0、必要说明

务必根据自己的系统、显卡、使用目的等情况，进行修改相关配置，才能保证环境搭建成功。
由于各种各样的原因，环境搭建过程找中会出现各种各样的问题，要有必要的心理准备。
如果出现问题，请使用Baidu/Google进行搜索，尝试解决，一般情况下能解决95%的问题。如果不行，记得重启一下，一般还能解决另外2%的问题。如果还是不行，可以考虑重装一下系统，一般还能解决另外2%的问题。如果仍然不行……那就是这个文档不行[狗头]。

1、禁用nouveau

在安装NVIDIA驱动以前需要禁止系统自带显卡驱动nouveau：可以先通过指令 **lsmod | grep nouveau** 查看nouveau驱动的启用情况，如果有输出表示nouveau驱动正在工作，如果没有内容输出则表示已经禁用了nouveau。下面对禁用nouveau进行配置。
首先，打开禁用设置文件

```
sudo gedit /etc/modprobe.d/blacklist-nouveau.conf
```

然后，将下面的内容复制粘贴到打开的blacklist-nouveau.conf中

```
blacklist nouveau
options nouveau modeset=0
```

保存并退出文件。使用命令更新配置，使之生效。

```
sudo update-initramfs -u
```

2、安装显卡驱动

首先，使用ubuntu-drivers devices查看推荐驱动版本（PS：不同的机器、显卡、系统会出现推荐结果不同的情况）。

```
(base) vcis1@vcis1:~$ ubuntu-drivers devices
== /sys/devices/pci0000:b2/0000:b2:00.0/0000:b3:00.0 ==
modalias : pci:v000010DEd00002204sv00001458sd00004043bc03sc00i00
vendor   : NVIDIA Corporation
manual install: True
driver   : nvidia-driver-510-server - distro non-free recommended
driver   : nvidia-driver-470-server - distro non-free
driver   : nvidia-driver-510 - distro non-free
driver   : nvidia-driver-470 - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
```

通过上述命令也可能返回为空什么都不显示，添加官方ppa的源，在更新以下就好了。

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt-get update
```

然后，再安装推荐的驱动版本。

```
(base) vcis1@vcis1:~$ sudo apt install nvidia-driver-510
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  clamav-base clamav-freshclam gyp libclamav9 libhttp-parser2.7.1 libjs-async li
  libtfm1 libuv1-dev node-abbrev node-ansi node-ansi-color-table node-archy node
  node-builtin-modules node-combined-stream node-concat-map node-cookie-jar node
  node-fstream node-fstream-ignore node-github-url-from-git node-glob node-grace
  node-is-builtin-module node-isexe node-json-stringify-safe node-lockfile node-
  node-node-uuid node-nopt node-normalize-package-data node-npmlog node-once nod
  node-read-package-json node-request node-retry node-rimraf node-semver node-sh
  node-spdx-license-ids node-tar node-tunnel-agent node-underscore node-validate
  nvidia-container-runtime shim
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  gcc-8-base:i386 libatomic1:i386 libbsd0:i386 libc6:i386 libdrm-amdgpu1:i386 li
  libedit2:i386 libelf1:i386 libexpat1 libexpat1:i386 libexpat1-dev libffi6:i386
  libglvnd0:i386 libglx-mesa0:i386 libglx0:i386 libllvm10:i386 libnvidia-cfg1-51
  libnvidia-decode-510 libnvidia-decode-510:i386 libnvidia-encode-510 libnvidia-
  libnvidia-fbc1-510:i386 libnvidia-gl-510 libnvidia-gl-510:i386 libpciaccess0:i
  libx11-xcb1:i386 libxau6:i386 libxcb-dri2-0:i386 libxcb-dri3-0:i386 libxcb-glx
  libxdamage1:i386 libxdmcp6:i386 libxext6:i386 libxfixes3:i386 libxnvctrl0 libx
  nvidia-dkms-510 nvidia-kernel-common-510 nvidia-kernel-source-510 nvidia-prime
  xserver-xorg-video-nvidia-510 zlib1g zlib1g:i386 zlib1g-dev
Suggested packages:
```

等待安装完毕后，使用 nvidia_smi 查看显卡和驱动信息。

```
(base) vcis1@vcis1:~$ nvidia-smi
Thu Apr 28 17:02:29 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.67       Driver Version: 460.67       CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| 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  GeForce RTX 3090    Off  | 00000000:65:00.0 Off |                  N/A |
| 34%   45C    P8    32W / 350W |     19MiB / 24265MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  GeForce RTX 3090    Off  | 00000000:B3:00.0 Off |                  N/A |
| 59%   52C    P2   132W / 370W |   2298MiB / 24268MiB |     17%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1250      G   /usr/lib/xorg/Xorg                 14MiB |
|    1   N/A  N/A     31980      C   python                           2295MiB |
+-----------------------------------------------------------------------------+
```

3、安装CUDA

首先，查看nvidia驱动对应的CUDA版本（链接：https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/）。

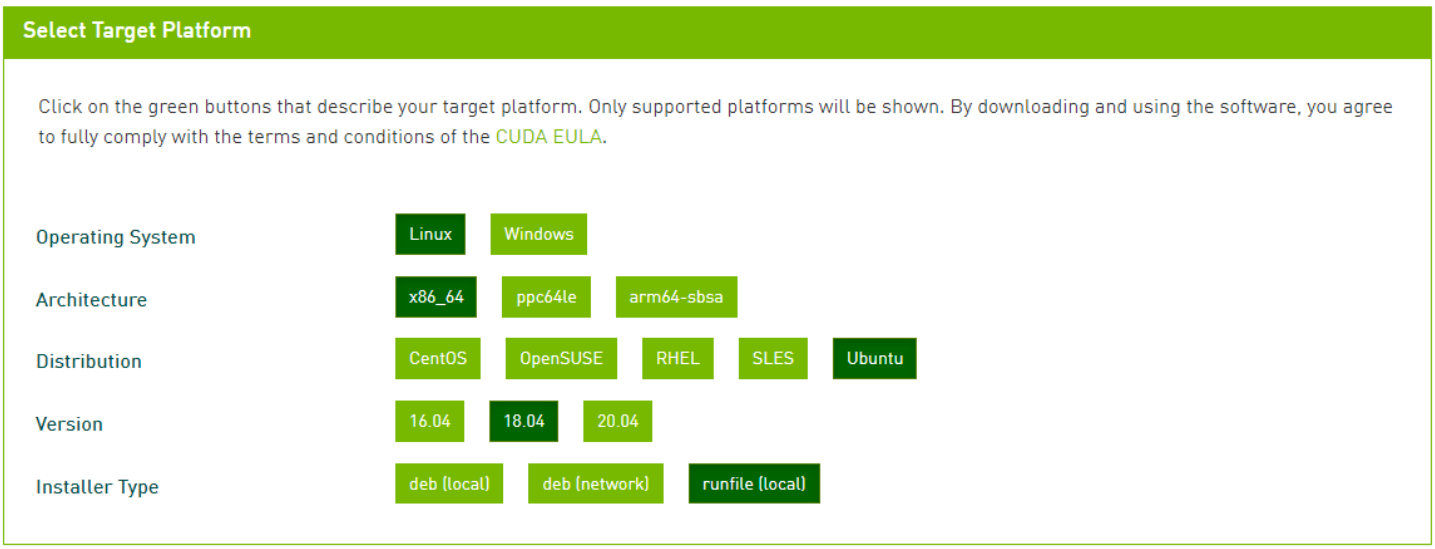
Table 3. CUDA Toolkit and Corresponding Driver Versions

CUDA Toolkit	Toolkit Driver Version	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 11.6 Update 2	>=510.47.03	>=511.65
CUDA 11.6 Update 1	>=510.47.03	>=511.65
CUDA 11.6 GA	>=510.39.01	>=511.23
CUDA 11.5 Update 2	>=495.29.05	>=496.13
CUDA 11.5 Update 1	>=495.29.05	>=496.13
CUDA 11.5 GA	>=495.29.05	>=496.04
CUDA 11.4 Update 4	>=470.82.01	>=472.50
CUDA 11.4 Update 3	>=470.82.01	>=472.50
CUDA 11.4 Update 2	>=470.57.02	>=471.41
CUDA 11.4 Update 1	>=470.57.02	>=471.41
CUDA 11.4.0 GA	>=470.42.01	>=471.11
CUDA 11.3.1 Update 1	>=465.19.01	>=465.89
CUDA 11.3.0 GA	>=465.19.01	>=465.89
CUDA 11.2.2 Update 2	>=460.32.03	>=461.33
CUDA 11.2.1 Update 1	>=460.32.03	>=461.09
CUDA 11.2.0 GA	>=460.27.03	>=460.82
CUDA 11.1.1 Update 1	>=455.32	>=456.81
CUDA 11.1 GA	>=455.23	>=456.38
CUDA 11.0.3 Update 1	>= 450.51.06	>= 451.82
CUDA 11.0.2 GA	>= 450.51.05	>= 451.48
CUDA 11.0.1 RC	>= 450.36.06	>= 451.22
CUDA 10.2.89	>= 440.33	>= 441.22
CUDA 10.1 (10.1.105 general release, and updates)	>= 418.39	>= 418.96
CUDA 10.0.130	>= 410.48	>= 411.31

然后，选择对应版本的CUDA（链接：https://developer.nvidia.com/cuda-toolkit-archive）。

CUDA Toolkit 11.4.0 (June 2021), Versioned Online Documentation
CUDA Toolkit 11.3.1 (May 2021), Versioned Online Documentation
CUDA Toolkit 11.3.0 (April 2021), Versioned Online Documentation
CUDA Toolkit 11.2.2 (March 2021), Versioned Online Documentation
CUDA Toolkit 11.2.1 (February 2021), Versioned Online Documentation
CUDA Toolkit 11.2.0 (December 2020), Versioned Online Documentation
CUDA Toolkit 11.1.1 (October 2020), Versioned Online Documentation
CUDA Toolkit 11.1.0 (September 2020), Versioned Online Documentation
CUDA Toolkit 11.0.3 (August 2020), Versioned Online Documentation
CUDA Toolkit 11.0.2 (July 2020), Versioned Online Documentation
CUDA Toolkit 11.0.1 (June 2020), Versioned Online Documentation
CUDA Toolkit 11.0.0 (March 2020), Versioned Online Documentation
CUDA Toolkit 10.2 (Nov 2019), Versioned Online Documentation
CUDA Toolkit 10.1 update2 (Aug 2019), Versioned Online Documentation
CUDA Toolkit 10.1 update1 (May 2019), Versioned Online Documentation
CUDA Toolkit 10.1 (Feb 2019), Online Documentation
CUDA Toolkit 10.0 (Sept 2018), Online Documentation

根据自己的系统情况，进行下载。



如果使用wget下载困难，将下载链接复制出来，使用其他下载器下载。

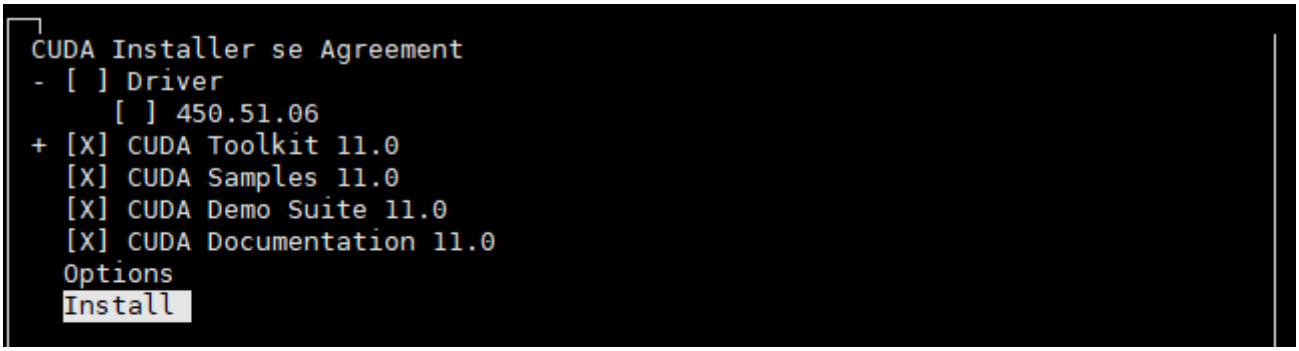
在安装CUDA之前需要首先安装一些相互依赖的库文件。

```
sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev libxi-dev
libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev
```

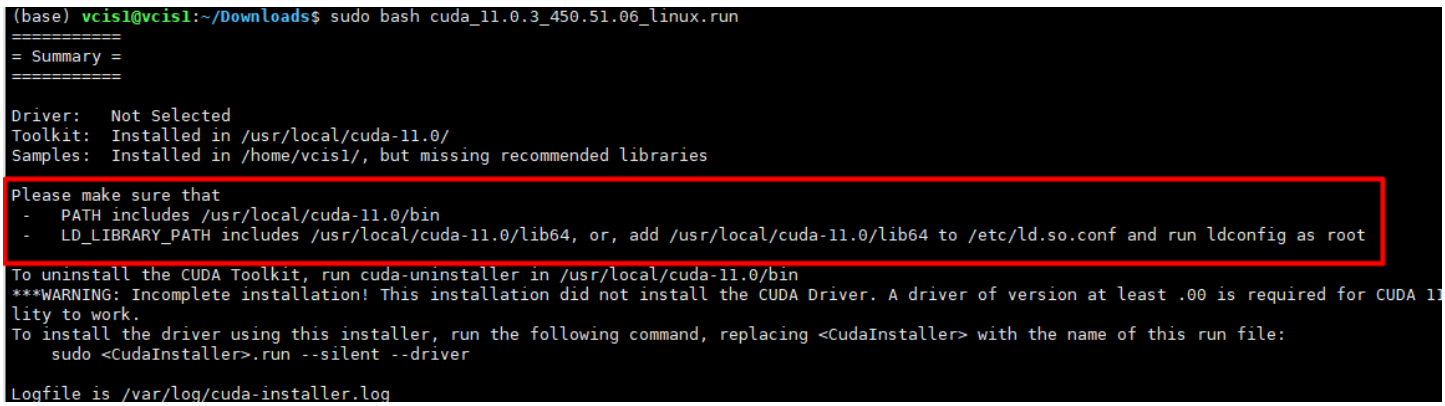
然后，执行以下命令来运行下载的安装包，按照提示进行安装。

```
sudo sh cuda_11.0.3_450.51.06_linux.run
```

注意！ 询问你是否安装Driver的时候的选择no！ 驱动前面已经安装过了，而且此时 安装显卡驱动非常容易出错。 Toolkit必选，其他的随意。



经过漫长的等待之后，输出以下界面代表安装完毕（稍微留意一下红框中的信息）。



然后配置CUDA环境变量，打开bashrc文件，并在文件尾部追加内容。

```
export PATH=/usr/local/cuda-11.0/bin:${PATH}
export LD_LIBRARY_PATH=/usr/local/cuda-11.0/lib64:${LD_LIBRARY_PATH}
export CUDA_HOME=$CUDA_HOME:/usr/local/cuda-11.0/
```

注意！务必修改CUDA版本（下图红框选中的内容），按照CUDA安装完成后输出的内容进行适配。

```
export PATH=/usr/local/cuda-11.0/bin:${PATH}
export LD_LIBRARY_PATH=/usr/local/cuda-11.0/lib64:${LD_LIBRARY_PATH}
```

添加完成后，保存退出文件。

当然还需要 source ~/.bashrc 一下把新添加的环境激活一下。如果激活成功的话继续下一步，如果失败看看是不是自己路径错了通过which或者whereis去查找一下。排除错误。

最后，通过 nvcc -V 判断cuda是否安装成功，如果返回版本信息就是成功了。

```
(base) vcis1@vcis1:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Wed_Jul_22_19:09:09_PDT_2020
Cuda compilation tools, release 11.0, V11.0.221
Build cuda_11.0_bu.TC445_37.28845127_0
```

或者通过编译cuda sample来进行测试

```
cd /usr/local/cuda-11.0/samples/1_Utilities/deviceQuery
sudo make
sudo ./deviceQuery
```

输出以下内容，也表示CUDA安装成功。

（PS:有兴趣的可以了解一下CUDA Driver Version 和 CUDA Runtime Version 的区别）

4、安装cuDNN

首先，下载对应 cuda 版本的 cudnn 压缩包及安装文件（链接：https://developer.nvidia.com/rdp/cudnn-archive）。

Download cuDNN v8.2.1 (June 7th, 2021), for CUDA 11.x

Library for Windows and Linux, Ubuntu(x86_64, armsbsa, PPC architecture)

cuDNN Library for Linux (aarch64sbsa)

cuDNN Library for Linux (x86_64)

cuDNN Library for Linux (PPC)

cuDNN Library for Windows (x86)

cuDNN Runtime Library for Ubuntu20.04 x86_64 (Deb)

cuDNN Developer Library for Ubuntu20.04 x86_64 (Deb)

cuDNN Code Samples and User Guide for Ubuntu20.04 x86_64 (Deb)

cuDNN Runtime Library for Ubuntu20.04 aarch64sbsa (Deb)

cuDNN Developer Library for Ubuntu20.04 aarch64sbsa (Deb)

cuDNN Code Samples and User Guide for Ubuntu20.04 aarch64sbsa (Deb)

cuDNN Cross-compile Library for Ubuntu20.04 aarch64sbsa (Deb)

cuDNN Developer Cross-compile Library for Ubuntu20.04 aarch64sbsa (Deb)

cuDNN Runtime Library for Ubuntu18.04 x86_64 (Deb)

cuDNN Developer Library for Ubuntu18.04 x86_64 (Deb)

cuDNN Code Samples and User Guide for Ubuntu18.04 x86_64 (Deb)

cuDNN Runtime Library for Ubuntu16.04 x86_64 (Deb)

首先，执行解压命令，将下载的cudnn库文件解压。

```
tar -xf cudnn-11.3-linux-x64-v8.2.1.32.tgz
```

```
(base) vcis1@vcis1:~/Downloads/cuda$ ls
include  lib64  NVIDIA_SLA_cuDNN_Support.txt
```

然后，用解压文件覆盖CUDA中头文件和库，并生成新的软链（注意：根据系统及CUDA版本，修改相关路径及文件名）。

```
sudo cp cuda/include/cudnn.h /usr/local/cuda-11.0/include
sudo cp cuda/lib64/libcudnn* /usr/local/cuda-11.0/lib64
sudo chmod a+r /usr/local/cuda-11.0/include/cudnn.h
sudo chmod a+r /usr/local/cuda-11.0/lib64/libcudnn*
```

拷贝完成后，我们可以使用如下的命令查看cuDNN的信息：

```
cat /usr/local/cuda-11.0/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
```

```
(base) vcisl@vcisl:~$ cat /usr/local/cuda-11.0/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 8
#define CUDNN_MINOR 2
#define CUDNN_PATCHLEVEL 1
--
#define CUDNN_VERSION (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)

#endif /* CUDNN_VERSION_H */
```

然后对cuDNN进行检测，安装下载的安装包。

```
sudo dpkg -i libcudnn8_8.2.1.32-1+cuda11.3_amd64.deb
sudo dpkg -i libcudnn8-dev_8.2.1.32-1+cuda11.3_amd64.deb
sudo dpkg -i libcudnn8-samples_8.2.1.32-1+cuda11.3_amd64.deb
```

cuDNN的测试文件会自动安装在系统的 **/usr/src/cudnn_samples_v8/** 文件夹下。

```
(base) vcisl@vcisl:/usr/src/cudnn_samples_v8$ ls
conv_sample  mnistCUDNN  multiHeadAttention  RNN  RNN_v8.0  samples_common.mk
```

进行测试之前首先安装必要的库：

```
sudo apt-get install libfreeimage3 libfreeimage-dev
```

然后，进入mnistCUDNN下，执行命令

```
 cd mnistCUDNN/
sudo make clean && sudo make
```

```
(base) vcisl@vcisl:/usr/src/cudnn_samples_v8/mnistCUDNN$ sudo make clean && sudo make
rm -rf *o
rm -rf mnistCUDNN
CUDA_VERSION is 11000
Linking agains cublasLt = true
CUDA VERSION: 11000
TARGET ARCH: x86_64
HOST_ARCH: x86_64
TARGET OS: linux
SMS: 35 50 53 60 61 62 70 72 75 80
/usr/local/cuda/bin/nvcc -ccbin g++ -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64l
pute_35,code=sm_35 -gencode arch=compute_50,code=sm_50 -gencode arch=compute_53,code=sm_53 -ge
=sm_61 -gencode arch=compute_62,code=sm_62 -gencode arch=compute_70,code=sm_70 -gencode arch=c
ode arch=compute_80,code=sm_80 -gencode arch=compute_80,code=compute_80 -o fp16_dev.o -c fp16_
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' arch
se (Use -Wno-deprecated-gpu-targets to suppress warning).
g++ -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64le-linux/include -IFreeImage/incl
g++ -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64le-linux/include -IFreeImage/incl
/usr/local/cuda/bin/nvcc -ccbin g++   -m64      -gencode arch=compute_35,code=sm_35 -gencode a
-gencode arch=compute_60,code=sm_60 -gencode arch=compute_61,code=sm_61 -gencode arch=compute_
h=compute_72,code=sm_72 -gencode arch=compute_75,code=sm_75 -gencode arch=compute_80,code=sm_8
dev.o fp16_emu.o mnistCUDNN.o -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64le-linu
ocal/cuda/targets/ppc64le-linux/lib -lcublasLt -LFreeImage/lib/linux/x86_64 -LFreeImage/lib/li
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' arch
se (Use -Wno-deprecated-gpu-targets to suppress warning).
```

然后执行 ./mnistCUDNN，出现以下结果，代表cuDNN正常。

```
Testing cudnnGetConvolutionForwardAlgorithm_v7 ...
^^^^ CUDNN_STATUS_SUCCESS for Algo 1: -1.000000 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 0: -1.000000 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 2: -1.000000 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 4: -1.000000 time requiring 2450080 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 5: -1.000000 time requiring 4656640 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 7: -1.000000 time requiring 1433120 memory
^^^^ CUDNN_STATUS_NOT_SUPPORTED for Algo 6: -1.000000 time requiring 0 memory
^^^^ CUDNN_STATUS_NOT_SUPPORTED for Algo 3: -1.000000 time requiring 0 memory
Testing cudnnFindConvolutionForwardAlgorithm ...
^^^^ CUDNN_STATUS_SUCCESS for Algo 0: 0.038912 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 1: 0.039936 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 4: 0.041984 time requiring 2450080 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 5: 0.049152 time requiring 4656640 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 7: 0.054272 time requiring 1433120 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 2: 0.070656 time requiring 0 memory
^^^^ CUDNN_STATUS_NOT_SUPPORTED for Algo 6: -1.000000 time requiring 0 memory
^^^^ CUDNN_STATUS_NOT_SUPPORTED for Algo 3: -1.000000 time requiring 0 memory
Resulting weights from Softmax:
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000714 0.0000000 0.0000000 0.0000000 0.0000000
Loading image data/five_28x28.pgm
Performing forward propagation ...
Resulting weights from Softmax:
0.0000000 0.0000008 0.0000000 0.0000002 0.0000000 1.0000000 0.0000154 0.0000000 0.0000012 0.0000006

Result of classification: 1 3 5

Test passed!
```

如果遇到以下类似问题

```
/sbin/ldconfig.real: /usr/local/cuda-11.0/targets/x86_64-linux/lib/libcudnn_cnn_infer.so.8 is not a symbolic link

/sbin/ldconfig.real: /usr/local/cuda-11.0/targets/x86_64-linux/lib/libcudnn_adv_infer.so.8 is not a symbolic link

/sbin/ldconfig.real: /usr/local/cuda-11.0/targets/x86_64-linux/lib/libcudnn.so.8 is not a symbolic link

/sbin/ldconfig.real: /usr/local/cuda-11.0/targets/x86_64-linux/lib/libcudnn_adv_train.so.8 is not a symbolic link

/sbin/ldconfig.real: /usr/local/cuda-11.0/targets/x86_64-linux/lib/libcudnn_ops_infer.so.8 is not a symbolic link

/sbin/ldconfig.real: /usr/local/cuda-11.0/targets/x86_64-linux/lib/libcudnn_ops_train.so.8 is not a symbolic link

/sbin/ldconfig.real: /usr/local/cuda-11.0/targets/x86_64-linux/lib/libcudnn_cnn_train.so.8 is not a symbolic link
```

可以执行以下命令，建立相应文件的软链接，可解决问题。

```

cd /usr/local/cuda-11.0/lib64/

# 删除原有动态文件
sudo rm -rf libcudnn.so libcudnn.so.8  libcudnn_cnn_train.so.8  libcudnn_cnn_infer.so.8 libcudnn_adv_infer.so.8 libcudnn_adv_train.so.8 libcudnn_ops_infer.so.8 libcudnn_ops_train.so.8
# 当前是8.2.1版本，生成软链接
```

```
sudo ln -s libcudnn.so.8.2.1 libcudnn.so.8
sudo ln -s libcudnn_cnn_train.so.8.2.1 libcudnn_cnn_train.so.8
sudo ln -s libcudnn_cnn_infer.so.8.2.1 libcudnn_cnn_infer.so.8
sudo ln -s libcudnn_adv_infer.so.8.2.1 libcudnn_adv_infer.so.8
sudo ln -s libcudnn_adv_train.so.8.2.1 libcudnn_adv_train.so.8

sudo ln -s libcudnn_ops_infer.so.8.2.1 libcudnn_ops_infer.so.8
sudo ln -s libcudnn_ops_train.so.8.2.1 libcudnn_ops_train.so.8
sudo ln -s libcudnn.so.8 libcudnn.so
```