in cuda 8 what is nv smi archtecture number -13rdparty/stb/include -DGPU -I/usr/local/cuda/include/ -Wall -Wfatal-errors -Wno-unusedresult -Wno-unknown-pragmas -fPIC -Ofast -DGPU -c ./src/conv_lstm_layer.c -o obj/conv_lstm_layer.o gcc -linclude/ -I3rdparty/stb/include -DGPU -I/usr/local/cuda/include/ -Wall-Wfatal-errors-Wno-unused-result-Wno-unknown-pragmasfPIC -Ofast -DGPU -c ./src/scale_channels_layer.c -o obj/scale_channels_layer.o gcc -linclude/ -I3rdparty/stb/include -DGPU -l/usr/local/cuda/include/ -Wall-Wfatal-errors-Wno-unused-result-Wno-unknown-pragmasfPIC -Ofast -DGPU -c ./src/sam_layer.c -o obj/sam_layer.o nvcc -gencode arch=compute_30,code=sm_30 -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code= [sm_50,compute_50] -gencode arch=compute_52,code= [sm_52,compute_52] -gencode arch=compute_61,code= [sm_61,compute_61] -linclude/ -l3rdparty/stb/include -DGPU -I/usr/local/cuda/include/ --compiler-options "-Wall -Wfatal-errors -Wno-unused-result -Wno-unknown-pragmas -fPIC -Ofast -DGPU" -c ./src/convolutional kernels.cu -o obj/convolutional kernels.o In file included from /usr/local/cuda/include/cuda_runtime.h:78:0, from <command-line>:0: /usr/local/cuda/include/host_config.h:119:2: error: #error -unsupported GNU version! gcc versions later than 5 are not supported! #error -- unsupported GNU version! gcc versions later than 5 are not supported!

compilation terminated due to -Wfatal-errors. Makefile:177: recipe for target 'obj/convolutional_kernels.o' failed make: *** [obj/convolutional_kernels.o] Error 1



You're encountering two distinct issues while compiling CUDA code:

1. Your actual error: GCC version too high for CUDA 8

You're using CUDA 8, which is not compatible with GCC versions later than 5.



You must use an older GCC version, such as 4.9 or 5.4. Here's how you can install and switch to GCC 5:

bash		Сору	to Edit
sudo apt install gcc-5 g++-5 sudo updat /usr/bin/gcc-5 50 sudo update-alternati			
To switch:			
bash		Сору	* Edit
sudo update-alternativesconfig gcc s	sudo update-alternativesco	nfig g++	

Verify: Сору bash */ Edit gcc --version

2. Your question: What SM architectures does CUDA 8 support?

CUDA 8 supports up to:

- sm_61 (Pascal e.g., GTX 1080)
- It officially supports:
 - sm_30 , sm_35 , sm_50 , sm_52 , sm_60 , sm_61

So your nvcc flags are mostly okay:

```
-gencode arch=compute_30,code=sm_30 \ -gencode arch=compute_35,code=sm_35 \ -gencode arch=compute_50,code=[sm_50,compute_50] \ -gencode arch=compute_52,code=[sm_52,compute_52] \ -gencode arch=compute_61,code=[sm_61,compute_61]
```

That's fine as long as you're targeting Pascal and earlier architectures. If you're on a newer GPU (e.g., Turing, Ampere), CUDA 8 will **not work**.

Summary:

- Downgrade your GCC version to 5.x
- CUDA 8 supports up to sm_61
- Use newer CUDA versions if you're targeting GPUs beyond Pascal (GTX 10xx)

Would you like help installing a newer CUDA version that supports your current GPU and compiler?



gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0 Copyright (C) 2017 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A

Since you're on GCC 7.5, and CUDA 8 only supports GCC ≤ 5, you need to install and switch to GCC 5.x (like GCC 5.4) on your Ubuntu 18.04 system. Here's a step-by-step guide to downgrade GCC temporarily without affecting your system:

PARTICULAR PURPOSE. downgrade





Step 2: Configure alternatives

This lets you easily switch between multiple GCC versions.



CUDA 8 GCC Compatibility Step 3: Switch to GCC 5 * Edit hash Copy sudo update-alternatives --config gcc # Choose gcc-5 from the list sudo updatealternatives --config g++ # Choose g++-5 from the list Step 4: Verify Copy */ Edit gcc --version g++ --version You should see something like: Copy * Edit SCSS gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 Step 5: Recompile your CUDA 8 project Copy * Edit hash make clean make Let me know if you'd like a script that automatically switches to GCC 5, compiles, and then restores GCC 7. 日の句のかのいく \$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 50 update-alternatives: error: alternative path /usr/bin/gcc-5 doesn't exist (base) sakthees@sakthees-monk:~/Videos/YOLOV4-C-official-AlexeyAB-master\$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 50 update-alternatives: error: alternative path /usr/bin/g++-5 doesn't (base) sakthees@sakthees-monk:~/Videos/YOLOV4-C-official-AlexeyAB-master\$ The error means that GCC 5 and G++ 5 are not yet installed on your system — the paths /usr/bin/gcc-5 and /usr/bin/g++-5 don't exist. Let's install them first. Step-by-step Fix: 1. Install GCC 5 and G++ 5:

This will install them at /usr/bin/gcc-5 and /usr/bin/g++-5.

sudo apt update sudo apt install gcc-5 g++-5

bash

Copy

* Edit

2. Verify the install:

ls /usr/bin/gcc-5 ls /usr/bin/g++-5

If those files now exist, you're good to continue.

3. Then set up alternatives again:

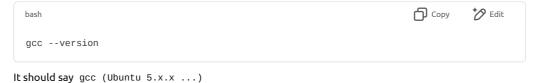


4. Switch to GCC 5:



Choose option for gcc-5 and g++-5.

5. Confirm version:



Let me know if you'd like to restore GCC 7 automatically after compilation.

日内の中ではく

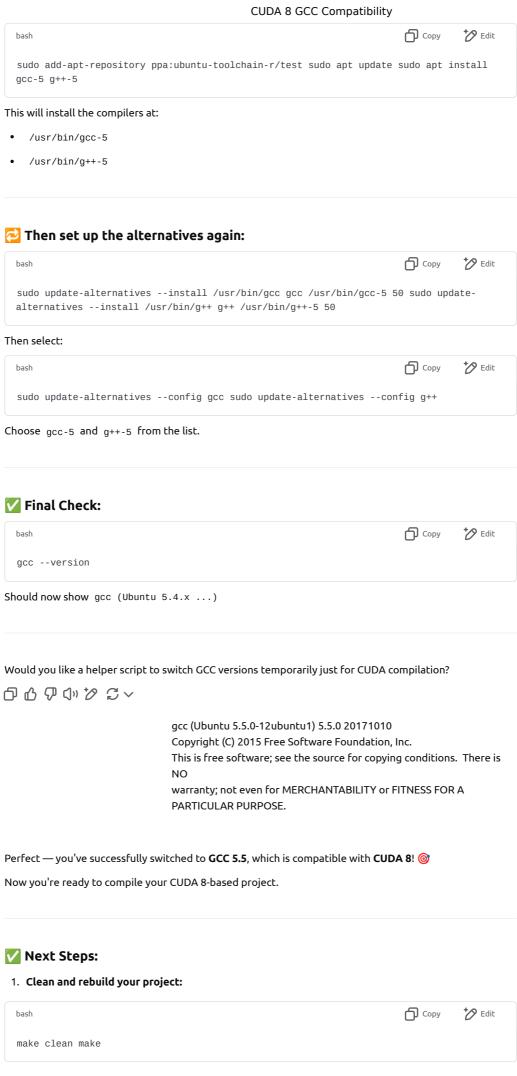
s@sakthees-monk:/bin\$ ls /usr/bin/gcc-5 ls: cannot access '/usr/bin/gcc-5': No such file or directory (base) sakthees@sakthees-monk:/bin\$ ls /usr/bin/g++-5 ls: cannot access '/usr/bin/g++-5': No such file or directory (base) sakthees@sakthees-monk:/bin\$

Since /usr/bin/gcc-5 and /usr/bin/g++-5 don't exist, it means **GCC 5 is not yet installed** on your system.

Let's install it manually on Ubuntu 18.04:



Run this:



If you're still in the YOLOV4-C-official-AlexeyAB-master directory, this will attempt to recompile with your new compiler setup.

2. If you're using nvcc directly, ensure your architecture flags are within CUDA 8's limits:



Optional: Switch back to GCC 7 after build

If other apps depend on newer GCC:

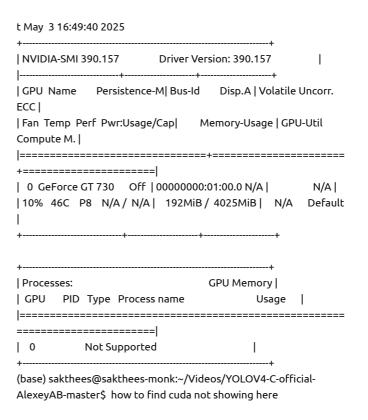
```
bash

Sudo update-alternatives --config gcc sudo update-alternatives --config g++
```

Then choose gcc-7 and g++-7 again from the menu.

Would you like a shell script that automatically switches GCC versions for the build process and then switches back?





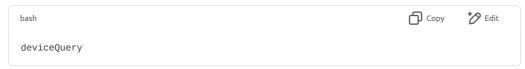
It seems that you have a **GeForce GT 730**, which supports **CUDA** (but might be limited in performance). However, if **CUDA is not showing** or is not being detected correctly in your setup, here are a few things you can check:

1. Check CUDA Installation:

Run the following command to check if CUDA is installed:

-		
bash	Сору	* Edit
nvccversion		
If CUDA is installed, it should return something like:		
arduino	Сору	* Edit
Cuda compilation tools, release 8.0, V8.0.61		
If the command is not found, CUDA may not be installed correctly.		
☑ 2. Check CUDA Device Query:		

Run the CUDA device query tool to check if your GPU is being recognized by CUDA:



This tool is typically found in the extras/demo_suite/ directory of your CUDA installation.

If deviceQuery doesn't run, you may need to ensure that CUDA is properly installed and configured.

3. Ensure Correct Driver Version for Your GPU:

Your NVIDIA driver version 390.157 is a bit older and might not fully support CUDA 8 with all modern

• GeForce GT 730 is compatible with CUDA 3.0 and up, but you might want to consider updating your NVIDIA driver to a version that supports newer CUDA versions, especially if you plan to use advanced CUDA features.

To update the driver:

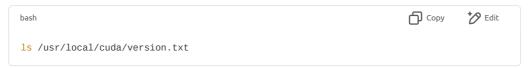


4. Check for CUDA Toolkit Version Compatibility:

Ensure that your CUDA toolkit version matches the driver version and is compatible with your GPU. CUDA 8 is old, and you may want to upgrade to a newer version of the toolkit, as newer CUDA versions provide better support for modern GPUs.

5. Check if GPU is Detected by nvidia-smi:

If nvidia-smi is showing your GPU but CUDA isn't recognized, it might indicate an issue with your CUDA installation. You can check the installed CUDA version with:



It should display the installed version.

6. Test a Simple CUDA Program:

You can also compile and run a simple CUDA program to see if your setup works. Create a file called simple_cuda.cu with the following contents:

```
#include <iostream> #include <cuda_runtime.h> __global__ void hello_cuda() {
printf("Hello, CUDA!\n"); } int main() { hello_cuda<<<1, 1>>>();
cudaDeviceSynchronize(); return 0; }
```

Then compile it using nvcc:

```
bash

nvcc simple_cuda.cu -o simple_cuda ./simple_cuda
```

If CUDA is properly set up, it should print Hello, CUDA!.

Let me know what steps you need help with or if you find any issues!