NVIDIA TensorRT 8.5.10 Release Notes

for DRIVE OS | NVIDIA Docs

SWE-SWDOCTRT-005-RELN | February 2023

Table of Contents

Revision History

This is the revision history of the *NVIDIA TensorRT 8.5.10 Release Notes for DRIVE OS*.

Document revision history

Date	Summary of Change
October 25, 2022	Initial draft
October 26, 2022	Start of review
December 9, 2022	End of review
December 12, 2022	Approval review

List of Tables

Chapter 1. TensorRT for DRIVE OS

1.1. DRIVE OS Linux "Standard"

The NVIDIA® TensorRT™ 8.5.10 for DRIVE® OS release includes a TensorRT Standard+Proxy package. The Standard+Proxy package for NVIDIA DRIVE OS users of TensorRT, which is available on all platforms except QNX safety, 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 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 8.5.10 for DRIVE OS release includes a TensorRT Standard+Safety Proxy package. The 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, GPU, and DLA.

DRIVE OS QNX for Safety

The safety package is available in the NVIDIA DRIVE OS 6.0.6.0 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.

1.3. 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. Deprecations in this Release

The following items have been deprecated in this release.

Note: These deprecations are still supported in this current release. They will be removed in the next major TensorRT 9.0 release.

Table 1. Deprecations in TensorRT 8.5.10

Summary	Impact
Two DLA Safety samples, dlaSafetyBuilder and dlaSafetyRuntime have been removed from the TensorRT 8.5.10 release.	**Module:** TensorRT samples **Action:** Since TensorRT safety runtime does not support DLA, this removal does not have any impact for TensorRT safety users. You can check TensorRT standard runtime for the DLA usage.
`ICudaEngine::getNbBindings`	**Module:** TensorRT runtime **Action:** Since the TensorRT standard runtime was updated to `enqueueV3()` in the TensorRT 8.5.1 release, the `enqueueV3()` in the TensorRT safety runtime reduces the API changes when migrating from the standard runtime to the safety runtime. Name-based functions have been added to `safe::ICudaEngine`.
`ICudaEngine::getBindingIndex`	
`ICudaEngine::getBindingName`	
`ICudaEngine::bindingIsInput`	
`ICudaEngine::getBindingDimensions`	
`ICudaEngine::getBindingDataType`	
`ICudaEngine::getBindingBytesPerComponent`	
`ICudaEngine::getBindingComponentsPerElement`	
`ICudaEngine::getBindingFormat`	
`ICudaEngine::getBindingVectorizedDim`	

Summary	Impact
`IExecutionContext::getStrides`	**Module:** TensorRT safety execution context
`IExecutionContext::enqueueV2`	

2.2. Planned Upcoming Changes

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

`IGatherLayer` Support

The TensorRT safety and proxy runtimes will add support for `IGatherLayer` in a future release. This is to complement the `IGatherLayer` functionality in the TensorRT standard runtime. `IGatherLayer` support in the TensorRT safety and proxy runtimes will be designed to support the Gather mode for the first DRIVE OS 6.0 safety cycle. Refer to the NVIDIA TensorRT 8.5.10 API Reference for DRIVE OS or the NVIDIA TensorRT Operator's Reference documentation to get more information and limitations.

`IMatrixMultiplyLayer` Support

TensorRT will add support for `IMatrixMultiplyLayer` in a future release. This will allow users of TensorRT safety to use the ONNX GEMM or MatMul operators as inputs to the TensorRT builder. Refer to the NVIDIA TensorRT 8.5.10 API Reference for DRIVE OS or the NVIDIA TensorRT Operator's Reference documentation to get more information and limitations.

Safe Plugin Registry Interface Updates

In a future release, TensorRT will split the existing `nvinfer1::IPluginRegistry` interface in `NvinferRuntimeCommon.h` into two new interfaces:

- `nvinfer1::IPluginRegistry` in `NvInferRuntime.h`, which will be used in TensorRT standard.
- `nvinfer1::safe::IPluginRegistry` in `NvInferSafeRuntime.h`, which will be used in TensorRT safety.

The two interfaces will be completely separate and shall not derive from a common base class.

Standard code that uses `nvinfer1::IPluginRegistry` is expected to compile without modifications. Automotive safety code that uses plugin registries may require the following changes to continue to compile:

- The explicit use of `nvinfer1::IPluginRegistry` must be replaced by `nvinfer1::safe::IPluginRegistry`.

- Instead of directly including the `NvInferRuntimeCommon.h` header, the user code must include `NvInferSafeRuntime.h` instead.
- The use of `getBuilderPluginRegistry()` should be replaced by `getBuilderSafePluginRegistry()`.

Chapter 3. New Features and Enhancements

This release includes support for these new features and enhancements.

API Changes

The following table provides a summary of the TensorRT API changes for the NVIDIA DRIVE OS 6.0.6 release. Any changes that affect the safety runtime will also affect the proxy runtime.

Table 2. API Changes for DRIVE OS 6.0.6

Interface	Impact
`ICudaEngine::getTensorShape`	**Affected:** Binding index-based functions have been deprecated. Name-based functions have been added. **Action:** Refer to the *NVIDIA TensorRT 8.5.10 API Reference for DRIVE OS*. Use `enqueueV3()` for asynchronous inference execution. Use name-based functions.
`ICudaEngine::getTensorDataType`	
`ICudaEngine::getTensorIOMode`	
`ICudaEngine::getTensorBytesPerComponent`	
`ICudaEngine::getTensorComponentsPerElement`	
`ICudaEngine::getTensorVectorizedDim`	
`ICudaEngine::getNbIOTensors`	
`ICudaEngine::getIOTensorName`	
`IExecutionContext::getTensorStrides`	
`IExecutionContext::setInputTensorAddress`	
`IExecutionContext::setOutputTensorAddress`	
`IExecutionContext::setInputConsumedEvent`	
`IExecutionContext::getInputConsumedEvent`	
`IExecutionContext::getInputTensorAddress`	
`IExecutionContext::getOutputTensorAddress`	
`IExecutionContext::enqueueV3`	

TensorRT Standard Build

The TensorRT 8.5 release includes changes to the standard builder and runtime that appear in TensorRT for DRIVE OS 6.0. For more information, refer to the TensorRT 8.5.1 Release Notes.

Documentation Changes

The TensorRT 8.5.10 documentation has been updated accordingly:

- The *NVIDIA TensorRT 8.5.10 Developer Guide for DRIVE OS* is based on the enterprise TensorRT 8.5.x release. We have modified the TensorRT 8.5.x Developer Guide documentation for DRIVE OS 6.0.6 accuracy. The TensorRT safety content has been removed.
- The TensorRT safety content is in the *NVIDIA TensorRT 8.5.10 Safety Developer Guide Supplement for DRIVE OS*. Refer to this PDF for all TensorRT safety specific documentation.

Safety Samples Update

New safety samples have been added to TensorRT 8.5.10. These samples focus on TensorRT safety features and functionality. Refer to the *NVIDIA TensorRT 8.5.10 Safety Developer Guide Supplement for DRIVE OS* for more information.

You can find the safety samples in the `/usr/src/tensorrt/samples` package directory. For more information on running samples, refer to the `README.md` file included with the sample.

Two DLA Safety samples, dlaSafetyBuilder and dlaSafetyRuntime have been removed from the TensorRT 8.5.10 release.

`ISliceLayer` Support

The TensorRT 8.5.10 release supports a new layer called `ISliceLayer`, which enables the slice operation of input tensors along all axes. Slices must be FP32, FP16, or INT8 precision and must meet the requirements listed in the *NVIDIA TensorRT 8.5.10 Developer Guide for DRIVE OS*. Refer to ONNX slice Op definition and the *NVIDIA TensorRT 8.5.10 API Reference for DRIVE OS* documentation for more information.

Equal Operation Support in `IElementWiseLayer`

The TensorRT 8.5.10 release extends `IElementWiseLayer` to support equal operation (`ElementWiseOperation::kEQUAL`). This is the first ElementWise logical operation that DLA supports, so there are several restrictions and requirements imposed when adopting this operation in DLA.

One such requirement is that you must explicitly set the device type of the ElementWise equal layer to `DLA`. `trtexec` now supports a flag `--layerDeviceTypes` to let you explicitly specify the device type for individual layers. Refer to the *NVIDIA TensorRT 8.5.10 Developer Guide for DRIVE OS* documentation for more information on the above changes.

Opportunistic TensorRT Safe Engine Version Forward Compatibility

Opportunistic TensorRT Safe Engine Version Forward Compatibility allows users to run safe engines generated by some older TensorRT versions with the current TensorRT safety runtime under specific conditions. This is only possible when the safety runtime does not change materially between releases, which would generally be limited to the safety and stabilization phase of a safety cycle leading up to safety assessment.

While we strive to ensure safe engine version forward compatibility opportunistically, safe engines generated from previous TensorRT versions are not forward compatible with the current TensorRT 8.5.10 safety runtime due to material changes in the runtime from new functionality. Similarly, safe engines generated from the current release will not be forward compatible with the TensorRT 8.6.10 runtime. Absent bugs and safety-related refactoring that would force us to do otherwise, our goal is to support safe engines generated from TensorRT 8.6.10+ for usage in later releases throughout the remainder of TensorRT 8 development.

Refer to the NVIDIA TensorRT 8.5.10 Safety Developer Guide Supplement for DRIVE OS and the Release Notes of each release for the supported forward compatible safe engine versions and limitations.

`enqueueV3()`

The TensorRT 8.5.10 release added a new function called `enqueueV3()` to support asynchronous inference execution. `enqueueV3()` adds support for constant input buffers, which is a safety requirement. Since the TensorRT standard runtime was updated to `enqueueV3()` in the TensorRT 8.5.1 release, the `enqueueV3()` in the TensorRT safety runtime reduces the API changes when migrating from the standard runtime to the safety runtime. Binding index-based functions have been deprecated and name-based functions have been added to `safe::ICudaEngine`. For more information regarding API changes, refer to the *NVIDIA TensorRT 8.5.10 API Reference for DRIVE OS*.

TensorRT Consistency Checker

The TensorRT 8.5.10 release of the consistency checker performs most checks to ensure that engines can be run in the safety runtime without invoking undefined or nondeterministic behavior. Operations within the safety scope are checked, tensor sizes and formats are checked, and inputs to each layer are analyzed to ensure no uninitialized values are read from memory. Some tactics require specialized kernels and internal data structures. Most, but not all, of these internal data structures are validated in the release.

Chapter 4. Fixed Issues

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

Table 3. Fixed Issues in TensorRT 8.5.10

Feature	Module	Description
3785919	TensorRT safety package	When installing files from Debians on the same system, some files installed by NVIDIA DRIVE OS QNX safety TensorRT Debian would be in the same location as the NVIDIA DRIVE OS QNX proxy TensorRT Debian. This limitation has been fixed in this release.
3698033	DLA	Some networks would fail to build DLA INT8 loadable in `DLA_STANDALONE` mode with INT8 calibrator. This bug has been fixed in this release.
3698054	TensorRT builder	In some cases, the TensorRT builder would allow input and output tensors in HWC16 format in FP16 precision. This format is outside the safety scope. HWC16 format has been removed from available formats in this release.

Feature	Module	Description
3657753	TensorRT runtime	There would sometimes be issues with large channel sizes with structured sparsity convolution kernels (seen at size 4096). This bug has been fixed in this release.
3689094	TensorRT builder	TensorRT would take some dense weights as sparse, if they match some special pattern. This bug has been fixed in this release.
3448473	TensorRT builder	The DLA compilation process in NVIDIA DRIVE OS 6.0.5.0 had a deep recursive call which required a lot of stack memory. On QNX, this may have exceeded the available stack space, leading to memory faults. This bug has been fixed in this release.

Chapter 5. 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 8.5.10 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 chosen automatically based on the shape of the input tensor, where:

Feature	Module	Description
		▶ the first mode triggers when all non-batch, non-axis dimensions are 1, and ▶ the second mode triggers in other cases if valid. The second of the two modes is supported only for DLA 3.9.0 and later. It involves approximations which may result in errors of a small degree. Also, batch size greater than 1 is supported only for DLA 3.9.0 and later. Refer to the *NVIDIA TensorRT 8.5.10 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.
Layers	TensorRT	For a list of safety-specific layer limitations, refer to the *NVIDIA TensorRT 8.5.10 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 total execution context memory size is limited to 4 GiB due to internal safety constraints. Error will be reported if the engine plans require memory that is more than 4 GiB.
Execution context	TensorRT	Users of DRIVE OS must ensure that `enqueueV3()` is not called concurrently by different execution contexts created from the same engine.

Feature	Module	Description
Restricted mode	TensorRT	Users who do not set layer precisions explicitly, `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

Feature	Module	Description
3494734	DLA	**What is the issue?** Some networks may produce incorrect outputs when run on DLA with large batch sizes. **How does it impact the customer?** Running networks on DLA with batch sizes larger than 32 may produce incorrect outputs. **If there is a workaround, what is it?** To work around this issue, use a batch size smaller than 32. **When can we expect the fix?** The issue will be fixed in a future DLA release. **Is it for Standard/Safety, SDK/PDK?** Standard, SDK
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

Feature	Module	Description
		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
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

Feature	Module	Description
		with NVIDIA and provide proxy networks as early as possible that demonstrate key performance metrics close to actual production networks. Future releases will target performance improvements for networks within the safety scope. **Is it for Standard/Safety, SDK/PDK?** Standard, SDK
3827883	Samples	**What is the issue?** The `trtexec` binary shipped with TensorRT has an unnecessary dependency on deprecated NVMedia libraries. **How does it impact the customer?** The binary will not be usable if the deprecated NVMedia libraries are missing. **If there is a workaround, what is it?** Building `trtexec` from source will result in a binary without the extra dependency. Refer to the samples README for details on how to do so. **When can we expect the fix?** This issue is not expected to be fixed in a future release. **Is it for Standard/Safety, SDK/PDK?** Safety, PDK

Chapter 7. TensorRT Release Properties

The following table describes the release properties and software versions.

	Linux x86-64	Linux AArch64	QNX AArch64	
			QNX Safety	QNX Standard
Supported NVIDIA CUDA® versions	11.4	11.4	11.4	11.4
Supported NVIDIA cuDNN versions	8.6.0	8.6.0	No	8.6.0
TensorRT Python API	Yes	Yes	No	No
NvUffParser	Deprecated	Deprecated	No	Deprecated
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 8.5.10 Safety Developer Guide Supplement for DRIVE OS*. In the NVIDIA TensorRT 8.5.10 safety runtime, version numbers for major, minor, and patch must be earlier or equal to the runtime version numbers, and later than or equal to 8.5.10.

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 “If I build the engine on one GPU and run the engine on another GPU, will this work?” question in the FAQ section in the *NVIDIA TensorRT 8.5.10 Developer Guide for DRIVE OS*.

Table 4. Hardware and precision support for TensorRT 8.5.10

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 5. Software versions per platform for TensorRT 8.5.10

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 8.5.10 has been tested with the following:

- CUDA 11.4.20
- cuDNN 8.6.0

- TensorFlow 1.15.0
- PyTorch 1.9.0
- ONNX 1.9.0 and opset 13
- DLA 3.12
- ElementWise 2.5.0

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