

# 在Ubuntu22.04LTS上配置深度学习环境

下划线部分带有超链接，可以直接访问 / 下载

## 1 配置Clash for Linux

能够解决很多问题，比如某些软件源位于墙外

### 1.1 下载Clash for Windows

直接到github下载CFW，这里我们选择下载适用于x64的Linux的版本：

**v 0.20.20** Latest

- update Electron v22.3.5
- add `Hide time-out proxies` button in Proxies module
- add `GUI Data Folder` setting
- break connections when proxy changes in tray menu
- persist filter keyword in Proxies module
- fix some other glitches

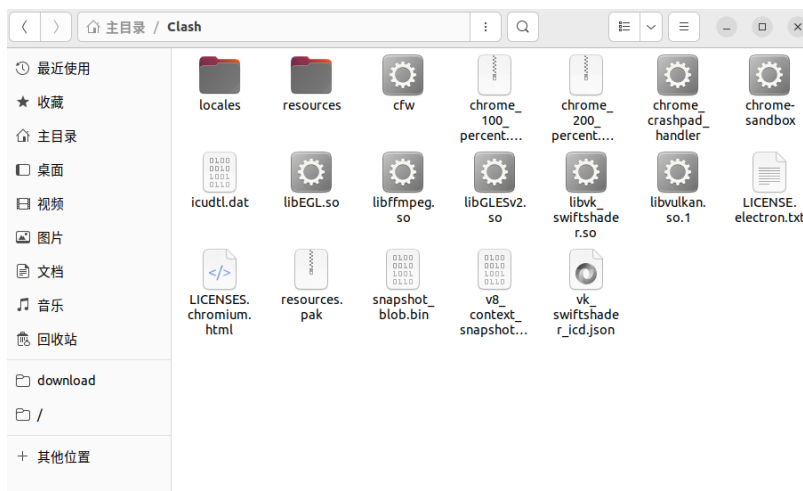
**Assets** 15

<a href="#">Clash.for.Windows-0.20.20-arm64-linux.tar.gz</a>	104 MB	last week
<a href="#">Clash.for.Windows-0.20.20-arm64-mac.7z</a>	69.6 MB	last week
<a href="#">Clash.for.Windows-0.20.20-arm64-win.7z</a>	74.2 MB	last week
<a href="#">Clash.for.Windows-0.20.20-arm64.dmg</a>	103 MB	last week
<a href="#">Clash.for.Windows-0.20.20-ia32-win.7z</a>	69.7 MB	last week
<a href="#">Clash.for.Windows-0.20.20-mac.7z</a>	69.9 MB	last week
<a href="#">Clash.for.Windows-0.20.20-win.7z</a>	73.5 MB	last week
<a href="#">Clash.for.Windows-0.20.20-x64-linux.tar.gz</a>	105 MB	last week
<a href="#">Clash.for.Windows-0.20.20.dmg</a>	108 MB	last week
<a href="#">Clash.for.Windows.Setup.0.20.20.arm64.exe</a>	81.1 MB	last week
<a href="#">Source code</a> (zip)		Feb 21
<a href="#">Source code</a> (tar.gz)		Feb 21

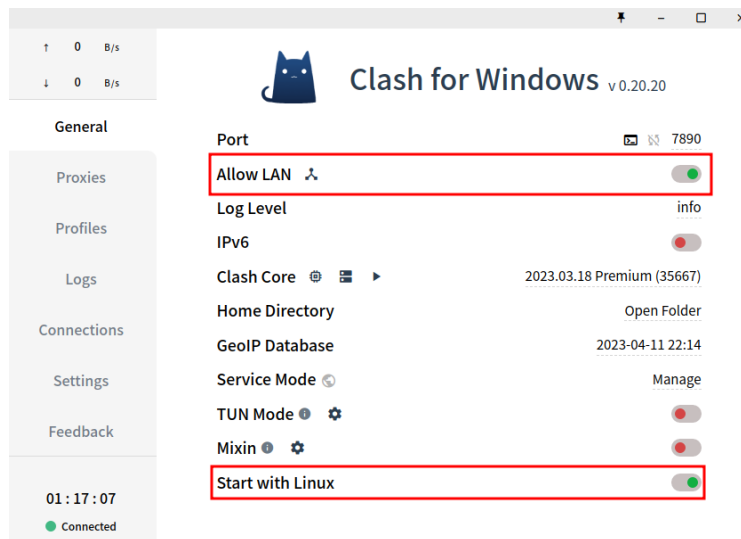
Show all 15 assets

92 7 8 12 8 111 people reacted

下载好后，我们在主目录中创建Clash文件夹，将压缩包解压到该文件夹：



随后我们运行cfw，将会出现Clash的GUI：



大家在第一次安装时应该和上图有些不同，请确保General页面与我的设置保持一致。

随后，我们的操作和在Windows上配置Clash一样（导入订阅链接、选择节点等等），此处不做赘述。

## 1.2 配置代理

要注意的是，Clash for Linux并没有“System Proxy”的开关，因此我们需要自己设置代理。

打开终端，输入如下命令将文件读写状态改为可读可写：

```
1 sudo chmod 666 /etc/environment
```

随后我们对文件进行编辑：

```
1 sudo nano /etc/environment
```

其中nano编辑器是Ubuntu自带的一款文本编辑器，其操作相比于老牌的Vim来说更加简单，学习成本较低。

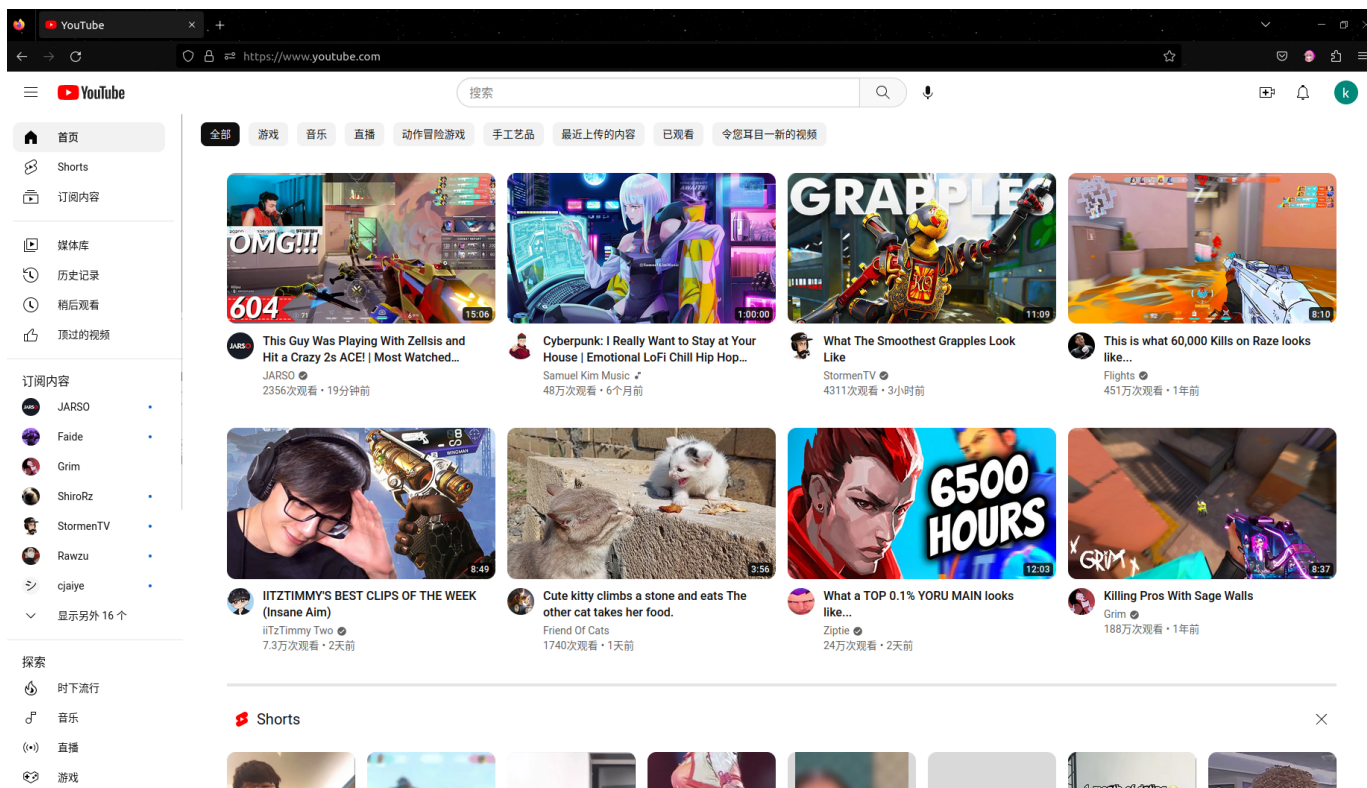
我们在该文件中加入如下内容：

```
1 http_proxy=http://127.0.0.1:7890/
2 https_proxy=http://127.0.0.1:7890/
3 ftp_proxy=http://127.0.0.1:7890/
4 HTTP_PROXY=http://127.0.0.1:7890/
5 HTTPS_PROXY=http://127.0.0.1:7890/
6 FTP_PROXY=http://127.0.0.1:7890/
```

随后依次按下Ctrl+O（写入），Enter（保存），Ctrl+X（退出），回到终端运行如下命令将文件读写状态改为只读并重启PC：

```
1 sudo chmod 444 /etc/environment
2 sudo reboot
```

PC重启后，我们可以尝试打开一些本来打不开的网站，若可以正常使用，则说明配置完成。



## 2 安装显卡驱动

做深度学习必然是要用到GPU的，CPU的算力后期肯定不够用

### 2.1 查看推荐驱动

打开终端，输入以下命令查看显卡可用的驱动：

```
1 ubuntu-drivers devices
```

可以看到以下界面：

```
1 (base) sakana@sakana:~$ ubuntu-drivers devices
2 == /sys/devices/pci0000:00/0000:00:01.0/0000:01:00.0 ==
3 modalias : pci:v000010DEd000024C9sv00001462sd00005058bc03sc00i00
4 vendor   : NVIDIA Corporation
5 manual_install: True
6 driver   : nvidia-driver-515-server - distro non-free
7 driver   : nvidia-driver-515 - distro non-free
8 driver   : nvidia-driver-525-open - distro non-free recommended
9 driver   : nvidia-driver-525 - distro non-free
10 driver  : nvidia-driver-525-server - distro non-free
11 driver  : nvidia-driver-515-open - distro non-free
12 driver  : nvidia-driver-510 - distro non-free
13 driver  : xserver-xorg-video-nouveau - distro free builtin
```

其中，可以看见第7行的：

```
1 driver   : nvidia-driver-525-open - distro non-free recommended
```

这行输出中带有“recommended”字样，证明这是系统推荐安装的驱动版本，我们需要记录下来。

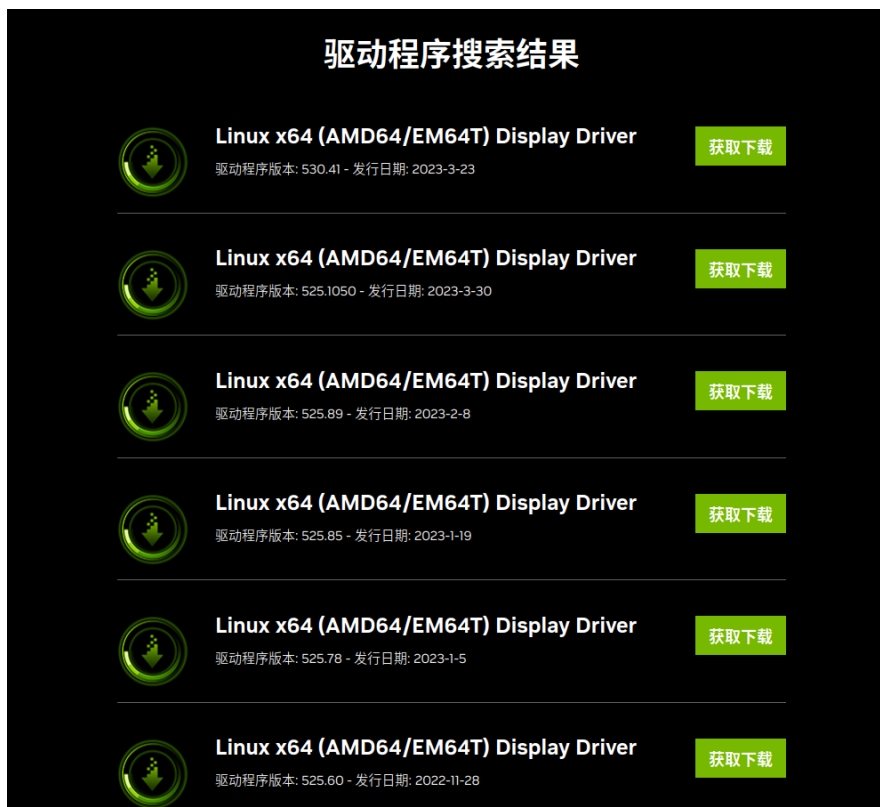
## 2.2 下载驱动

没什么好说的，去NV官网下载需要对应版本的显卡驱动，注意系统类型的选择，这里以64位的Linux系统为例：



The screenshot shows the 'Manual Search for Drivers' (手动搜索驱动程序) interface on the NVIDIA website. It features a series of dropdown menus for selecting the product type (GeForce), product series (GeForce RTX 30 Series), specific product (GeForce RTX 3060 Ti), operating system (Linux 64-bit), language (Chinese (Simplified)), and download type (All). A green 'Start Search' (开始搜索) button is at the bottom right.

搜索结果如下，注意选择之前记录下来的推荐版本。推荐版本可能有很多细分的版本，选择不太新也不太旧的就好：



The screenshot displays the 'Driver Search Results' (驱动程序搜索结果) page. It lists six different driver versions for Linux x64 (AMD64/EM64T) Display Driver. Each entry includes a download icon, the driver name, the version number, the release date, and a green 'Get Download' (获取下载) button. The versions listed are 530.41, 525.105.0, 525.89, 525.85, 525.78, and 525.60.

Driver Name	Version	Release Date	Action
Linux x64 (AMD64/EM64T) Display Driver	530.41	2023-3-23	获取下载
Linux x64 (AMD64/EM64T) Display Driver	525.105.0	2023-3-30	获取下载
Linux x64 (AMD64/EM64T) Display Driver	525.89	2023-2-8	获取下载
Linux x64 (AMD64/EM64T) Display Driver	525.85	2023-1-19	获取下载
Linux x64 (AMD64/EM64T) Display Driver	525.78	2023-1-5	获取下载
Linux x64 (AMD64/EM64T) Display Driver	525.60	2022-11-28	获取下载

这里我选择的是525.78版本，我们将其下载到本地目录，并记录其存放的路径和文件名。

要注意的是如果使用火狐浏览器默认的下载目录，且安装的Ubuntu系统为中文，驱动程序的路径中会带有中文，这在之后的安装中会带来很大的麻烦。

解决办法1：将“下载”文件夹更名为“download”

解决办法2：将下载好后的驱动程序放置到一个不带中文的路径中

## 2.3 安装依赖

在终端中运行如下命令：

```
1 sudo apt-get update
2 sudo apt-get upgrade
```

等待软件包安装完成后，继续运行如下命令：

```
1 sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-
  dev protobuf-compiler
2 sudo apt-get install --no-install-recommends libboost-all-dev
3 sudo apt-get install libopenblas-dev liblapack-dev libatlas-base-dev
4 sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev
```

如果安装失败，则先进行pip3的升级更新，运行如下命令：

```
1 # 未安装pip3
2 sudo apt-get install python3-pip
3 # 安装的pip3版本较低
4 sudo pip3 install --upgrade pip
```

## 2.4 禁用已有显卡驱动

Ubuntu系统在安装完成后会使用自带的开源第三方显卡驱动nouveau，在安装驱动前我们需要将其禁用。

在终端中运行如下命令：

```
1 sudo nano /etc/modprobe.d/blacklist.conf
```

打开文件后，我们在文件的末尾写入：

```
1 blacklist nouveau
2 options nouveau modeset=0
```

随后保存，回到终端运行如下命令进行更新并重启PC：

```
1 sudo update-initramfs -u
2 sudo reboot
```

PC重启后，在终端中运行如下命令检验nouveau是否正确禁用：

```
1 lsmod | grep nouveau
```

若无输出结果，说明nouveau已被正确禁用，可以继续执行后续操作。

## 2.5 配置环境变量

与Windows一样的是，我们在Linux中同样需要配置环境变量。在终端中输入如下命令打开配置文件：

```
1 sudo nano ~/.bashrc
```

值得注意的是，.bashrc是一个隐藏文件，我们在文件管理器中勾选“显示隐藏文件”后便可以在主目录中看见它。

我们在末尾写入：

```
1 export LD_LIBRARY_PATH=/usr/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH
2 export LD_LIBRARY_PATH=/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH
```

随后更新环境变量：

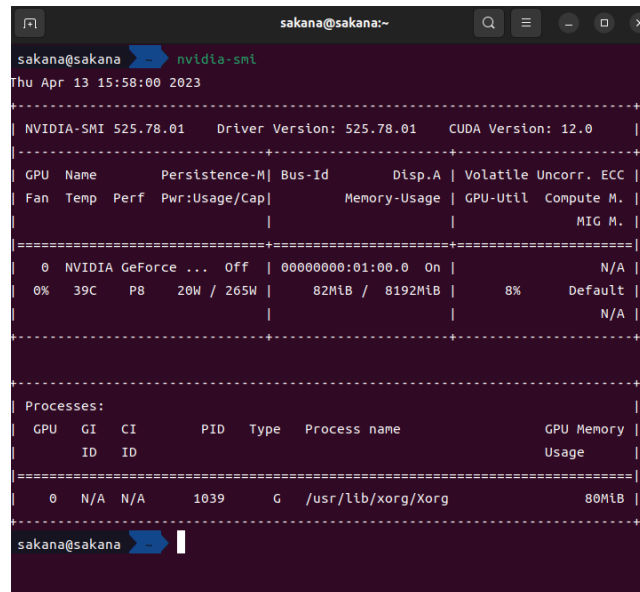
```
1 source ~/.bashrc
```

## 2.6 卸载已有驱动

在往下看前，请先在终端中运行如下命令：

```
1 nvidia-smi
```

若提示未找到命令，那么请直接跳转至2.7进行后续操作，若出现下图所示的结果，请继续阅读。

A terminal window titled 'sakana@sakana:~' showing the output of the 'nvidia-smi' command. The output displays NVIDIA-SMI 525.78.01, Driver Version: 525.78.01, and CUDA Version: 12.0. It also shows a table of GPU information for a single NVIDIA GeForce GPU, including its name, persistence mode, bus ID, display mode, memory usage, and temperature. Below this, it shows a table of processes running on the GPU, with one process listed: /usr/lib/xorg/Xorg with PID 1039 and GPU Memory Usage of 80MiB.

```
sakana@sakana ~$ nvidia-smi
Thu Apr 13 15:58:00 2023

+-----+
| NVIDIA-SMI 525.78.01      Driver Version: 525.78.01   CUDA Version: 12.0     |
+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+
|  0  NVIDIA GeForce ...    Off      | 00000000:01:00.0  On  |          N/A         |
| 0%   39C   P8     20W / 265W |  82MiB /  8192MiB |      8%    Default  |
|                                           |          N/A         |
+-----+-----+

Processes:
+-----+
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
|  ID   ID                                     Usage      |
+-----+
|  0   N/A  N/A       1039    G   /usr/lib/xorg/Xorg                  80MiB     |
+-----+
```

我们在安装适合版本的驱动时，需要先卸载现有的驱动，在终端中运行如下命令：

```
1 sudo apt-get --purge remove nvidia*
2 sudo apt autoremove
```

随后再运行nvidia-smi，确保提示未找到命令。

## 2.7 退出图形化界面

Ubuntu22.04LTS的x-server为tty1-6，图形化界面为tty7，因此我们按住Ctrl+Alt+F1（1-6均可），进入x-server并运行如下命令关闭图形化界面：

```
1 | sudo service lightdm stop
```

若提示该服务未装载，则先安装Lightdm：

```
1 | sudo apt install lightdm
```

安装完成后会跳出一个界面勾选默认的显示管理器（Display Manager），我们选择Lightdm作为默认的显示管理器。

随后再次运行第一条命令，关闭图形化界面。

## 2.8 驱动安装

首先给驱动程序权限，并执行安装：

```
1 | sudo chmod +x 驱动名称.run*  
2 | sudo sh 驱动名称.run* --no-opengl-files --no-x-check --no-nouveau-check
```

值得注意的是，在图形化界面关闭后我们就没有办法直观的看到文件的名称了，且由于x-server模式下我们没办法输入中文，所以之前需要大家将驱动转移到没有中文路径的地方并记住它的文件名。

在安装过程中会出现一些勾选项，包括但不限于：

```
1 | 1.The distribution-provided pre-install script failed! Are you sure you want to continue?  
2 |  
3 | "Yes"  
4 |  
5 | 2.Would you like to register the kernel module sources with DKMS? This will allow DKMS to  
6 | automatically build a new module, if you install a different kernel later?  
7 |  
8 | "No"  
9 |  
9 | 3.Nvidia's 32-bit compatibility libraries?  
10 |  
11 | "No"  
12 |  
13 | 4.Would you like to run the nvidia-xconfig utility to automatically update your x configuration  
14 | so that the NVIDIA x driver will be used when you restart x? Any pre-existing x config file will be  
15 | backed up.  
14 |  
15 | "Yes"
```

上述为一些重要的勾选项，根据机器不同可能有不同的项目，**切记不要直接乱选！！！！**如果遇到上述勾选项以外的请上网搜索，一旦勾选错可能导致黑屏等问题，需要从头来过。

## 2.9 验证

安装完成后，我们会回到x-server界面，此时我们先挂载NVIDIA驱动：

```
1 modprobe nvidia
```

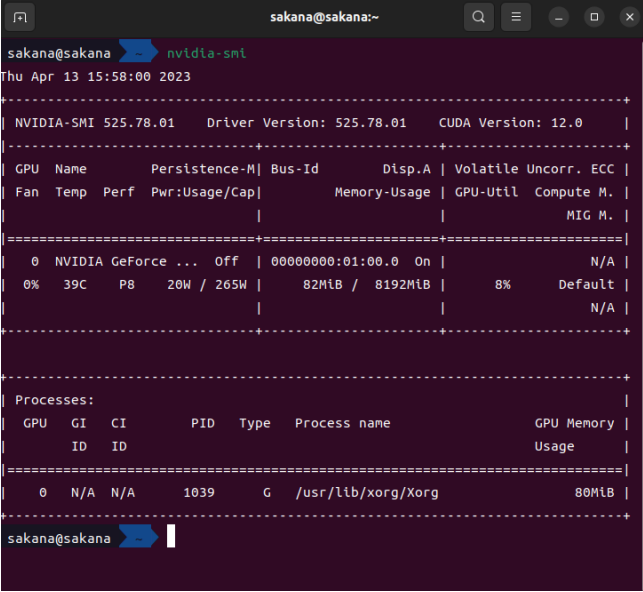
随后我们回到图形化界面并重启PC：

```
1 sudo service lightdm start
2 sudo reboot
```

PC重启后，我们在终端中运行如下命令：

```
1 nvidia-smi
```

若出现如下输出，说明安装成功。



```
sakana@sakana:~$ nvidia-smi
Thu Apr 13 15:58:00 2023

+-----+
| NVIDIA-SMI 525.78.01      Driver Version: 525.78.01   CUDA Version: 12.0     |
+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.     |
+-----+
| 0   NVIDIA GeForce ...   Off   | 00000000:01:00.0  On   |          N/A         |
| 0%   39C    P8      20W / 265W |  82MiB /  8192MiB |      8%      Default |
|                                           MIG M.     |
+-----+

Processes:
+-----+
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
| ID   ID   ID          |              |           Usage              |
+-----+
| 0   N/A  N/A       1039    G   /usr/lib/xorg/Xorg                  80MiB |
+-----+
```

至此显卡驱动安装完毕。

## 3 安装CUDA Toolkit

**统一计算设备架构**（Compute Unified Device Architecture, **CUDA**），是由**NVIDIA**推出的通用并行计算架构。解决的是用更加廉价的设备资源，实现更高效的并行计算。

### 3.1 下载CUDA Toolkit

在NV官网可以下载CUDA Toolkit，在此之前我们需要先利用nvidia-smi命令获知我们的设备最高支持的CUDA版本。如下图所示，我稍后要安装的CUDA Toolkit必须低于12.0。



```
sakana@sakana:~$ nvidia-smi
Thu Apr 13 16:25:54 2023

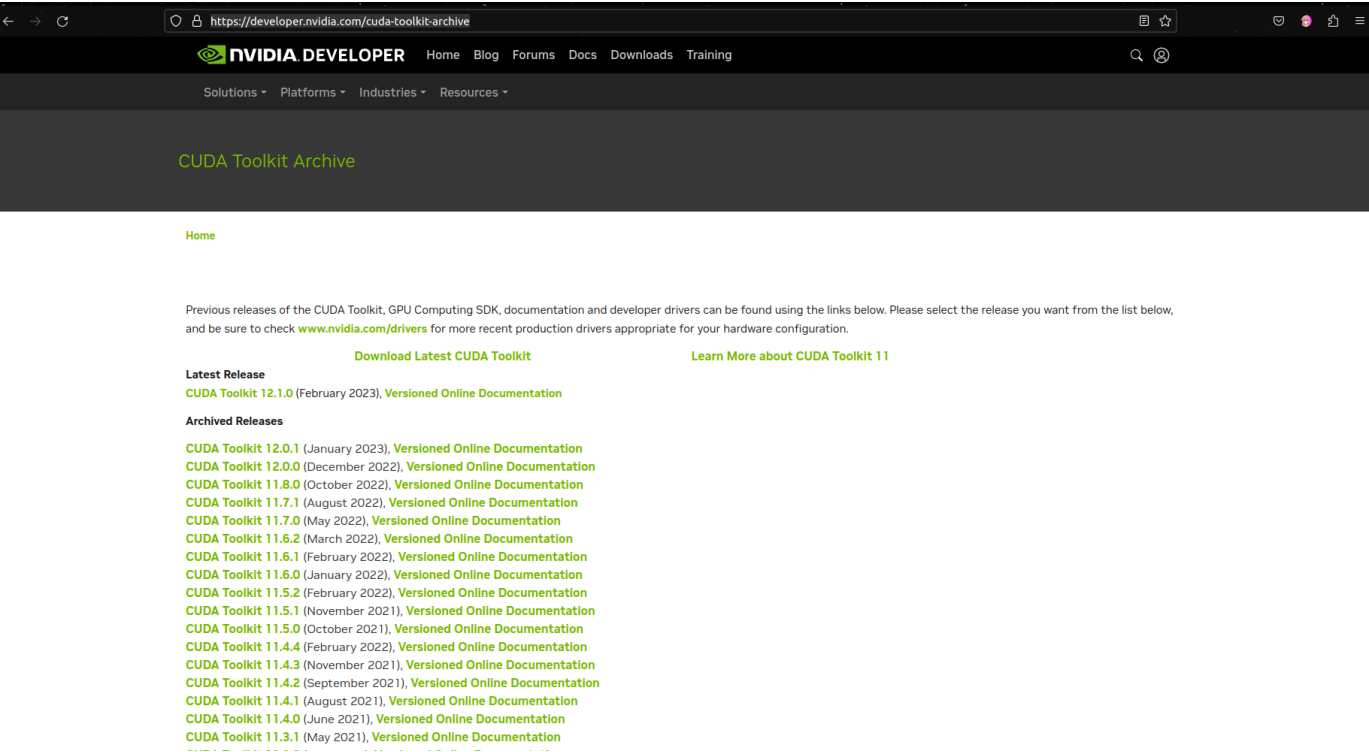
+-----+
| NVIDIA-SMI 525.78.01      Driver Version: 525.78.01   CUDA Version: 12.0     |
+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|=====+-----+
| 0     NVIDIA GeForce ...   Off      | 00000000:01:00.0  On |          N/A         |
| 0%    38C    P8      18W / 265W |  92MiB /  8192MiB |      2%    Default   |
|=====+-----+
|                          |                      |                      |
+-----+

Processes:
+-----+
| GPU   GI    CI          PID    Type   Process name                      GPU Memory |
| ID    ID                                     Usage      |
+-----+
| 0     N/A   N/A         1039    G     /usr/lib/xorg/Xorg                90MiB     |
+-----+

sakana@sakana:~$
```

需要注意的是，这一步获取的CUDA版本是我们设备支持的CUDA上限，能够显示并不代表我们已经安装了CUDA。

在了解了这一点之后，我们到官网找到对应的工具包。



在这里，我选择CUDA 11.8.0。在选择版本后，我们需要查看CUDA的安装文档。安装文档中的第一张表格会给出我们安装CUDA需要的一些依赖及其对应版本，请务必确保自己的有相关依赖且版本适配。

Table 1. Native Linux Distribution Support in CUDA 11.8									
Distribution	Kernel <sup>1</sup>	Default GCC	GLIBC	GCC <sup>2,3</sup>	ICC <sup>3</sup>	NVHPC <sup>3</sup>	XLC <sup>3</sup>	CLANG	Arm C/C++
x86_64									
RHEL 9.0	5.14.0-70.13.1	11.2.1	2.34	11	2021	22.3	NO	14.0	NO
RHEL 8.y (y <= 6)	4.18.0-372.9.1	8.5.0	2.28						
RHEL 7.y (y <= 9)	3.10.0-1160	6.x	2.17						
CentOS 7.y (y <= 9)	3.10.0-1160	6.x	2.17						
OpenSUSE Leap 15.y (y <= 4)	5.14.21-150400.22	7.5.0	2.31						
Rocky Linux 9.0	5.14.0-70.13.1	11.2.1	2.34						
Rocky Linux 8.y (y <= 6)	4.18.0-372.9.1	8.5.0	2.28						
SUSE SLES 15.y (y <= 4)	5.14.21-150400.22	7.5.0	2.31						
Ubuntu 22.04 LTS	5.15.0-25	11.2.0	2.35						
Ubuntu 20.04.z (z <= 4) LTS	5.13.0-30	9.3.0	2.31						
Ubuntu 18.04.z (z <= 6) LTS	5.4.0-89	7.5.0	2.27						
Debian 11.4	5.10.0-16	10.2.1	2.31						
Fedora 35	5.14.10	11.2.1	2.34						
KylinOS V10 SP2	4.19.90-25.14.v2101.ky10	7.3.0	2.28						
Arm64 sbasa									
RHEL 9.0	5.14.0-70.13.1	11.2.1	2.34	11	NO	22.3	NO	NO	21.1
RHEL 8.y (y <= 6)	4.18.0-372.9.1	8.5.0	2.28						
SUSE SLES 15.y (y <= 4)	5.14.21-150400.22	7.5.0	2.31						
Ubuntu 22.04 LTS	5.15.0-25	11.2.0	2.35						
Ubuntu 20.04.z (z <= 4) LTS	5.4.0-100	9.3.0	2.31						
Arm64 Jetson (dGPU)									
L4T <sup>4</sup> 20.04.z (z <= 4)	5.10.65-tegra	9.4.0	2.31	NO	NO	NO	NO	NO	NO
Arm64 Jetson (iGPU)									
L4T <sup>4</sup> 20.04.z (z <= 4)	5.10.104-tegra	9.4.0	2.31	NO	NO	NO	NO	NO	NO
POWER 9									
RHEL 8.y (y <= 6)	4.18.0-372.9.1	8.5.0	2.28	11	NO	22.3	16.1.x	14.0	NO

随后，在选择对应的项目后，选择runfile安装，在终端中运行给出命令的第一条即可完成CUDA Toolkit的下载。（先不要执行第二条命令）

我们同样将其移动至一个没有中文路径的位置，并记录其路径和文件名。

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the [CUDA EULA](#).

Operating System

Linux

Windows

Architecture

x86\_64

ppc64le

arm64-sbsa

aarch64-jetson

Distribution

CentOS

Debian

Fedora

KylinOS

OpenSUSE

RHEL

Rocky

SLES

Ubuntu

WSL-Ubuntu

Version

18.04

20.04

22.04

Installer Type

deb (local)

deb (network)

runfile (local)

Download Installer for Linux Ubuntu 22.04 x86\_64

The base installer is available for download below.

>Base Installer

Installation Instructions:

\$ wget https://developer.download.nvidia.com/compute/cuda/11.8.0/local\_installers/cuda\_11.8.0\_520.61.05\_linux.run

\$ sudo sh cuda\_11.8.0\_520.61.05\_linux.run

## 3.2 退出图形化界面

同样的，由于CUDA Toolkit设计图形界面，我们需要退出图形化界面。

按住Ctrl+Alt+F1（1-6均可），进入x-server并运行如下命令关闭图形化界面：

```
1 | sudo service lightdm stop
```

## 3.3 CUDA安装

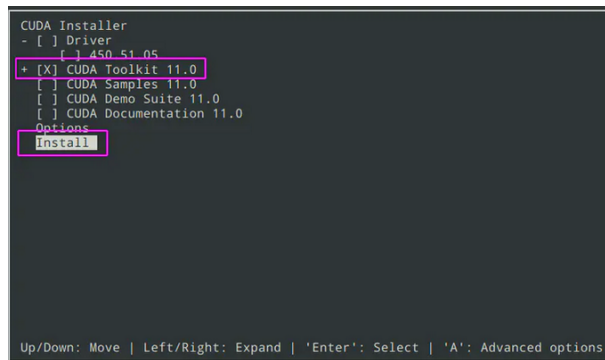
我们在x-server界面中，给CUDA Toolkit赋予执行权限：

```
1 | chmod +x CUDA程序.run
```

随后运行安装包：

```
1 | sudo ./CUDA程序.run
```

注意在开始安装后，会让我们选择要安装的组件，如下图（网图，除了版本信息外没有区别）所示。其中除了CUDA Toolkit外都无需勾选，**Driver必须不勾选**，其他随意，之后Install即可。



随后等待安装，完成后会有安装总结。总结里面会有两行提示你修改PATH变量和LD\_LIBRARY\_PATH变量，将这部分内容进行记录。

## 3.4 配置环境变量

回到图形化界面，在终端中运行如下命令：

```
1 | sudo nano ~/.bashrc
```

我们在文件结尾加入如下命令：

```
1 | export CUDA_HOME=/usr/local/cuda-11.8
2 | export LD_LIBRARY_PATH=${CUDA_HOME}/lib64
3 | export PATH=${CUDA_HOME}/bin:${PATH}
```

随后更新环境变量并重启PC：

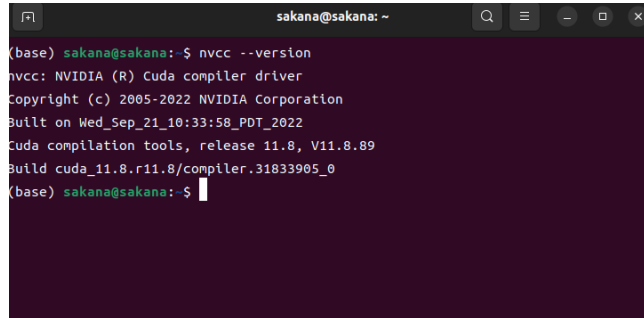
```
1 source ~/.bashrc
2 sudo reboot
```

## 3.5 验证

我们在终端中运行如下命令：

```
1 nvcc --version
```

若出现下图所示结果，则说明安装成功，至此CUDA Toolkit的安装全部完成。



```
(base) sakana@sakana:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
(base) sakana@sakana:~$
```

## 4 安装CUDNN

NVIDIA CUDA深度神经网络库 (cuDNN) 是一个 GPU 加速的深度神经网络基元库，能够以高度优化的方式实现标准例程（如前向和反向卷积、池化层、归一化和激活层）。

CUDNN更多的是对CUDA的一个补充，所以CUDNN的安装相对简单，我们从NV官网下载即可，这里要注意CUDNN对应的CUDA版本。

# cuDNN Archive

**NVIDIA cuDNN** is a GPU-accelerated library of primitives for deep neural networks.

[Download cuDNN v8.8.1 \(March 8th, 2023\), for CUDA 12.x](#)

[Download cuDNN v8.8.1 \(March 8th, 2023\), for CUDA 11.x](#)

### Local Installers for Windows and Linux, Ubuntu(x86\_64, armsbsa)

[Local Installer for Windows \(Zip\)](#)

[Local Installer for Linux x86\\_64 \(Tar\)](#)

[Local Installer for Linux PPC \(Tar\)](#)

[Local Installer for Linux SBSA \(Tar\)](#)

[Local Installer for Debian 11 \(Deb\)](#)

[Local Installer for Ubuntu18.04 x86\\_64 \(Deb\)](#)

[Local Installer for Ubuntu20.04 x86\\_64 \(Deb\)](#)

[Local Installer for Ubuntu22.04 x86\\_64 \(Deb\)](#)

[Local Installer for Ubuntu20.04 aarch64sbsa \(Deb\)](#)

[Local Installer for Ubuntu22.04 aarch64sbsa \(Deb\)](#)

[Local Installer for Ubuntu20.04 cross-sbsa \(Deb\)](#)

[Local Installer for Ubuntu22.04 cross-sbsa \(Deb\)](#)

### Local Installers for Red Hat (x86\_64, armsbsa, Power architecture)

[Local Installer for RedHat/Centos 7.1 x86\\_64 \(RPM\)](#)

这里我们选择针对Ubuntu22.04的64位机版本，下载.deb文件。.deb文件的好处在于可以直接通过Ubuntu自带的安装器安装，右键选择其他程序打开后进行安装即可。

## 5 安装Anaconda

大部分都推荐装Miniconda，比较轻量化，我个人还是喜欢Anaconda多一点

首先，访问[Anaconda官网](#)，直接点击下载即可。随后在下载目录中打开终端,运行如下代码：

```
1 | sudo sh Anaconda程序.sh
```

等待安装后即可。

## 6 安装Pytorch

我看的书使用的是Pytorch，所以此处以Pytorch为例

首先，访问[Pytorch官网](#)，选择对应的选项，查找命令。

### START LOCALLY

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also [install previous versions of PyTorch](#). Note that LibTorch is only available for C++.

PyTorch Build	Stable (2.0.0)		Preview (Nightly)	
Your OS	Linux		Mac	Windows
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 11.7	CUDA 11.8	ROCm 5.4.2	CPU
Run this Command:	conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia			

**NOTE:** PyTorch LTS has been deprecated. For more information, see [this blog](#).

随后，我们在终端中（如果要安装到特定的环境，先conda activate那个环境）运行给出的命令。

由于要下载的包很多且部分容量较大，需要耐心等待一段时间。

安装完成后，在终端中运行如下命令：

```
1 | (dlsite) sakana@sakana:~$ python
2 | Python 3.9.16 (main, Mar 8 2023, 14:00:05)
3 | [GCC 11.2.0] :: Anaconda, Inc. on linux
4 | Type "help", "copyright", "credits" or "license" for more information.
5 | >>> import torch
6 | >>> torch.cuda.is_available()
7 | True
8 | >>> torch.cuda.get_device_name()
9 | 'NVIDIA GeForce RTX 3060 Ti'
10 | >>>
```

出现True则代表Pytorch可以调用你的GPU，并可以正确输出GPU型号，安装成功。

## 7 结语

到此为止，深度学习的环境就基本搭建完毕了。文章有一定省去部分，但都相对简单（安装IDE等），故不做赘述。

Sakannnnna