NVIDIA TensorRT 8.6.11 Release Notes

for DRIVE OS | NVIDIA Docs

SWE-SWDOCTRT-005-RELN | October 2023

Table of Contents

Revision History

This is the revision history of the NVIDIA TensorRT 8.6.11 Release Notes for DRIVE OS.

Document Revision History

Date	Summary of Change
April 18, 2022	Initial draft
May 1, 2022	Start of review
July 7, 2023	End of review
July 10, 2023	Approval review

Chapter 2 Updates

Date	Summary of Change
June 7, 2023	▶ Included the Breaking API Changes topic. ▶ Included the Planned Upcoming Changes topic.

Chapter 3 Updates

Date	Summary of Change
June 14, 2023	Included the New Features and Enhancements topic.

Chapter 4 Updates

Date	Summary of Change
June 21, 2023	Included the Fixed Issues topic.

Chapter 6 Updates

Date	Summary of Change
June 28, 2023	Added known issues 3793130, 4125845, 4138970, and 4157177.

Chapter 7 Updates

Date	Summary of Change
July 7, 2023	Updated supported CUDA, opset, and DLA versions.

List of Tables

Chapter 1. TensorRT for DRIVE OS

1.1. DRIVE OS Linux "Standard"

The NVIDIA® TensorRT™ 8.6.11 for DRIVE® OS release includes a TensorRT Standard+Safety Proxy package. The Linux Standard+Safety Proxy package for NVIDIA DRIVE OS users of TensorRT, contains the builder, standard runtime, proxy runtime, consistency checker, parsers, Python bindings, sample code, standard and safety headers, and documentation. The builder can create engines suitable for the standard runtime, proxy runtime, and DLA. This release includes safety headers and the capability to build standard engines restricted to the scope of operations that will be supported by the safety and proxy runtimes in this and future NVIDIA DRIVE OS 6.0 releases.

1.2. DRIVE OS QNX "Standard"

The NVIDIA TensorRT 8.6.11 for DRIVE OS release includes a TensorRT Standard+Safety Proxy package. The QNX Standard+Safety Proxy package for NVIDIA DRIVE OS users of TensorRT contains the builder, standard runtime, proxy runtime, consistency checker, parsers, sample code, standard and safety headers, and documentation. The builder can create engines suitable for the standard runtime, proxy runtime, safety runtime, and DLA.

1.3. DRIVE OS QNX for Safety

The safety package is available in the NVIDIA DRIVE OS 6.0.8.0 release. The safety package for NVIDIA DRIVE OS users of TensorRT, which is only available on QNX safety, contains the safety runtime, safety headers only, and the API documentation specific to the safety runtime.

1.4. DRIVE OS for Safety Proxy

Proxy runtime

The TensorRT proxy runtime is a version of the safety runtime for platforms that are not safety certified. This includes NVIDIA DRIVE OS x86 SDK, NVIDIA DRIVE OS Linux SDK, NVIDIA DRIVE OS Linux PDK, NVIDIA DRIVE OS QNX SDK and NVIDIA DRIVE OS QNX PDK. The proxy runtime is part of the development flow for safety but it is not certified itself. The proxy runtime only supports engines with engine capability `kSAFETY` (safe engines).

Safety headers

Headers allow applications to compile against the proxy runtime and the safety runtime.

Safety runtime

The safety runtime is also a library that allows applications to load serialized engine plans and perform inference. It is only available for QNX safety. The safety runtime only supports engines with engine capability `kSAFETY` (safe engines).

Chapter 2. Release Highlights

2.1. Breaking API Changes

- `struct FloatingPointErrorInformation` has been replaced with `struct RuntimeErrorInformation`.

2.2. Planned Upcoming Changes

The following sections describe planned, upcoming changes for a future release.

`ILayer` Scope Relaxation

The TensorRT safety runtime is planning to relax the restrictions on the following `ILayer` in the future release:

- `IActivationLayer`: the minimum rank of the input and output tensors for `IActivationLayer` will be relaxed to 0.
- `IConstantLayer`: the batch, channel, and spatial dimension restrictions for `IConstantLayer` will be removed.
- `IElementWiseLayer`: the minimum rank of the input and output tensors for `IElementWiseLayer` will be relaxed to 0.
- `IGatherLayer`: the minimum rank of the input and output tensors for `IGatherLayer` will be relaxed to 0.
- `IIdentityLayer`: the minimum rank of the input and output tensors for `IIdentityLayer` will be relaxed to 0.
- `IPluginV2Layer`: the minimum rank of the input and output tensors for `IPluginV2Layer` will be relaxed to 0. Index tensors with precision INT32 will be supported.
- `IScaleLayer`: the batch, channel, and spatial dimension restrictions for `IScaleLayer` will be removed.

- `IShuffleLayer`: the minimum rank of the output tensor for `IShuffleLayer` will be relaxed to 0.

Add DLA support for `IReduceLayer`

The TensorRT 8.6.12 release will add DLA support for `IReduceLayer` with the Max operation where any combination of the CHW is reduced.

Add DLA support for the following `IElementWiseLayer` operations: `Div`, `Pow`, `Greater`, and `Less`

The TensorRT 8.6.12 release will add DLA support for `IElementWiseLayer` with the `Div`, `Pow`, `Greater`, and `Less` operations.

DLA will support broadcasting for `IElementWiseLayer`

The TensorRT 8.6.12 release will support broadcasting for `IElementWiseLayer`.

Add DLA support for the following `IUnaryLayer` operations: `Sin`, `Cos`, and `Atan`

The TensorRT 8.6.12 release will add DLA support for the following `IUnaryLayer` operations: `Sin`, `Cos`, and `Atan`.

Chapter 3. New Features and Enhancements

This release includes support for these new features and enhancements.

`RuntimeErrorInformation` Update

The TensorRT safety and proxy runtimes replaced `FloatingPointErrorInformation` with a more generalized struct `RuntimeErrorInformation`. The `RunTimeErrorInformation` provides a more generalized method for asynchronous error reporting during runtime. The same API interface can be used to interact with the new struct but the underlying structure has been changed to a bitmap to support more types of runtime error. The TensorRT runtime will set a flag when a supported error type occurs in the runtime instead of counting the number of errors like the old `FloatingPointErrorInformation`. Refer to the NVIDIA TensorRT 8.6.11 API Reference for DRIVE OS for more information.

`IConstantLayer` Output Tensor Rank Relaxation

The TensorRT safety runtime has updated the output tensor rank constraint of `IConstantLayer`, eliminating the previous limit of 4 and now allowing any rank. Refer to the NVIDIA TensorRT 8.6.11 Safety Developer Guide Supplement for DRIVE OS for more information.

API Changes

The following table provides a summary of the TensorRT API changes for the NVIDIA DRIVE OS 6.0.8 release. Any changes that affect the safety runtime will also affect the proxy runtime.

Table 1. API Changes for DRIVE OS 6.0.8

Interface	Impact

`struct RuntimeErrorInformation`
 	
`enum class RuntimeErrorType`
 	
`virtual void setErrorBuffer(RuntimeErrorInformation* const buffer) noexcept = 0;`
 	
`virtual RuntimeErrorInformation* getErrorBuffer() const noexcept = 0;`
	Affected: The `FloatingPointErrorInformation` has been replaced with `RuntimeErrorInformation`.
	
	Action: Refer to the Breaking API Changes, New Features and Enhancements, and the NVIDIA TensorRT 8.6.11 API Reference for DRIVE OS document for more information.

TensorRT Standard Build

The TensorRT 8.6 release includes changes to the TensorRT 8.6.1 standard builder and runtime that appear in TensorRT for DRIVE OS 6.0. For more information, refer to the NVIDIA TensorRT 8.6.1 Release Notes.

Documentation Changes

The TensorRT 8.6.11 documentation has been updated accordingly:

- The NVIDIA TensorRT 8.6.11 Developer Guide for DRIVE OS is based on the enterprise TensorRT 8.6.1 release. We have modified the TensorRT 8.6.1 Developer Guide documentation for DRIVE OS 6.0.8 accuracy. The TensorRT safety content has been removed.
- The TensorRT safety content is in the NVIDIA TensorRT 8.6.11 Safety Developer Guide Supplement for DRIVE OS. Refer to this PDF for all TensorRT safety specific documentation.

Chapter 4. Fixed Issues

The following NVIDIA DRIVE OS issues from the previous release are resolved in this release.

Table 2. Fixed Issues in TensorRT 8.6.11

Feature	Module	Description
4064008	TensorRT runtime	The Resize layer generates inconsistent results under specific configurations in the safety runtime and could potentially lead to an accuracy drop compared to the standard runtime. This issue has been fixed in this release.
4065495	TensorRT builder and consistency checker	The dimension constraint for `ILayers` in TensorRT safety releases may not correspond with the range specified in the NVIDIA TensorRT Safety Developer Guide Supplement for DRIVE OS. This discrepancy has been resolved in this release.
3988897	TensorRT runtime	The INT8 accuracy of the safety runtime decreased ~5% in the Top1/Top5 results compared to the standard runtime for some networks such as ResNet, DenseNet, and GoogleNet. This issue has been fixed in this release.

Feature	Module	Description
4001076	TensorRT builder	ASCII control characters are not written correctly using unicode escape sequences for JSON writers. This issue has been fixed in this release.
3995364	DLA	Setting the DLA SRAM pool size to 0 can cause hangs or memory faults. This issue has been fixed in this release.

Chapter 5. Known Limitations

Table 3. Known Limitations

Feature	Module	Description
DLA	TensorRT	DLA is not supported through the TensorRT safety runtime. The DLA loadables for standard and safety can be consumed by the cuDLA runtime and the NvMedia runtime.
DLA	TensorRT	When running on DLA, various layers have restrictions on supported parameters and input shapes. Some existing limitations for the convolution, fully connected, concatenation, and pooling layers were newly documented in this release. Refer to the NVIDIA TensorRT 8.6.11 Developer Guide for DRIVE OS for details.
DLA	TensorRT	When running INT8 networks on DLA using TensorRT, avoid marking intermediate tensors as network outputs to reduce quantization errors by allowing layers to be fused and retain higher precision for intermediate results.
DLA	TensorRT	There are two modes of SoftMax where the mode is

Feature	Module	Description
		chosen automatically based on the shape of the input tensor, where: ▶ the first mode triggers when all non-batch, non-axis dimensions are 1, and ▶ the second mode triggers in other cases if valid. Refer to the NVIDIA TensorRT 8.6.11 Developer Guide for DRIVE OS for details.
DLA	TensorRT	The DLA compiler can remove identity transposes, but it cannot fuse multiple adjacent transpose layers into a single transpose layer. Likewise, for reshape. For example, given a TensorRT `IShuffleLayer` consisting of two non-trivial transposes and an identity reshape in between, the shuffle layer will be translated into two consecutive DLA transpose layers, unless you merge the transposes together manually in the model definition in advance.
DLA	TensorRT	Running networks on DLA with large batch sizes may produce incorrect outputs. It is suggested to use batch size up to 64 to run networks on DLA.
Layers	TensorRT	For a list of safety-specific layer limitations, refer to the NVIDIA TensorRT 8.6.11 Safety Developer Guide Supplement for DRIVE OS.

Feature	Module	Description
I/O Formats	TensorRT	When using vectorized I/O formats, the extent of a tensor in a vectorized dimension might not be a multiple of the vector length. Elements in a partially occupied vector that are not within the tensor are referred to here as *vector-padding*. ▶ For input tensors, the application shall set vector-padding elements to zero. ▶ For output tensors, the value of vector-padding elements is undefined. In a future release, TensorRT will support setting them to zero.
Safety samples	TensorRT	We cannot use `-Xcompiler -Wno-deprecated-declarations` options for safety samples; that is a standard certified option. We only add it for standard builds. Seeing the deprecated warnings during the build is expected for this case.
Execution context	TensorRT	The GPU memory allocated to each execution context is limited to 4 GiB. An error will be reported if more GPU memory is required.
Execution context	TensorRT	Users of DRIVE OS must ensure that `enqueueV3()` is not called concurrently by multiple execution contexts created from the same engine instance.
Restricted mode	TensorRT	If layer precision is not explicitly set,

Feature	Module	Description
		`IBuilder::isNetworkSupported` may return `True` and building a standard engine with the `kSAFETY_SCOPE` flag may pass while building a safe engine fails with the same network.

Chapter 6. Known Issues

Table 4. Known Issues

Feature	Module	Description
3656116	TensorRT runtime	**What is the issue?** There is an up to 7% performance regression for the 3D-UNet networks compared to TensorRT 8.4 EA when running in INT8 precision on NVIDIA Orin due to a functionality fix. **How does it impact the customer?** When running 3D-UNet networks in INT8 precision, the latency will be up to 7% longer than in TensorRT 8.4 EA. **If there is a workaround, what is it?** To work around this issue, set the input type and format to kINT8 and kCHW32, respectively. **When can we expect the fix?** We do not plan to fix this performance regression since it was caused by a necessary fix for an accuracy issue. **Is it for Standard/Safety, SDK/PDK?** Standard, SDK

Feature	Module	Description
3263411	TensorRT builder	**What is the issue?** For some networks, building and running an engine in the standard runtime will have better performance than the safety runtime. This can be due to various limitations in scope of the safety runtime including more limited tactics, tensor size limits, and operations supported in the safety scope. **How does it impact the customer?** Inference in the safety runtime may be significantly slower than in the standard runtime. **If there is a workaround, what is it?** Depending on the network, it may or may not be possible to reorganize operations into a more efficient form matching the safety runtime scope. **What is the recommendation?** It is recommended to work with NVIDIA and provide proxy networks as early as possible that demonstrate key performance metrics close to actual production networks. **Is it for Standard/Safety, SDK/PDK?** Safety, SDK
3793130	TensorRT runtime	**What is the issue?** Enabling the CUDA-graph option may cause the safety runtime to perform less efficiently compared to the proxy runtime for some networks. This discrepancy is due to the

Feature	Module	Description
		different objectives of the safety and proxy runtime. The safety runtime has more restrictive constraints to fulfill safety goals, resulting in different implementations between safety and proxy runtime. **How does it impact the customer?** Using the CUDA-graph for inference in the safety runtime may result in slower performance compared to the proxy runtime. However, this can vary depending on the inference network. **If there is a workaround, what is it?** It is recommended to check whether enabling CUDA-graph improves performance on the networks in production. Since the safety implementation with CUDA-graph comes with additional error checking and more deterministic execution, it is recommended to conduct cost-benefit analysis to decide if using CUDA-graph is beneficial to the use case. It is also recommended to work with NVIDIA and provide proxy networks as early as possible that demonstrate key performance metrics close to actual production networks. **When can we expect the fix?** In order to achieve safety, the implementation might require further support on error-checking and robustness

Feature	Module	Description
		measures. This could demand extra CPU/GPU cycles. However, in certain scenarios, the safety implementation might be faster since it does not support some features in proxy runtime. The performance parity will continue to improve in the future releases but it might not be completely realized. **Is it for Standard/Safety, SDK/PDK?** Safety SDK
4125845	TensorRT builder	**What is the issue?** Some networks with Convolution layers may fail to build when the `builderOptimizationLevel` is set to 4 or 5. **How does it impact the customer?** For specific networks, customers may not build engines with 4 or 5 builder optimization levels. The builder optimization level is a new feature and this issue does not break previous behavior. **If there is a workaround, what is it?** No, the only way is to use builder optimization level under 3. **When can we expect the fix?** This issue is expected to be fixed in a future release. **Is it for Standard/Safety, SDK/PDK?** Standard, Safety SDK
4138970	Consistency checker	**What is the issue?** The consistency checker will report

Feature	Module	Description
		an error when a standalone `IScaleLayer` is not fused with other layers and the TensorRT builder selects INT8 formats for its I/O tensor. **How does it impact the customer?** The consistency checker will report the unexpected error when the standalone IScaleLayer with INT8 I/O tensor format occurs in the network. **If there is a workaround, what is it?** Use `setPrecision()` to set the precision of the corresponding `IScaleLayer` to FP32 or FP16. **When can we expect the fix?** This issue is expected to be fixed in a future release. **Is it for Standard/Safety, SDK/PDK?** Safety, SDK
4157177	TensorRT builder	**What is the issue?** An assertion error will occur from the TensorRT builder if an `IActivationLayer` serves as the input for an `IElementwiseLayer` and is also the input for another layer. **How does it impact the customer?** If the model is structured such that an `IActivationLayer` is connected as an input to both an `IElementwiseLayer` and a different layer, it might result in the failure from the TensorRT builder.

Feature	Module	Description
		If there is a workaround, what is it? You can clone the `IActivationLayer` and connect one `IActivationLayer` to the `IElementwiseLayer` and the other `IActivationLayer` to the other layer. **When can we expect the fix?** This issue is expected to be fixed in a future release. **Is it for Standard/Safety, SDK/PDK?** Safety SDK

Chapter 7. TensorRT Release Properties

The following table describes the release properties and software versions.

Table 5. TensorRT Release Properties

	Linux x86-64	Linux AArch64	QNX AArch64	
			QNX Safety	QNX Standard
Supported NVIDIA CUDA® versions	11.4.24	11.4.24	11.4.24	11.4.24
Supported NVIDIA cuDNN versions	8.9.0	8.9.0	No	8.9.0
TensorRT Python API	Yes	Yes	No	No
NvOnnxParser	Yes	Yes	No	Yes

Note: With the exception of QNX safety, which requires engines to be built and serialized on QNX standard, serialized engines are not generally portable across platforms or TensorRT versions. In the standard runtime, version numbers must match (in major, minor, patch, and build) for the previously generated serialized engine to be minimally compatible. For more information, refer to the NVIDIA TensorRT 8.6.11 Safety Developer Guide Supplement for DRIVE OS. In the NVIDIA TensorRT 8.6.11 safety runtime, version numbers for major, minor, and patch must be equal to the runtime version numbers, and equal to 8.6.11.

7.1. Hardware Precision

The following table lists NVIDIA hardware and which precision modes each hardware supports. It also lists availability of Deep Learning Accelerator (DLA) on this hardware.

For standard runtime, TensorRT supports SM 7.x or SM 8.x. For proxy runtime, TensorRT supports all hardware with capability of 8.x. For safety runtime, TensorRT supports hardware with capability of 8.7.

For more information, refer to the FAQ section in the NVIDIA TensorRT 8.6.11 Developer Guide for DRIVE OS.

Table 6. Hardware and Precision Support for TensorRT 8.6.11

CUDA Compute Capabilit	Example Device	TF32	FP32	FP16	INT8	FP16 Tensor Cores	INT8 Tensor Cores	DLA
8.7	NVIDIA Orin	No (TensorRT safe) Yes (TensorRT standard)	Yes	Yes	Yes	Yes	Yes	Yes
8.6	NVIDIA A10	Yes	Yes	Yes	Yes	Yes	Yes	No
8.0	NVIDIA PG199	Yes	Yes	Yes	Yes	Yes	Yes	No

7.2. Software Versions Per Platform

Table 7. Software Versions per Platform for TensorRT 8.6.11

Platform	Compiler Version	Python Version
Ubuntu 20.04 x86-64	gcc 9.3.0	3.8
Ubuntu 20.04 AArch64	gcc 9.3.0	3.8
QNX AArch64	QNX 7.1.0 Q++ 8.3.0	N/A

7.3. Compatibility

TensorRT 8.6.11 has been tested with the following:

- CUDA 11.4.24
- cuDNN 8.9.0
- TensorFlow 1.15.5

- PyTorch 1.13.1
- ONNX 1.12.0 and opset 16
- DLA 3.14

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

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 CUDA, DALI, DGX-1, DRIVE, JetPack, Orin, Pegasus, TensorRT, Triton and Xavier 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