NVIDIA TensorRT 8.6.13 Release Notes

for DRIVE OS | NVIDIA Docs

SWE-SWDOCTRT-005-RELN | March 2024

Table of Contents

Revision History

This is the revision history of the NVIDIA TensorRT 8.6.13 Release Notes for DRIVE OS.

Document Revision History

Date	Summary of Change
December 12, 2023	Initial draft
December 13, 2023	Start of review
March 15, 2024	End of review
March 15, 2024	Approval review

List of Tables

Chapter 1. TensorRT for DRIVE OS

1.1. DRIVE OS QNX "Standard"

The NVIDIA TensorRT 8.6.13 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.

1.2. DRIVE OS QNX for Safety

The safety package is available in the NVIDIA DRIVE OS 6.0.9.2 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. New Features and Enhancements

This release includes support for these new features and enhancements.

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 8.6.13 documentation has been updated accordingly:

- The NVIDIA TensorRT 8.6.13 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 6.0.9.2 accuracy. The TensorRT safety content has been removed.
- The TensorRT safety content is in the NVIDIA TensorRT 8.6.13 Safety Developer Guide Supplement for DRIVE OS. Refer to this PDF for all TensorRT safety specific documentation.

Chapter 3. Fixed Issues

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

Table 1. Fixed Issues in TensorRT 8.6.13

Reference ID	Module	Description
4323665	TensorRT runtime	The TensorRT safety runtime does not set the CUDA API mode by invoking the API `cudaSafeEXSelectAPIMode()` at the initialization state. This bug has been fixed in this release.
4350817	TensorRT runtime	Some networks containing the softmax layer may fail with `SafeCaskError` when switching to the Operational State using NvDVMS (DRIVE OS VM State Management), which helps to verify DOS_RES_107. This bug has been fixed in this release.
4369190	TensorRT safety samples	The sample `sampleSafeINT8` cannot run on the safety runtime due to a bug in the inference phase. This bug has been fixed in this release.

Chapter 4. Known Limitations

Table 2. 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 8.6.13 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 8.6.13 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 8.6.13 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 5. Known Issues

Table 3. Known Issues

Reference ID	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

Reference ID	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

Reference ID	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

Reference ID	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. We do not intend to address this issue within the DRIVE OS 6.0 release timeframe. **Is it for Standard/Safety, SDK/PDK?** Safety, SDK
4489498	TensorRT safety samples	**What is the issue?** The `trtexec_safe` built with `make TRT_STATIC=1` may report error when loading the dynamically built plugin `.so` file. **How does it impact the customer?** The error may occur when customers use the manually built `trtexec_safe` with option `make TRT_STATIC=1` to load the dynamically built plugin file. **If there is a workaround, what is it?** If the customer uses the `trtexec_safe` in the package, or uses the `make` without `TRT_STATIC=1` option, the error will not occur. **When can we expect the fix?** The `trtexec_safe` will not be fixed for the future release. The safety team will provide the workflow for the plugin and the use case in the bug will not be triggered.

Reference ID	Module	Description
		Is it for Standard/Safety, SDK/PDK? Safety, SDK

Chapter 6. TensorRT Release Properties

The following table describes the release properties and software versions.

Table 4. TensorRT Release Properties

	QNX AArch64	
	QNX Safety	**QNX Standard**
Supported NVIDIA CUDA® versions	11.4.28	11.4.28
Supported NVIDIA cuDNN versions	No	8.9.2.19
TensorRT Python API	No	No
NvOnnxParser	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 8.6.13 Safety Developer Guide Supplement for DRIVE OS. In the NVIDIA TensorRT 8.6.13 safety runtime, engine version numbers for major, minor, and patch must be equal to the runtime version numbers, and equal to 8.6.13.

6.1. 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 8.6.13 Developer Guide for DRIVE OS.

Table 5. Hardware and Precision Support for TensorRT 8.6.13

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

6.2. Software Versions Per Platform

Table 6. Software Versions per Platform for TensorRT 8.6.13

Platform	Compiler Version	Python Version
QNX AArch64	QNX 7.1.0 Q++ 8.3.0	N/A

6.3. Compatibility

TensorRT 8.6.13 has been tested with the following:

- CUDA 11.4.28
- cuDNN 8.9.2.19
- TensorFlow 1.15.5
- PyTorch 1.13.1
- ONNX 1.12.0 and opset 17
- DLA 3.14.3

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