



SUPPORT MATRIX FOR TENSORRT

SWE-SWDOCTRT-001-SPMT_vTensorRT 6.0.1 | August 2019

Support Guide



TABLE OF CONTENTS

Chapter 1. Features For Platforms And Software..... 1

Chapter 2. Layers And Features..... 2

Chapter 3. Layers And Precision.....5

Chapter 4. Hardware And Precision.....8

Chapter 5. Software Versions Per Platform..... 9

Chapter 6. Supported Ops..... 10

Chapter 1.

FEATURES FOR PLATFORMS AND SOFTWARE

Table 1 List of supported features per platform.

	Linux x86-64	Windows x64	Linux ppc64le	Linux AArch64	QNX AArch64
Supported CUDA versions	<ul style="list-style-type: none">▶ 10.2▶ 10.1▶ 10.0▶ 9.0	<ul style="list-style-type: none">▶ 10.1▶ 10.0▶ 9.0	10.1	<ul style="list-style-type: none">▶ 10.2▶ 10.0	10.2
Supported cuDNN versions	7.6.3	7.6.3	7.6.3	7.6.3	7.6.3
TensorRT Python API	Yes	No	Yes	Yes ¹	No
NvUffParser	Yes	Yes	Yes	Yes	Yes
NvOnnxParser	Yes	Yes	Yes	Yes	Yes



Serialized engines are not portable across platforms or TensorRT versions.

¹ Python is not supported on automotive platforms.

Chapter 2.

LAYERS AND FEATURES

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 (see Note 1)	Supports broadcast across batch (see Note 2)
IActivationLayer	0-7 dimensions	0-7 dimensions	No	No	No
IConcatenationLayer	1-7 dimensions	1-7 dimensions	No	No	No
IConstantLayer	0-7 dimensions	0-7 dimensions	No	No	Always
IConvolutionLayer > 2D Convolution	3 or more dimensions	3 or more dimensions	Yes	No	No
IConvolutionLayer > 3D Convolution	4 or more dimensions	4 or more dimensions	No	No	No
IDeconvolutionLayer > 2D Deconvolution	3 or more dimensions	3 or more dimensions	Yes	No	No
IDeconvolutionLayer > 3D Deconvolution	4 or more dimensions	4 or more dimensions	No	No	No
IElementWiseLayer	0-7 dimensions	0-7 dimensions	No	Yes	Yes

Layer	Dimensions of input tensor	Dimensions of output tensor	Does the operation apply to only the innermost 3 dimensions?	Supports broadcast (see Note 1)	Supports broadcast across batch (see Note 2)
IFullyConnectedLayer	3 or more dimensions	3 or more dimensions	Yes	No	No
IGatherLayer	<ul style="list-style-type: none"> Input1: 1-7 dimensions Input2: 0-7 dimensions 	0-7 dimensions	No	No	Yes
IIdentityLayer	0-7 dimensions	0-7 dimensions	No	No	No
ILRNLayer	3 or more dimensions	3 or more dimensions	Yes	No	No
IMatrixMultiplyLayer	2 or more dimensions	2 or more dimensions	No	Yes	Yes
IPaddingLayer	3 or more dimensions	3 or more dimensions	Yes	No	No
IPluginLayer	User defined	User defined	User defined	User defined	User defined
IPluginV2Layer	User defined	User defined	User defined	User defined	User defined
IPoolingLayer > 2D Pooling	3 or more dimensions	3 or more dimensions	Yes	Yes	Yes
IPoolingLayer > 3D Pooling	4 or more dimensions	4 or more dimensions	No	Yes	Yes
IRaggedSoftMaxLayer	<ul style="list-style-type: none"> Input: 2 dimensions Bounds: 2 dimensions 	2 or more dimensions	No	No	Yes
IReduceLayer	1-7 dimensions	0-7 dimensions	No	No	No
IResizeLayer	1-7 dimensions	1-7 dimensions	No	No	No
IRNNLayer	3 dimensions	3 dimensions	No	No	No

Layer	Dimensions of input tensor	Dimensions of output tensor	Does the operation apply to only the innermost 3 dimensions?	Supports broadcast (see Note 1)	Supports broadcast across batch (see Note 2)
IRNNv2Layer	<ul style="list-style-type: none"> ▶ Data/Hidden/Cell: 2 or more dimensions ▶ SeqLen: 0 or more dimensions 	Data/Hidden/Cell: 2 or more dimensions	No	No	No
IScaleLayer	3 or more dimensions	3 or more dimensions	Yes	No	No
IShapeLayer	1 or more dimensions	1 dimension	No	No	No
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	<ul style="list-style-type: none"> ▶ Output1: 1-7 dimensions ▶ Output2: 1-7 dimensions 	Yes	No	Yes
IUnaryLayer	0-7 dimensions	0-7 dimensions	No	No	No



1. 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 out be [1, 5, 4, 3]. Note: The second input tensor has been broadcast in the innermost 2 dimensions.
2. Indicates support for broadcast across the batch dimension.

For more information about each of the TensorRT layers, see [TensorRT Layers](#).

Chapter 3.

LAYERS AND PRECISION

The following table 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 [TensorRT API](#) documentation.

Table 3 List of supported precision mode per TensorRT layer.

Layer	FP32	FP16	INT8	INT32	DLA FP16	DLA INT8
IActivationLayer	Yes	Yes	Yes	No	Yes ²	Yes ³
IConcatenationLayer	Yes	Yes	Yes	Yes	Yes ⁴	Yes ³
IConstantLayer	Yes	Yes	Yes	Yes	No	No
IConvolutionLayer > 2D Convolution	Yes	Yes	Yes	No	Yes	Yes ⁵
IConvolutionLayer > 3D Convolution	Yes	Yes	No	No	No	No
IDeconvolutionLayer > 2D Deconvolution	Yes	Yes	Yes	No	Yes	Yes ⁶

² Partial support. Yes for ReLU, sigmoid and TanH activation types only.

³ Partial support. Yes for ReLU activation type only.

⁴ Partial support. Yes for concatenation across C dimension only.

⁵ Partial support. Yes for ungrouped convolutions and No for grouped.

⁶ Partial support. Yes for ungrouped deconvolutions and No for grouped.

Layer	FP32	FP16	INT8	INT32	DLA FP16	DLA INT8
IDeconvolutionLayer > 3D Deconvolution	Yes	Yes	No	No	No	No
IElementwiseLayer	Yes	Yes	No	Yes	Yes ⁷	Yes ⁸
IFullyConnectedLayer	Yes	Yes	Yes	No	Yes	Yes
IGatherLayer	Yes	Yes	No	Yes	No	No
IIdentityLayer	Yes	Yes	Yes	Yes	No	No
IPluginV2Layer	Yes	Yes	Yes	No	No	No
ILRNLayer	Yes	Yes	Yes	No	Yes	No
IMatrixMultiplyLayer	Yes	Yes	No	No	No	No
IPaddingLayer	Yes	Yes	Yes	No	No	No
IPluginLayer	Yes	Yes	No	No	No	No
IPoolingLayer > 2D Pooling	Yes	Yes	Yes	No	Yes ⁹	Yes ⁸
IPoolingLayer > 3D Pooling	Yes	Yes	No	No	No	No
IRaggedSoftMaxLayer	Yes	No	No	No	No	No
IReduceLayer	Yes	Yes	No	No	No	No
IResizeLayer	Yes	Yes	No	No	No	No
IRNNLayer	Yes	Yes	No	No	No	No
IRNNv2Layer	Yes	Yes	No	No	No	No
IScaleLayer	Yes	Yes	Yes	No	Yes ¹⁰	Yes ⁹
IShapeLayer ¹¹	Yes	Yes	Yes	Yes	No	No
IShuffleLayer	Yes	Yes	Yes	Yes	No	No
ISliceLayer	Yes	Yes	No ¹²	Yes	No	No
ISoftMaxLayer	Yes	Yes	No	No	No	No

⁷ Partial support. Yes for **sum**, **sub**, **prod**, **min** and **max** **elementwise** operations only.

⁸ Partial support. Yes for **sum** **elementwise** operation only.

⁹ Partial support. Yes for **max** and average pooling type only.

¹⁰ Partial support. DLA does not support power on scale layer.

¹¹ Output is always INT32.

¹² Partial support. Yes for unstrided Slice and No for strided.

Layer	FP32	FP16	INT8	INT32	DLA FP16	DLA INT8
ITopKLayer	Yes	Yes	No	No	No	No
IUnaryLayer	Yes	Yes	No	No	No	No



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

Chapter 4.

HARDWARE AND PRECISION

The following table lists NVIDIA hardware and which precision modes each hardware supports. It also lists availability of Deep Learning Accelerator (DLA) on these hardware. TensorRT supports all NVIDIA hardware with capability SM 3.0 or higher.

Table 4 List of supported precision mode per hardware.

CUDA Compute Capability	Example Device	FP32	FP16	INT8	FP16 Tensor Cores	INT8 Tensor Cores	DLA
7.5	Tesla T4	Yes	Yes	Yes	Yes	Yes	No
7.2	Jetson AGX Xavier	Yes	Yes	Yes	Yes	Yes	Yes
7.0	Tesla V100	Yes	Yes	Yes	Yes	No	No
6.2	Jetson TX2	Yes	Yes	No	No	No	No
6.1	Tesla P4	Yes	No	Yes	No	No	No
6.0	Tesla P100	Yes	Yes	No	No	No	No
5.3	Jetson TX1	Yes	Yes	No	No	No	No
5.2	Tesla M4	Yes	No	No	No	No	No
5.0	Quadro K2200	Yes	No	No	No	No	No
3.7	Tesla K80	Yes	No	No	No	No	No
3.5	Tesla K40	Yes	No	No	No	No	No
3.0	Tesla K10	Yes	No	No	No	No	No

Chapter 5.

SOFTWARE VERSIONS PER PLATFORM

Table 5 List of supported platforms per software version.

	Compiler version	Python version
Ubuntu 14.04 x86-64	gcc 4.8.4	2.7, 3.4
Ubuntu 16.04 x86-64	gcc 5.4.0	2.7, 3.5
Ubuntu 18.04 x86-64	gcc 7.4.0	2.7, 3.6
CentOS 7.5 x86-64	gcc 4.8.5	2.7, 3.6
Windows 10 x64	CUDA 10.0, 10.1 MSVC 2017u5 CUDA 9.0 MSVC 2017u3	
Ubuntu 18.04 ppc64le	gcc 7.4.0	2.7, 3.6
CentOS 7.5 ppc64le	gcc 4.8.5	2.7, 3.6
Ubuntu 18.04 AArch64	gcc 7.4.0	2.7, 3.6
QNX AArch64	gcc 5.4.0	

Chapter 6.

SUPPORTED OPS

The following lists describe the operations that are supported in a Caffe or TensorFlow framework and in the ONNX TensorRT parser:

Caffe

These are the operations that are supported in a Caffe framework:

- ▶ **BatchNormalization**
- ▶ **BNLL**
- ▶ **Clip**¹³
- ▶ **Concatenation**
- ▶ **Convolution**
- ▶ **Crop**
- ▶ **Deconvolution**
- ▶ **Dropout**
- ▶ **ElementWise**
- ▶ **ELU**
- ▶ **InnerProduct**
- ▶ **Input**
- ▶ **LeakyReLU**
- ▶ **LRN**
- ▶ **Permute**
- ▶ **Pooling**
- ▶ **Power**
- ▶ **Reduction**
- ▶ **ReLU, TanH, and Sigmoid**
- ▶ **Reshape**

¹³ When using the **Clip** operation, Caffe users must serialize their layers using **ditcaffe.pb.h** instead of **caffe.pb.h** in order to import the layer into TensorRT.

- ▶ **SoftMax**
- ▶ **Scale**

TensorFlow

These are the operations that are supported in a TensorFlow framework:

- ▶ **Add, Sub, Mul, Div, Minimum and Maximum**
- ▶ **ArgMax**
- ▶ **ArgMin**
- ▶ **AvgPool**
- ▶ **BiasAdd**
- ▶ **Clip**
- ▶ **ConcatV2**
- ▶ **Const**
- ▶ **Conv2D**
- ▶ **ConvTranspose2D**
- ▶ **DepthwiseConv2dNative**
- ▶ **Elu**
- ▶ **ExpandDims**
- ▶ **FusedBatchNorm**
- ▶ **Identity**
- ▶ **LeakyReLU**
- ▶ **MaxPool**
- ▶ **Mean**
- ▶ **Negative, Abs, Sqrt, Recip, Rsqrt, Pow, Exp and Log**
- ▶ **Pad** is supported if followed by one of these TensorFlow layers: **Conv2D**, **DepthwiseConv2dNative**, **MaxPool**, and **AvgPool**.
- ▶ **Placeholder**
- ▶ **ReLU, TanH, and Sigmoid**
- ▶ **Relu6**
- ▶ **Reshape**
- ▶ **ResizeBilinear, ResizeNearestNeighbor**
- ▶ **Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Asinh, Acosh, Atanh, Ceil and Floor**
- ▶ **Selu**
- ▶ **Slice**
- ▶ **SoftMax**



If the input to a TensorFlow **SoftMax** op is not **NHWC**, TensorFlow will automatically insert a transpose layer with a non-constant permutation, causing

the UFF converter to fail. It is therefore advisable to manually transpose **SoftMax** inputs to **NHWC** using a constant permutation.

- ▶ **Softplus**
- ▶ **Softsign**
- ▶ **Transpose**

ONNX

Since the ONNX parser is an open source project, the most up-to-date information regarding the supported operations can be found in [GitHub: ONNX TensorRT](#).

These are the operations that are supported in the ONNX framework:

- ▶ **Abs**
- ▶ **Add**
- ▶ **ArgMax**
- ▶ **ArgMin**
- ▶ **AveragePool**
- ▶ **BatchNormalization**
- ▶ **Cast**
- ▶ **Ceil**
- ▶ **Clip**
- ▶ **Concat**
- ▶ **Constant**
- ▶ **Conv**
- ▶ **ConvTranspose**
- ▶ **DepthToSpace**
- ▶ **Div**
- ▶ **Dropout**
- ▶ **Elu**
- ▶ **Exp**
- ▶ **Flatten**
- ▶ **Floor**
- ▶ **Gather**
- ▶ **Gemm**
- ▶ **GlobalAveragePool**
- ▶ **GlobalMaxPool**
- ▶ **HardSigmoid**
- ▶ **Identity**
- ▶ **ImageScaler**
- ▶ **InstanceNormalization**
- ▶ **LRN**

- ▶ LeakyRelU
- ▶ Log
- ▶ LogSoftmax
- ▶ MatMul
- ▶ Max
- ▶ MaxPool
- ▶ Mean
- ▶ Min
- ▶ Mul
- ▶ Neg
- ▶ Pad
- ▶ ParametricSoftplus
- ▶ Pow
- ▶ Reciprocal
- ▶ ReduceL1
- ▶ ReduceL2
- ▶ ReduceLogSum
- ▶ ReduceLogSumExp
- ▶ ReduceMax
- ▶ ReduceMean
- ▶ ReduceMin
- ▶ ReduceProd
- ▶ ReduceSum
- ▶ ReduceSumSquare
- ▶ Relu
- ▶ Reshape
- ▶ Resize
- ▶ ScaledTanh
- ▶ Selu
- ▶ Shape
- ▶ Sigmoid
- ▶ Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Asinh, Acosh, and Atanh
- ▶ Size
- ▶ Slice
- ▶ Softmax
- ▶ Softplus
- ▶ Softsign
- ▶ SpaceToDepth
- ▶ Split

- ▶ Squeeze
- ▶ Sub
- ▶ Sum
- ▶ Tanh
- ▶ ThresholdedRelu
- ▶ TopK
- ▶ Transpose
- ▶ Unsqueeze
- ▶ Upsample

Notice

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

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

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

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

Trademarks

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

Copyright

© 2019 NVIDIA Corporation. All rights reserved.