

Installing Tensorflow with CUDA, cuDNN and GPU support on Windows 10

Pimp Up your PC for Deep Learning — Part 2



Dr. Joanne Kitson

Follow

Apr 3 · 11 min read

In Part 1 of this series, I discussed how you can upgrade your PC hardware to incorporate a CUDA Toolkit compatible graphics processing card, such as an Nvidia GPU. This Part 2 covers the installation of CUDA, cuDNN and Tensorflow on Windows 10. This article below assumes that you have a CUDA-compatible GPU already installed on your PC; but if you haven't got this already, Part 1 of this series will help you get that hardware set up, ready for these steps.

Step 1: Check the software you will need to install

Assuming that Windows is already installed on your PC, the additional bits of software you will install as part of these steps are:-

- Microsoft Visual Studio
- the NVIDIA CUDA Toolkit
- NVIDIA cuDNN
- Python
- Tensorflow (with GPU support)

. . .

Step 2: Download Visual Studio Express

Visual Studio is a Prerequisite for CUDA Toolkit

Visual studio is required for the installation of Nvidia CUDA Toolkit (this prerequisite is referred to here). If you attempt to download and install CUDA Toolkit for Windows without having first installed Visual Studio, you get the message shown in Fig. 1.

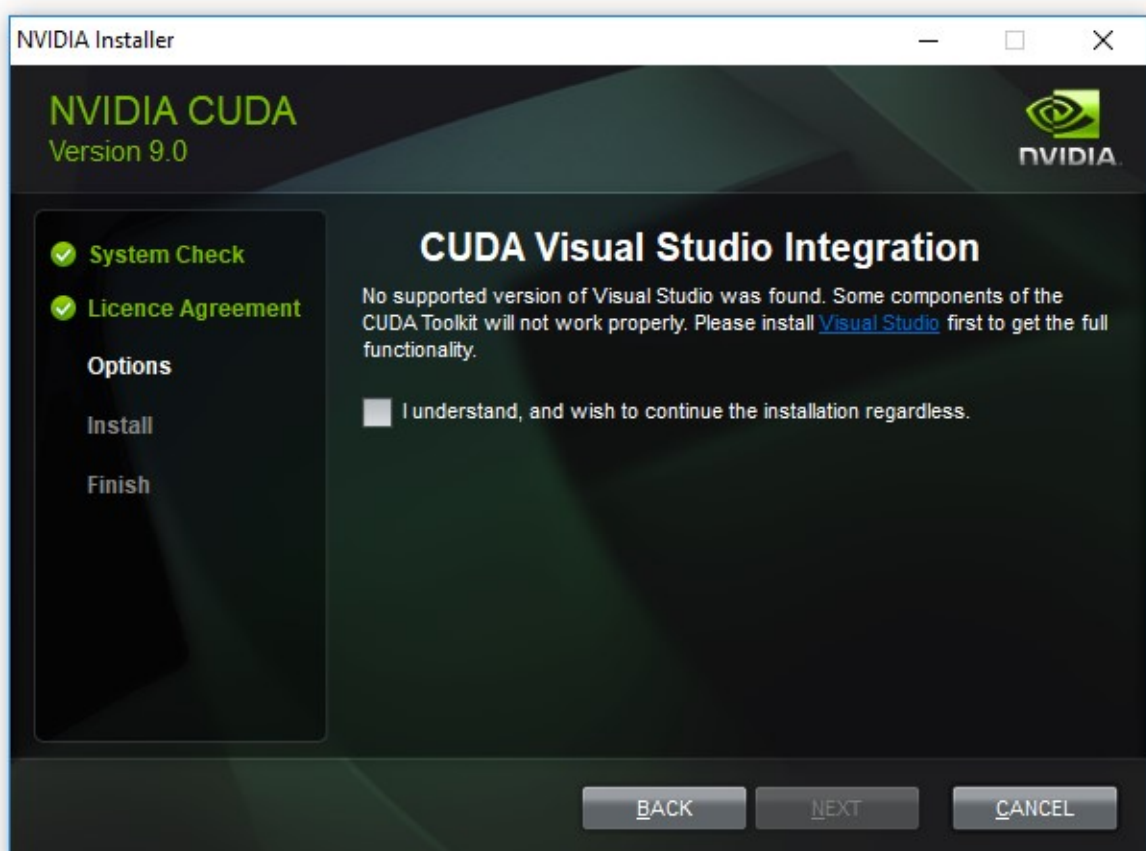


Fig 1: Message when attempting to install CUDA Toolkit without Visual Studio

Selecting and downloading Visual Studio Express

At the time of writing, the most recent version of Visual Studio (which is free) is the Visual Studio Express Community Version 2017, shown in Fig 2. You can get previous versions of Visual Studio for free by joining “Visual Studio Dev Essentials” and then searching for the version of Visual Studio you want.



Fig 2: Visual Studio Community 2017 (free)

Installing Visual Studio Express

Once you have downloaded Visual Studio Express, its installation is straightforward. Fig 3 shows the executable file you receive as a download.

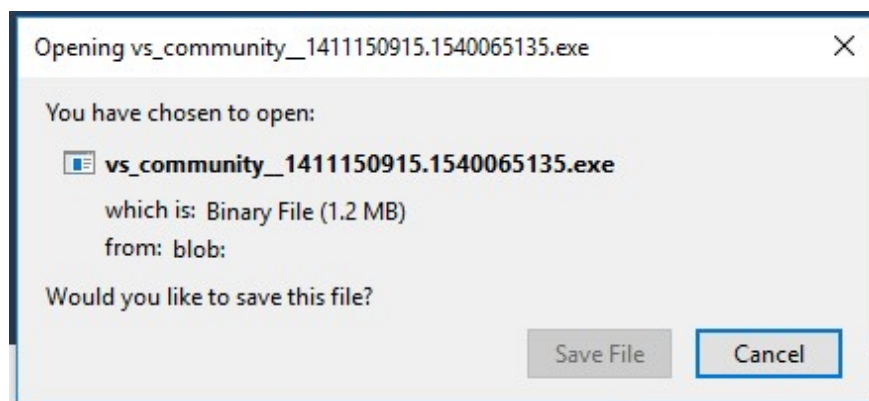
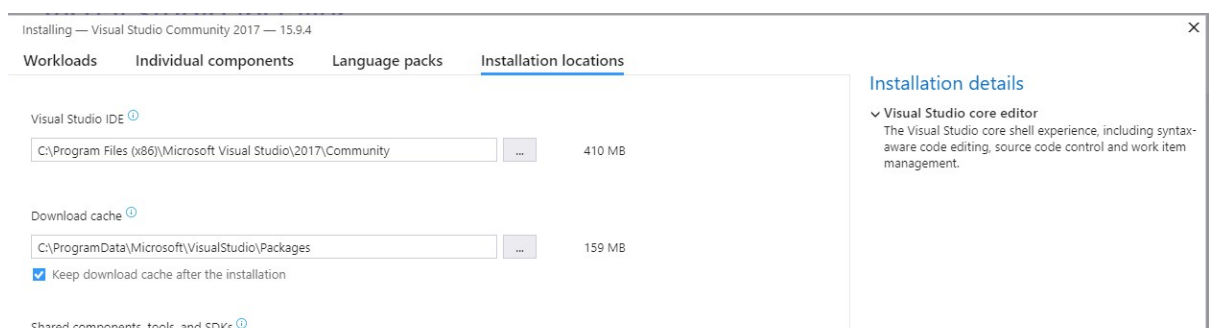


Fig 3: The Visual Studio Community executable file

When you press the 'save file' option on Fig 3, the window in Fig 4 will appear where you can set installation options (or just leave them as they are by default, as I did).



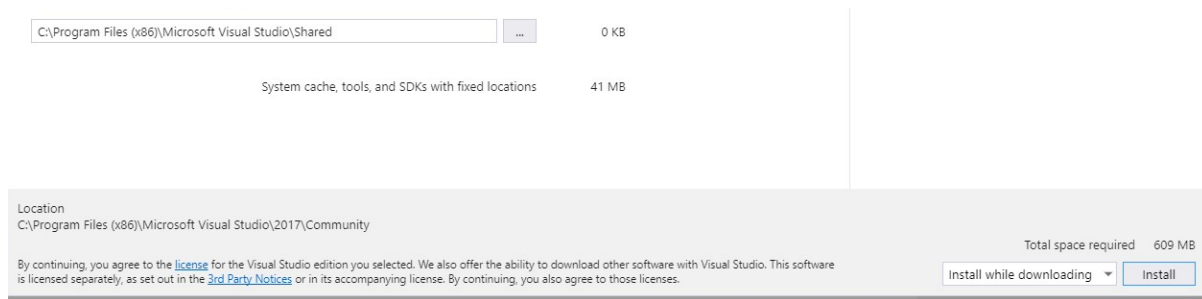


Fig 4: installation window for Visual Studio Community 2017

During installation, Visual Studio prompts you to as whether you ‘*want to continue without workloads*’. I pressed ‘continue’ here, as I had no intention of using workloads at all.

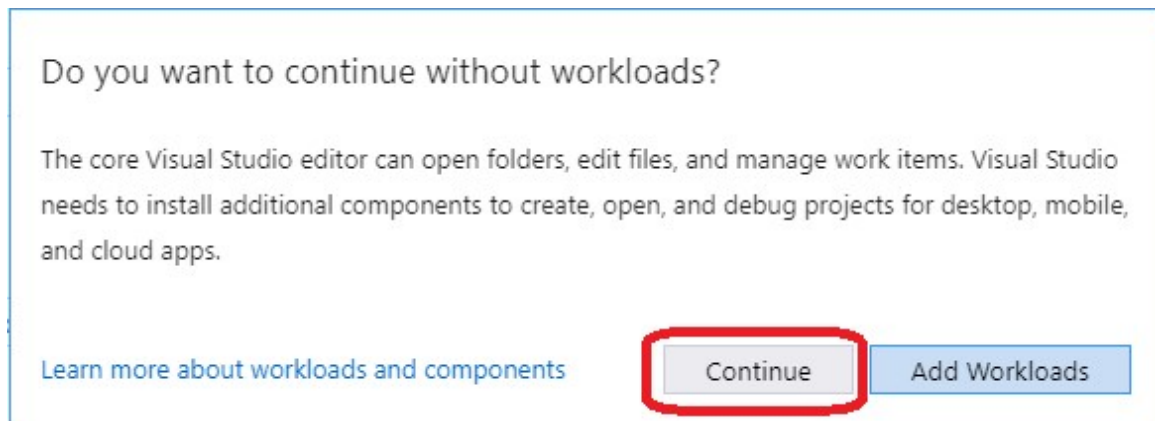


Fig 5: I didn’t add workloads on prompting by Visual Studio

A PC reboot after installation of Visual Studio May be Required

After installing Visual Studio, I initially moved straight on to downloading and attempting to install CUDA Toolkit for Windows — that step is **Step 3** which I will be describing next. I got a message that *Visual Studio was still operating and was preventing the installation of CUDA Toolkit*. Rebooting my PC before attempting to install CUDA Toolkit again solved this problem.

• • •

Step 3: Download CUDA Toolkit for Windows 10

These CUDA installation steps are loosely based on the Nvidia CUDA installation guide for windows. The CUDA Toolkit (free) can be downloaded from the Nvidia website here.

At the time of writing, the default version of CUDA Toolkit offered is version 10.0, as shown in Fig 6. However, you should check which version of CUDA Toolkit you choose for download and installation to ensure compatibility with Tensorflow (looking ahead to **Step 7** of this process). When you go onto the Tensorflow website, the latest version of Tensorflow available (1.12.0) requires **CUDA 9.0**, not CUDA 10.0. To find CUDA 9.0, you need to navigate to the “Legacy Releases” on the bottom right hand side of Fig 6.

Select Target Platform ⓘ

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

Operating System: **Windows** Linux Mac OSX

Architecture ⓘ: **x86_64**

Version: **10** 8.1 7 Server 2016 Server 2012 R2

Installer Type ⓘ: **exe (network)** exe (local)

Download Installer for Windows 10 x86_64

The base installer is available for download below.

Base Installer Download [2.1 GB] ⓘ

Installation Instructions:

1. Double click cuda_10.0.130_411.31_win10.exe
2. Follow on-screen prompts

The checksums for the installer and patches can be found in [Installer Checksums](#).
For further information, see the [Installation Guide for Microsoft Windows](#) and the [CUDA Quick Start Guide](#).

Documentation > Release Notes > Code Samples > **Legacy Releases >**

Fig 6: The default (most recent) version of CUDA for Windows is cuda_10.0.130_411.31_win10.exe.
For CUDA 9.0, choose “Legacy Releases”

• • •

Step 3.1: Downloading CUDA 9.0 from the CUDA Toolkit Archive

Choosing “Legacy Releases” takes you to the CUDA Toolkit Archive. Based on Tensorflow installation guidance, the CUDA version required is 9.0, as listed in Fig 7.

CUDA Toolkit Archive

Previous releases of the CUDA Toolkit, GPU Computing SDK, documents you want from the list below, and be sure to check www.nvidia.com/docs

[Download CUDA Toolkit 10.0](#)

Latest Release

[CUDA Toolkit 10.0](#) (Sept 2018), [Versioned Online Documentation](#)

Archived Releases

[CUDA Toolkit 9.2](#) (May 2018), [Online Documentation](#)

[CUDA Toolkit 9.1](#) (Dec 2017), [Online Documentation](#)

[CUDA Toolkit 9.0](#) (Sept 2017), [Online Documentation](#)

[CUDA Toolkit 8.0 GA2](#) (Feb 2017), [Online Documentation](#)



Fig 7: List of archived CUDA releases still available for download — CUDA Toolkit 9.0 (Sept 2017)

. . .

Step 3.2: Installing CUDA 9.0

CUDA 9.0 comes as a base installation and four patches; the base installation of CUDA 9.0 must be installed first, followed by the patches. The options for the base install which I selected are shown in Fig 8.

CUDA Toolkit 9.0 Downloads

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

Operating System

Windows

Linux

Mac OSX

Architecture ⓘ

x86_64

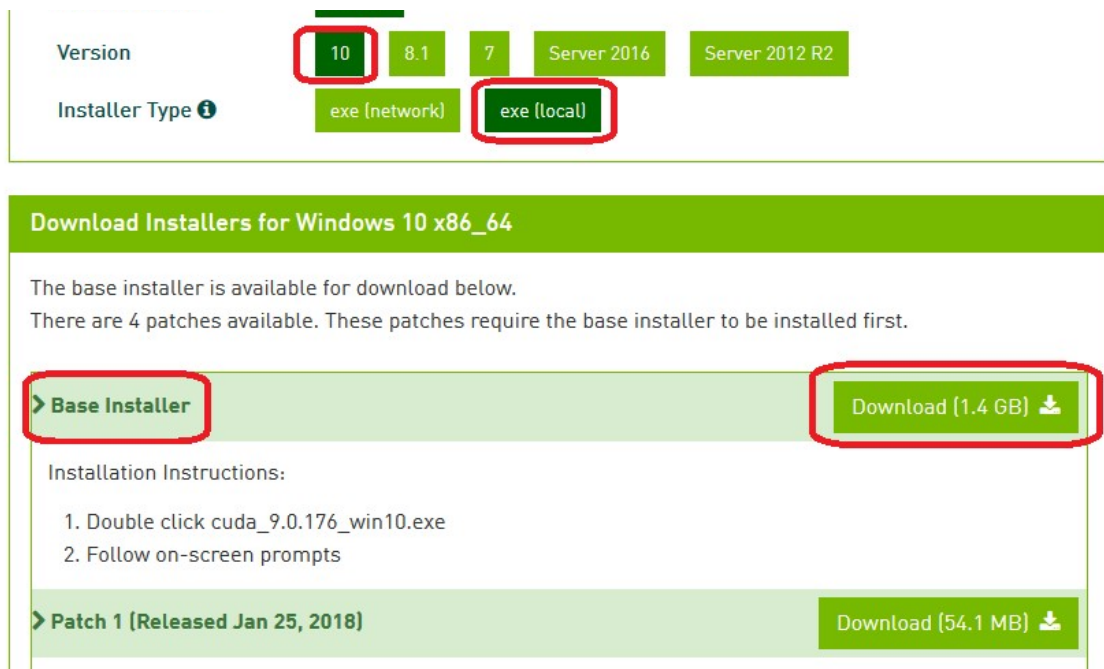


Fig 8: Options chosen for the base installation of CUDA 9.0 for Windows base installer

Running the base installer which has just been downloaded will produce the CUDA Setup Package window, as shown in Fig 9.

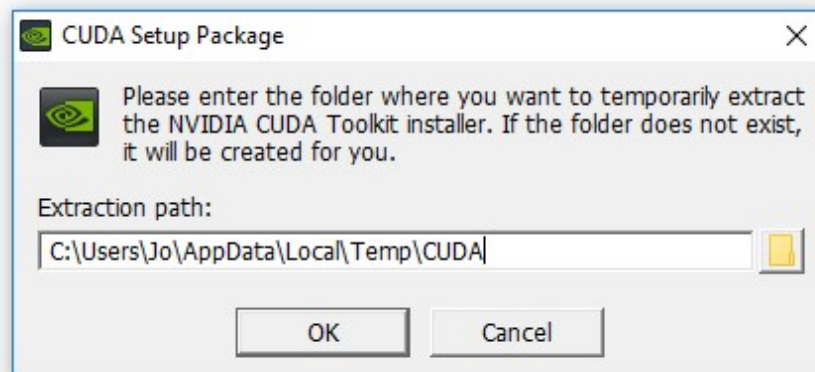


Fig 9: CUDA Setup Package for CUDA base installer

The CUDA installer extracts to your PC and, when complete, the NVIDIA CUDA Toolkit installation will start; you will get a message to that effect. The resulting NVIDIA Installer windows throughout the installation process are shown at Fig 10 — Fig 13. I chose the express installation option (Fig. 10).



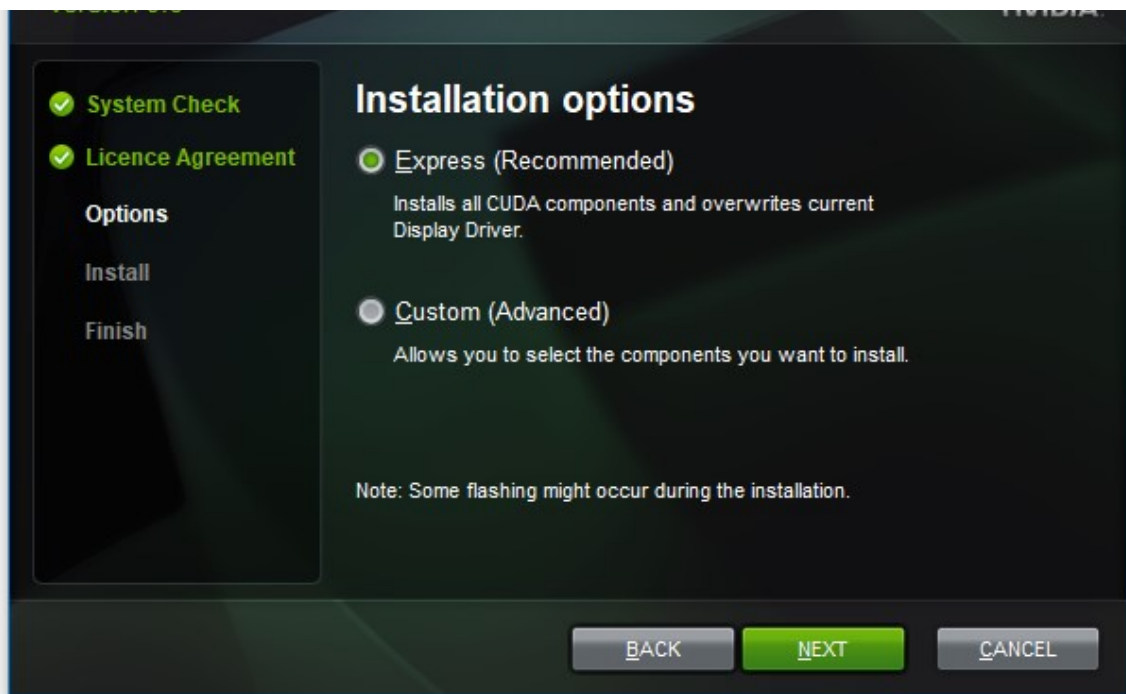
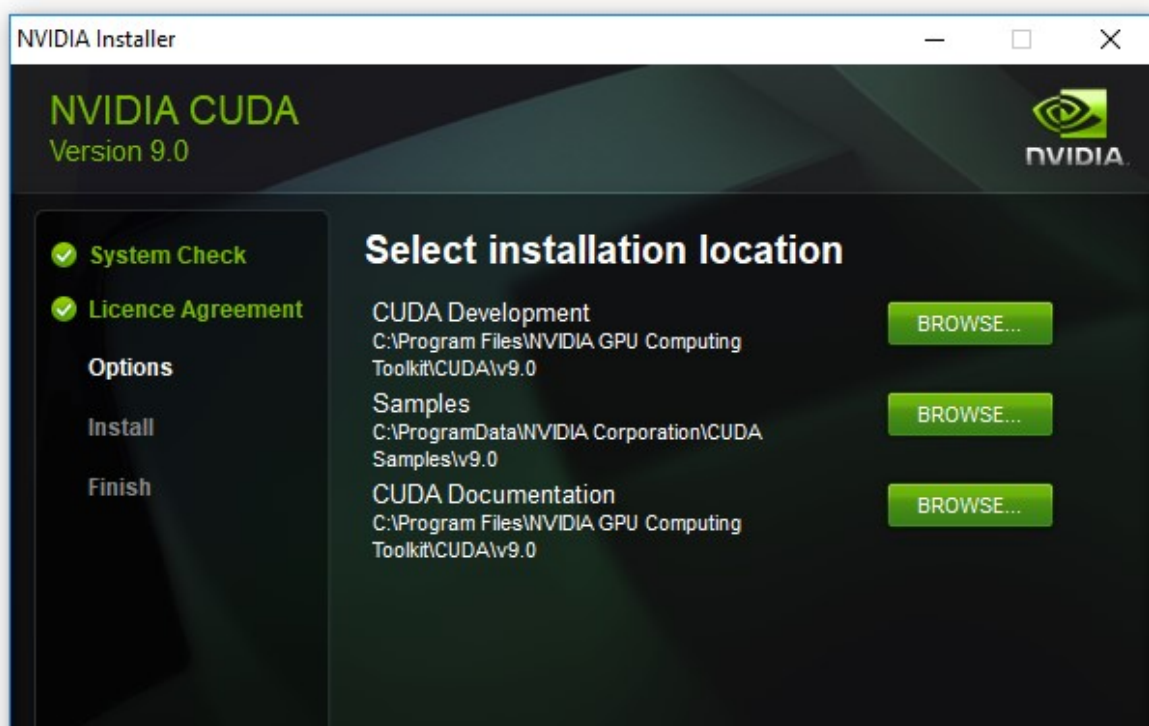


Fig 10: Installation options for CUDA 9.0 base installer — I chose the Express option

Fig. 11 provides the opportunity to select installation location; I chose the default locations provided, which for CUDA is :

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0



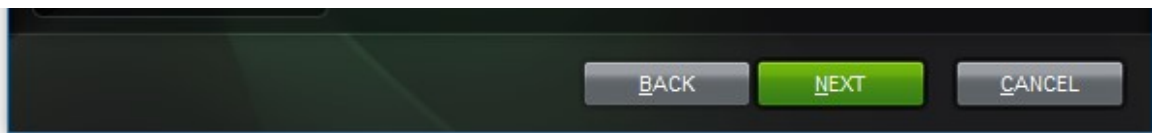


Fig 11 : CUDA 9.0 base installation — selecting CUDA installation location

Fig. 12 below shows the CUDA installations which rely on Visual studio, previously installed in **Step 1**.

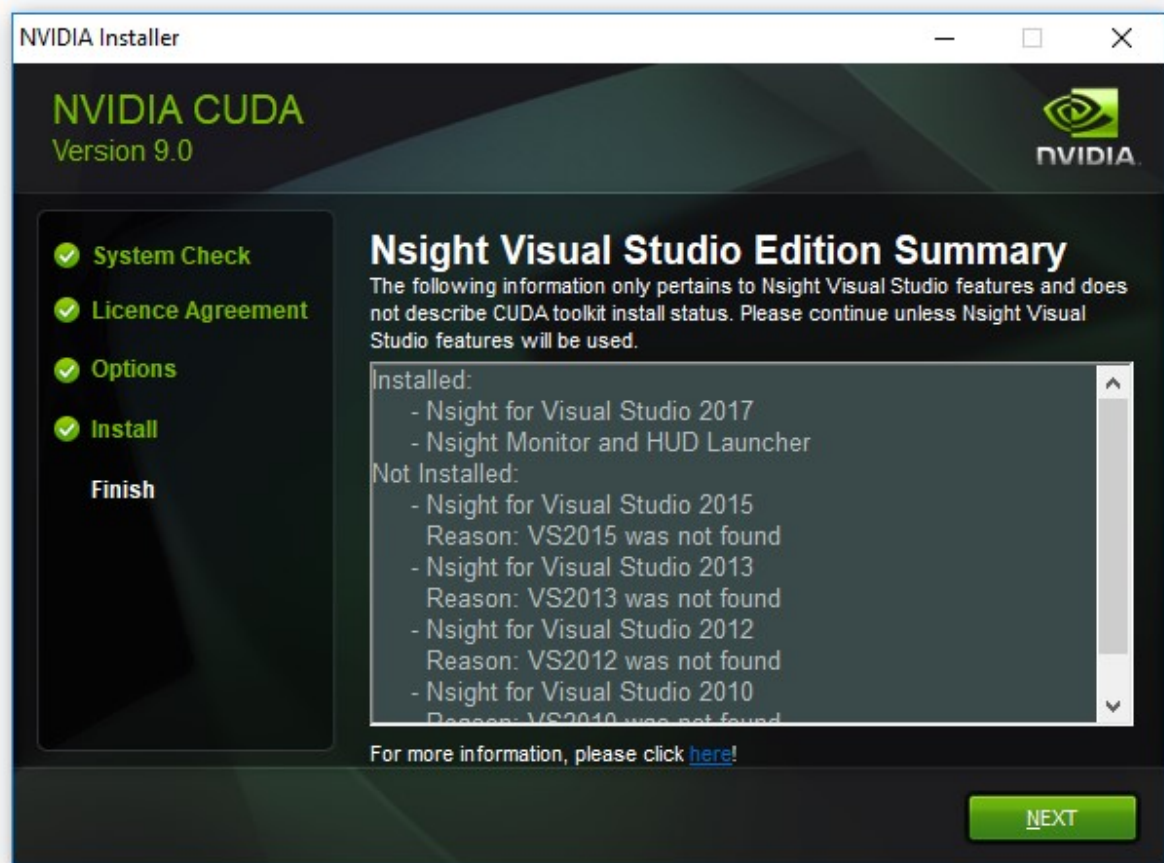


Fig 12: CUDA 9.0 base installation process — window showing installations relying on Visual Studio

Pressing 'next' at the window shown in Fig. 12 above, gives the final installation window, shown as Fig.13 below, where the NVIDIA installer is marked as finished.

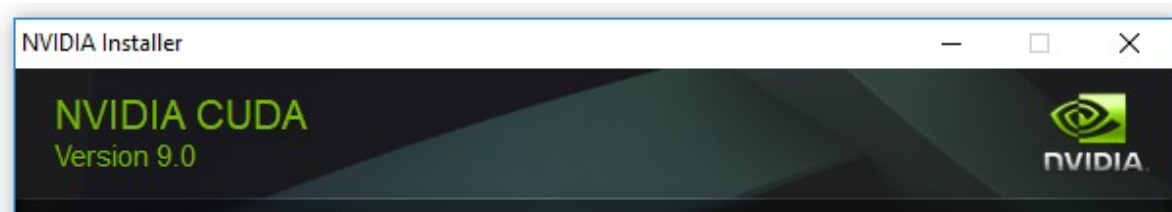




Fig 13: Final installation window for CUDA 9.0 base installer

. . .

Step 4: Download Windows 10 CUDA patches

At the time of writing, there are four CUDA patches to get (as well as the base installer), so let's go and download these. They are shown in Fig.14.

> Base Installer

Download [1.4 GB]

Installation Instructions:

1. Double click cuda_9.0.176_win10.exe
2. Follow on-screen prompts

> Patch 1 (Released Jan 25, 2018)

Download [54.1 MB]

cuBLAS Patch Update: This update to CUDA 9.0 includes new GEMM kernels optimized for the Volta architecture and improved heuristics to select GEMM kernels for given input sizes.

> Patch 2 (Released Mar 5, 2018)

Download [54.7 MB]

cuBLAS Patch Update: This update to CUDA 9 includes GEMM heuristics improvements to selects the most optimized algorithms for input sizes commonly used in Deep Learning RNNs. The update also includes other bug-fixes and performance enhancements.

> Patch 3 (Released Jan 7, 2018)

Download [54.2 MB]

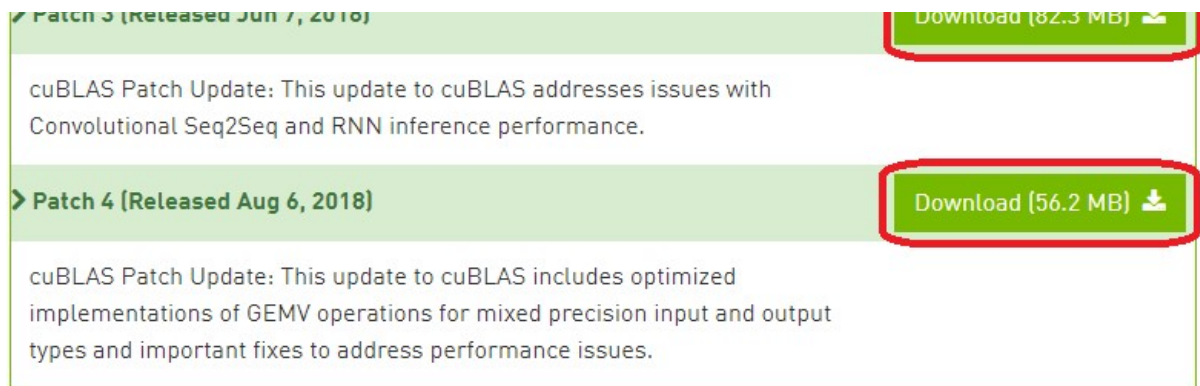


Fig 14: Downloading and installing the additional four patches for CUDA 9.0

When the four patches are downloaded, they can be installed in the same way as the base installer — with installation windows providing guidance through the process.

. . .

Step 5: Download and Install cuDNN

Having installed CUDA 9.0 base installer and its four patches, the next step is to find a compatible version of CuDNN. Based on the information on the Tensorflow website, Tensorflow with GPU support requires a cuDNN version of at least 7.2.

Step 5.1: Downloading cuDNN

In order to download CuDNN, you have to register to become a member of the NVIDIA Developer Program (which is free).

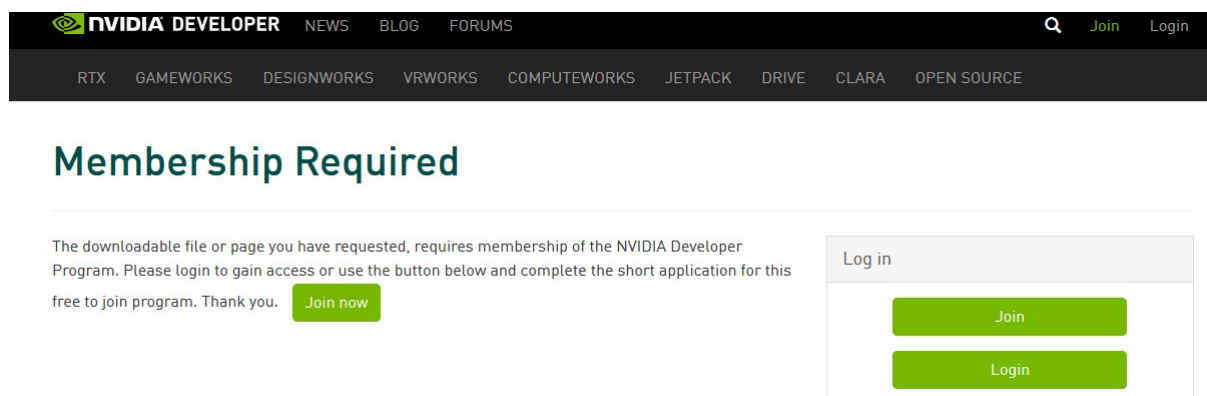


Fig 15: Creating a free membership account in order to download cuDNN

When you create an account, login and fill out some other required details about why you are using the account, you get the download page shown in Fig. 16.

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 v7.4.2 \[Dec 14, 2018\], for CUDA 10.0](#)

[Download cuDNN v7.4.2 \[Dec 14, 2018\], for CUDA 9.2](#)

[Download cuDNN v7.4.2 \[Dec 14, 2018\], for CUDA 9.0](#)

[Archived cuDNN Releases](#)



Fig 16: cuDNN download page with selection of cuDNN v.7.4

As I have downloaded CUDA 9.0, the corresponding version of cuDNN is version 7.4.2. Choosing cuDNN version 7.4.2 enables the download as a zip file named as follows:

```
cuda-9.0-windows10-x64-v7.zip
```

Step 5.2: Unzipping cuDNN files and copying to CUDA folders

Instructions at Nvidia provide support for windows cuDNN installation, as do instructions on the Tensorflow website ; I have reproduced these instructions in distilled form, based on my implementation of them. In my case, I downloaded the cuDNN .zip file named above into a folder which has the following path on my PC (your path will no doubt be different).

```
C:\Users\jo\Documents\cuDNN_downloads\
```

In the instructions below, I refer to the folder path “*C:\Users\jo\Documents\cuDNN_downloads*” (referred to just above) as “<downloadpath>”, such that the zip file is now in the path:

```
<downloadpath>\cudnn-9.0-windows10-x64-v7.5.0.56.zip
```

I unzipped the cuDNN “.zip” file where I downloaded it, hence the unzipped folder structure which will contain the required cuDNN files is now:-

```
<downloadpath>\cudnn-9.0-windows10-x64-v7.5.0.56\
```

There are three files in the unzipped cuDNN folder subdirectories which are to be copied into the CUDA Toolkit directories. These are cudnn64_7.dll, cudnn.h and :

1. cudnn64_7.dll

cudnn64_7.dll can be found in the following path within the downloaded cuDNN files:

```
<downloadpath>\cudnn-9.0-windows10-x64-v7.5.0.56\cuda\bin\cudnn64_7.dll
```

Assuming that you installed CUDA 9.0 to its default path (as I did at **Step 2.3**), namely the following default path:

```
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0
```

you can copy the *cudnn64_7.dll* file directly into the CUDA folder’s *bin* folder path (note: you don’t need to create any new subfolders):

```
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin\
```

2. cudnn.h

As with the cudnn64_7.dll file above, after downloading and unzipping the cuDNN folder, the header file *cudnn64.h* can be found in the path:

```
<downloadpath>\cudnn-9.0-windows10-x64-v7.5.0.56\cuda\  
include\cudnn.h
```

Again, assuming that you installed CUDA 9.0 into the default path as I did at **Step 2.3**, copy *cudnn.h* directly into the CUDA folder with the following path (no new subfolders are necessary):

```
C:\Program Files\NVIDIA GPU Computing  
Toolkit\CUDA\v9.0\include\
```

3. cudnn.lib

The .lib file *cudnn.lib* can be found in the downloaded cuDNN path:

```
<downloadpath>\cudnn-9.0-windows10-x64-  
v7.5.0.56\cuda\lib\x64\cudnn.lib
```

Copy cudnn.lib directly into the CUDA folder with the following path:

```
C:\Program Files\NVIDIA GPU Computing  
Toolkit\CUDA\v9.0\lib\x64\
```

Step 5.3: Checking CUDA environment variables are set in Windows

Finally, the instructions at Nvidia direct that you ensure that the CUDA environment variable has previously been set up, as follows:

```
Variable Name: CUDA_PATH  
Variable Value: C:\Program Files\NVIDIA GPU Computing  
Toolkit\CUDA\v9.0
```

In Windows 10, the Environment Variables can be found by choosing:

Control Panel -> System and Security->System->Advanced System settings.

This opens up a window called “System Properties” (Fig 17), at which point the “Environment Variables” button should be chosen.

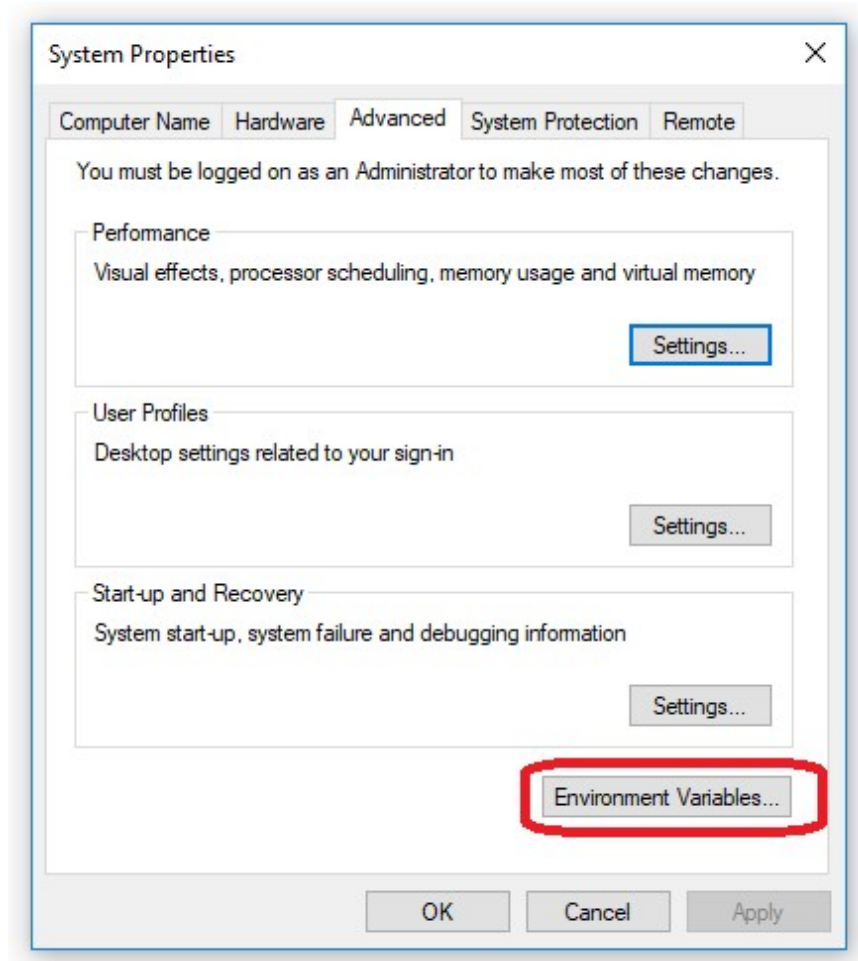


Fig 17: Environment Variables button (in System Properties window) for setting and checking CUDA paths

When the Environment Variables window then appears, within “system variables” (in the bottom half of the window), click on “Path” and choose the button “edit”. A new window will appear, called “Edit environment variable” as shown in Fig 18 below.

On checking the Environment Variables, I found the installation process which determines the CUDA installation path — **Step 3.2**, see Fig. 11 — had already added two paths to CUDA . These paths are shown in Fig 18 below, so I found I did not need to add a further CUDA path.

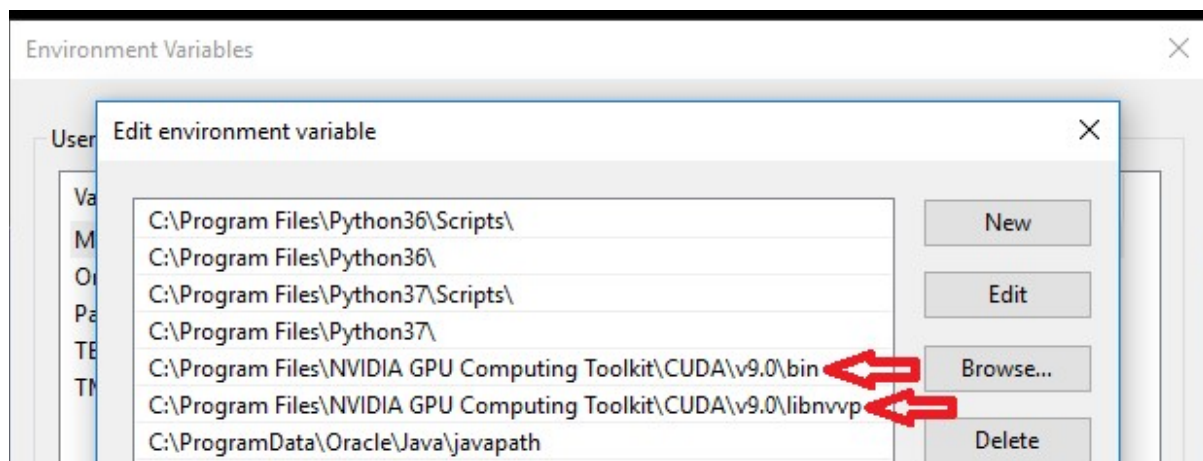
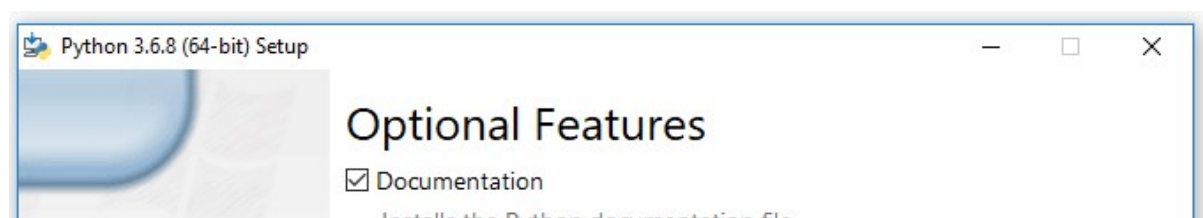


Fig 18: Default paths previously created during CUDA 9.0 installation process

. . .

Step 6: Install Python (if you don't already have it)

Now that CUDA and cuDNN are installed, it is time to install Python to enable Tensorflow to be installed later on. At the time of writing, the most up to date version of Python 3 available is Python 3.7, but the Python 3 versions required for Tensorflow are 3.4, 3.5 or 3.6. Python 3.6 can be downloaded for Windows 10 from [here](#). When you run the Python installer for windows, the setup window in Fig 19 will appear.



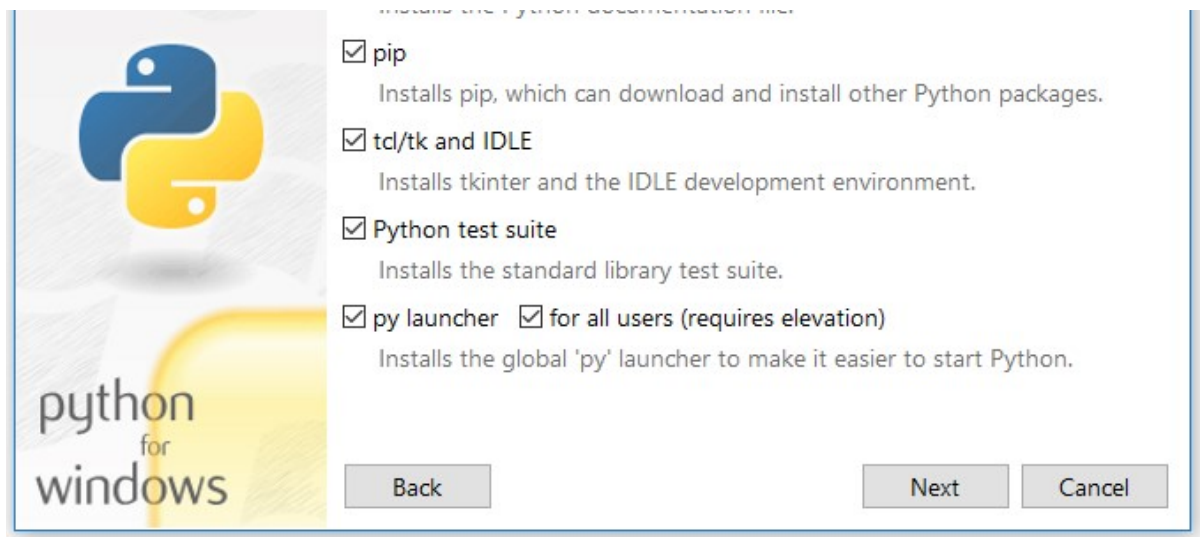


Fig 19: Python 3.6 installation screen; I choose all the optional features on this screen (pip and IDLE are both used in subsequent steps)

Of the options in Fig. 19 above during Python installation, I chose to select all of them. These options are useful: Python's 'pip' installer is used at **Step 7.2** of this guide to install Tensorflow. Additionally, I use the IDE (integrated development environment for writing and running python code) called "IDLE" at **Step 8**.

In the "Advanced Options" (shown at Fig 20 below), I chose the installation of Python for all users (which was not ticked by default); this gives the more useful system wide installation.

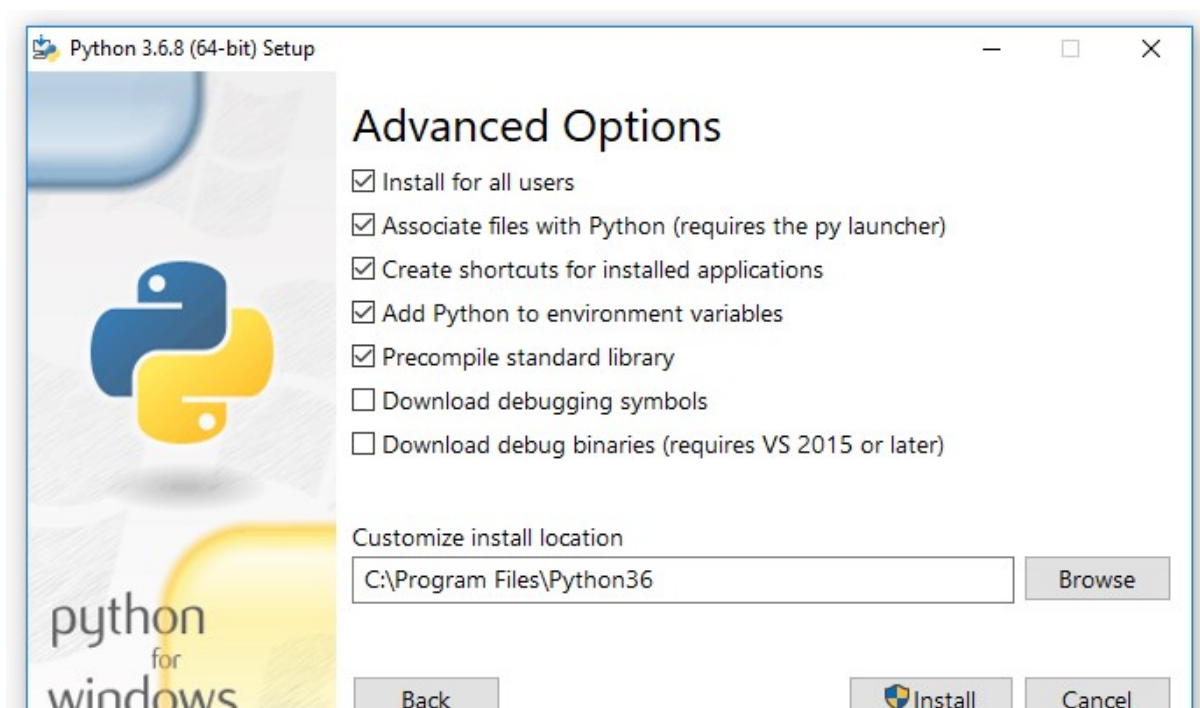




Fig 20: Advanced options available when installing Python 3.6

. . .

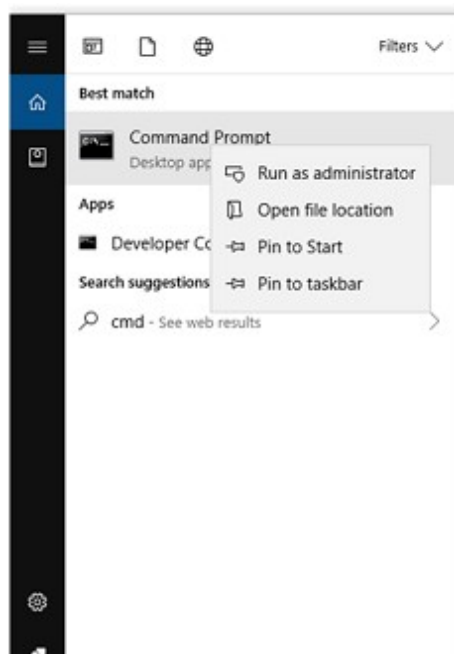
Step 7: Install Tensorflow with GPU support

Tensorflow provides instructions for checking that CUDA, cuDNN and (optional: CUPTI) installation directories are correctly added to the PATH environmental variables. As the three cuDNN files were copied into the subfolders of CUDA, I did not update the existing CUDA environmental variables path.

Step 7.1: Calling up the command prompt with administration rights

In this step a system-wide install of Tensorflow is carried out, not a per user install. System wide installation of Tensorflow requires administrative rights, therefore, accordingly the command prompt should be run with administrative rights.

Open up the command prompt by running '`cmd`' in the search bar, and then right clicking on command prompt to choose 'run as administrator'. This opens up Administrator: Command Prompt as shown in Fig 21.



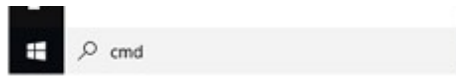


Fig 21: Running the command prompt as administrator from the Windows 10 search bar

Step 7.2: System wide install of Tensorflow via python pip

Having opened the Command Prompt, the system-wide installation command for Tensorflow with GPU support is as follows:

```
pip3 install --upgrade tensorflow-gpu
```

The “pip3” command (as opposed to “pip”) is required as the installation is to Python 3. Execution of this command in the command prompt is shown in Fig 22.

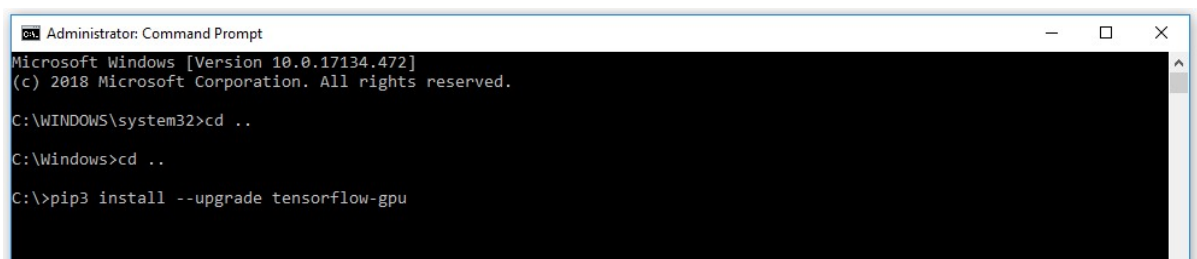
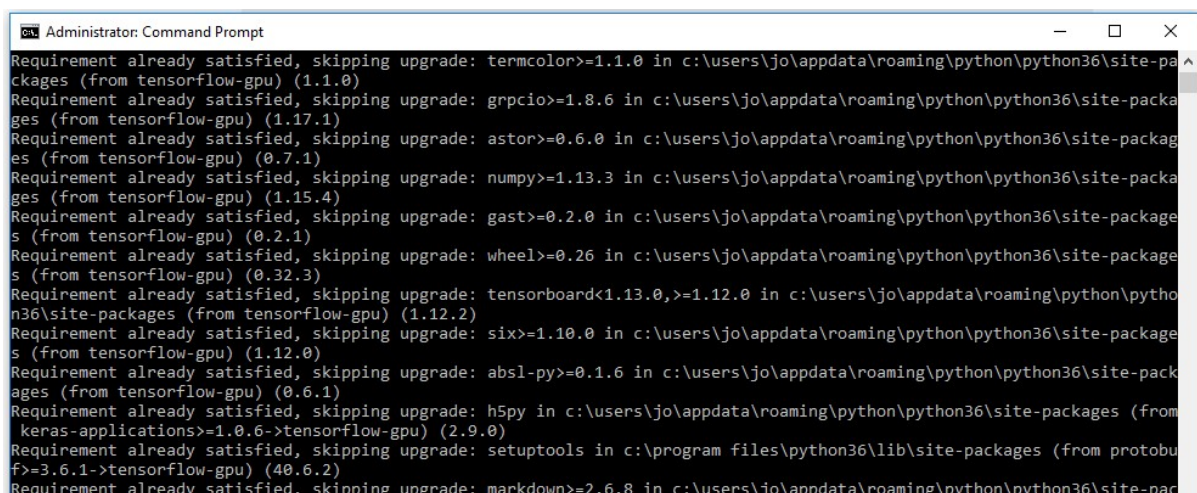


Fig 22: Pip install command for Tensorflow with GPU support

The resulting output from the command is shown in Fig 23, and if all goes to plan there should eventually be a message to confirm that Tensorflow has been installed successfully.



```
kages (from tensorboard<1.13.0,>=1.12.0->tensorflow-gpu) (3.0.1)
Requirement already satisfied, skipping upgrade: werkzeug>=0.11.10 in c:\users\jo\appdata\roaming\python\python36\site-p
ackages (from tensorboard<1.13.0,>=1.12.0->tensorflow-gpu) (0.14.1)
Installing collected packages: tensorflow-gpu
Successfully installed tensorflow-gpu-1.12.0
C:\>
```

Fig 23: Command prompt messages shown when Tensorflow GPU 1.12.0 installed successfully.

. . .

Step 8: Test Installation of TensorFlow and its access to GPU

Go to the start menu in windows and search for the IDE called 'idle', which will be installed as part of your python installation if you selected as I did at **Step 6**. A Python window should appear labelled *Python 3.6.x Shell*. At the prompt (denoted by '>>>'), import the Tensorflow package. This will check that Tensorflow has been installed (as you can import it). The command for the IDLE shell to import the tensorflow package is as follows:

```
# importing the tensorflow package
import tensorflow as tf
```

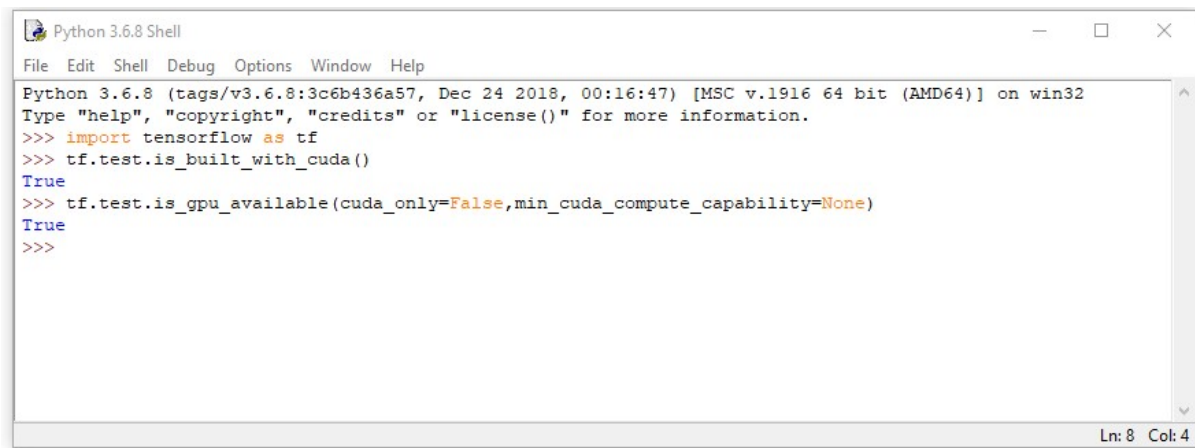
To test CUDA support for your Tensorflow installation, you can run the following command in the shell:

```
tf.test.is_built_with_cuda()
```

Finally, to confirm that the GPU is available to Tensorflow, you can test using a built-in utility function in TensorFlow as shown here:

```
tf.test.is_gpu_available(cuda_only=False,
min_cuda_compute_capability=None)
```


It takes a few minutes to return a result from this; when it is finished it returns **True**, and then the prompt `>>>` appears again. Importing tensorflow and these tests are all shown in Fig 24 in the Python IDLE Shell.

A screenshot of the Python 3.6.8 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following code and output:

```
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import tensorflow as tf
>>> tf.test.is_built_with_cuda()
True
>>> tf.test.is_gpu_available(cuda_only=False,min_cuda_compute_capability=None)
True
>>>
```

The status bar at the bottom right indicates 'Ln: 8 Col: 4'.

Fig 24: Using the IDLE python IDE to check that Tensorflow has been built with CUDA and that the GPU is available

Conclusions

These were the steps I took to install Visual Studio, CUDA Toolkit, CuDNN and Python 3.6, all with the ultimate aim of installing Tensorflow with GPU support on Windows 10. To date, my GPU based machine learning and deep learning work has been on Linux Ubuntu machines; by the same token, much of the machine learning community support online focuses on Ubuntu.

For Machine Learning, the major drawback to using Windows is that it is necessary to build more things from source (for example using Cmake) than on Linux, and also to install additional software for the build processes, such as Visual Studio. For example, if you were to install Caffe2 on Windows, there are no pre-built binaries and Windows build is in testing and beta mode. I installed CUDA and cuDNN on Windows 10 more out of curiosity than anything else, just to see how straightforward (or otherwise) it was.

As I intimated in Part 1, now that CUDA, cuDNN and Tensorflow are successfully installed on Windows 10 and I have checked Tensorflow's access to GPU, I am going to sweep the whole Windows 10 operating system away in

order to make a fresh installation of Ubuntu 18.04 LTS. This new installation of Ubuntu will be covered in Part 3 of this series.

This article has also been published here on my own blog.

[TensorFlow](#)

[Installation](#)

[Windows 10](#)

[Gpu](#)

[Cuda](#)

[About](#)

[Help](#)

[Legal](#)