

Installing CUDA Toolkit and cuDNN for Deep Learning:

Step 1:

Download Cuda toolkit 10.0

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System

Windows Linux

Architecture

x86_64

Version

10 Server 2019 Server 2016

Installer Type

exe (network) exe (local)

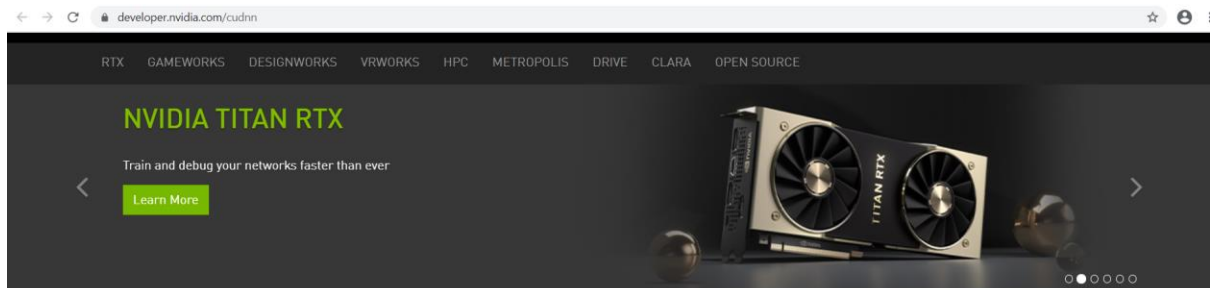
After downloaded the file look like below and the file size comes around 2.0 GB

 cuda_10.0.130_411.31_win10

Install the Cuda toolkit.

Step 2:

Next we need to install Nvidia Cuda Deep neural network library (NVIDIA cuDNN)

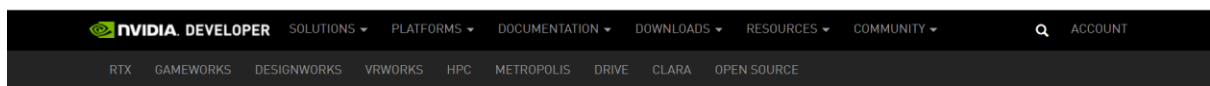


[Home](#) > [Deep Learning](#) > [Deep Learning Software](#) > NVIDIA cuDNN

NVIDIA cuDNN

The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for [deep neural networks](#). cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers.

Deep learning researchers and framework developers worldwide rely on cuDNN for high-performance GPU acceleration. It allows them to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning. cuDNN accelerates widely used deep learning frameworks, including [Caffe2](#), [Chainer](#), [Keras](#), [MATLAB](#), [MxNet](#), [PyTorch](#), and [TensorFlow](#). For access to NVIDIA optimized deep learning framework containers that have cuDNN integrated into frameworks, visit [NVIDIA GPU CLOUD](#) to learn more and get started.

[Download cuDNN >](#)[GTC2020 >](#)[Developer Guide >](#)[Forums >](#)

NVIDIA Developer Program Membership Required

The file or page you have requested requires membership in the NVIDIA Developer Program. Please either log in or join the program to access this material. You can [learn more](#) about the benefits of the NVIDIA Developer Program here.

[Login](#)[Join now](#)

If already an existing user login with your credentials and download the package. If you are a new user create a new account then verify the account and then download the package.

cuDNN Download

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

☒ I Agree To the Terms of the [cuDNN Software License Agreement](#)

Note: Please refer to the [Installation Guide](#) for release prerequisites, including supported GPU architectures and compute capabilities, before downloading.

For more information, refer to the cuDNN Developer Guide, Installation Guide and Release Notes on the [Deep Learning SDK Documentation](#) web page.

Download cuDNN v8.0.0 RC (May 28th, 2020), for CUDA 11.0
Download cuDNN v8.0.0 RC (May 28th, 2020), for CUDA 10.2
Download cuDNN v7.6.5 (November 18th, 2019), for CUDA 10.2
Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 10.1
Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 10.0
Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 9.2
Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 9.0

Library for Windows, Mac, Linux, Ubuntu and RedHat/Centos(x86_64architectures)


[cuDNN Library for Windows 7](#)

[cuDNN Library for Windows 10](#)





[cuDNN Library for Linux](#)

[cuDNN Library for OSX](#)

After the OS version which you have chosen and the download automatically starts and the downloaded file look like below and the size of the file around 286MB





 cudnn-10.0-windows10-x64-v7.6.5.32

Now extract the file and you will be able to see the below mentioned files inside the extracted cuDNN zip file.

-  bin
-  include
-  lib
-  NVIDIA_SLA_cuDNN_Support

Now copy the three folders and paste it into the c:\

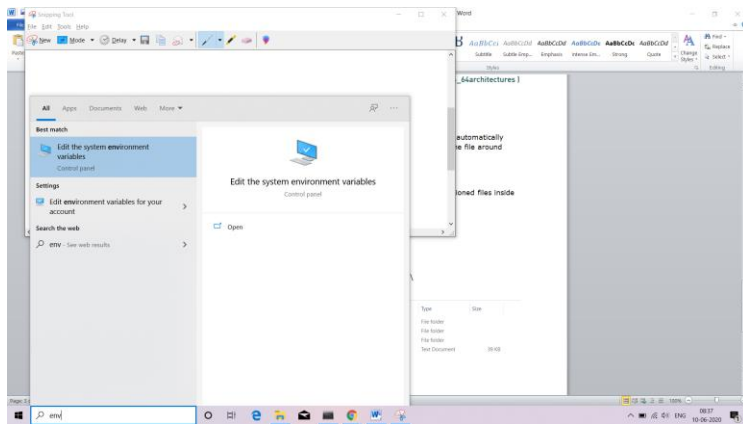
> This PC > OS (C:) > cuda

	Name	Date modified	Type	Size
	bin	09-06-2020 09:32	File folder	
	include	09-06-2020 09:32	File folder	
	lib	09-06-2020 09:32	File folder	
	NVIDIA_SLA_cuDNN_Support	27-10-2019 21:37	Text Document	39 KB

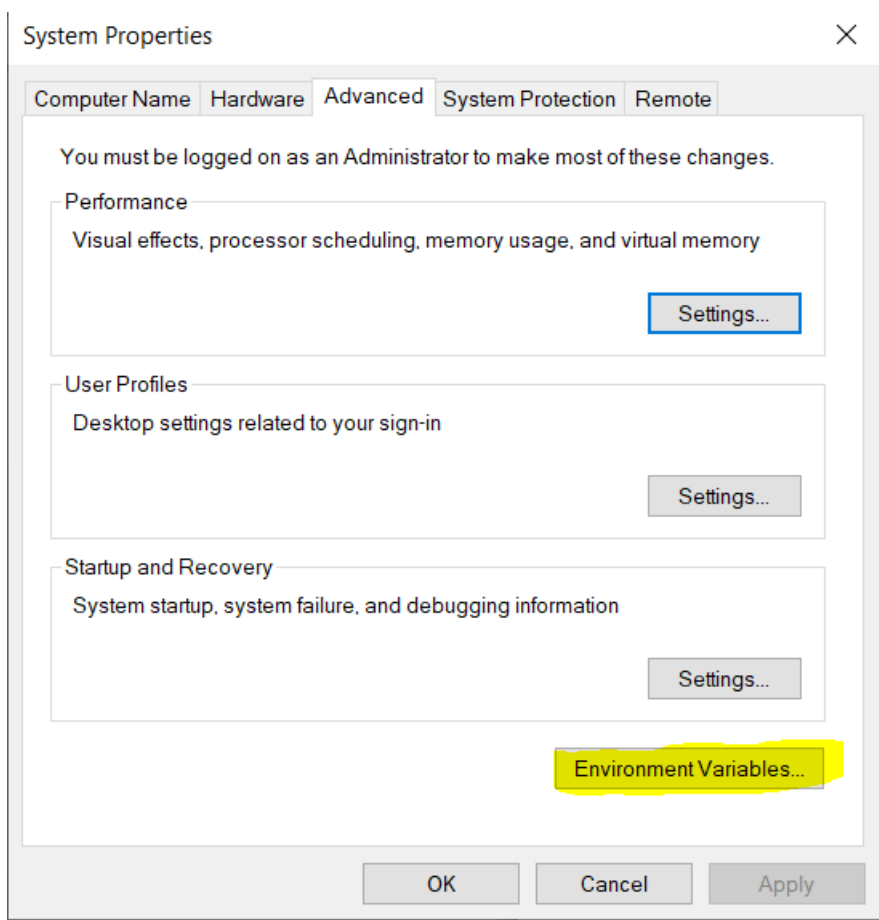
Step 3:

Setup the path variable

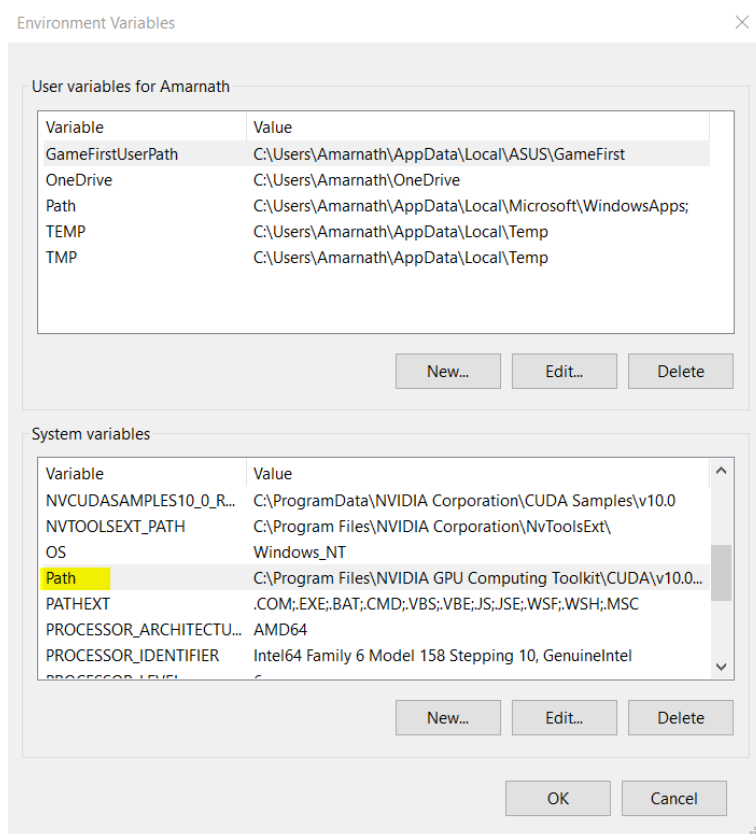
Now go to the environment variable



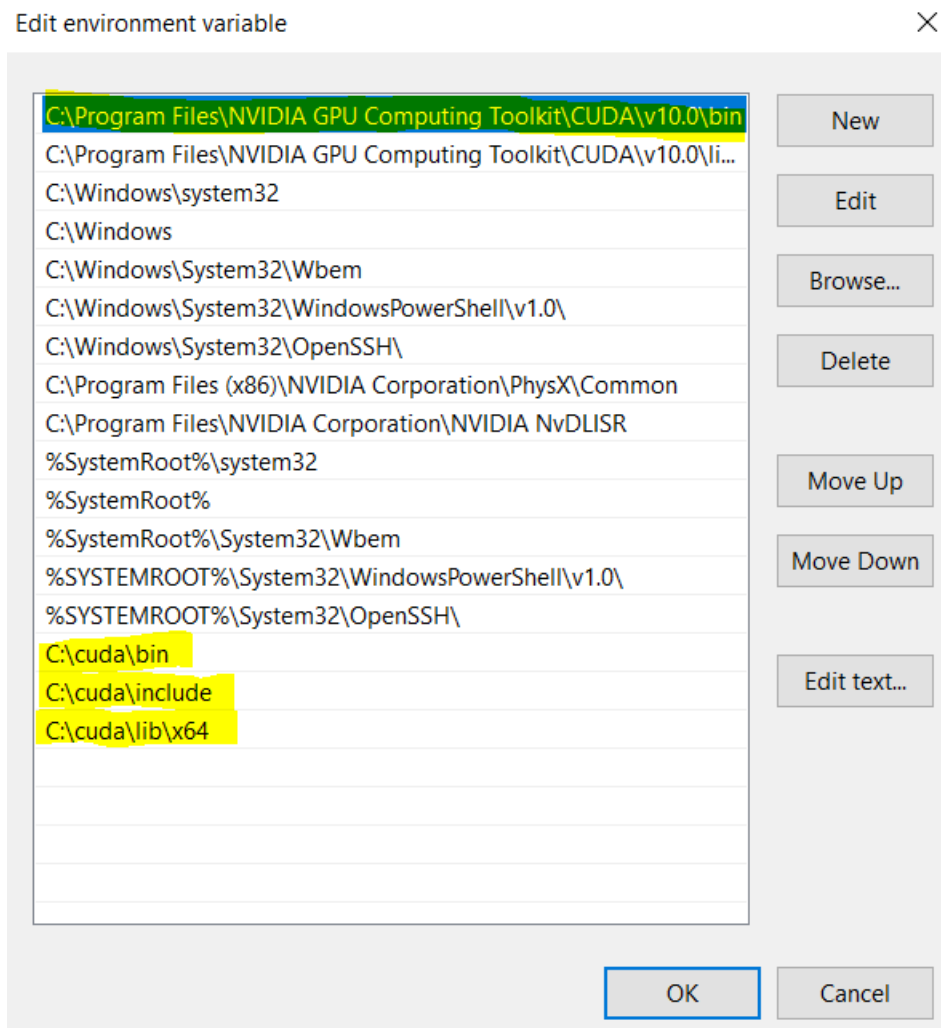
Then click Environment variable



Under system variables double click the path which displayed as below.



Make sure the NVIDIA GPU toolkit automatically updated in the environment variable and we need to add manually the three cuDNN library (Bin, Include, lib x64 path) into the environment variable



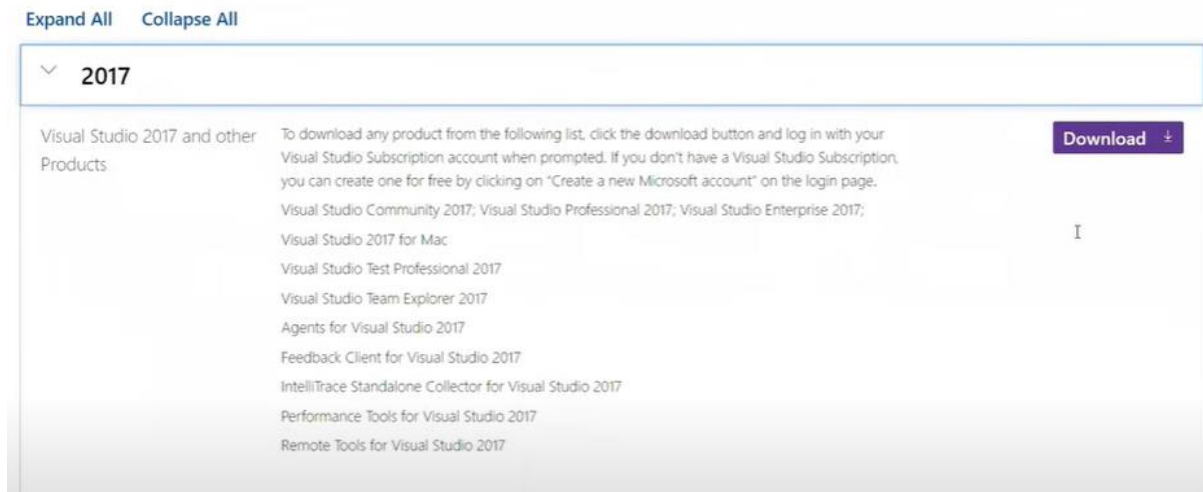
Then click ok after adding the cuDNN libraries

Step 4:

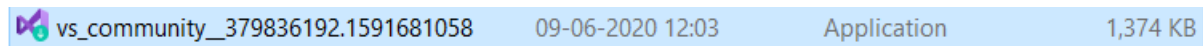
Install Visual studio community version.

To take all the c++ libraries

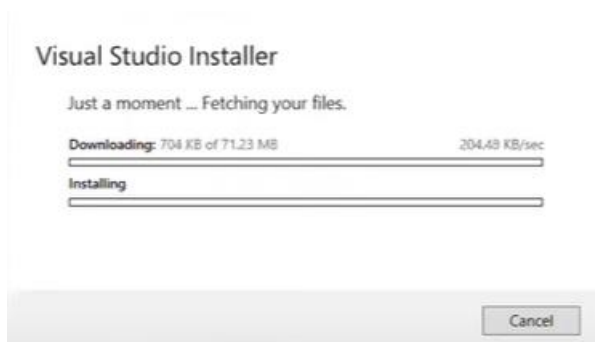




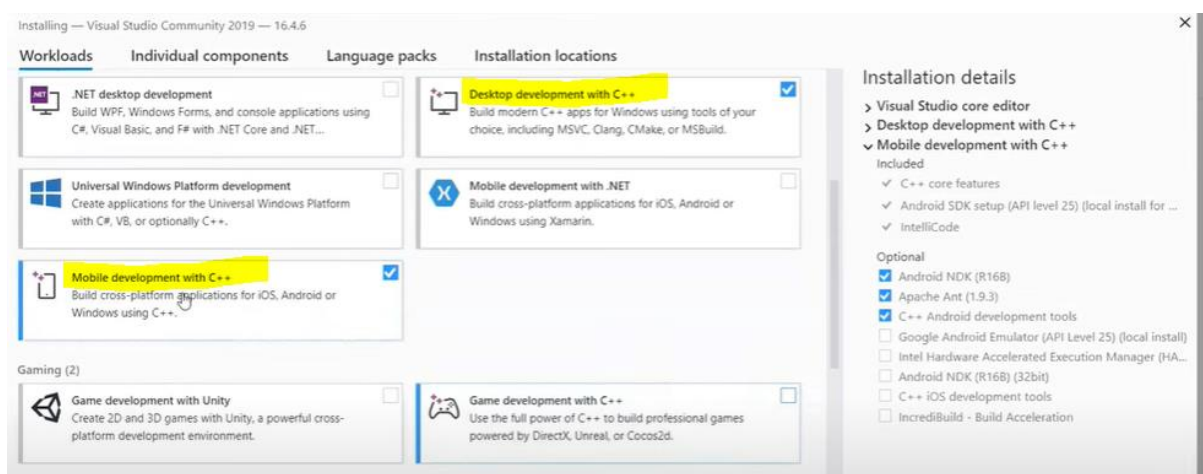
Then file looks like below



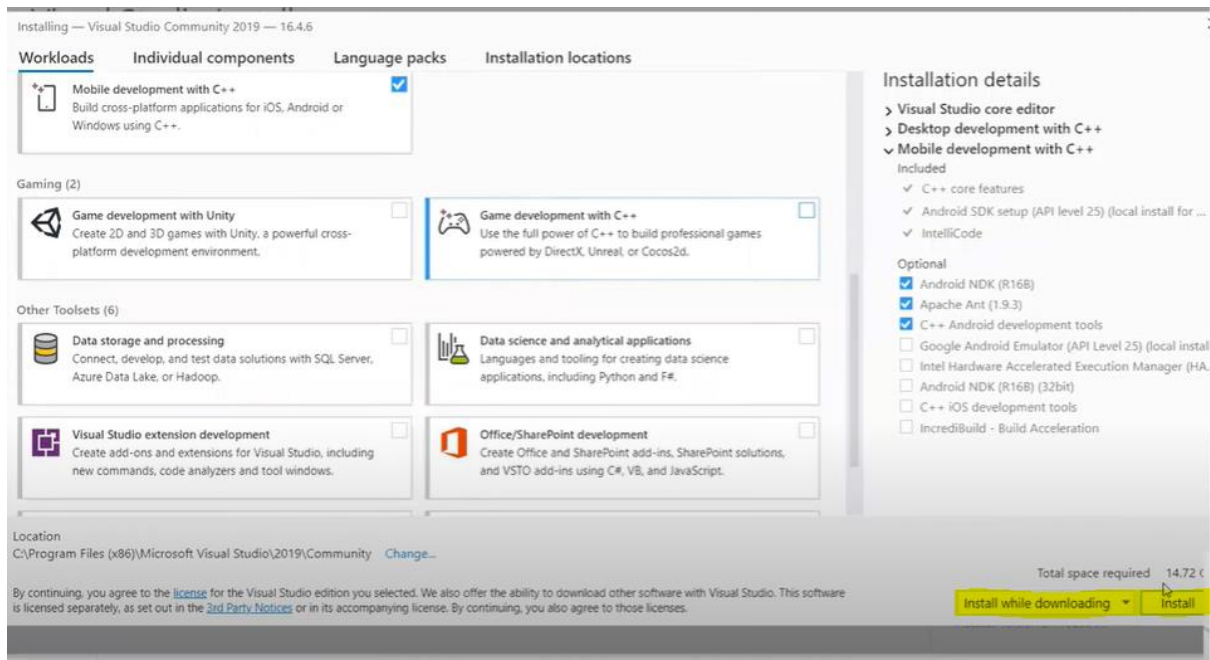
Now install



After some time then you will be able to see the screen like below and make sure our goal is to install the c++ libraries, so we need to select the desktop development c++ and mobile development with c++



To install this it requires around 14 GB space to install



Then click the desire option to install.

Step 5:

Create a new environment variable

Open Anaconda prompt

```
(base) C:\Users\Amarnath>conda create -n GPUenv python=3.6
```

Make sure the Python version should be only 3.6

```
Downloading and Extracting Packages
wheel-0.34.2 | 66 KB | ##### | 100%
setuptools-47.1.1 | 530 KB | ##### | 100%
python-3.6.10 | 15.9 MB | ##### | 100%
vs2015_runtime-14.16 | 1.2 MB | ##### | 100%
certifi-2020.4.5.1 | 156 KB | ##### | 100%
zlib-1.2.11 | 113 KB | ##### | 100%
winertstore-0.2 | 14 KB | ##### | 100%
pip-20.0.2 | 1.7 MB | ##### | 100%
sqlite-3.31.1 | 804 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Environment created

Now we need to activate the environment

```
(base) C:\Users\Amarnath>conda activate GPUenv
```

After activating the environment then we need to install the tensorflow with GPU in our newly created environment

```
(GPUenv) C:\Users\Amarnath>pip install tensorflow-gpu==2.0.0
```


Here I have installed the tensorflow GPU for the version 2.0 if you want to install tensorflow for gpu version 1.15

```
pip install tensorflow==1.15 # CPU
pip install tensorflow-gpu==1.15 # GPU
```

```
(GPUEnv) C:\Users\Amarnath>pip install tensorflow-gpu==2.0.0
Collecting tensorflow-gpu==2.0.0
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out. (read timeout=15)",)': /packages/d5/cf/83a8b5aa711adcdbe5496618663ed99878b28e843593cb6b2fae6a59181/tensorflow_gpu-2.0.0-cp36-cp36m-win_amd64.whl
  Downloading tensorflow_gpu-2.0.0-cp36-cp36m-win_amd64.whl (285.3 MB)
    | 285.3 MB 16 kB/s
Collecting absl-py>=0.7.0
  Downloading absl-py-0.9.0.tar.gz (104 kB)
    | 104 kB 6.8 MB/s
Requirement already satisfied: wheel>=0.26 in d:\anaconda\envs\gpuenv\lib\site-packages (from tensorflow-gpu==2.0.0) (0.34.2)
Collecting google-pasta>=0.1.6
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
    | 57 kB 128 kB/s
Collecting keras-applications>=1.0.8
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
    | 50 kB 355 kB/s
Collecting keras-preprocessing>=1.0.5
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
    | 42 kB 102 kB/s
Collecting tensorboard<2.1.0,>=2.0.0
  Downloading tensorboard-2.0.2-py3-none-any.whl (3.8 MB)
    | 3.8 MB 156 kB/s
Collecting wrapt>=1.11.1
  Downloading wrapt-1.12.1.tar.gz (27 kB)
Collecting opt-einsum>=2.3.2
```

After successful installation of Tensorflow with GPU in our environment then we need to check whether it is successfully installed or not. In order to do that open python in the command prompt and then type import tensorflow

```
(GPUEnv) C:\Users\Amarnath>python
Python 3.6.10 [Anaconda, Inc.] (default, May 7 2020, 19:46:08) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
2020-06-10 08:05:36.084599: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_100.dll
```

Then you will be see this dll file. This is the dll file which we try to run inside the c:\ drive -- > bin which helps us to communicate with GPU cuda.

This PC > OS (C:) > cuda > bin				
Name		Date modified	Type	Size
cudnn64_7.dll		28-10-2019 03:44	Application extens...	3,76,152 KB

How to check whether our GPU is initialized or not in our system

In order to do

Try to execute below 3 commands

1. import tensorflow as tf
2. print(tf.test.is_gpu_available())

3. `print(tf.test.is_built_with_cuda())`

```
Anaconda Prompt (Anaconda) - python
(GPUenv) C:\Users\Amarnath>import tensorflow as tf
'import' is not recognized as an internal or external command,
operable program or batch file.

(GPUenv) C:\Users\Amarnath>python
Python 3.6.10 [Anaconda, Inc.] (default, May 7 2020, 19:46:08) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import tensorflow as tf
2020-06-10 09:16:38.532306: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_100.dll
>>> print(tf.test.is_gpu_available())
2020-06-10 09:17:38.498622: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2020-06-10 09:17:38.530690: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
2020-06-10 09:17:39.701499: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
name: GeForce GTX 1050 major: 6 minor: 1 memoryClockRate(GHz): 1.493
pciBusID: 0000:01:00.0
2020-06-10 09:17:39.723829: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries are statically linked, skip dlopen check.
2020-06-10 09:17:39.739694: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1746] Adding visible gpu devices: 0
2020-06-10 09:17:41.670501: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1159] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-06-10 09:17:41.686937: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1165]      0
2020-06-10 09:17:41.694879: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 0:  N
2020-06-10 09:17:41.739304: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/device:GPU:0 with 3001 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1050, pci bus id:
0000:01:00.0, compute capability: 6.1)
True
>>> print(tf.test.is_built_with_cuda())
True
>>>
```

Cuda Toolkit: <https://developer.nvidia.com/cuda-10...>

cuDnn: <https://developer.nvidia.com/rdp/cudn...>