一个档案可以有多个名称，而软链接的方式则是产生一个特殊的档案，该档案的内容是指向另一个档案的位置。硬链接是存在同一个文件系统中，而软链接却可以跨越不同的文件系统

删除软链接

删除软链接时只需删除软链接的名，**千万别加/**，它会把软链接映射的原文件夹下面的文件删除掉

如果两个文件夹都有东西就不要创建软链接（当然这可能创建不了，一般是一个空文件夹访问一个有东西的文件夹）

rm -rf  软件链接名（别加/）递归无提示强制删除

这个命令会把没有软链接的普通文件夹删掉

fsaf环境以及conda基础环境文件夹都被破坏

18. 'ld'把一定量的目标文件跟档案文件连接起来,并重定位它们的数据,连接符号引用.一般,在编译一个程序  
时,最后一步就是运行'ld'.

19. verbose：日志显示

20.ll命令

“ls -l”或者“ls -al”

21.vi命令

22.shell历史命令 history

23.vim命令

i 切换到输入模式，以输入字符。

ESC，退出输入模式，切换到命令模式（最初始状态就是这个模式）

 :wq 储存后离开 vi

:q!强制退出

24.~是用户目录

**计算机基础**

1. path和dir的区别

path表示路径，既可以指向文件，也可以指向目录，如目录\tmp文件名\tmp\img.png

dir一般表示目录

1. 随机种子

一种以随机数作为对象的以真随机数（种子）为初始条件的随机数。一般计算机的[随机数](https://baike.baidu.com/item/%E9%9A%8F%E6%9C%BA%E6%95%B0/2454368)都是[伪随机数](https://baike.baidu.com/item/%E4%BC%AA%E9%9A%8F%E6%9C%BA%E6%95%B0/104358)，以一个真随机数（种子）作为初始条件，然后用一定的算法不停[迭代](https://baike.baidu.com/item/%E8%BF%AD%E4%BB%A3/8415523)产生随机数。

3.绝对路径和相对路径

绝对路径 cd的时候最前面加/

如cd /usr

相对路径，如访问当前目录下的文件夹config，则 cd config 或者 cd ./config

不能cd /config 这是绝对路径会报错

1. 将软件放D盘，代码放F盘

设置安装路径时要新建个文件夹，不然D盘会很乱

1. 软件卸载失败（或者安装不上）

以xftp为例，一般是注册表出现了问题

先cmd后输入regedit打开注册表编辑器

将 RegistrySizeLimit（REG\_DWORD类型的） 的值改为 FFFFFFFF （10进制就是 4294967295，原值是8ca7290）如果不存在就创建一个  
重新启动计算机2次

**分屏操作**

1.ctrl+B所有快捷键命令的前缀，只有先按这个键，再其他键，命令才有效

2.shift + 5竖着分屏

3.shift + '横着分屏

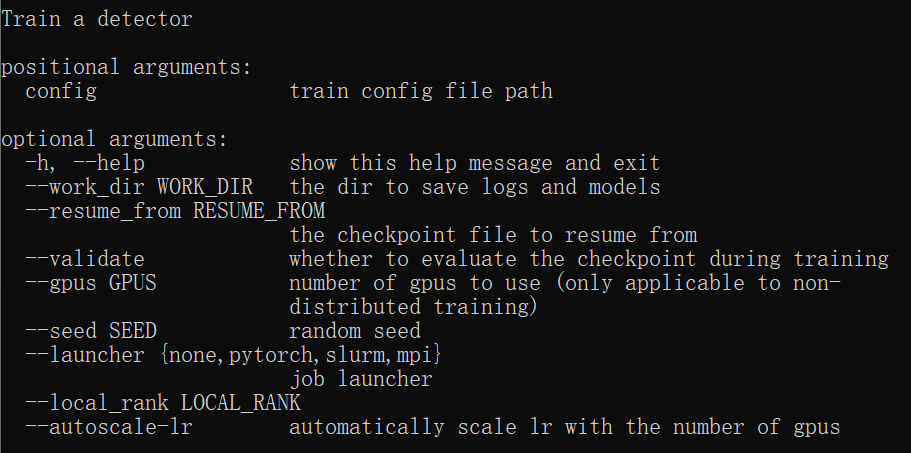
4.d 离开session

5.tmux ls 查看session列表

6.tmux select-pane -L 将光标移动到其他窗格这里为左

**FSAF代码（环境名fsaf-new） fsaf环境已经废掉**

1. train.py的argparse的-h列表



config参数传入的是模型的路径

1. 要知道.sh文件脚本在做什么事

我们也可以手动一步一步做出来

我的理解是

在coco数据集上跑这个模型

3.demo

多GPU

./tools/dist\_train\_retinanet\_r50\_400\_050x.sh

单GPU

CUDA\_VISIBLE\_DEVICES=0 python tools/train.py configs/fsaf/fsaf\_r50\_fpn\_1x\_coco.py

4.mmcv配置要求

mmcv>=1.1.5, <=1.3.

5.报错处理

安装setup.py文件报错

成功路径/usr/lib/x86\_64-linux-gnu/libpthread.so

报错路径

/data/lihongxi/anaconda3/envs/fsaf/compiler\_compat/

替换

6.错误操作（把lptread当成了lcudart）

ln -s /usr/lib/x86\_64-linux-gnu/libpthread.so /data/lihongxi/anaconda3/envs/fsaf/compiler\_compat/

**COCO数据集**

官网下载

<https://cocodataset.org/#download>

COCO2017数据集下载链接（每个文件夹的压缩包）

http://images.cocodataset.org/zips/train2017.zip

http://images.cocodataset.org/annotations/annotations\_trainval2017.zip

目标检测的annotation文件一般下载

http://images.cocodataset.org/annotations/annotations\_trainval2017.zip

http://images.cocodataset.org/zips/val2017.zip

ttp://images.cocodataset.org/annotations/stuff\_annotations\_trainval2017.zip

http://images.cocodataset.org/zips/test2017.zip

http://images.cocodataset.org/annotations/image\_info\_test2017.zip

**GPU**

1.cudDNN是用于深度神经网络的GPU加速库

2.cuda的存放路径

安装路径没有单独形成cuda文件夹，都是统一存放在envs下的虚拟环境lib、include文件夹下。

**计算机视觉顶会截稿日期**

A会.

ICLR2022 截稿日期2021.10.5 <https://iclr.cc/>

CVPR2022 截稿日期2021.11.16<http://cvpr2022.thecvf.com/>

B会

ICASSP2022 截稿日期2021.10.1 https://2022.ieeeicassp.org/

AAMAS2022截稿日期2021.10.2 https://aamas2021.soton.ac.uk/

ECCV2022 截稿日期TBD

C会

ALT2022 截稿日期2021.10.5 http://algorithmiclearningtheory.org/alt2022/

AISTATS2022 截稿日期2021.10.8 https://aistats.org/aistats2022/index.html

其他会议

BMVC

WACV

**error总结**

1. 解决需要Xmanager软件来处理X11转发需求（不安装Xmanager）

通过设置xshell解决，参考链接

https://blog.csdn.net/OnlyLove\_longshao/article/details/108147111

2.pycharm学生激活不了，因为此前用其他方法破解过，卸了重装就好

3.在mmdet环境(mmcv-full=1.2.4)中安装FSAF的requirement.txt，里面有mmcv=1.3.12

这就导致了我的mmdetection-2.7.0项目没法跑了，卸了mmcv=1.3.12也不行，如果想跑这个项目只能重新配环境了

4.subprocess.CalledProcessError: Command '['/data/lihongxi/anaconda3/envs/mmdet/bin/python', '-u', './tools/train.py', '--local\_rank=3', './configs/retinanet\_r50\_400\_050x.py', '--launcher', 'pytorch']' returned non-zero exit status 1.

# 5. OSError: CUDA\_HOME environment variable is not set. Please set it to your CUDA install root.

一次性export CUDA\_HOME=/usr/local/cuda-X.X

永久解决

sudo vi ~/.bashrc

CUDA\_HOME=/usr/local/cuda-X.X

6.ld: warning: cannot find entry symbol \_start; not setting start address

/usr/bin/ld: cannot find -l\* 错误的解决方法

服务器中有这个库的情况,使用软连接

ld -l\* –verbose (这里的l\*代表以l开头的)

 找到**succeeded**路径，作为**path1**

从错误信息中获得path2

ln -s path1 path2

服务器中没有库的情况，安装库

参考博文：https://blog.csdn.net/liuxiao723846/article/details/100128209?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_title~default-0.no\_search\_link&spm=1001.2101.3001.4242

7.新建一个虚拟环境（python3.7的），安装库时显示已经装过这个库（在./anaconda3/lib/python3.8/site-packages文件夹下），但是虚拟环境文件夹下却没有这个包

法1：使用了软链接

ln -s /data/lihongxi/anaconda3/envs/mmcv\_124 /data/lihongxi/anaconda3/lib/python3.8/site-packages

rm -rf  /data/lihongxi/anaconda3/lib/python3.8/site-packages

目前解决不了问题

法2：pip指定目录安装

pip install torch==1.6.0+cu101 torchvision==0.7.0+cu101 -f <https://download.pytorch.org/whl/torch_stable.html> --target=/data/lihongxi/anaconda3/envs/mmcv\_124/lib/python3.7/site-packages

成功解决

# 8. ModuleNotFoundError: No module named xxx 的原因和解决办法

# 一种是没有这个模块

# 一种是多个文件夹都有这个模块(安装mmdetection时pycocotools报错)

9.进入了新的虚拟环境（python=3.7）却仍然使用了python3.8.8

重装conda后解决这个问题

11. RuntimeError:CUDA error:out of memory

默认使用的那gpu内存不足

12.PIL无法显示图片

https://blog.csdn.net/u014114109/article/details/79559946

**训练技巧**

1.batch\_size: 批大小。在深度学习中，一般采用SGD训练，即每次训练在训练集中取batchsize个样本训练

iteration: 1个iteration等于使用batchsize个样本训练一次

epoch:1个epoch等于使用训练集中的全部样本训练一次

1.当数据量足够大的时候可以适当的减小batch\_size,由于数据量太大，内存不够。但盲目减少会导致无法收敛，batch\_size=1时为在线学习，也是标准的SGD，这样学习，如果数据量不大，noise数据存在时，模型容易被noise带偏，如果数据量足够大，noise的影响会被“冲淡”，对模型几乎不影响。2.batch的选择，首先决定的是下降方向，如果数据集比较小，则完全可以采用全数据集的形式。这样做的好处有两点

2.linux跑代码将推理后的图片保存

https://blog.csdn.net/qq\_36756866/article/details/116762837

3.shell连接服务器突然断开

Socket error Event: 32 Error: 10053.

Connection closing...Socket close.

因为连接断开后模型仍在服务器上跑，如果log文件不再更新了则训练终止

传统的Mimic过程，一般使用一个已经训练好的大模型，固定该模型的weights不变，设计一个小模型，学习大模型的soft targets 或者logits的输出；大模型学习到有效的信息可以传递给小模型，使得小模型也可以有较为不错的性能表现，其Loss函数如下：

C++

1.vector用法

a.创建vector对象

vector<int> vec;

push\_back()函数

函数将一个新的元素加到vector的最后面，位置为当前最后一个元素的下一个元素

相当于python里的列表的append方法

2.stack用法（是一个容器类的改编）

empty() 堆栈为空则返回真

pop() 移除栈顶元素（删除）

push() 在栈顶增加元素（增加）

size() 返回栈中元素数目

top() 返回栈顶元素，不删除（获取）

**力扣**

1.从尾到头打印单链表（利用栈）

把该链表的值从头到尾依次入栈，再出栈

classSolution {

public:

vector<int>reversePrint(ListNode\* head) {

if(!head)

return {};

vector<int> a=reversePrint(head->next);

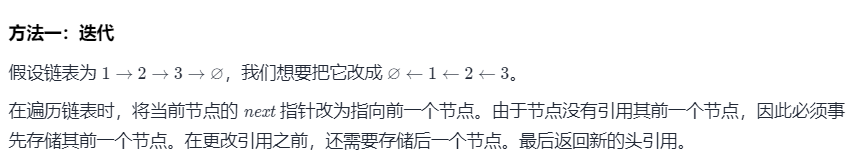
a.push\_back(head->val);

return a;

}

};

2.反转链表



时间复杂度O(n)，空间复杂度O(1)

3.复制复杂链表



class Solution {

public:

Node\* copyRandomList(Node\* head) {

if (head == nullptr) {

return nullptr;

}

为每个结点S添加对应的复制后的结点S’

for (Node\* node = head; node != nullptr; node = node->next->next) {

Node\* nodeNew = new Node(node->val);

nodeNew->next = node->next;

node->next = nodeNew;

}

#随机指针的赋值，因为原结点的随机指针还在，直接找原结点的随机指针

for (Node\* node = head; node != nullptr; node = node->next->next) {

Node\* nodeNew = node->next;

nodeNew->random = (node->random != nullptr) ? node->random->next : nullptr;//node->random->next才是新节点要连接的结点

}

Node\* headNew = head->next;

#新链表S’的串接（之前算单独的结点，先在用next串起来）

for (Node\* node = head; node != nullptr; node = node->next) {

Node\* nodeNew = node->next;

node->next = node->next->next;//原链表连回原来的结点，！！因为是复制链表，因此新节点上也不能有多余的指针指向它

nodeNew->next = (nodeNew->next != nullptr) ? nodeNew->next->next : nullptr;

}

return headNew;

}

};

算法 acwing

**ATSS**

CUDA\_VISIBLE\_DEVICES=1 python tools/train.py configs/atss/atss\_r50\_fpn\_1x\_coco.py --resume-from ./checkpoints/epoch\_11.pth

CUDA\_VISIBLE\_DEVICES=0 python tools/test.py configs/atss/atss\_r50\_fpn\_1x\_coco.py checkpoints/atss\_r50\_fpn\_1x\_coco\_20200209-985f7bd0.pth

FSAF

CUDA\_VISIBLE\_DEVICES=3 python tools/train.py configs/fsaf/fsaf\_r50\_fpn\_1x\_coco.py --resume-from ./checkpoints/atss\_r50\_fpn\_1x\_coco\_20200209-985f7bd0.pth

mmdetection测试

python tools/test.py configs/faster\_rcnn/faster\_rcnn\_r50\_fpn\_1x\_coco.py \

checkpoints/faster\_rcnn\_r50\_fpn\_1x\_coco\_20200130-047c8118.pth \

--show

python tools/test.py configs/ faster\_rcnn\_r50\_fpn\_1x\_coco.py checkpoints/faster\_rcnn\_r50\_fpn\_1x\_coco\_20200130-047c8118.pth

python tools/test.py configs/atss/atss\_r50\_fpn\_1x\_coco.py checkpoints/atss\_r50\_fpn\_1x\_coco\_20200209-985f7bd0.pth

CUDA\_VISIBLE\_DEVICES=3 python tools/test.py configs/faster\_rcnn/faster\_rcnn\_r50\_fpn\_1x\_coco.py checkpoints/faster\_rcnn\_r50\_fpn\_1x\_coco\_20200130-047c8118.pth –show

**Pycharm**

远程开发

<https://www.cnblogs.com/xuegqcto/p/8621689.html>

修改本地代码会改变远程代码

而修改远程代码运行会报错，解决方法是用远程的覆盖本地代码

如果要新增文件夹或者添加文件，则需要手动添加

在pycharm左边项目栏里新增python文件则远程也会新增

远程调试显示图像

下载xming

设置xshell

更改服务器的系统文件

https://blog.csdn.net/qq\_35189360/article/details/96453135

远程显示图像

必须打开xshell和xming（先打开xshell再打开xming）

Anaconda

由于先装的miniconda，没有装python

Conda的python路径直接加入到了系统变量中

**MMdetection用法**

初始化一个检测器

model = init\_detector(config\_file, checkpoint\_file, device=device)