NVIDIA TensorRT

Support Matrix | NVIDIA Docs

TRM-09025-001 _v8.4.1 | June 2022

Table of Contents

Chapter 1. Features for Platforms and Software

This section lists the supported NVIDIA® TensorRT™ features based on which platform and software.

Table 1. List of Supported Features per Platform

	Linux x86-64	Windows x64	Linux ppc64le	Linux AArch64
	8.4.x	8.4.x	8.0.x	8.4.x
Supported NVIDIA CUDA® versions	11.7[1] 11.6 update 2 11.5 update 2[2] 11.4 update 4 11.3 update 1[3] 11.2 update 2[4] 11.1 update 1[5]	11.7[7] 11.6 update 2 11.5 update 2[8] 11.4 update 4[9] 11.3 update 1[10] 11.2 update 2[11] 11.1 update 1[12]	11.3 update 1	11.6 update 2 11.4

[1] CUDA 11.7 added a feature called Lazy loading, however, this feature is not supported by TensorRT 8.4 because the CUDA 11.x binaries were built with CUDA Toolkit 11.6.

[7] CUDA 11.7 added a feature called Lazy loading, however, this feature is not supported by TensorRT 8.4 because the CUDA 11.x binaries were built with CUDA Toolkit 11.6.

[2] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[8] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[9] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[3] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[10] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[4] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[11] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[5] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

	Linux x86-64	Windows x64	Linux ppc64le	Linux AArch64
	8.4.x	8.4.x	8.0.x	8.4.x
	11.0 update 1[6] 10.2	11.0 update 1[13] 10.2		
Supported cuBLAS versions	11.10.1.x 11.9.2.x 11.7.4.6 11.6.5.2 11.5.1.109 11.4.1.1043 11.3.0.106 11.2.0.252 10.2.3.254	11.10.1.x 11.9.2.x 11.7.4.6 11.6.5.2 11.5.1.109 11.4.1.1043 11.3.0.106 11.2.0.252 10.2.3.254	11.5.1.109	11.9.2.x 11.6.5.x
Supported cuDNN versions	cuDNN 8.4.1	cuDNN 8.4.1	cuDNN 8.2.1	cuDNN 8.3.3
TensorRT Python API	Yes	Yes	Yes	Yes
NvUffParser	Yes	Yes	Yes	Yes
NvOnnxParser	Yes	Yes	Yes	Yes
Loops	Yes	Yes	Yes	Yes

Note:

- Serialized engines are not portable across platforms or TensorRT versions.
- Refer to the minimum compatible driver versions in the *NVIDIA CUDA Release Notes* for specific NVIDIA Driver versions.

[6] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

[13] These CUDA versions are supported using a single build, built with CUDA toolkit 11.6. It is compatible with all CUDA 11.x versions and only requires driver 450.x.

Chapter 2. Layers and Features

The section lists the supported TensorRT layers and each of the features.

About this task

Note:

- **Supports broadcast** indicates support for broadcast in this layer. This layer allows its two input tensors to be of dimensions [1, 5, 4, 3] and [1, 5, 1, 1], and its output is [1, 5, 4, 3]. The second input tensor has been broadcast in the innermost two dimensions.
- **Supports broadcast across batch** indicates support for broadcast across the batch dimension. "NA" in this column means it is not allowed in networks with an implicit batch dimension.

Table 2. List of Supported Features per TensorRT Layer

Layer	Dimensions of input tensor	Dimensions of output tensor	Does the operation apply to only the innermost 3 dimensions?	Supports broadcast	Supports broadcast across batch
IActivationLayer	0-7 dimensions	0-7 dimensions	No	No	No
IAssertionLayer	0-1 dimensions	No output	No	No	No
IConcatenationLayer	1-7 dimensions	1-7 dimensions	No	No	No
IConditionLayer	Zero	No output	No	No	No
IConstantLayer	Has no inputs	0-7 dimensions	No	No	Always
IConvolutionLayer > 2D Convolution	Three or more dimensions	Three or more dimensions	Yes	No	No

Layer	Dimensions of input tensor	Dimensions of output tensor	Does the operation apply to only the innermost 3 dimensions?	Supports broadcast	Supports broadcast across batch
IConvolutionLayer > 3D Convolution	Four or more dimensions	Four or more dimensions	No	No	No
IDeconvolutionLayer > 2D Deconvolution	Three or more dimensions	Three or more dimensions	Yes	No	No
IDeconvolutionLayer > 3D Deconvolution	Four or more dimensions	Four or more dimensions	No	No	No
IDequantizeLayer	Two or more dimensions	Two or more dimensions	Yes	No	No
IEinsumLayer	0-7 dimensions	0-7 dimensions	No	No	Yes
IElementWiseLayer	0-7 dimensions	0-7 dimensions	No	Yes	Yes
IFillLayer	One dimension	0-7 dimensions	No	Not Applicable	Not Applicable
IFullyConnectedLayer	Three or more dimensions	Three or more dimensions	Yes	No	No
IGatherLayer	▶ Input1: 1-7 dimensions ▶ Input2: 0-7 dimensions	0-7 dimensions	No	No	Yes
IIdentityLayer	0-7 dimensions	0-7 dimensions	No	No	No
IIfConditionalOutputLayer	0-7 dimensions	0-7 dimensions	No	No	No
IIfConditionalInputLayer	0-7 dimensions	0-7 dimensions	No	No	No
IIteratorLayer	1-7 dimensions	0-6 dimensions	No	No	Not Applicable
ILoopOutputLayer	0-7 dimensions	0-7 dimensions	No	No	Not Applicable
ILRNLayer	Three or more dimensions	Three or more dimensions	Yes	No	No
IMatrixMultiplyLayer	Two or more dimensions	Two or more dimensions	No	Yes	Yes
IPaddingLayer	Three or more dimensions	Three or more dimensions	Yes	No	No

Layer	Dimensions of input tensor	Dimensions of output tensor	Does the operation apply to only the innermost 3 dimensions?	Supports broadcast	Supports broadcast across batch
IParametricReluLayer	1-7 dimensions	1-7 dimensions	No	No	No
IPluginV2Layer	User defined	User defined	User defined	User defined	User defined
IPoolingLayer > 2D Pooling	Three or more dimensions	Three or more dimensions	Yes	Yes	Yes
IPoolingLayer > 3D Pooling	Four or more dimensions	Four or more dimensions	No	Yes	Yes
IQuantizeLayer	Two or more dimensions	Two or more dimensions	Yes	No	No
IRaggedSoftMaxLayer	▶ Input: Two dimensions ▶ Bounds: Two dimensions	Two or more dimensions	No	No	Yes
IRecurrenceLayer	0-7 dimensions	0-7 dimensions	No	No	Not Applicable
IReduceLayer	1-7 dimensions	0-7 dimensions	No	No	No
IResizeLayer	1-7 dimensions	1-7 dimensions	No	No	No
IRNNv2Layer	▶ Data/ Hidden/ Cell: Two or more dimensions ▶ Seqlen: Zero or more dimensions	Data/Hidden/ Cell: Two or more dimensions	No	No	No
IScaleLayer	Three or more dimensions	Three or more dimensions	Yes	No	No
IScatterLayer	0-7 dimensions	0-7 dimensions	No	No	No
ISelectLayer	0-7 dimensions	0-7 dimensions	No	Yes	Not Applicable
IShapeLayer	One or more dimensions	One dimension	No	No	Not Applicable

Layer	Dimensions of input tensor	Dimensions of output tensor	Does the operation apply to only the innermost 3 dimensions?	Supports broadcast	Supports broadcast across batch
IShuffleLayer	0-7 dimensions	0-7 dimensions	No	No	No
ISliceLayer	1-7 dimensions	1-7 dimensions	No	No	Yes
ISoftMaxLayer	1-7 dimensions	1-7 dimensions	No	No	Yes
ITopKLayer	1-7 dimensions	▶ Output1: 1-7 dimensions ▶ Output2: 1-7 dimensions	Yes	No	Yes
ITripLimitLayer	Zero dimensions	Has no outputs	No	No	Not Applicable
IUnaryLayer	1-7 dimensions	1-7 dimensions	No	No	No

For more information about each of the TensorRT layers, see TensorRT Layers.

Chapter 3. Layers and Precision

The section lists the TensorRT layers and the precision modes that each layer supports. It also lists the ability of the layer to run on Deep Learning Accelerator (DLA).

For more information about additional constraints, see DLA Supported Layers.

For more information about each of the TensorRT layers, see TensorRT Layers. To view a list of the specific attributes that are supported by each layer, refer to the *NVIDIA TensorRT API Reference* documentation.

Table 3. List of Supported Precision Modes per TensorRT Layer

Layer	FP32	FP16	INT8	INT32	Bool	DLA FP16	DLA INT8
IActivationLayer	Yes	Yes	Yes	No	No	Yes[14]	Yes[15]
IAssertionLayer	No	No	No	No	Yes	No	No
IConcatenationLayer	Yes	Yes	Yes	Yes	No	Yes[16]	Yes[5]
IConditionLayer	No	No	No	No	Yes	No	No
IConstantLayer	Yes	Yes	Yes	Yes	No	No	No
IConvolutionLayer > 2D Convolution	Yes	Yes	Yes	No	No	Yes	Yes
IConvolutionLayer > 3D Convolution	Yes	Yes	No	No	No	No	No
IDeconvolutionLayer > 2D Deconvolution	Yes	Yes	Yes	No	No	Yes	Yes[17]

[14] Partial support. *Yes* for `ReLU`, `Clipped ReLU`, `Leaky ReLU`, `Sigmoid`, and `TanH` activation types only.
[15] Partial support. *Yes* for `ReLU`, `Clipped ReLU`, `Leaky ReLU`, `Sigmoid`, and `TanH` activation types only.
[16] Partial support. *Yes* for concatenation across `C` dimension only.
[17] Partial support. *Yes* for ungrouped deconvolutions and *No* for grouped.

Layer	FP32	FP16	INT8	INT32	Bool	DLA FP16	DLA INT8
IDeconvolutionLayer > 3D Deconvolution	Yes	Yes	No	No	No	No	No
IDequantizeLayer	Yes	No	Yes	No	No	No	No
IEinsumLayer	Yes	Yes	No	No	No	No	No
IElementWiseLayer	Yes	Yes	Yes	Yes	Yes	Yes[18]	Yes[19]
IFillLayer	Yes	No	No	Yes	No	No	No
IFullyConnectedLayer	Yes	Yes	Yes	No	No	Yes	Yes
IGatherLayer	Yes	Yes	No	Yes	No	No	No
IIdentityLayer	Yes	Yes	Yes	Yes	No	No	No
IIfConditionalOutputLayer	Yes	Yes	No	Yes	Yes	No	No
IIfConditionalInputLayer	Yes	Yes	No	Yes	Yes	No	No
IIteratorLayer	Yes	Yes	No	Yes	No	No	No
ILoopOutputLayer	Yes	Yes	No	Yes	No	No	No
ILRNLayer	Yes	Yes	Yes	No	No	Yes	No
IMatrixMultiplyLayer	Yes	Yes	Yes[20]	No	No	No	No
IPaddingLayer	Yes	Yes	Yes	No	No	No	No
IParametricReLULayer	Yes	Yes	Yes	No	No	Yes	Yes
IPluginV2Layer	Yes	Yes	Yes	No	No	No	No
IPoolingLayer > 2D Pooling	Yes	Yes	Yes	No	No	Yes[21]	Yes[9]
IPoolingLayer > 3D Pooling	Yes	Yes	No	No	No	No	No
IQuantizeLayer	Yes	No	No	No	No	No	No
IRaggedSoftMaxLayer	Yes	No	No	No	No	No	No
IRecurrenceLayer	Yes	Yes	No	Yes	Yes	No	No
IReduceLayer	Yes	Yes	Yes	Yes	No	No	No
IResizeLayer	Yes	Yes	Yes	No	No	Yes	Yes

[18] Partial support. *Yes* for `sum`, `sub`, `prod`, `min` , and `max` elementwise operations only.
[19] Partial support. *Yes* for `sum`, `sub`, `prod`, `min` , and `max` elementwise operations only.
[20] Partial support. *Yes* for the case the second input is build-time constant and the first input is not transposed - either produced by a Shuffle layer or `opA == kTRANSPOSE`.
[21] Partial support. *Yes* for `max` and average padding inclusive pooling type only.

Layer	FP32	FP16	INT8	INT32	Bool	DLA FP16	DLA INT8
IRNNv2Layer	Yes	Yes	No	No	No	No	No
IScaleLayer	Yes	Yes	Yes	No	No	Yes[22]	Yes[10]
IScatterLayer	Yes	Yes	Yes	Yes	No	No	No
ISelectLayer	Yes	Yes	No	Yes	Yes	No	No
IShapeLayer[23]	Yes	Yes	Yes	Yes	Yes	No	No
IShuffleLayer	Yes	Yes	Yes	Yes	No	No	No
ISliceLayer	Yes	Yes	No[24]	Yes	No	Yes	No
ISoftMaxLayer	Yes	Yes	No	No	No	Yes	No
ITopKLayer	Yes	Yes	No	No	No	No	No
ITripLimitLayer	Yes	Yes	No	Yes	Yes	No	No
IUnaryLayer[25]	Yes	Yes	Yes	Yes	Yes	No	No

Note: DLA with FP16/INT8 precision with some restrictions on layer parameters.

[22] Partial support. DLA does not support power on the scale layer.
[23] Output is always INT32.
[24] Partial support. *Yes* for unstrided Slice and *No* for strided.
[25] Datatype support is limited to the type of unary operation used. Refer to the Unary Layer for more information.

Chapter 4. Hardware and Precision

The following table lists NVIDIA hardware and which precision modes that each hardware supports. TensorRT supports all NVIDIA hardware with capability SM 5.0 or higher. It also lists the availability of DLA on this hardware. Refer to the following tables for the specifics.

Note: Support for CUDA compute capability version 3.0 has been removed. Support for CUDA compute capability versions below 5.0 may be removed in a future release and is now deprecated.

Table 4. Supported Hardware

CUDA Compute Capability	Example Device	TF32	FP32	FP16	INT8	FP16 Tensor Cores	INT8 Tensor Cores	DLA
8.7	NVIDIA DRIVE AGX Orin™	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8.6	NVIDIA A10	Yes	Yes	Yes	Yes	Yes	Yes	No
8.0	NVIDIA A100/ GA100 GPU	Yes	Yes	Yes	Yes	Yes	Yes	No
7.5	NVIDIA T4	No	Yes	Yes	Yes	Yes	Yes	No
7.2	Jetson AGX Xavier	No	Yes	Yes	Yes	Yes	Yes	Yes
7.0	NVIDIA V100	No	Yes	Yes	Yes	Yes	No	No

CUDA Compute Capability	Example Device	TF32	FP32	FP16	INT8	FP16 Tensor Cores	INT8 Tensor Cores	DLA
6.1	NVIDIA P4	No	Yes	No	Yes	No	No	No
6.0	NVIDIA P100	No	Yes	Yes	No	No	No	No
5.2	NVIDIA M4	No	Yes	No	No	No	No	No
5.0	Quadro K2200	No	Yes	No	No	No	No	No

Deprecated Hardware

Table 5. List of Supported Precision Mode per Hardware

CUDA Compute Capability	Example Device	FP32	FP16	INT8	FP16 Tensor Cores	INT8 Tensor Cores	DLA
3.7	NVIDIA K80	Yes	No	No	No	No	No
3.5	NVIDIA K40	Yes	No	No	No	No	No

Removed Hardware

Table 6. List of Supported Precision Mode per Hardware

CUDA Compute Capability	Example Device	FP32	FP16	INT8	FP16 Tensor Cores	INT8 Tensor Cores	DLA
3.0	NVIDIA K10	Yes	No	No	No	No	No

Chapter 5. Layers for Flow-Control Constructs

The following table lists the TensorRT layers that can be used as interior layers in TensorRT flow-control constructs.

Currently, TensorRT supports loop constructs (using `ILoopLayer`) and ternary conditional constructs (using `IIfConditionalLayer`). Interior layers are layers that include the body of a loop or one of the two branches of an if-conditional.

An `ILoopLayer` interior layer may contain other loops and if-conditionals. An `IIfConditionalLayer` branch may contain other if-conditionals and loops.

Flow-control constructs do not support INT8 calibration and interior-layers cannot employ implicit-quantization (INT8 is supported only in explicit-quantization mode).

Table 7. List of TensorRT Layers that are Supported as Interior Layers of Flow-control Constructs

Layer	Supported
IActivationLayer	Yes, when the operation is one of: `kRELU`, `kSIGMOID`, `kTANH`, `kELU`
IAssertionLayer	Yes
IConcatenationLayer	Yes
IConditionLayer	Yes (for nested conditionals)
IConstantLayer	Yes
IConvolutionLayer > 2D Convolution	singleton channel and spatial dims, that are, the dimensions must be static or have a single value in each optimization profile
IConvolutionLayer > 3D Convolution	singleton channel and spatial dims
IDeconvolutionLayer > 2D Deconvolution	No
IDeconvolutionLayer > 3D Deconvolution	No
IDequantizeLayer	No

Layer	Supported
IEinsumLayer	Yes
IElementWiseLayer	Yes
IFillLayer	`kRANDOM_UNIFORM` only
IFullyConnectedLayer	Yes
IGatherLayer	Yes
IIdentityLayer	Yes
IIfConditionalOutputLayer	Yes (for nested conditionals)
IIfConditionalInputLayer	Yes (for nested conditionals)
IIteratorLayer	Yes (for nested loops)
ILoopOutputLayer	Yes (for nested loops)
ILRNLayer	No
IMatrixMultiplyLayer	Yes
IPaddingLayer	No
IParametricReluLayer	No
IPluginV2Layer	Yes
IPoolingLayer > 2D Pooling	No
IPoolingLayer > 3D Pooling	No
IQuantizeLayer	No
IRaggedSoftMaxLayer	No
IRecurrenceLayer	Yes
IReduceLayer	Yes
IResizeLayer	No
IRNNv2Layer	No
IScaleLayer	Yes
IScatterLayer	Yes
ISelectLayer	Yes
IShapeLayer	Yes
IShuffleLayer	Yes
ISliceLayer	Yes
ISoftMaxLayer	Yes
ITopKLayer	No
ITripLimitLayer	Yes

Layer	Supported
IUnaryLayer	Yes, when the operation is one of: `kABS`, `kCEIL`, `kERF`, `kEXP`, `kFLOOR`, `kLOG`, `kNEG`, `kNOT`, `kRECIP`, `kROUND`, `kSIGN`, `kSQRT`, `kSIN`, `kCOS`, `kATAN`

Chapter 6. Compute Capability Per Platform

The section lists the supported compute capability based on platform.

Table 8. Compute Capability per Platform

Platform	Compute capability
Linux x86-64	3.5, 3.7, 5.0, 5.2, 6.0, 6.1, 7.0, 7.5, 8.0[26], 8.6[27]
Windows 10 x64	3.5, 3.7, 5.0, 5.2, 6.0, 6.1, 7.0, 7.5, 8.0[13], 8.6[14]
CentOS 8.3 ppc64le	7.0, 7.5, 8.0, 8.6
Ubuntu 20.04 SBSA	7.0, 7.5, 8.0, 8.6
NVIDIA JetPack AArch64	5.3, 6.2, 7.2, 8.7

[26] Requires CUDA toolkit 11.0 or newer and a TensorRT CUDA 11.x build.
[27] Requires CUDA toolkit 11.1 or newer and a TensorRT CUDA 11.x build.

Chapter 7. Software Versions Per Platform

The section lists the supported software versions based on platform.

Table 9. List of Supported Platforms per Software Version

Platform	Compiler version	Python version
Ubuntu 18.04 x86-64	gcc 8.3.1	3.6
Ubuntu 20.04 x86-64	gcc 8.3.1	3.8
CentOS 7.9 x86-64	gcc 8.3.1	3.6
CentOS 8.3 x86-64	gcc 8.3.1	3.8
SLES 15 x86-64	gcc 8.3.1	Not Applicable
Windows 10 x64	MSVC 2017u8	Not Applicable
CentOS 8.3 ppc64le	Clang 10.0.1	3.8
Ubuntu 20.04 SBSA	gcc 8.4.0	3.8
NVIDIA JetPack AArch64	gcc 9.3.0	3.8

Note: Python versions supported when using Debian or RPM packages. When using Python wheel files, versions 3.6, 3.7, 3.8, 3.9, and 3.10 are supported.

Chapter 8. ONNX Operator Support

The ONNX operator support list for TensorRT can be found here.

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

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, JetPack, Jetson AGX Xavier, Jetson Nano, Kepler, 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

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