NVIDIA TensorRT

Installation Guide | NVIDIA Docs

DI-08731-001_v8.0.3 | April 2024

Table of Contents

List of Tables

Chapter 1. Overview

The core of NVIDIA® TensorRT™ is a C++ library that facilitates high-performance inference on NVIDIA graphics processing units (GPUs). TensorRT takes a trained network, which consists of a network definition and a set of trained parameters, and produces a highly optimized runtime engine that performs inference for that network.

TensorRT provides API's via C++ and Python that help to express deep learning models via the Network Definition API or load a pre-defined model via the parsers that allow TensorRT to optimize and run them on an NVIDIA GPU. TensorRT applies graph optimizations, layer fusion, among other optimizations, while also finding the fastest implementation of that model leveraging a diverse collection of highly optimized kernels. TensorRT also supplies a runtime that you can use to execute this network on all of NVIDIA's GPU's from the Kepler generation onwards.

TensorRT also includes optional high speed mixed precision capabilities introduced in the Tegra™ X1, and extended with the Pascal™, Volta™, Turing™, and NVIDIA® Ampere GPU architectures.

Chapter 2. Getting Started

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

- The Windows zip package for TensorRT does not provide Python support. Python may be supported in the future.
- If you are using the TensorRT Python API and PyCUDA isn't already installed on your system, see Installing PyCUDA. If you encounter any issues with PyCUDA usage, you may 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; versions 10.2, 11.0 update 1, 11.1 update 1, 11.2 update 2, and 11.3 update 1 are supported.
- The TensorFlow to TensorRT model export requires TensorFlow 1.15.5.
- The PyTorch examples have been tested with PyTorch 1.8.1, but may work with older versions.
- The TensorRT ONNX parser has been tested with ONNX 1.8.0 and supports opset 11.
- If the target 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 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 or RPM 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 UFF files from TensorFlow models.

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 Now.
3. Select the version of TensorRT that you are interested in.
4. Select the check-box to agree to the license terms.
5. Click the package you want to install. Your download begins.

Chapter 4. Installing TensorRT

You can choose between the following installation options when installing TensorRT; Debian or RPM packages, a `pip` wheel file, a tar file, or a zip file.

The Debian and RPM installations automatically install any dependencies, however, it:

- requires `sudo` or root privileges to install
- provides no flexibility as to which location TensorRT is installed into
- requires that the CUDA Toolkit and cuDNN have also been installed using Debian or RPM packages.
- does not allow more than one minor version of TensorRT to be installed at the same time

The tar file provides more flexibility, such as installing multiple versions of TensorRT at the same time. However, you need to ensure that you have the necessary dependencies already installed and you must manage `LD_LIBRARY_PATH` yourself. For more information, see Tar File Installation.

The zip file is the only option currently for Windows. It does not support any other platforms besides Windows. Ensure that you have the necessary dependencies already installed. For more information, see Zip File Installation.

TensorRT versions: TensorRT is a product made up of separately versioned components. The version of 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	8.0.1	8.0.3	+1.0 when significant new

Product or Component		Previously Released Version	Current Version	Version Description
				capabilities are added. +0.1 when capabilities have been improved.
`nvinfer` libraries, headers, samples, and documentation.		8.0.1	8.0.3	+1.0 when the API or ABI changes in a non-compatible way. +0.1 when the API or ABI changes are backward compatible
UFF	`uff-converter-tf` Debian and RPM packages	8.0.1	8.0.3	+0.1 while we are developing the core functionality. Set to 1.0 when we have all base functionality in place.
	`uff-*.whl` file	0.6.8	0.6.9	
`graphsurgeon`	`graphsurgeon-tf` Debian and RPM packages	8.0.1	8.0.3	+0.1 while we are developing the core functionality. Set to 1.0 when we have all base functionality in place.
	`graphsurgeon-*.whl` file	0.4.4	0.4.5	
`onnx-graphsurgeon`	`onnx-graphsurgeon` Debian and RPM packages	8.0.1	8.0.3	+0.1 while we are developing the core functionality. Set to 1.0 when we have all base functionality in place.
	`onnx_graphsurgeon-*.whl` file	0.2.6	0.3.10	

Product or Component		Previously Released Version	Current Version	Version Description
`libnvinfer` Python packages[1]	▶ `python3-libnvinfer` ▶ `python3-libnvinfer-dev` Debian and RPM packages	8.0.1	8.0.3	+1.0 when the API or ABI changes in a non-compatible way. +0.1 when the API or ABI changes are backward compatible.
	`tensorrt.whl` file	8.0.1	8.0.3	

4.1. Debian Installation

This section contains instructions for a developer installation. This installation method is for new users or users who want the complete developer installation, including samples and documentation for both the C++ and Python APIs.

For advanced users who are already familiar with TensorRT and want to get their application running quickly, are using an NVIDIA CUDA container, or want to set up automation, follow the network repo installation instructions (refer to Using The NVIDIA CUDA Network Repo For Debian Installation).

Note:

- The following commands are examples for `amd64`, however, the commands are identical for `arm64`.
- When installing Python packages using this method, you must install dependencies manually with `pip`.

Ensure that you have the following dependencies installed.

- CUDA 11.0 update 3, 11.1 update 1, 11.2 update 2, 11.3 update 1, 11.4 update 4, 11.5 update 2, 11.6 update 2, 11.7 update 1, 11.8, 12.0 update 1, 12.1 update 1, 12.2 update 2, 12.3 update 2, or 12.4 update 1
- cuDNN 8.9.7 (Optional and not required for lean or dispatch runtime installations.)

1. Install CUDA according to the CUDA installation instructions.
2. Download the TensorRT local repo file that matches the Ubuntu version and CPU architecture that you are using.

[1] These components are not included in the zip file installation for Windows.

3. Install TensorRT from the Debian local repo package. Replace `ubuntuxx04`, `10.x.x`, and `cuda-x.x` with your specific OS version, TensorRT version, and CUDA version.

```
os="ubuntuxx04"
tag="10.x.x-cuda-x.x"
sudo dpkg -i nv-tensorrt-local-repo-${os}-${tag}_1.0-1_amd64.deb
sudo cp /var/nv-tensorrt-local-repo-${os}-${tag}/*-keyring.gpg /usr/share/keyrings/
sudo apt-get update
```

For the full C++ and Python runtimes

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

For the lean runtime only, instead of `tensorrt`

```
sudo apt-get install libnvinfer-lean10
sudo apt-get install libnvinfer-vc-plugin10
```

For lean runtime Python package

```
sudo apt-get install python3-libnvinfer-lean
```

For the dispatch runtime only, instead of `tensorrt`

```
sudo apt-get install libnvinfer-dispatch10
sudo apt-get install libnvinfer-vc-plugin10
```

For dispatch runtime Python package

```
sudo apt-get install python3-libnvinfer-dispatch
```

For all TensorRT Python packages without samples

```
python3 -m pip install numpy
sudo apt-get install python3-libnvinfer-dev
```

The following additional packages will be installed:

```
python3-libnvinfer
python3-libnvinfer-lean
python3-libnvinfer-dispatch
```

If you want to install Python packages for the lean or dispatch runtime *only*, specify these individually rather than installing the `dev` package.

If you want to run samples that require `onnx-graphsurgeon` or use the Python module for your own project

```
python3 -m pip install numpy onnx
sudo apt-get install onnx-graphsurgeon
```

4. Verify the installation.

```
dpkg-query -W tensorrt
```

You should see something similar to the following:

```
tensorrt 10.0.1.x-1+cuda12.4
```

4.1.1. Using The NVIDIA CUDA Network Repo For Debian Installation

This installation method is for advanced users who are already familiar with TensorRT and want to get their application running quickly or to set up automation, such as when using containers. New users or users who want the complete installation, including samples and documentation, should follow the local repo installation instructions (refer to Debian Installation).

Note: If you are using an NVIDIA CUDA container with cuDNN included, then the NVIDIA CUDA network repository will already be set up and you can skip step 1.

1. To install the NVIDIA CUDA network repository, follow the instructions at the CUDA Toolkit Download page.
   a). Select the Linux operating system.
   b). Select the desired architecture.
   c). Select the Ubuntu distribution.
   d). Select the desired Ubuntu version.
   e). Select the “deb (network)” installer type.
   f). Enter the commands provided into your terminal.

   You can omit the final `apt-get install` command if you do not require the entire CUDA toolkit. While installing TensorRT, `apt` downloads the required CUDA and cuDNN dependencies for you automatically.
2. Install the TensorRT package that fits your particular needs.
   a). For only running TensorRT C++ applications:
   ```
   sudo apt-get install libnvinfer8 libnvonnxparsers8 libnvparsers8 libnvinfer-plugin8
   ```
   b). For also building TensorRT C++ applications:
   ```
   sudo apt-get install libnvinfer-dev libnvonnxparsers-dev
    libnvparsers-dev libnvinfer-plugin-dev
   ```
   c). For running TensorRT Python applications:
   ```
   sudo apt-get install python3-libnvinfer
   ```
3. When using the NVIDIA CUDA network repository, Ubuntu will by default install TensorRT for the latest CUDA version. The following commands will install `libnvinfer8` for an older CUDA version and hold the `libnvinfer8` package at this version. Replace `8.x.x` with your version of TensorRT and `cudax.x` with your CUDA version for your install.
   ```
   version="8.x.x-1+cudax.x"
   sudo apt-get install libnvinfer8=${version} libnvonnxparsers8=${version} libnvparsers8=
   ${version} libnvinfer-plugin8=${version} libnvinfer-dev=${version} libnvonnxparsers-
   dev=${version} libnvparsers-dev=${version} libnvinfer-plugin-dev=${version} python3-
   libnvinfer=${version}

   sudo apt-mark hold libnvinfer8 libnvonnxparsers8 libnvparsers8 libnvinfer-plugin8
    libnvinfer-dev libnvonnxparsers-dev libnvparsers-dev libnvinfer-plugin-dev python3-
   libnvinfer
   ```
   If you want to upgrade to the latest version of TensorRT or the latest version of CUDA, then you can unhold the `libnvinfer8` package using the following command.
   ```
   sudo apt-mark unhold libnvinfer8 libnvonnxparsers8 libnvparsers8 libnvinfer-plugin8
    libnvinfer-dev libnvonnxparsers-dev libnvparsers-dev libnvinfer-plugin-dev python3-
   libnvinfer
   ```
   You may need to repeat these steps for `libcudnn8` to prevent cuDNN from being updated to the latest CUDA version. Refer to the TensorRT Release Notes for the specific version of cuDNN that was tested with your version of TensorRT. Example commands for downgrading and holding the cuDNN version can be found in Upgrading TensorRT. See the cuDNN Installation Guide for additional information.

   If the NVIDIA CUDA network repository and a TensorRT local repository are enabled at the same time you may observe package conflicts with either TensorRT or cuDNN. You will need to configure APT so that it prefers local packages over network packages. You can do this by creating a new file at `/etc/apt/preferences.d/local-repo` with the following lines:

```
Package: *
Pin: origin ""
Pin-Priority: 1001
```

Note: This preference change will affect more than just TensorRT in the unlikely event that you have other repositories which are also not downloaded over HTTP(S). To revert APT to its original behavior simply remove the newly created file.

4.2. App Server Installation

This type of installation is for cloud users or container users who will be going to production.

If you are going to be deploying the application to a server and running an already existing application in a minimal or standalone environment, then this type of installation allows you to set up a runtime environment instead of a full development environment. It provides a simple list of packages you can install if you want to run an application you've already developed.

When setting up servers which will host TensorRT powered applications, you can simply install any of the following Debian packages using `apt-get`:

- the `libnvinfer8` package (C++) plus any additional library packages you require, or
- the `python3-libnvinfer` package (Python 3.x).

4.3. RPM Installation

This section contains instructions for installing TensorRT from an RPM package. This installation method is for new users or users who want the complete installation, including samples and documentation.

For advanced users who are already familiar with TensorRT and want to get their application running quickly or to set up automation, follow the network repo installation instructions (see Using The NVIDIA CUDA Network Repo For RPM Installation).

Note:

- Before issuing the following commands, you'll need to replace `cudax.x`, `trt8.x.x.x`, and `yyyymmdd` with your specific CUDA version, TensorRT version, and package date.
- The following example commands are for `x86_64`, but the commands should be identical for `ppc64le`.

1. Download the TensorRT local repo file that matches the RHEL/CentOS version and CPU architecture you are using.
2. Install TensorRT from the RPM local repo package.

```
os="rhelx"
```

```
tag="cudax.x-trt8.x.x.x-yyyymmdd"
sudo rpm -Uvh nv-tensorrt-repo-${os}-${tag}-1-1.x86_64.rpm
sudo yum clean expire-cache
```

The packages which can be installed are:

```
graphsurgeon-tf.x86_64
libnvinfer-bin.x86_64
libnvinfer-devel.x86_64
libnvinfer-doc.x86_64
libnvinfer-plugin-devel.x86_64
libnvinfer-plugin8.x86_64
libnvinfer-samples.x86_64
libnvinfer8.x86_64
libnvonnxparsers-devel.x86_64
libnvonnxparsers8.x86_64
libnvparsers-devel.x86_64
libnvparsers8.x86_64
python3-libnvinfer.x86_64
python3-libnvinfer-devel.x86_64
tensorrt.x86_64
uff-converter-tf.x86_64
onnx-graphsurgeon.x86_64
```

Install TensorRT.

```
sudo yum install tensorrt
```

If using Python 3.x:

```
sudo yum install python3-libnvinfer-devel
```

The following additional packages will be installed:

```
python3-libnvinfer
```

For the UFF converter (only required if you plan to use TensorRT with TensorFlow):

```
sudo yum install uff-converter-tf
```

The `graphsurgeon-tf` package will also be installed with the above command.

If you would like to run the samples that require ONNX `graphsurgeon` or use the Python module for your own project, run:

```
sudo yum install onnx-graphsurgeon
```

3. Verify the installation.
   a). Run:

   ```
   rpm -qa | grep tensorrt
   ```

   You should see something similar to the following:

   ```
   tensorrt-8.0.3.x-1.cuda11.3.x86_64
   ```

   b). Run:

   ```
   rpm -qa | grep -e libnvinfer -e libnv.*parsers
   ```

   You should see something similar to the following:

   ```
   libnvinfer-doc-8.0.3-1.cuda11.3.x86_64
   libnvinfer-plugin8-8.0.3-1.cuda11.3.x86_64
   libnvinfer-devel-8.0.3-1.cuda11.3.x86_64
   libnvinfer-bin-8.0.3-1.cuda11.3.x86_64
   libnvinfer8-8.0.3-1.cuda11.3.x86_64
   libnvinfer-samples-8.0.3-1.cuda11.3.x86_64
   libnvinfer-plugin-devel-8.0.3-1.cuda11.3.x86_64
   libnvonnxparsers8-8.0.3-1.cuda11.3.x86_64
   libnvonnxparsers-devel-8.0.3-1.cuda11.3.x86_64
   libnvparsers8-8.0.3-1.cuda11.3.x86_64
   libnvparsers-devel-8.0.3-1.cuda11.3.x86_64
   python3-libnvinfer-8.0.3-1.cuda11.3.x86_64
   ```

```
python3-libnvinfer-devel-8.0.3-1.cuda11.3.x86_64
```

c). Run:

```
rpm -qa | grep graphsurgeon-tf
```

You should see something similar to the following:

```
graphsurgeon-tf-8.0.3-1.cuda11.3.x86_64
```

d). Run:

```
rpm -qa | grep uff-converter-tf
```

You should see something similar to the following:

```
uff-converter-tf-8.0.3-1.cuda11.3.x86_64
```

e). Run:

```
rpm -qa | grep onnx-graphsurgeon
```

You should see something similar to the following:

```
onnx-graphsurgeon-8.0.3-1.cuda11.3.x86_64
```

4.3.1. Using The NVIDIA CUDA Network Repo For RPM Installation

This installation method is for advanced users who are already familiar with TensorRT and want to get their application running quickly or to set up automation. New users or users who want the complete installation, including samples and documentation, should follow the local repo installation instructions (see RPM Installation).

Note: If you are using an NVIDIA CUDA container with cuDNN included, then the NVIDIA CUDA network repository will already be set up and you can skip step 1.

1. To install the NVIDIA CUDA network repository, follow the instructions at the CUDA Toolkit Download page for the latest CUDA version.
   a). Select the Linux operating system.
   b). Select the desired architecture.
   c). Select the CentOS or RHEL distribution.
   d). Select the desired CentOS/RHEL version.
   e). Select the “rpm (network)” installer type.
   f). Enter the commands provided into your terminal.

   You can omit the final `yum/dnf install` command if you do not require the entire CUDA toolkit. While installing TensorRT, `yum/dnf` downloads the required CUDA and cuDNN dependencies for you automatically.
2. Install the TensorRT package that fits your particular needs. When using the NVIDIA CUDA network repository, RHEL will by default install TensorRT for the latest CUDA version. If you need the libraries for other CUDA versions, refer to next step.
   a). For only running TensorRT C++ applications:

   ```
   sudo yum install libnvinfer8 libnvparsers8 libnvonnxparsers8 libnvinfer-plugin8
   ```

   b). For also building TensorRT C++ applications:

   ```
   sudo yum install libnvinfer-devel libnvparsers-devel libnvonnxparsers-devel
    libnvinfer-plugin-devel
   ```

c). For running TensorRT Python applications:

```
sudo yum install python3-libnvinfer
```

3. The following commands install `libnvinfer8` for an older CUDA version and hold the `libnvinfer8` package at this version. Replace `8.x.x` with your version of TensorRT and `cudax.x` with your CUDA version for your install.

```
version="8.x.x-1.cudax.x"
sudo yum downgrade libnvinfer8-${version} libnvparsers8-${version} libnvonnxparsers8-
${version} libnvinfer-plugin8-${version} libnvinfer-devel-${version} libnvparsers-devel-
${version} libnvonnxparsers-devel-${version} libnvinfer-plugin-devel-${version} python3-
libnvinfer-${version}


sudo yum install yum-plugin-versionlock
sudo yum versionlock libnvinfer8 libnvparsers8 libnvonnxparsers8 libnvinfer-plugin8
 libnvinfer-devel libnvparsers-devel libnvonnxparsers-devel libnvinfer-plugin-devel
 python3-libnvinfer
```

If you want to upgrade to the latest version of TensorRT or the latest version of CUDA, then you can unhold the `libnvinfer8` package using the following command.

```
sudo yum versionlock delete libnvinfer8 libnvparsers8 libnvonnxparsers8 libnvinfer-
plugin8 libnvinfer-devel libnvparsers-devel libnvonnxparsers-devel libnvinfer-plugin-
devel python3-libnvinfer
```

You may need to repeat these steps for `libcudnn8` to prevent cuDNN from being updated to the latest CUDA version. Refer to the TensorRT Release Notes for the specific version of cuDNN that was tested with your version of TensorRT. Example commands for downgrading and holding the cuDNN version can be found in Upgrading TensorRT. See the cuDNN Installation Guide for additional information.

4.4. Python Package Index Installation

This section contains instructions for installing TensorRT from the Python Package Index.

When installing TensorRT from the Python Package Index, you're not required to install TensorRT from a `.tar`, `.deb`, or `.rpm` package. All required libraries are included in the Python package. However, the header files, which may be needed if you want to access TensorRT C++ APIs or to compile plugins written in C++, are not included. Additionally, if you already have the TensorRT C++ library installed, using the Python package index version will install a redundant copy of this library, which may not be desirable. Refer to Tar File Installation for information on how to manually install TensorRT wheels that do not bundle the C++ libraries. You can stop after this section if you only need Python support.

The `tensorrt` Python wheel files only support Python versions 3.8 to 3.12 at this time and will not work with other Python versions. Only the Linux operating system and

x86_64 CPU architecture is currently supported. These Python wheel files are expected to work on RHEL 8 or newer and Ubuntu 20.04 or newer.

Note: If you do not have root access, you are running outside a Python virtual environment, or for any other reason you would prefer a user installation, then append `--user` to any of the `pip` commands provided.

1. Ensure the `pip` Python module is up-to-date and the `wheel` Python module is installed before proceeding or you may encounter issues during the TensorRT Python installation.

```
python3 -m pip install --upgrade pip
python3 -m pip install wheel
```

2. Install the TensorRT Python wheel.

```
python3 -m pip install --upgrade tensorrt
```

   The above `pip` command will pull in all the required CUDA libraries in Python wheel format from PyPI because they are dependencies of the TensorRT Python wheel. Also, it will upgrade `tensorrt` to the latest version if you had a previous version installed.

   A TensorRT Python Package Index installation is split into multiple modules:

   - TensorRT libraries (`tensorrt_libs`)
   - Python bindings matching the Python version in use (`tensorrt_bindings`)
   - Frontend source package, which pulls in the correct version of dependent TensorRT modules from pypi.nvidia.com (`tensorrt`)
   - You can append `-cu11` or `-cu12` to any of the Python modules if you require a different CUDA major version. When unspecified, the TensorRT Python meta-packages default to the CUDA 12.x variants, which is the latest CUDA version supported by TensorRT.

   Optionally, install the TensorRT lean or dispatch runtime wheels, which are similarly split into multiple Python modules. If you are only using TensorRT to run pre-built version compatible engines, you can install these wheels without installing the regular TensorRT wheel.

```
python3 -m pip install --upgrade tensorrt_lean
python3 -m pip install --upgrade tensorrt_dispatch
```

3. To verify that your installation is working, use the following Python commands to:

   - Import the `tensorrt` Python module.
   - Confirm that the correct version of TensorRT has been installed.
   - Create a `Builder` object to verify that your CUDA installation is working.

```
python3
>>> import tensorrt
>>> print(tensorrt.__version__)
>>> assert tensorrt.Builder(tensorrt.Logger())
```

   Use a similar procedure to verify that the lean and dispatch modules work as expected:

```
python3
```

```
>>> import tensorrt_lean as trt
>>> print(trt.__version__)
>>> assert trt.Runtime(trt.Logger())

python3
>>> import tensorrt_dispatch as trt
>>> print(trt.__version__)
>>> assert trt.Runtime(trt.Logger())
```

If the final Python command fails with an error message similar to the error message below, then you may not have the NVIDIA driver installed or the NVIDIA driver may not be working properly. If you are running inside a container, then try starting from one of the `nvidia/cuda:x.y-base-<os>` containers.

```
[TensorRT] ERROR: CUDA initialization failure with error 100. Please check your CUDA
 installation: ...
```

If the preceding Python commands worked, then you should now be able to run any of the TensorRT Python samples to further confirm that your TensorRT installation is working. For more information about TensorRT samples, refer to the NVIDIA TensorRT Sample Support Guide.

4.5. Tar File Installation

1. Install the following dependencies, if not already present:
   - CUDA 10.2, 11.0 update 1, 11.1 update 1, 11.2 update 2, or 11.3 update 1
   - cuDNN 8.2.1
   - Python 3 (Optional)
2. Download the TensorRT tar file that matches the CPU architecture and CUDA version you are using.
3. Choose where you want to install TensorRT. This tar file will install everything into a subdirectory called `TensorRT-8.x.x.x`.
4. Unpack the tar file.

   ```
   version="8.x.x.x"
   arch=$(uname -m)
   cuda="cuda-x.x"
   cudnn="cudnn8.x"
   tar xzvf TensorRT-${version}.Linux.${arch}-gnu.${cuda}.${cudnn}.tar.gz
   ```

   Where:

   - `8.x.x.x` is your TensorRT version
   - `cuda-x.x` is CUDA version `10.2` or `11.3`
   - `cudnn8.x` is cuDNN version `8.2`

   This directory will have sub-directories like `lib`, `include`, `data`, etc...

   ```
   ls TensorRT-${version}
   bin  data  doc  graphsurgeon  include  lib  onnx_graphsurgeon  python  samples  targets
    TensorRT-Release-Notes.pdf  uff
   ```

5. Add the absolute path to the TensorRT `lib` directory to the environment variable `LD_LIBRARY_PATH`:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<TensorRT-${version}/lib>
```

6. Install the Python TensorRT wheel file.

```
cd TensorRT-${version}/python

python3 -m pip install tensorrt-*-cp3x-none-linux_x86_64.whl
```

7. Install the Python UFF wheel file. This is only required if you plan to use TensorRT with TensorFlow.

```
cd TensorRT-${version}/uff

python3 -m pip install uff-0.6.9-py2.py3-none-any.whl
```

   Check the installation with:

```
which convert-to-uff
```

8. Install the Python `graphsurgeon` wheel file.

```
cd TensorRT-${version}/graphsurgeon

python3 -m pip install graphsurgeon-0.4.5-py2.py3-none-any.whl
```

9. Install the Python `onnx-graphsurgeon` wheel file.

```
cd TensorRT-${version}/onnx_graphsurgeon

python3 -m pip install onnx_graphsurgeon-0.3.10-py2.py3-none-any.whl
```

10. 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. You should be able to compile and execute the sample without additional settings. For more information, see the "Hello World" For TensorRT (sampleMNIST).
    c). The Python samples are in the `samples/python` directory.

4.6. Zip File Installation

This section contains instructions for installing TensorRT from a zip package on Windows 10.

Ensure that you have the following dependencies installed.

- CUDA 10.2, 11.0 update 1, 11.1 update 1, 11.2 update 2, or 11.3 update 1
- cuDNN 8.2.1

1. Download the TensorRT zip file that matches the Windows version you are using.
2. Choose where you want to install TensorRT. The zip file will install everything into a subdirectory called `TensorRT-8.x.x.x`. This new subdirectory will be referred to as `<installpath>` in the steps below.
3. Unzip the `TensorRT-8.x.x.x.Windows10.x86_64.cuda-x.x.cudnn8.x.zip` file to the location that you chose.

   Where:

- `8.x.x.x` is your TensorRT version
- `cuda-x.x` is CUDA version `10.2` or `11.3`
- `cudnn8.x` is cuDNN version `8.2`

4. Add the TensorRT library files to your system `PATH`. There are two ways to accomplish this task:
   a). Leave the DLL files where they were unzipped and add `<installpath>/lib` to your system `PATH`. You can add a new path to your system `PATH` using the steps below.
      i. Press the Windows key and search for "environment variables" which should present you with the option Edit the system environment variables and click it.
      ii. Click Environment Variables... at the bottom of the window.
      iii. Under System variables, select Path and click Edit....
      iv. Click either New or Browse to add a new item that contains `<installpath>/lib`.
      v. Continue to click OK until all the newly opened windows are closed.
      vi. If your cuDNN libraries were not copied to the CUDA installation directory and instead left where they were unzipped, then repeat the above steps for the cuDNN `bin` directory.
   b). Copy the DLL files from `<installpath>/lib` to your CUDA installation directory, for example, `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vX.Y\bin`, where `vX.Y` is your CUDA version. The CUDA installer should have already added the CUDA path to your system `PATH`.
5. To verify that your installation is working you should open a Visual Studio Solution file from one of the samples, such as "Hello World" For TensorRT (sampleMNIST), and confirm that you are able to build and run the sample.

   If you want to use TensorRT in your own project, ensure that the following is present in your Visual Studio Solution project properties:
   a). `<installpath>/lib` has been added to your `PATH` variable and is present under VC ++ Directories > Executable Directories.
   b). `<installpath>/include` is present under C/C++ > General > AdditionalDirectories.
   c). `nvinfer.lib` and any other `LIB` files that your project requires are present under Linker > Input > Additional Dependencies.

   > Note: In order to build the included samples, you should have Visual Studio 2017 (https://visualstudio.microsoft.com/downloads/) installed. The community edition is sufficient to build the TensorRT samples.

6. If you are using TensorFlow or PyTorch, install the `uff`, `graphsurgeon`, and `onnx_graphsurgeon` wheel packages. You must prepare the Python environment before installing `uff`, `graphsurgeon` or `onnx_graphsurgeon`.

   If using Python 3.x:

```
python3 -m pip install <installpath>\graphsurgeon\graphsurgeon-0.4.5-py2.py3-none-any.whl
python3 -m pip install <installpath>\uff\uff-0.6.9-py2.py3-none-any.whl
```

```
python3 -m pip install <installpath>\onnx_graphsurgeon\onnx_graphsurgeon-0.3.10-py2.py3-none-any.whl
```

4.7. Additional Installation Methods

Aside from installing TensorRT from the product package, you can also install TensorRT from the following locations.

TensorRT container
The TensorRT container provides an easy method for deploying TensorRT with all necessary dependencies already packaged in the container. For information about installing TensorRT via a container, see the TensorRT Container Release Notes.

JetPack
JetPack bundles all Jetson platform software, including TensorRT. Use it to flash your Jetson Developer Kit with the latest OS image, install NVIDIA SDKs, and jump-start your development environment. For information about installing TensorRT through JetPack, see the JetPack documentation.

For JetPack downloads, see Develop: Jetpack.

NVIDIA DriveWorks
With every release, TensorRT delivers features to make the DRIVE Development Platform an excellent computing platform for Autonomous Driving. For more information about installing TensorRT through DriveWorks, see the DriveWorks documentation.

For DriveWorks downloads, see NVIDIA Developer: Drive Downloads.

Chapter 5. Upgrading TensorRT

Upgrading TensorRT to the latest version is only supported when the currently installed TensorRT version is equal to or newer than the last two public GA releases. For example, TensorRT 8.0.x supports upgrading from TensorRT 7.1.x and 7.2.x. If you want to upgrade from an unsupported version, then you should upgrade incrementally until you reach the latest version of TensorRT. If you have an EA version of TensorRT installed, you should first upgrade to the corresponding GA version.

5.1. Ubuntu And Windows Users

The following section provides step-by-step instructions for upgrading TensorRT for Ubuntu and Windows users.

5.1.1. Upgrading From TensorRT 7.x.x To TensorRT 8.0.x

These upgrade instructions are for Ubuntu and Windows users only. When upgrading from TensorRT 7.x.x to TensorRT 8.0.x, ensure you are familiar with the following.

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. If you installed TensorRT 7.x.x via a Debian package and you upgrade to TensorRT 8.0.x, your documentation, samples, and headers will all be updated to the TensorRT 8.0.x content. After you have downloaded the new local repo, use `apt-get` to upgrade your system to the new version of TensorRT.

```
os="ubuntuxx04"
tag="cudax.x-trt8.x.x.x-yyyymmdd"
sudo dpkg -i nv-tensorrt-repo-${os}-${tag}_1-1_amd64.deb

sudo apt-get update
sudo apt-get install tensorrt libcudnn8
```

  Python 2.7 is no longer supported and can be removed:

```
sudo apt-get purge python-libnvinfer
```

  If using Python 3.x:

```
sudo apt-get install python3-libnvinfer-dev
```

- If you are using the `uff-converter` and/or `graphsurgeon`, then you should also upgrade those Debian packages to the latest versions.

```
sudo apt-get install uff-converter-tf graphsurgeon-tf onnx-graphsurgeon
```

- After you upgrade, ensure you have a directory `/usr/src/tensorrt` and the corresponding version shown by the `dpkg -l tensorrt` command is `8.x.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 libraries.
- If you are currently or were previously using the NVIDIA CUDA network repository, then it may conflict with the version of `libcudnn8` that is expected to be installed from the local repository for TensorRT. The following commands will change `libcudnn8` to version 8.2.x.x, which is supported and tested with TensorRT 8.0.x, and hold the `libcudnn8` package at this version. Replace `cudax.x` with the appropriate CUDA version for your install.

```
version="8.2.x.x-1+cudax.x"
sudo apt-get install libcudnn8=${version} libcudnn8-dev=${version}
sudo apt-mark hold libcudnn8 libcudnn8-dev
```

Using a tar file

- If you are upgrading using the tar file installation method, then install TensorRT into a new location. Tar file installations can support multiple use cases including having a full installation of TensorRT 7.x.x with headers and documentation side-by-side with a full installation of TensorRT 8.0.x. 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 installation will not remove the previously installed packages. Unless a side-by-side installation is desired, it would be best to remove the previously installed `libnvinfer8`, `libnvinfer-dev`, `libnvinfer-samples` and other related packages to avoid confusion.

Using a zip file

- If you are upgrading using the zip file installation method, then install TensorRT into a new location. Zip file installations can support multiple use cases including having a full installation of TensorRT 7.x.x with headers and documentation side-by-side with a full installation of TensorRT 8.0.x. 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.
- After unzipping the new version of TensorRT you will need to either update the `PATH` environment variable to point to the new install location or copy the DLL files to the location where you previously installed the TensorRT libraries. Refer to Zip File Installation for more information about setting the `PATH` environment variable.

5.2. RedHat And CentOS Users

The following section provides step-by-step instructions for upgrading TensorRT for RedHat and CentOS users.

5.2.1. Upgrading From TensorRT 7.x.x To TensorRT 8.0.x

These upgrade instructions are for Red Hat Enterprise Linux (RHEL) and CentOS users only. When upgrading from TensorRT 7.x.x to TensorRT 8.0.x, ensure you are familiar with the following.

Using an RPM file

- The RPM packages are designed to upgrade your development environment without removing any runtime components that other packages and programs might rely on. If you installed TensorRT 7.x.x via an RPM package and you want to upgrade to TensorRT 8.0.x, your documentation, samples, and headers will all be updated to the TensorRT 8.0.x content. After you have downloaded the new local repo, issue:

```
os="rhelx"
tag="cudax.x-trt8.x.x.x-yyyymmdd"
sudo rpm -Uvh nv-tensorrt-repo-${os}-${tag}-1-1.x86_64.rpm
sudo yum clean expire-cache
sudo yum install tensorrt libcudnn8
```

  Python 2.7 is no longer supported and can be removed:

```
sudo yum erase python-libnvinfer
```

  If using Python 3.x:

```
sudo yum install python3-libnvinfer-devel
```

- If using `uff-converter` and/or `graphsurgeon`:

```
sudo yum install uff-converter-tf graphsurgeon-tf onnx-graphsurgeon
```

- After you upgrade, ensure you see the `/usr/src/tensorrt` directory and the corresponding version shown by the `rpm -qa tensorrt` command is `8.x.x.x`.
- If you are currently or were previously using the NVIDIA CUDA network repository, then it may conflict with the version of `libcudnn8` that is expected to be installed from the local repository for TensorRT. The following commands will change `libcudnn8` to version 8.2.x.x, which is supported and tested with TensorRT 8.0.x, and hold the `libcudnn8` package at this version. Replace `cudax.x` with the appropriate CUDA version for your install.

```
version="8.2.x.x-1.cudax.x"
sudo yum downgrade libcudnn8-${version} libcudnn8-devel-${version}
sudo yum install yum-plugin-versionlock
sudo yum versionlock libcudnn8 libcudnn8-devel
```

Chapter 6. Uninstalling TensorRT

This section provides step-by-step instructions for ways in which you can uninstall TensorRT.

To uninstall TensorRT using the untarred file, simply delete the tar files and reset `LD_LIBRARY_PATH` to its original value.

To uninstall TensorRT using the zip file, simply delete the unzipped files and remove the newly added path from the `PATH` environment variable.

To uninstall TensorRT using the Debian or RPM packages, follow these steps:

1. Uninstall `libnvinfer8` which was installed using the Debian or RPM packages.
   ```
   sudo apt-get purge "libnvinfer*"
   ```
   Or
   ```
   sudo yum erase "libnvinfer*"
   sudo yum erase "nv-tensorrt-repo*"
   ```
2. Uninstall `uff-converter-tf`, `graphsurgeon-tf`, and `onnx-graphsurgeon` which were also installed using the Debian or RPM packages.
   ```
   sudo apt-get purge graphsurgeon-tf onnx-graphsurgeon
   ```
   Or
   ```
   sudo yum erase graphsurgeon-tf onnx-graphsurgeon
   ```
   The `uff-converter-tf` package will also be removed with the above command.

   You can use the following command to uninstall `uff-converter-tf` and not remove `graphsurgeon-tf`, however, it is no longer required.
   ```
   sudo apt-get purge uff-converter-tf
   ```
   Or
   ```
   sudo yum erase uff-converter-tf
   ```
   You can later use `autoremove` to uninstall `graphsurgeon-tf` as well.
   ```
   sudo apt-get autoremove
   ```
   Or
   ```
   sudo yum autoremove
   ```
3. Uninstall the Python TensorRT wheel file.

   If using Python 3.x:
   ```
   sudo pip3 uninstall tensorrt
   ```
4. Uninstall the Python UFF wheel file.

   If using Python 3.x:
   ```
   sudo pip3 uninstall uff
   ```

5. Uninstall the Python GraphSurgeon wheel file.

   If using Python 3.x:

   ```
   sudo pip3 uninstall graphsurgeon
   ```

6. Uninstall the Python ONNX GraphSurgeon wheel file.

   If using Python 3.x:

   ```
   sudo pip3 uninstall onnx-graphsurgeon
   ```

Chapter 7. Installing PyCUDA

This section provides useful information regarding PyCUDA including how to install.

> **!** 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 first, 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.

To install PyCUDA first make sure `nvcc` is in your `PATH`, then issue the following command:

```
pip install 'pycuda<2021.1'
```

If you encounter any issues with PyCUDA usage after installing PyCUDA with the above command, you may need to recompile it yourself. For more information, see Installing PyCUDA on Linux.

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 NVIDIA CUDA Installation Guide.
3. Install PyCUDA. To install PyCUDA, issue the following command:

```
pip install 'pycuda<2021.1'
```

Chapter 8. Troubleshooting

For troubleshooting support refer to your support engineer or post your questions onto the NVIDIA Developer Forum.

NVIDIA Developer Forum

Appendix A. Appendix

The following section provides our list of acknowledgements.

A.1. ACKNOWLEDGEMENTS

TensorRT uses elements from the following software, whose licenses are reproduced below.

Google Protobuf

This license applies to all parts of Protocol Buffers except the following:

- Atomicops support for generic gcc, located in `src/google/protobuf/stubs/atomicops_internals_generic_gcc.h`. This file is copyrighted by Red Hat Inc.
- Atomicops support for AIX/POWER, located in `src/google/protobuf/stubs/atomicops_internals_power.h`. This file is copyrighted by Bloomberg Finance LP.

Copyright 2014, Google Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner of the input file used when generating it. This code is not standalone and requires a support library to be linked with it. This support library is itself covered by the above license.

Google Flatbuffers

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not

include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   a). You must give any other recipients of the Work or Derivative Works a copy of this License; and

   b). You must cause any modified files to carry prominent notices stating that You changed the files; and

c). You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

d). If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor

be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright 2014 Google Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at: http://www.apache.org/licenses/LICENSE-2.0.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

BVLC Caffe

COPYRIGHT

All contributions by the University of California:

Copyright (c) 2014, 2015, The Regents of the University of California (Regents) All rights reserved.

All other contributions:

Copyright (c) 2014, 2015, the respective contributors All rights reserved.

Caffe uses a shared copyright model: each contributor holds copyright over their contributions to Caffe. The project versioning records all such contribution and copyright details. If a contributor wants to further mark their specific copyright on a particular contribution, they should indicate their copyright solely in the commit message of the change when it is committed.

LICENSE

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CONTRIBUTION AGREEMENT

By contributing to the BVLC/Caffe repository through pull-request, comment, or otherwise, the contributor releases their content to the license and copyright terms herein.

half.h

Copyright (c) 2012-2017 Christian Rau <rauy@users.sourceforge.net>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

jQuery.js

jQuery.js is generated automatically under doxygen.

In all cases TensorRT uses the functions under the MIT license.

CRC

policies, either expressed or implied, of the Regents of the University of California.

Note: The copyright of UC Berkeley's Berkeley Software Distribution ("BSD") source has been updated. The copyright addendum may be found at ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change and is

William Hoskins

Director, Office of Technology Licensing

University of California, Berkeley

getopt.c

Copyright (c) 2002 Todd C. Miller <Todd.Miller@courtesan.com>

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F39502-99-1-0512.

Copyright (c) 2000 The NetBSD Foundation, Inc.

All rights reserved.

This code is derived from software contributed to The NetBSD Foundation by Dieter Baron and Thomas Klausner.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform 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 document. NVIDIA accepts no 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 document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Arm

Arm, AMBA and Arm Powered are registered trademarks of Arm Limited. Cortex, MPCore and Mali are trademarks of Arm Limited. "Arm" is used to represent Arm Holdings plc; its operating company Arm Limited; and the regional subsidiaries Arm Inc.; Arm KK; Arm Korea Limited.; Arm Taiwan Limited; Arm France SAS; Arm Consulting (Shanghai) Co. Ltd.; Arm Germany GmbH; Arm Embedded Technologies Pvt. Ltd.; Arm Norway, AS and Arm Sweden AB.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

Blackberry/QNX

Copyright © 2020 BlackBerry Limited. All rights reserved.

Trademarks, including but not limited to BLACKBERRY, EMBLEM Design, QNX, AVIAGE, MOMENTICS, NEUTRINO and QNX CAR are the trademarks or registered trademarks of BlackBerry Limited, used under license, and the exclusive rights to such trademarks are expressly reserved.

Google

Android, Android TV, Google Play and the Google Play logo are trademarks of Google, Inc.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com

Trademarks

NVIDIA, the NVIDIA logo, and BlueField, CUDA, DALI, DRIVE, Hopper, JetPack, Jetson AGX Xavier, Jetson Nano, Maxwell, NGC, Nsight, Orin, Pascal, Quadro, Tegra, TensorRT, Triton, Turing and Volta 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

© 2017-2024 NVIDIA Corporation & affiliates. All rights reserved.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com