



TENSORRT 3.0

DU-08731-001_v3.0 | May 2018

Installation Guide



TABLE OF CONTENTS

Chapter 1. Overview.....	1
Chapter 2. Getting Started.....	2
Chapter 3. Downloading TensorRT.....	4
Chapter 4. Installing TensorRT.....	5
4.1. Debian Installation.....	6
4.2. Tar File Installation.....	8
Chapter 5. Upgrading from TensorRT 2.1 to TensorRT 3.0.....	10
Chapter 6. Uninstalling TensorRT.....	12
Chapter 7. Installing PyCUDA.....	13
7.1. Updating CUDA.....	13
Chapter 8. Troubleshooting.....	15

Chapter 1.

OVERVIEW

NVIDIA[®] TensorRT[™] is a C++ library that facilitates high performance inference on NVIDIA graphics processing units (GPUs). TensorRT takes a network definition and optimizes it by merging tensors and layers, transforming weights, choosing efficient intermediate data formats, and selecting from a large kernel catalog based on layer parameters and measured performance.

TensorRT consists of import methods to help you express your trained deep learning model for TensorRT to optimize and run. It is an optimization tool that applies graph optimization and layer fusion and finds the fastest implementation of that model leveraging a diverse collection of highly optimized kernels, and a runtime that you can use to execute this network in an inference context.

TensorRT includes an infrastructure that allows you to leverage high speed reduced precision capabilities of Pascal[™] GPUs as an optional optimization.

TensorRT is built with [gcc 4.8](#).

Chapter 2.

GETTING STARTED

Ensure you are familiar with the following installation requirements and notes.

- ▶ If you are using the TensorRT python API and PyCUDA isn't already installed on your system, see [Installing PyCUDA](#). If you are testing on V100, or if you encounter any issues with PyCUDA usage, you will almost certainly need to recompile it yourself. For more information, see [Installing PyCUDA on Linux](#).
- ▶ Ensure you are familiar with the Release Notes. The current version of the release notes can be found online at [TensorRT Release Notes](#).
- ▶ Verify that you have the CUDA Toolkit installed
 - ▶ For TensorRT 3.0.2 and later, install CUDA Toolkit 8.0 or 9.0.
 - ▶ For TensorRT 3.0.3, install CUDA Toolkit 9.1.
- ▶ , release 8.0 or 9.0.
- ▶ The TensorFlow to TensorRT model export requires TensorFlow v1.3 with GPU acceleration enabled.
- ▶ If a target install system has both TensorRT and one or more training frameworks installed on it, the simplest strategy is to use the same version of cuDNN for the training frameworks as the one that TensorRT ships with. If this is not possible, or for some reason strongly undesirable, be careful to properly manage the side-by-side installation of cuDNN on the single system. In some cases, depending on the training framework being used, this may not be possible without patching the training framework sources.
- ▶ The `libnvcaffe_parser.so` library file from previous versions is now called `libnvparsers.so` in TensorRT 3. The installed symbolic link for `libnvcaffe_parser.so` is updated to point to the new `libnvparsers.so` library. The static library `libnvcaffe_parser.a` is also symbolically linked to the new `libnvparsers.a`.
- ▶ The installation instructions below assume you want the full TensorRT; both the C++ and TensorRT python APIs. In some environments and use cases, you may not want to install the Python functionality. In which case, simply don't install the debian packages labeled Python or the whl files. None of the C++ API functionality depends on Python. You would need to install the UFF whl file if you want to export the UFF file from TensorFlow.

- ▶ On Ubuntu 14.04 systems, if you want to use the TensorRT Python packages, ensure that the version of [SWIG](#) on the target system is 3.0 or higher. This is not the default version that comes with the Ubuntu 14.04 distribution. This version of SWIG will need to be installed before installing the `python-libnvinfer` package.
- ▶ If you have trouble installing the TensorRT Python modules on Ubuntu 14.04, refer to the steps on installing `swig` to resolve the issue. For installation instructions, see [Unix Installation](#).

Chapter 3.

DOWNLOADING TENSORRT

Ensure you are a member of the NVIDIA Developer Program. If not, follow the prompts to gain access.


1. Go to: <https://developer.nvidia.com/tensorrt>.
2. Click **Download**.
3. Complete the TensorRT Download Survey.
4. Select the checkbox to agree to the license terms.
5. Follow the Quick Start Instructions and choose which method you want to install the package.
6. Click the package you want to install. Your download begins.
7. Verify that your download was successful. The download can be verified by comparing the MD5 checksum file with that of the download file. If either of the checksums differ, the downloaded file is corrupt and needs to be downloaded again. To calculate the MD5 checksum of the downloaded file, run the following:

```
$ md5sum <file>
```

Chapter 4.

INSTALLING TENSORRT

You can choose between two installation options when installing TensorRT; a debian package or tar file.

 The tar file installation is the only option for Ubuntu 14.04 x86_64 target systems.

The debian installation automatically installs any dependencies, but:

- ▶ requires **sudo** root privileges to install
- ▶ provides no flexibility as to which location TensorRT is installed into
- ▶ requires that the CUDA Toolkit has also been installed with a debian package.

The tar file provides more flexibility, however, you need to ensure that you have the necessary dependencies already installed.

TensorRT versions: TensorRT is a product made up of separately versioned components. The version on the product conveys important information about the significance of new features while the library version conveys information about the compatibility or incompatibility of the API. The following table shows the versioning of the TensorRT components.

Table 1 Versioning of TensorRT components

Product or Component	Previously Released Version	Current Version	Version Description
TensorRT product	2.1.2	3.0.3	+1.0 when significant new capabilities are added. +0.1 when capabilities have been improved.

Product or Component	Previously Released Version	Current Version	Version Description
nvinfer library, headers, samples, and documentation.	3.0.2	4.0.2	+1.0 when the API changes in a non-compatible way. +0.1 when the API changes are backward compatible
UFF	new	0.2.0	+0.1 while we are developing the core functionality. Set to 1.0 when we have all base functionality in place.
libnvinfer python package	new	3.0.2	+1.0 when the API changes in a non-compatible way. +0.1 when the API changes are backward compatible.

4.1. Debian Installation

This section contains instructions for a developer installation and an app server installation. Choose which installation best fits your needs.

Developer Installation: The following instructions sets up a full TensorRT development environment with samples, documentation and both the C++ and Python API.



Attention If only the C++ development environment is desired, you can modify the following instructions and simply not install the Python and UFF packages.



Before issuing the following commands, you'll need to replace 3.x.x with your specific TensorRT version. The following commands are examples.

1. Install TensorRT from the debian package, for example:

```
$ sudo dpkg -i
nv-tensorrt-repo-ubuntu1604-ga-cuda9.0-trt3.x.x-20180108_1-1_amd64.deb
$ sudo apt-get update
```



```
$ sudo apt-get install tensorrt
```

where **3.x.x** is your TensorRT version

If using Python 2.7:

```
$ sudo apt-get install python-libnvinfer-doc
```

The following additional packages will be installed:

```
python-libnvinfer python-libnvinfer-dev swig swig3.0
```

If using Python 3.5:

```
$ sudo apt-get install python3-libnvinfer-doc
```

The following additional packages will be installed:

```
python3-libnvinfer python3-libnvinfer-dev
```

In either case:

```
$ sudo apt-get install uff-converter-tf
```



If installing on Ubuntu 14.04, `swig` will not install. For instructions on installing `swig`, see [Unix Installation](#).

2. Verify the installation:

```
dpkg -l | grep TensorRT
```

You should see something similar to the following:

```
c$ dpkg -l | grep TensorRT
ii libnvinfer-dev                    4.0.2-1+cuda9.0
    amd64 TensorRT development libraries
    and headers
ii libnvinfer-samples              4.0.2-1+cuda9.0
    amd64 TensorRT samples and
    documentation
ii libnvinfer4                    4.0.2-1+cuda9.0
    amd64 TensorRT runtime libraries
ii python3-libnvinfer              4.0.2-1+cuda9.0
    amd64 Python 3 bindings for TensorRT
ii python3-libnvinfer-dev          4.0.2-1+cuda9.0
    amd64 Python 3 development package for
    TensorRT
ii python3-libnvinfer-doc          4.0.2-1+cuda9.0
    amd64 Documention and samples of python
    bindings for TensorRT
ii tensorrt                        3.x.x-1+cuda9.0
    amd64 Meta package of TensorRT
ii uff-converter-tf                4.0.2-1+cuda9.0
    amd64 UFF converter for TensorRT pack
```

App Server Installation: When setting up servers which will host TensorRT powered applications, you can simply install any of the following:

- ▶ the `libnvinfer` package (C++), or
- ▶ the `python-libnvinfer` package (Python), or
- ▶ the `python3-libnvinfer` package (Python).

Issue the following commands if you want to run an application that was built with TensorRT. Install TensorRT from the debian package:

```
$ sudo dpkg -i
nv-tensorrt-repo-ubuntu1604-ga-cuda9.0-trt3.0-20171128_1-1_amd64.deb

$ sudo apt-get update
$ sudo apt-get install libnvinfer
```

4.2. Tar File Installation



Before issuing the following commands, you'll need to replace `3.x.x` with your specific TensorRT version. The following commands are examples.

1. Install the following dependencies, if not already present:
 - ▶ If using TensorRT 3.0.2 or later, install the CUDA Toolkit v8.0 or 9.0
 - ▶ If using TensorRT 3.0.3, install the CUDA Toolkit v9.1
 - ▶ cuDNN 7.0.5
 - ▶ Python 2 or Python 3
2. Choose where you want to install. This tar file will install everything into a directory called `TensorRT-3.x.x`, where `3.x.x` is your TensorRT version. This directory will have sub-directories like `lib`, `include`, `data`, etc...
3. Unpack the tar file, for example:

```
$ tar xzvf TensorRT-3.x.x.Ubuntu-16.04.3.x86_64.cuda-9.0.cudnn7.0.tar.gz

$ ls TensorRT-3.x.x
bin data doc include lib python samples targets TensorRT-Release-
Notes.pdf uff
```

4. Add the absolute path of TensorRT `lib` to the environment variable `$LD_LIBRARY_PATH`, for example:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<eg:TensorRT-3.x.x/lib>
```

5. Install the Python TensorRT package, for example:

```
$ cd TensorRT-3.x.x/python
```

If using Python 2.7:

```
$ sudo pip2 install tensorrt-3.x.x-cp27-cp27mu-linux_x86_64.whl
```

If using Python 3.5:

```
$ sudo pip3 install
tensorrt-3.x.x-cp35-cp35m-linux_x86_64.whl
```

In either case:

```
$ which tensorrt
/usr/local/bin/tensorrt
```

6. Install the Python UFF package, for example:

```
$ cd TensorRT-3.x.x/uff
$ sudo pip2 install uff-0.2.0-py2.py3-none-any.whl
$ which convert-to-uff
/usr/local/bin/convert-to-uff
```

7. Update the `custom_plugins` example to point to the location that the tar package was installed into. For example, in the `<PYTHON_INSTALL_PATH>/tensorrt/examples/custom_layers/tensorrtplugins/setup.py` file, change the following:
 - ▶ Change `TENSORRT_INC_DIR` to point to the `<TAR_INSTALL_ROOT>/include` directory.
 - ▶ Change `TENSORRT_LIB_DIR` to point to `<TAR_INSTALL_ROOT>/lib` directory.
8. Verify the installation:
 - a) Ensure that the installed files are located in the correct directories. For example, run the `tree -d` command to check whether all supported installed files are in place in the `lib`, `include`, `data`, etc... directories.
 - b) Build and run one of the shipped samples, for example, `sampleMNIST` in the installed directory. The sample should be compiled and executable without additional settings. For more information about `sampleMNSIT`, see the [TensorRT Developer Guide](#).

Chapter 5.

UPGRADING FROM TENSORRT 2.1 TO TENSORRT 3.0

When upgrading from TensorRT 2.1 to TensorRT 3.0, ensure you are familiar with the following notes:

Using a debian file:

- ▶ The debian packages are designed to upgrade your development environment without removing any runtime components that other packages and programs might rely on. Therefore, if you installed TensorRT 2.1 via a debian package and you install TensorRT 3.0, your documentation, samples and headers will be updated to the TensorRT 3.0 content. In the case of an upgrade, you would expect to have two or more installed versions of `libnvinfer`.
- ▶ After you upgrade, ensure you have a package called `tensorrt` and the corresponding version shown by the `dpkg -l` command is `3.x.x`.
- ▶ If installing a debian package on a system where the previously installed version was from a tar file, note that the debian package will not remove the previously installed files. Unless a side-by-side installation is desired, it would be best to remove the older version before installing the new version to avoid compiling against outdated headers.

Using a tar file:

- ▶ When using the tar file installation method, install TensorRT into a new location. Tar file installations can support multiple use cases including having a full installation of TensorRT 2.1 with headers and documentation side-by-side with a full installation of TensorRT 3.0. If the intention is to have the new version of TensorRT replace the old version, then the old version should be removed once the new version is verified.
- ▶ If installing a tar file on a system where the previously installed version was from a debian package, note that the tar file install will not remove the previously installed

packages. Unless a side-by-side installation is desired, it would be best to remove the previously installed `nvinfer-dev` and `nvinfer-sample` packages to avoid confusion.

Chapter 6.

UNINSTALLING TENSORRT

1. Uninstall `libnvinfer4` which was installed through the debian file.

```
$ sudo apt-get purge "libnvinfer*"
```

2. Uninstall `uff-converter-tf`, which was also installed through the debian file.

```
$ sudo apt-get purge "uff-converter-tf"
```

3. Uninstall the Python TensorRT package.

If using Python 2.7:

```
$ sudo pip2 uninstall tensorrt
```

If using Python 3.5:

```
$ sudo pip3 uninstall tensorrt
```

4. Uninstall the Python UFF package.

If using Python 2.7:

```
$ sudo pip2 uninstall uff
```

If using Python 3.5:

```
$ sudo pip3 uninstall uff
```

Chapter 7.

INSTALLING PYCUDA



Attention If you have to update your CUDA version on your system, do not install PyCUDA at this time. Perform the steps in [Updating CUDA](#) instead, then install PyCUDA.

PyCUDA is used within Python wrappers to access NVIDIA's CUDA APIs. Some of the key features of PyCUDA include:

- ▶ Maps all of CUDA into Python.
- ▶ Enables run-time code generation (RTCG) for flexible, fast, automatically tuned codes.
- ▶ Added robustness: automatic management of object lifetimes, automatic error checking
- ▶ Added convenience: comes with ready-made on-GPU linear algebra, reduction, scan.
- ▶ Add-on packages for FFT and LAPACK available.
- ▶ Fast. Near-zero wrapping overhead.

1. Download [PyCUDA](#) and unpack it.

```
$ tar xzvf pycuda-VERSION.tar.gz
```

2. Configure, make and install PyCUDA.

```
$ cd pycuda-VERSION
$ ./configure.py --cuda-root=<CUDA Installation Path>
$ make
$ sudo make install
```

For more information, see [Installing PyCUDA on Linux](#). These instructions involve building and installing PyCUDA on the platform.

7.1. Updating CUDA

Existing installations of PyCUDA will not automatically work with a newly installed CUDA Toolkit. That is because PyCUDA will only work with a CUDA Toolkit that is already on the target system when PyCUDA was installed. This requires that PyCUDA be updated after the newer version of the CUDA Toolkit is installed. The steps below are the most reliable method to ensure that everything works in a compatible fashion after the CUDA Toolkit on your system has been upgraded.

1. Uninstall the existing PyCUDA installation.
2. Update CUDA. For more information, see the [CUDA Installation Guide](#).
3. Install PyCUDA using the new CUDA installation. For more information, see [Installing PyCUDA on Linux](#). Ensure you build PyCUDA with your updated CUDA version, by properly setting the `--cuda-root` option to point to the new CUDA installation when building PyCUDA.

Chapter 8. TROUBLESHOOTING

For troubleshooting support refer to your support engineer or post your questions onto the [NVIDIA Developer Forum](#).

Notice

THE INFORMATION IN THIS GUIDE AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION REFERENCED IN THIS GUIDE IS PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the product described in this guide shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

THE NVIDIA PRODUCT DESCRIBED IN THIS GUIDE IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE, AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE (INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

NVIDIA makes no representation or warranty that the product described in this guide will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this guide. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this guide, or (ii) customer product designs.

Other than the right for customer to use the information in this guide with the product, no other license, either expressed or implied, is hereby granted by NVIDIA under this guide. Reproduction of information in this guide is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, cuDNN, cuFFT, cuSPARSE, DIGITS, DGX, DGX-1, Jetson, Kepler, NVIDIA Maxwell, NCCL, NVLink, Pascal, Tegra, TensorRT, and Tesla are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2018 NVIDIA Corporation. All rights reserved.