cuDNN Installation Guide

DU-08670-001_v8.0.5 | November 2020

Table of Contents

Chapter 1. Overview

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

Deep learning researchers and framework developers worldwide rely on cuDNN for high-performance GPU acceleration. It allows them to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning. cuDNN accelerates widely used deep learning frameworks and is freely available to members of the NVIDIA Developer Program™.

Chapter 2. Installing cuDNN On Linux

2.1. Prerequisites

Ensure you meet the following requirements before you install cuDNN.

- For the latest compatibility software versions of the OS, CUDA, the CUDA driver, and the NVIDIA hardware, see the cuDNN Support Matrix.

2.1.1. Installing NVIDIA Graphics Drivers

About this task

Install up-to-date NVIDIA graphics drivers on your Linux system.

Procedure

1. Go to: NVIDIA download drivers
2. Select the GPU and OS version from the drop-down menus.
3. Download and install the NVIDIA graphics driver as indicated on that web page. For more information, select the **ADDITIONAL INFORMATION** tab for step-by-step instructions for installing a driver.
4. Restart your system to ensure the graphics driver takes effect.

2.1.2. Installing The CUDA Toolkit For Linux

About this task

Refer to the following instructions for installing CUDA on Linux, including the CUDA driver and toolkit: NVIDIA CUDA Installation Guide for Linux.

2.2. Downloading cuDNN For Linux

Before you begin

In order to download cuDNN, ensure you are registered for the NVIDIA Developer Program.

Procedure

1. Go to: NVIDIA cuDNN home page.
2. Click **Download**.
3. Complete the short survey and click **Submit**.
4. Accept the Terms and Conditions. A list of available download versions of cuDNN displays.
5. Select the cuDNN version you want to install. A list of available resources displays.

2.3. Installing cuDNN On Linux

About this task

The following steps describe how to build a cuDNN dependent program. Choose the installation method that meets your environment needs. For example, the tar file installation applies to all Linux platforms, and the Debian installation package applies to Ubuntu 16.04, 18.04 and 20.04.

In the following sections:

- your CUDA directory path is referred to as `/usr/local/cuda/`
- your cuDNN download path is referred to as `<cudnnpath>`

2.3.1. Installing From A Tar File

Before issuing the following commands, you'll need to replace `x.x` and `v8.x.x.x` with your specific CUDA version and cuDNN version and package date.

Procedure

1. Navigate to your `<cudnnpath>` directory containing the cuDNN Tar file.
2. Unzip the cuDNN package.

```
$ tar -xzvf cudnn-x.x-linux-x64-v8.x.x.x.tgz
```

or

```
$ tar -xzvf cudnn-x.x-linux-aarch64sbsa-v8.x.x.x.tgz
```

3. Copy the following files into the CUDA Toolkit directory, and change the file permissions.

```
$ sudo cp cuda/include/cudnn*.h /usr/local/cuda/include
$ sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64

$ sudo chmod a+r /usr/local/cuda/include/cudnn*.h /usr/local/cuda/lib64/libcudnn*
```

2.3.2. Installing From A Debian File

Before issuing the following commands, you'll need to replace `x.x` and `8.x.x.x` with your specific CUDA version and cuDNN version and package date.

About this task

Procedure

1. Navigate to your `<cudnnpath>` directory containing the cuDNN Debian file.
2. Install the runtime library, for example:

   ```
   sudo dpkg -i libcudnn8_x.x.x-1+cudax.x_amd64.deb
   ```

   or

   ```
   sudo dpkg -i libcudnn8_x.x.x-1+cudax.x_arm64.deb
   ```
3. Install the developer library, for example:

   ```
   sudo dpkg -i libcudnn8-dev_8.x.x.x-1+cudax.x_amd64.deb
   ```

   or

   ```
   sudo dpkg -i libcudnn8-dev_8.x.x.x-1+cudax.x_arm64.deb
   ```
4. Install the code samples and the cuDNN library documentation, for example:

   ```
   sudo dpkg -i libcudnn8-samples_8.x.x.x-1+cudax.x_amd64.deb
   ```

   or

   ```
   sudo dpkg -i libcudnn8-samples_8.x.x.x-1+cudax.x_arm64.deb
   ```

2.3.3. Installing From An RPM File

About this task

Procedure

1. Download the rpm package `libcudnn*.rpm` to the local path.
2. Install the rpm package from the local path. This will install the cuDNN libraries.

   ```
   rpm -ivh libcudnn8-*.x86_64.rpm
   rpm -ivh libcudnn8-devel-*.x86_64.rpm
   rpm -ivh libcudnn8-samples-*.x86_64.rpm
   ```

   or

   ```
   rpm -ivh libcudnn8-*.aarch64.rpm
   rpm -ivh libcudnn8-devel-*.aarch64.rpm
   rpm -ivh libcudnn8-samples-*.aarch64.rpm
   ```

2.4. Verifying The cuDNN Install On Linux

About this task

To verify that cuDNN is installed and is running properly, compile the `mnistCUDNN` sample located in the `/usr/src/cudnn_samples_v8` directory in the Debian file.

Procedure

1. Copy the cuDNN sample to a writable path.
   ```
   $cp -r /usr/src/cudnn_samples_v8/ $HOME
   ```
2. Go to the writable path.
   ```
   $ cd  $HOME/cudnn_samples_v8/mnistCUDNN
   ```
3. Compile the `mnistCUDNN` sample.
   ```
   $make clean && make
   ```
4. Run the `mnistCUDNN` sample.
   ```
   $ ./mnistCUDNN
   ```
   If cuDNN is properly installed and running on your Linux system, you will see a message similar to the following:
   ```
   Test passed!
   ```

2.5. Upgrading From v7 To v8

Since version 8 can coexist with previous versions of cuDNN, if the user has an older version of cuDNN such as v6 or v7, installing version 8 will not automatically delete an older revision. Therefore, if the user wants the latest version, install cuDNN version 8 by following the installation steps.

About this task

To upgrade from v7 to v8 for RHEL, run:

```
sudo rpm --upgrade *.rpm
```

To upgrade from v7 to v8 for Ubuntu, run:

```
sudo dpkg -i libcudnn*.deb
```

To switch between v7 and v8 installations, issue `sudo update-alternatives --config libcudnn` and choose the appropriate cuDNN version.

2.6. Troubleshooting

About this task

Join the NVIDIA Developer Forum to post questions and follow discussions.

Chapter 3. Installing cuDNN On Windows

3.1. Prerequisites

Ensure you meet the following requirements before you install cuDNN.

- For the latest compatibility software versions of the OS, CUDA, the CUDA driver, and the NVIDIA hardware, see the cuDNN Support Matrix.

3.1.1. Installing NVIDIA Graphic Drivers

Install up-to-date NVIDIA graphics drivers on your Windows system.

Procedure

1. Go to: NVIDIA download drivers
2. Select the GPU and OS version from the drop-down menus.
3. Download and install the NVIDIA driver as indicated on that web page. For more information, select the **ADDITIONAL INFORMATION** tab for step-by-step instructions for installing a driver.
4. Restart your system to ensure the graphics driver takes effect.

3.1.2. Installing The CUDA Toolkit For Windows

About this task

Refer to the following instructions for installing CUDA on Windows, including the CUDA driver and toolkit: NVIDIA CUDA Installation Guide for Windows.

3.2. Downloading cuDNN For Windows

Before you begin

In order to download cuDNN, ensure you are registered for the NVIDIA Developer Program.

Procedure

1. Go to: NVIDIA cuDNN home page.
2. Click **Download**.
3. Complete the short survey and click **Submit**.
4. Accept the Terms and Conditions. A list of available download versions of cuDNN displays.
5. Select the cuDNN version to want to install. A list of available resources displays.
6. Extract the cuDNN archive to a directory of your choice.

3.3. Installing cuDNN On Windows

The following steps describe how to build a cuDNN dependent program.

About this task

Before issuing the following commands, you'll need to replace `x.x` and `8.x.x.x` with your specific CUDA version and cuDNN version and package date.

In the following sections the CUDA v9.0 is used as example:

- Your CUDA directory path is referred to as `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vx.x`
- Your cuDNN directory path is referred to as `<installpath>`

Procedure

1. Navigate to your `<installpath>` directory containing cuDNN.
2. Unzip the cuDNN package.
   `cudnn-x.x-windows-x64-v8.x.x.x.zip`
   or
   `cudnn-x.x-windows10-x64-v8.x.x.x.zip`
3. Copy the following files into the CUDA Toolkit directory.
   a). Copy `<installpath>\cuda\bin\cudnn*.dll` to `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vx.x\bin`.
   b). Copy `<installpath>\cuda\include\cudnn*.h` to `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vx.x\include`.
   c). Copy `<installpath>\cuda\lib\x64\cudnn*.lib` to `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vx.x\lib\x64`.
4. Set the following environment variables to point to where cuDNN is located. To access the value of the `$(CUDA_PATH)` environment variable, perform the following steps:

a). Open a command prompt from the **Start** menu.
b). Type `Run` and hit **Enter**.
c). Issue the `control sysdm.cpl` command.
d). Select the **Advanced** tab at the top of the window.
e). Click **Environment Variables** at the bottom of the window.
f). Ensure the following values are set:

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

5. Include `cudnn.lib` in your Visual Studio project.
   a). Open the Visual Studio project and right-click on the project name.
   b). Click **Linker > Input > Additional Dependencies**.
   c). Add `cudnn.lib` and click **OK**.

3.4. Upgrading From v7 To v8

About this task

Navigate to your `<installpath>` directory containing cuDNN and delete the old cuDNN `lib` and `header` files. Reinstall the latest cuDNN version by following the steps in Installing cuDNN On Windows.

3.5. Troubleshooting

About this task

Join the NVIDIA Developer Forum to post questions and follow discussions.

Chapter 4. Cross-compiling cuDNN Samples

This section describes how to cross-compile cuDNN samples.

4.1. NVIDIA DRIVE OS Linux

Follow the below steps to cross-compile samples on NVIDIA DRIVE OS Linux.

4.1.1. Installing The For DRIVE OS

Before issuing the following commands, you'll need to replace `x-x` with your specific version.

1. Download the for Ubuntu package:`cuda*ubuntu*_amd64.deb`
2. Download the cross compile package: `cuda*-cross-aarch64*_all.deb`
3. Execute the following commands:

```
sudo dpkg -i cuda*ubuntu*_amd64.deb
sudo apt-get update
sudo apt-get install cuda-toolkit-x-x -y
sudo apt-get install cuda-cross-aarch64* -y
```

4.1.2. Installing For DRIVE OS

1. Download the Ubuntu package for your preferred version: `*libcudnn8-cross-aarch64_*.deb`
2. Download the cross compile package: `libcudnn8-dev-cross-aarch64_*.deb`
3. Execute the following commands:

```
sudo dpkg -i *libcudnn8-cross-aarch64_*.deb
sudo dpkg -i libcudnn8-dev-cross-aarch64_*.deb
```

4.1.3. Cross-compiling Samples For DRIVE OS

Copy the `cudnn_samples_v8` directory to your home directory:

```
$ cp -r /usr/src/cudnn_samples_v8 $HOME
```

For each sample, execute the following commands:

```
$ cd $HOME/cudnn_samples_v8/(each sample)
$ make TARGET_ARCH=aarch64
```

4.2. QNX

Follow the below steps to cross-compile cuDNN samples on QNX:

4.2.1. Installing The For QNX

Before issuing the following commands, you'll need to replace `x-x` with your specific version.

1. Download the for Ubuntu package:`cuda*ubuntu*_amd64.deb`
2. Download the cross compile package: `cuda*-cross-aarch64*_all.deb`
3. Execute the following commands:

```
sudo dpkg -i cuda*ubuntu*_amd64.deb
sudo dpkg -i cuda*-cross-aarch64*_all.deb
sudo apt-get update
sudo apt-get install cuda-toolkit-x-x -y
sudo apt-get install cuda-cross-qnx -y
```

4.2.2. Installing For QNX

1. Download the Ubuntu package for your preferred version: `*libcudnn8-cross-aarch64_*.deb`
2. Download the cross compile package: `libcudnn8-devel-cross-aarch64_*.deb`
3. Execute the following commands:

```
sudo dpkg -i *libcudnn8-cross-aarch64_*.deb
sudo dpkg -i libcudnn8-dev-cross-aarch64_*.deb
```

4.2.3. Set The Environment Variables

To set the environment variables, issue the following commands:

```
export CUDA_PATH={PATH}/install/cuda/
export QNX_HOST={PATH}/host/linux/x86_64
export QNX_TARGET={PATH}/target/qnx7
```

4.2.4. Cross-compiling Samples For QNX

Copy the `cudnn_samples_v8` directory to your home directory:

```
$ cp -r /usr/src/cudnn_samples_v8 $HOME
```

Before issuing the following commands, you'll need to replace `8.x.x` with your specific version.

For each sample, execute the following commands:

```
$ cd $HOME/cudnn_samples_v8/(each sample)
$ make TARGET_OS=QNX TARGET_ARCH=aarch64 HOST_COMPILER={SET FULL PATH to YOUR CROSS
 COMPILER}
(for example: make TARGET_OS=QNX TARGET_ARCH=aarch64 HOST_COMPILER=$QNX_HOST/usr/bin/aarch64-
unknown-nto-qnx8.x.x-g++)
```

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.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

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

ARM

ARM, AMBA and ARM Powered are registered trademarks of ARM Limited. Cortex, MPCore and Mali are trademarks of ARM Limited. All other brands or product names are the property of their respective holders. "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.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

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

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, CUDA Toolkit, cuDNN, DALI, DIGITS, DGX, DGX-1, DGX-2, DGX Station, DLProf, GPU, JetPack, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NVCaffe, NVIDIA Ampere GPU architecture, NVIDIA Deep Learning SDK, NVIDIA Developer Program, NVIDIA GPU Cloud, NVLink, NVSHMEM, PerfWorks, Pascal, SDK Manager, T4, Tegra, TensorRT, TensorRT Inference Server, Tesla, TF-TRT, Triton Inference Server, 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

© 2017-2020 NVIDIA Corporation. All rights reserved.

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