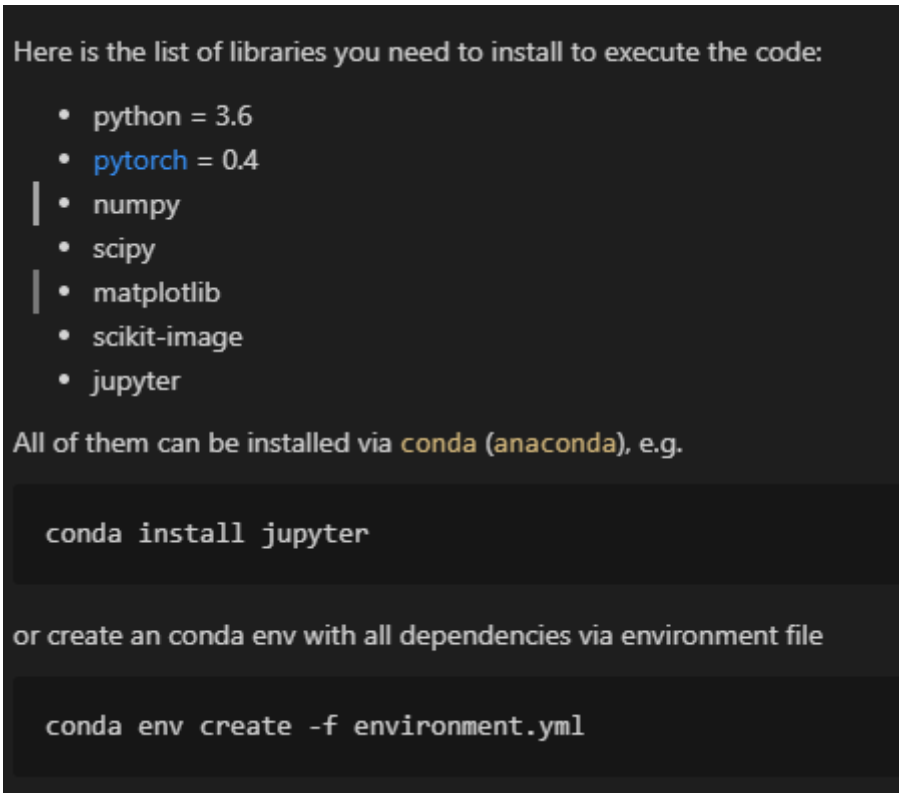
summary

简单介绍一下

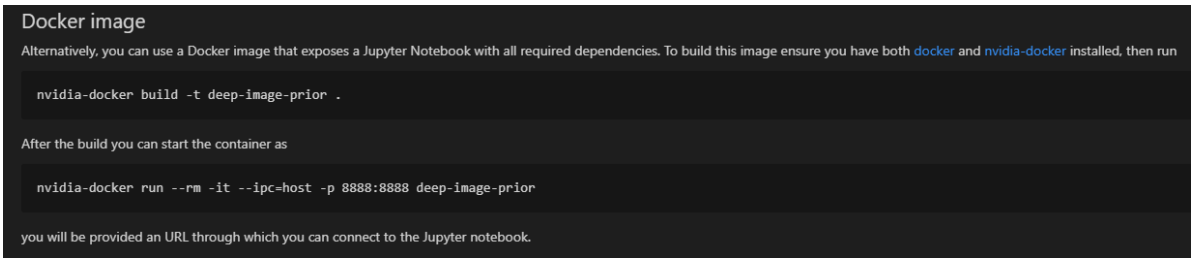
论文简单过了一遍，主要的时间都花在配置环境上了，过程中踩的坑以及解决方案总结如下：

GitHub上的代码作为算法的演示以`ipynb`文件的形式给出，跑起来还是相对容易的，并且在repo的`README.md`文件中作者很贴心地给出了三种运行方式:

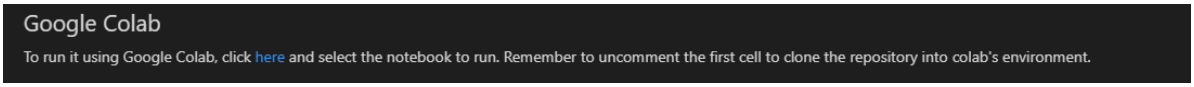
- anaconda（conda）



- docker



- google colab



conda和colab的方式我都试了一下，具体的一个一个说。

anaconda

一些anaconda的介绍和使用也可以提两句，最重要的操作莫过于换源（我一般用tuna镜像）和环境管理，查看文档点这里，也可以点这里看别人的总结

实际上用的是conda搭建环境，当然用pip也可以，区别在于conda会自动解决依赖，pip要手动解决（我记得pip能解决一些）

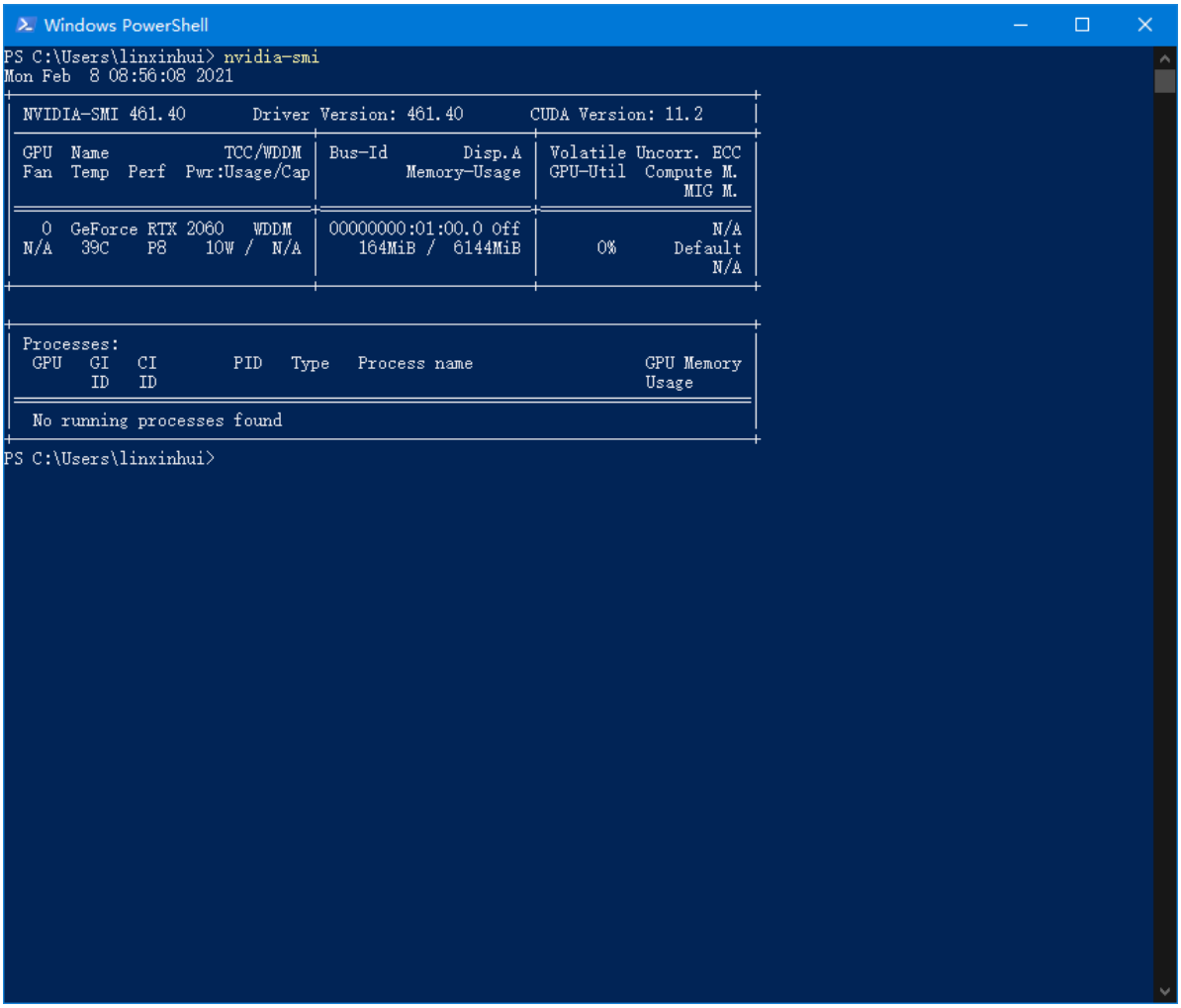
说回正题，Python（解释器）、numpy、jupyter之流安装起来都非常容易，搭建环境的核心在于安装pytorch，官网上给了详细的文档，当然为了下载速度还需要国内镜像

第一个问题

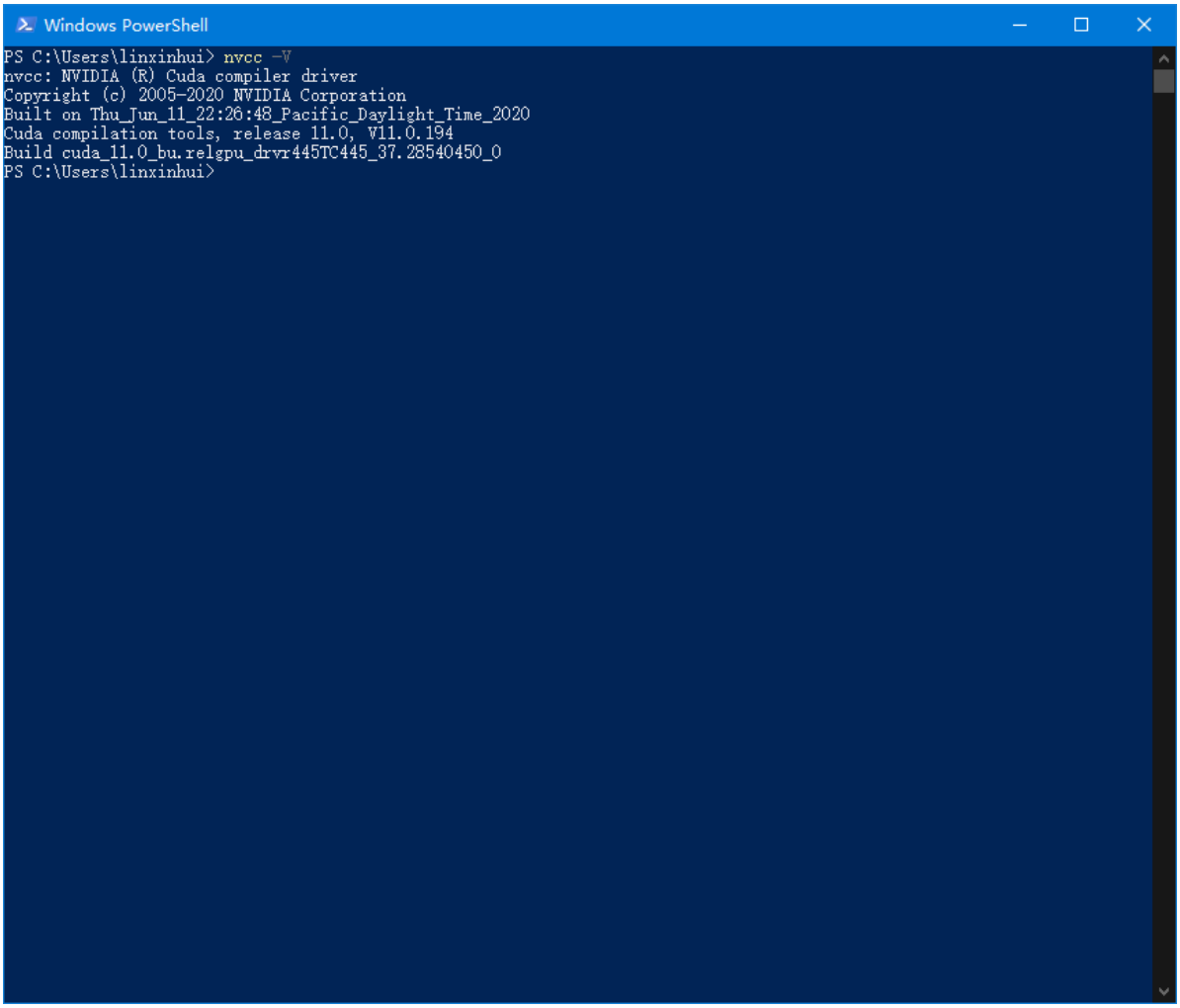
第一个问题是尽管pytorch有`cpu-only`和`gpu`的版本，但是要跑起来这里的代码只能用`gpu`版本，作者使用的环境是`pytorch0.4`+`cuda9.0`，但是cuda9.0和我的显卡不兼容，所以最后我选了最新的`pytorch1.7(stable)+cuda11.0+cuDNN11.2`（Python版本还是3.6），幸运的是代码也能运行

在安装cuda的时候一定要注意版本，首先要满足显卡兼容性（按照经验是不能用早于显卡出厂时间的版本），然后要满足框架（比如pytorch最高支持cuda11.0，而cuda的最新版本号是11.2），这种对应关系在pytorch等的官网上可以查到，当然还可以用百度搜索，当然还可以看镜像站上具体包的包名

用`nvidia-smi`检查显卡驱动（这里显示的cuda版本是最新版，不用管它）：



用`nvcc -v`检查cuda版本：

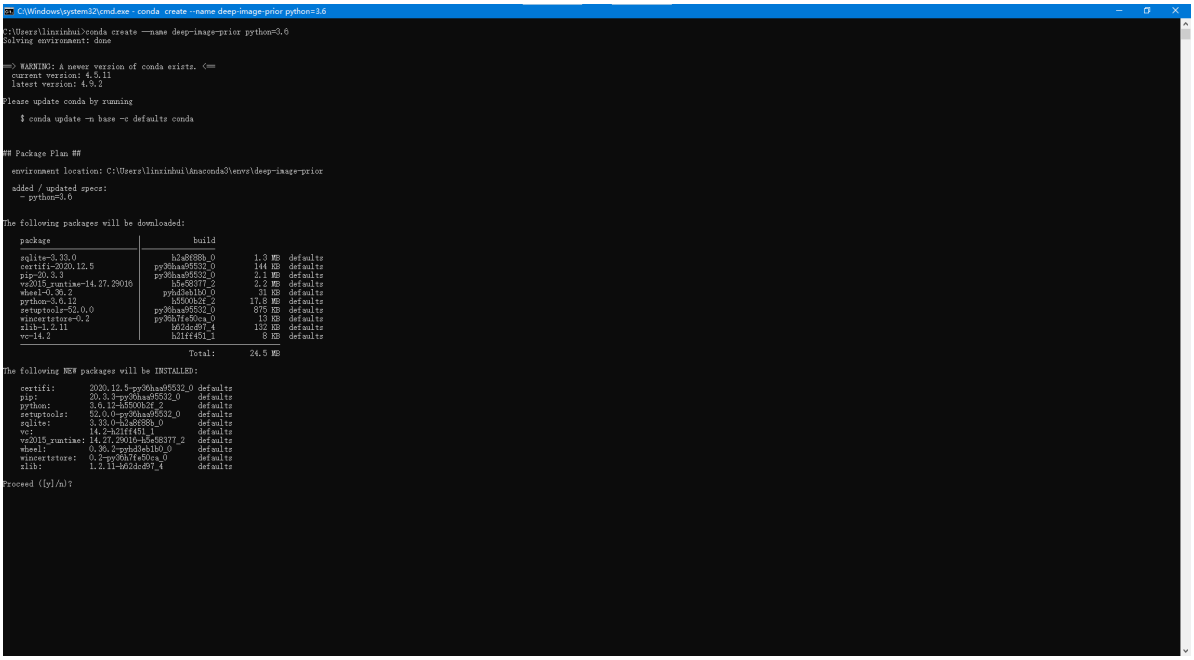


检查cuda能否与显卡建立连接的最严谨的方法是运行`deviceQuery.exe`和``，这也是nvidia官网guide中的方法

还有通过pytorch检查的，见下。

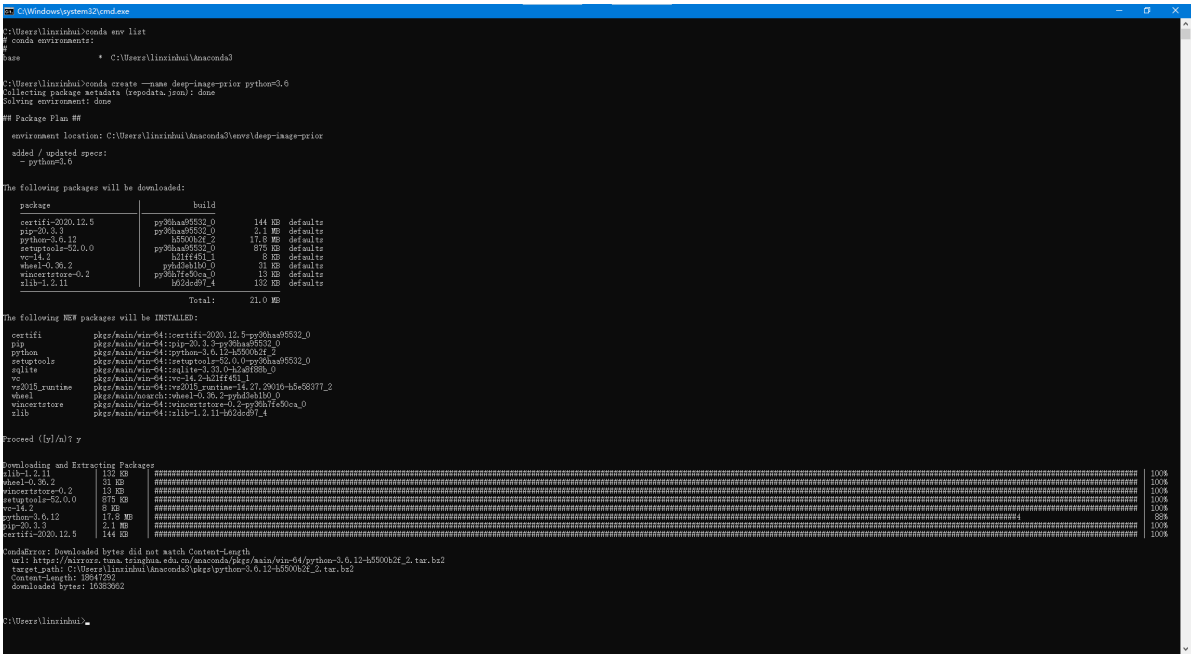
第二个问题

解决cuda和cuDNN的问题之后就可以用conda配置环境了，conda有导出环境的功能，但实质上是一个“配置单”，该下载的包还是要重新下载，而且由于上述问题，pytorch和cuda不会匹配，所以无视作者给的`environment.yml`，手动配置conda，首先创建环境：

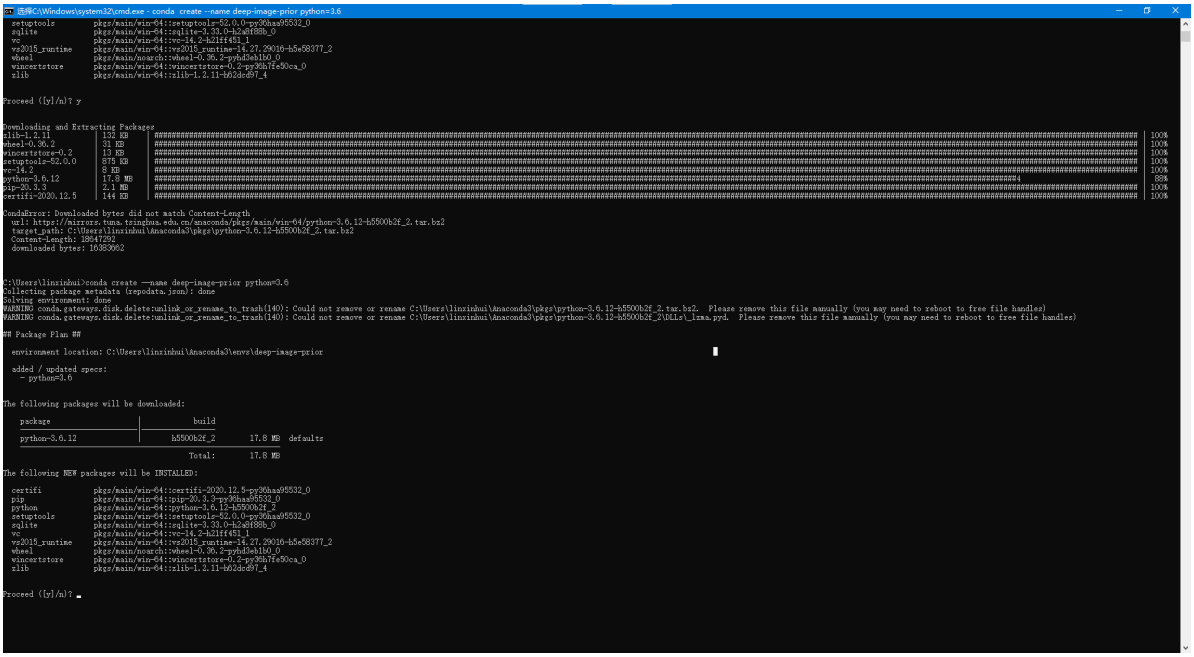


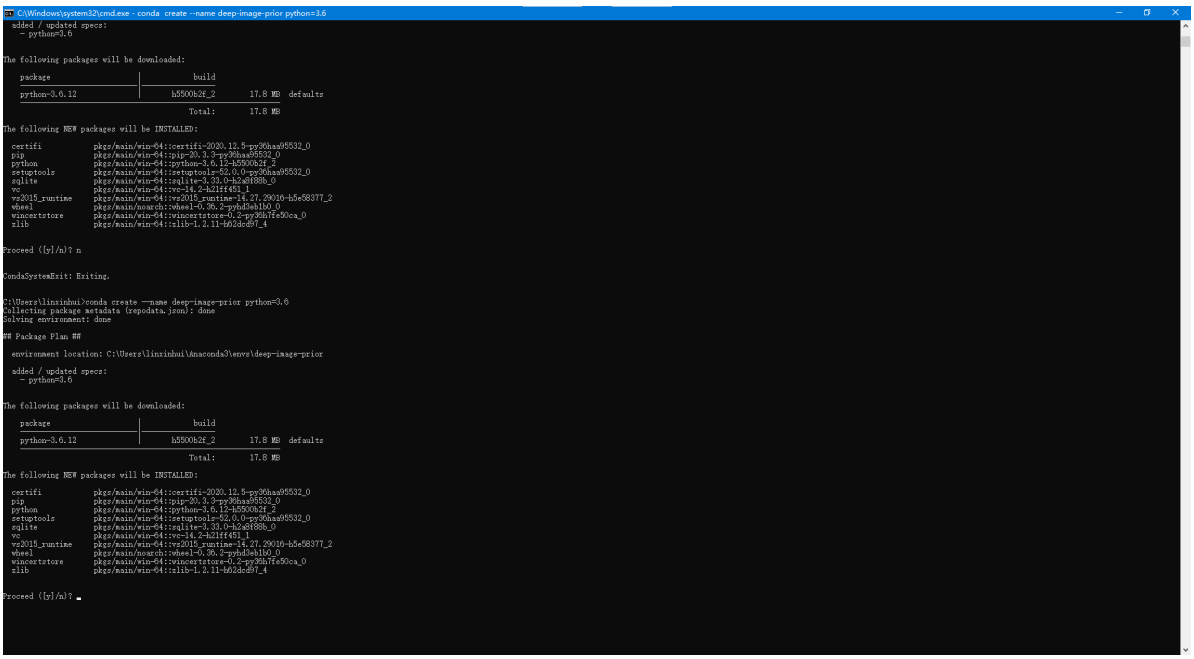
conda会自动解决包依赖的问题，打印所有需要下载的包。

第二个问题就是conda下载速度慢的问题，当下载量很大时，很有可能出现无法下载的情况：



conda没有断点续传功能，这时候需要手动删除已下载的**不完整的**包（必要时重启），然后重试下载：





下载速度慢毫无疑问是网络的问题，这时候当然首先考虑换源，然而操作的过程中发现即使换了国内镜像，下载速度还是很慢，大文件几乎成功不了，换源之后再挂上VPN效果不错。

当然这并不能完全解决问题并不能解决问题，要完全解决这个问题，可以采取如下方案：

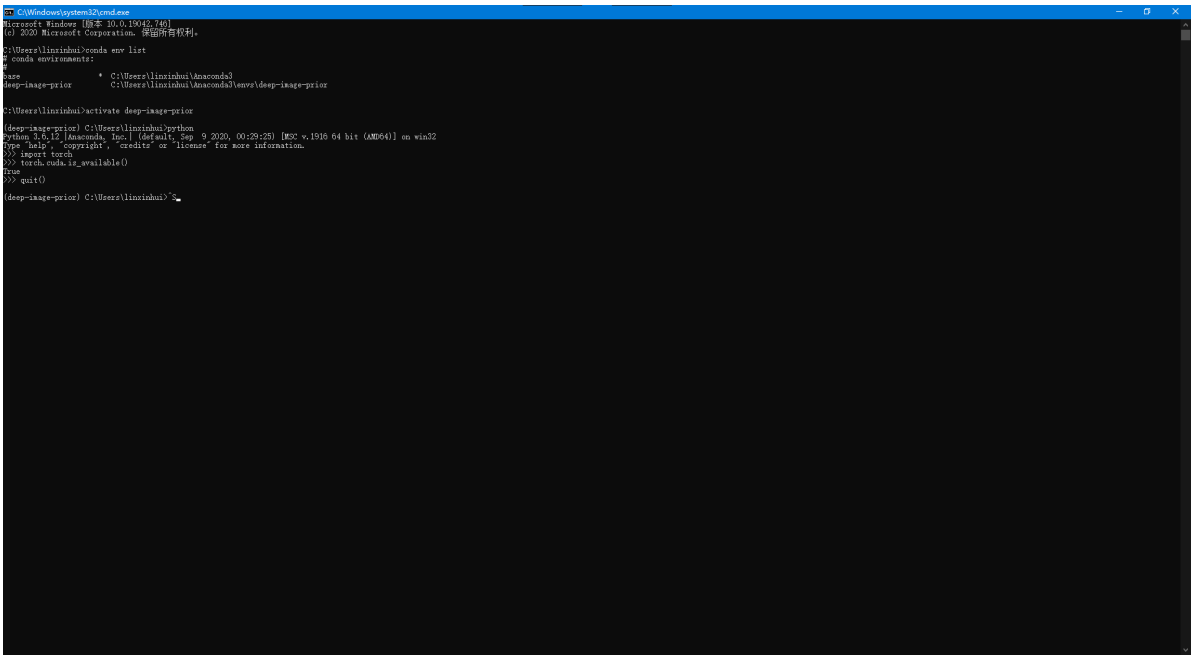
- 修改conda用于下载文件的函数的所在文件
  - 用`python -c "from conda.gateways.connection import download; print(download.__file__)"`找到目标文件
  - 找到`download()`函数，该函数的开始处加上如下代码：

```
from ...base.constants import CONDA_TEMP_EXTENSION

tmp_file_path = target_full_path + CONDA_TEMP_EXTENSION
if exists(tmp_file_path):
    print("\n[Download patch] file exists: %s", tmp_file_path)

    checksum_ok = True

    if sha256 or md5:
        builder = hashlib.new("sha256" if sha256 else "md5")
        checksum = sha256 if sha256 else md5

        with open(tmp_file_path, 'rb') as f:
            for chunk in iter(lambda: f.read(4096), b''):
                builder.update(chunk)

        actual_checksum = builder.hexdigest()
        if actual_checksum != checksum:
            print("\n[Download patch] cached file checksum mismatch: %s (%s
!= %s)",
                  checksum_type, actual_checksum, checksum)
        checksum_ok = actual_checksum == checksum

    if checksum_ok:
        from ..disk.update import backoff_rename
        backoff_rename(tmp_file_path, target_full_path, True)
        if progress_update_callback:
            progress_update_callback(1.0)
```

```
        print("\n[Download patch] using cached file instead of download",
target_full_path)
        return
```

- 从镜像站手动下载需要的包，放到conda的pkgs目录下
- （重启电脑）
- 重复下载过程，应该出现`[Download patch] using cached file instead of download`等字样，下载进度条直接跑到底，稍等片刻即可

最后检查安装情况：



这种方法本质上是离线安装，只适用于下载的包很少的情况（一般来说可以在线下载体积较小的包，像pytorch这样1GB多的包应该是少数情况）。

docker

暂时没用过，据说可以满足多下同切换和直接调GPU资源的需求。

Google Colab

要搭梯子。

这大概是跑这篇论文的代码的最简单的方式，作者已经把环境都配置好了（查了一下git log，后期不少提交都是关于colab的，可能作者也觉得环境不好整[doge]），注意看注释（要先把相关的仓库clone到colab环境中）。

据说colab的IO瓶颈是个大问题，当然网速也是，我想如果后期需要类似的平台训练可以用百度paddle之类国内平台。

还有一些问题

- 论文不能完全看懂

  这是知识储备的问题，数字图像处理、机器学习、深度学习、线性代数概率论等基础该学还得学

- 科学上网的问题

  科学上网快成刚需了，ExpressVPN，NordVPN这些服务商提供的VPN服务支持多人共用一个账号，我们三个人分摊一下差不多每个月每人二十到三十

我好奇这笔支出能报销么[doge]

- 环境切换和算力的问题

  需要在不同的环境（操作系统）下切换，包括Windows，Ubuntu16，Ubuntu18等，虚拟机和WSL（WSL1和WSL2中的一个甚至用不了GPU）的方案根据测试GPU性能的折损挺厉害的，双系统一次只能跑一个，也不是很好的方案，docker据说可以直接调用GPU，这个有待研究，比较靠谱的方法可能是用colab这样的平台，或者租服务器自己搭平台（费用挺高，并且我们没有相关的经验）