NVIDIA TensorRT 11.0.1 Release Notes

Release 7.2.5 for NVIDIA DriveOS

NVIDIA Corporation

Jul 02, 2026

Contents

These release notes cover new features, enhancements, fixed issues, deprecations, upcoming changes, known issues, known limitations, and platform compatibility information for NVIDIA TensorRT 11.0.1 on NVIDIA DriveOS 7.2.5.

Chapter 1. Release Notes Revision History

Table 1.1: Document Revision History

Date	Summary of Change
March 31, 2026	Initial draft for NVIDIA TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5
April 1, 2026	Start of review
June 21, 2026	End of review
June 22, 2026	Approval review

1.1. Document Changes

Table 1.2: Document Changes

Date	Summary of Change
May 11, 2026	Reorganized the release notes for clarity: reordered the table of contents into a standard release-notes flow (revision history, new features, fixed issues, deprecations, upcoming changes, known issues, known limitations, reference material); renamed the “Deprecated APIs” chapter to “Deprecations” so it can cover all deprecated items, not just APIs; renamed the “TensorRT for DriveOS” chapter to “Packages and Components” to reflect its actual content; and moved issue 5897100 (remote auto-tuning slowdown on Thor iGPU) from known issues to fixed issues now that it is resolved. * - May 11, 2026
May 12, 2026	▶ Reframed the *Features and Enhancements* introduction to clarify that this page lists only NVIDIA DriveOS- and safety-specific additions; general TensorRT 11.0.x features are inherited from the enterprise release notes (linked). * Renamed the “TensorRT Standard Build” subsection to “TensorRT Standard Build (Inherited from Enterprise)” to reinforce the same scope distinction. ▶ Renamed the “TensorRT Standard Build” subsection to “TensorRT Standard Build (Inherited from Enterprise)” to reinforce the same scope distinction.
May 13, 2026	▶ Added an NVFP4 column to the **Hardware and Precision Support for TensorRT** table in *TensorRT Release Properties* (SM 11.0 / NVIDIA DRIVE AGX Thor = Yes), aligning the table with the NVFP4 (E2M1) safety-runtime support documented in *Features and Enhancements*. ▶ Added **Ubuntu 24.04 x86-64** (gcc 13.2.0, Python 3.12) and **QNX SDP 8.0.4** (gcc 12.2.0, Python N/A) rows to the **Software Versions per Platform for TensorRT** table in *TensorRT Release Properties*; the table previously listed only Ubuntu 24.04 AArch64. ▶ Updated the *Tested Compatibility* list in *TensorRT Release Properties* to ONNX 1.20.0 and opset 25 (previously ONNX 1.18.0 and opset 20).
June 4, 2026	▶ Documented the QNX-safe Safety package in *Features and Enhancements* and *Packages and Components*; reframed known issue 6156821 from a QNX Safety non-functional statement to *CUTLASS Static Kernels Excluded on QNX Safety*. ▶ Revised the *Known Limitations* *Restricted mode* section to *Build-time safety scope and isNetworkSupported()* with a link to *Deprecations*.

Chapter 2. New Features and Enhancements

This section describes the features and enhancements that are specific to NVIDIA TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5. To avoid duplication, general features inherited from the enterprise TensorRT 11.0.x release are not repeated here.

For the full list of features, enhancements, and bug fixes in the underlying TensorRT 11.0.x release, refer to the NVIDIA TensorRT 11.0.0 Release Notes.

2.1. TensorRT Standard Build (Inherited from Enterprise)

TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5 includes updates to the standard builder and runtime based on the TensorRT 11.0.x enterprise release. These changes provide improved performance, bug fixes, and new capabilities for standard (non-safety) engine building and inference. For details, refer to the enterprise release notes linked above.

2.2. FP8 Datatype Support in the Safety Runtime

The TensorRT safety runtime now supports the FP8 (E4M3) datatype on NVIDIA Thor. Safety engines can use FP8 weights and FP8 internal tensors (activations).

Note

FP8 is **not** supported on network input or output tensors in this release. Network inputs and outputs must use a non-FP8 type; FP8 conversion occurs only at engine-internal layer boundaries.

For guidance on FP8 quantization and accuracy considerations, refer to the *NVIDIA TensorRT 11.0.1 Developer Guide for DriveOS*.

2.3. NVFP4 Datatype Support in the Safety Runtime

The TensorRT safety runtime now supports the NVIDIA FP4 (NVFP4) (E2M1) datatype on NVIDIA Thor. Safety engines can use NVFP4 for weights (including pre-quantized weights) and for internal tensors in supported layers (notably matrix multiplication / GEMM).

Because NVFP4 does not have IEEE NaN/Inf representations, the safety runtime defines explicit error handling for cases when NaN/Inf values would be stored into NVFP4 tensors. Refer to the *NVIDIA TensorRT 11.0.1 Safety Production Guide for DriveOS* for the NaN/Inf semantics and the per-layer NVFP4 support scope.

Note

NVFP4 is **not** supported on network input or output tensors in this release. Network inputs and outputs must use a non- NVFP4 type; conversion to NVFP4 occurs only at engine-internal layer boundaries (for example, through quantize/dequantize layers).

For guidance on NVFP4 quantization and accuracy considerations, refer to the *NVIDIA TensorRT 11.0.1 Developer Guide for DriveOS*.

2.4. Auxiliary CUDA Stream Support in the Safety Runtime

Important

Default builder behavior changed in TensorRT 11.x. Engines built with default settings may request one or more auxiliary streams. TensorRT 10.x safety application code that loads such an engine and calls `executeAsync()` without first registering the auxiliary streams may fail at runtime. To preserve 10.x single-stream behavior with no application-code changes, set `IBuilderConfig::setMaxAuxStreams(0)` (or pass `--maxAuxStreams=0` to `trtexec`). To adopt multi-stream execution, refer to Adapting to Auxiliary CUDA Stream Support in the *NVIDIA TensorRT 11.0.1 Migration Guide for DriveOS*.

The TensorRT safety runtime now supports auxiliary CUDA streams, lifting the TensorRT 10.x restriction that forced every safety engine to execute on a single stream. Applications query the number of auxiliary streams a serialized engine requires with `nvinfer2::safe::ITRTGraph::getNbAuxStreams()` and register them with `setAuxStreams()`. The builder side is controlled by `IBuilderConfig::setMaxAuxStreams()` and the equivalent `trtexec --maxAuxStreams=N` flag.

The performance impact of multi-stream execution is workload-dependent. Benchmark your workloads with both single-stream and multi-stream builds to determine the best configuration for your application.

2.5. QNX-safe Safety Package with Limited Support

TensorRT 11.0.1 ships a QNX-safe Safety package for DriveOS 7.2.5 with limited support. Certain workloads may succeed. The limited support means that usage may lead to builder errors, runtime errors, degraded accuracy, or degraded performance on QNX Safety.

For package contents, platform availability, and further information, refer to *DriveOS QNX Safety*.

2.6. Debug Overlay Filesystem for Safe Engine Building on QNX Safety

Starting in TensorRT 11.0.1, components of the TensorRT safety builder ship in the debug overlay filesystem on QNX Safety. The `timing_server` executable and certain dependent libraries are now mounted in the `/nvidia_overlay` directory rather than `/usr/libnvidia`, where they previously shipped. The production safety runtime is unaffected: the `libnvinfer_safe.so*` libraries remain available at `/usr/libnvidia`, and the debug overlay is not required at runtime. Other platforms, such as QNX Standard and x86, are unchanged.

To build safe engines, mount the debug overlay by passing `debug_sr` to `bind_partitions`, and point the `trtexec --remoteAutoTuningConfig` flag at `/nvidia_overlay`. For the full procedure, refer to Using Debug Overlay Filesystem for Safe Engine Building on QNX Safety in the *NVIDIA TensorRT 11.0.1 Migration Guide for DriveOS*.

2.7. Documentation Updates

The TensorRT 11.0.1 documentation has been reorganized for better clarity:

- **NVIDIA TensorRT 11.0.1 Developer Guide for DriveOS**: Contains standard TensorRT documentation based on the enterprise TensorRT 11.0.0 release, modified for NVIDIA DriveOS 7.2 accuracy. Safety-specific content has been removed from this guide.
- **NVIDIA TensorRT 11.0.1 Safety Production Guide for DriveOS**: A dedicated PDF document containing all TensorRT safety-specific documentation. Refer to this supplement for safety runtime APIs, safety features, and safety-related migration guides.
- **NVIDIA TensorRT Migration Guide**: Includes a Safety Runtime chapter documenting API changes between TensorRT 10.x and TensorRT 11.x safety runtimes. TensorRT 11.x safety runtime support will be available in an upcoming DriveOS 7.2 release.

2.8. Samples Installation

Samples are no longer included in release packages. To build TensorRT samples, fetch them from GitHub and build as shown below.

```
git clone -b release/11.0 https://github.com/NVIDIA/TensorRT.git
cd TensorRT
```

(continues on next page)

(continued from previous page)

```
mkdir build && cd build
cmake .. \
  -DTRT_LIB_DIR=$TRT_LIBPATH \
  -DTRT_OUT_DIR=`pwd`/out \
  -DBUILD_SAMPLES=ON \
  -DBUILD_PARSERS=OFF \
  -DBUILD_PLUGINS=OFF
cmake --build . --parallel 4
```

Chapter 3. Fixed Issues

This section describes issues that were resolved in NVIDIA TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5. Each entry includes details about the original problem and how it was fixed.

3.1. TensorRT - QNX Performance

3.1.1. Issue 5897100: Remote Auto-Tuning Significantly Slower on NVIDIA DRIVE AGX Thor iGPU Than Orin iGPU in QNX

Reference ID: 5897100

What was the issue?

TensorRT remote auto-tuning ran significantly slower on NVIDIA DRIVE AGX Thor iGPU compared to Orin iGPU. INT8 models showed up to 23.45x slowdown, with an average of 7.11x across common models. Applications relying on remote auto-tuning (for building safe engines) on Thor iGPU for QNX safety proxy experienced significantly longer build times, particularly for INT8 models, which could cause model builds to time out.

Resolution

The issue is no longer reproducible in this release.

Affected platforms

Standard, SDK, QNX only (Thor iGPU)

3.1.2. Issue 5703925: FP8 MHA Operations Not Fused on QNX

Reference ID: 5703925

What was the issue?

FP8 Multi-Head Attention (MHA) operations were not fused on QNX, resulting in suboptimal performance. Models using FP8 MHA did not achieve the expected performance improvements over FP16, with observed performance degradation of approximately 162ms in Vision Transformer (ViT) workloads.

Resolution

FP8 MHA operations are now fused on QNX. Models using FP8 MHA achieve the expected performance improvements over FP16.

Affected platforms

Standard, SDK, QNX only

Chapter 4. Deprecations

This section lists items deprecated in NVIDIA TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5. Deprecated items may be removed in a future release; migrate before upgrading.

4.1. API Deprecations

Table 4.1: Deprecations

Deprecated API	Impact	Replacement API
`INetwork->addNormaliza1`	May be removed in the next major release. Migrate before upgrading.	`INetwork->addNormalizationV2()`

4.2. Frontend Safety Scope and Restricted Engines

DriveOS 7.2.5 moves safety scope enforcement to build time. Refer to Adapting to Build-Time Safety Scope Validation in the *NVIDIA TensorRT 11.0.1 Migration Guide for DriveOS*.

Starting in NVIDIA DriveOS 7.2.5, TensorRT removes the pre-build safety scope validator and the concept of "restricted engines." The `BuilderFlag::kSAFETY_SCOPE` API flag is deprecated in TensorRT 11.0.1 and has no effect beginning in DriveOS 7.2.5. The `--restricted` flag in `trtexec` is also removed.

Safety enforcement is now handled exclusively by the engine builder, which is the authoritative source of truth for what can be built in a safety context. This change removes false-negative rejections caused by the pre-build static rule set, which could not account for optimization profiles, fusion decisions, or the full range of builder-certified kernels.

4.3. `isNetworkSupported()` No Longer Validates Safety Scope

`IBuilder::isNetworkSupported()` still runs for safety `EngineCapability` configurations. It deterministically checks whether the network satisfies the constraints implied by the builder configuration (`EngineCapability`, `BuilderFlag`, and `DeviceType`); including standard validation wher-

ever applicable and a limited set of safety-specific rules (for example, hardware compatibility must be kNONE for safety engines, and incompatible BuilderFlag combinations such as kSAFETY_SCOPE with kGPU_FALLBACK or kDLA_STANDALONE). A true return value does not guarantee that buildSerialized-Network() succeeds; per-layer safety scope and other restrictions are enforced at build time. In TensorRT 10.x, this API could be used as a pre-build safety-scope gate; that behavior is removed in 11.x.

Chapter 5. Upcoming Changes

This section describes changes scheduled for future NVIDIA TensorRT and DriveOS releases that are not yet active in TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5. Refer to this section to plan ahead for future migrations.

5.1. TensorRT Safety Builder Will Require Host C/C++ Compilers

In a future release of TensorRT, the build process for safe engines will require host C/C++ toolchains, including compilers and linkers. The builder uses these toolchains to compile generated CPU code that supports kernels created for safe engine execution. Install the toolchain on the machine where you run the TensorRT builder (for example, `trtexec`).

The toolchain must match the TensorRT target platform. An x86-64 toolchain with GCC is required for the TensorRT Safety Proxy on x86 hosts. For deployment to QNX Safety, install the BlackBerry® QNX® Software Development Platform (SDP) and make it available to the builder. For validated compiler versions, refer to **Software Versions per Platform** in *TensorRT Release Properties*.

TensorRT will document compiler configuration requirements in a future release.

5.2. Reference Checker

TensorRT plans to introduce a new reference checker verification tool that will enhance the safety of the TensorRT build flow and generate optimized engine blobs.

The TensorRT reference checker checks the numerical correctness of the generated engine. For each generated kernel in the engine plan, the reference checker executes the exact compiled kernel on a remote target board against a set of boundary test cases, and compares the kernel outputs against an independent CPU reference computed on the host. If the difference between a kernel's output and the reference exceeds the configured numerical tolerance, the tool reports the mismatch and identifies the affected layer. This tool does not check the performance of generated kernels.

For more information, refer to the NVIDIA Deep Learning Inference SEooC 2.2 Safety Tools Manual V0.2 for DriveOS 7.2.5.

Chapter 6. Known Issues

This section describes known issues in NVIDIA TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5. These are bugs or problems that have been identified and are being tracked for resolution. Each issue includes details about the problem, its impact, available workarounds, and expected resolution timeline.

6.1. TensorRT - QNX Builder

6.1.1. Issue 6360135: Remote Auto-Tuning Server Output Reports Incorrect TensorRT Version

Reference ID: 6360135

What is the issue?

During remote auto-tuning, the `timing_server` executable reports an incorrect TensorRT version. For example, the SSH remote command output reports:

```
SERVER_INFO: sm=-1 cudart_version=-1 os=QNX os_release=8.0.0 machine=ARMv9_
↪nVidia-Thor safety=1 dkg=0 trt_version=11.0.0.97
```

How does it impact the customer?

The reported version information might be misleading.

Workaround

To verify the TensorRT version in use, rely on other version outputs from `trtexec`, such as those reported by `trtexec --help`.

Expected fix

A fix is planned for a future TensorRT release.

Affected platforms

Standard, SDK, QNX only

6.2. TensorRT - Accuracy

6.2.1. Issue 6062305: Detection-Head Outputs Fall Below Cosine Similarity Threshold

Reference ID: 6062305

What is the issue?

For some object-detection models, a subset of the detection-head outputs can fall below the 0.95 cosine similarity threshold during accuracy validation, while the remaining outputs pass. The affected outputs are associated with ArgMax and TopK selection chains, where small numerical differences in close-scoring candidates can change which detections are selected. The highest-confidence detections remain accurate.

How does it impact the customer?

Models that depend on strict numerical accuracy (cosine similarity of at least 0.95) for detection-head outputs may not meet that threshold for the affected tensors, even though the top-ranked detections are correct.

Workaround

Evaluate detection quality using the top-ranked detections or application-level accuracy metrics in addition to full-tensor cosine similarity. Ensure reference outputs are generated with a consistent framework, toolchain, and precision.

Expected fix

A fix is planned for a future TensorRT release.

Affected platforms

Standard, SDK, QNX, and NVIDIA DriveOS Linux

6.3. TensorRT - QNX Safety Runtime

6.3.1. Issue 6249419: TensorRT Safe Runtime May Fail to Load Engine on QNX Safety

Reference ID: 6249419

What is the issue?

When loading a safe engine on QNX Safety, an error like the following may be encountered for some models:

```
[FAILED_INITIALIZATION] CUDR Error: [cuModuleLoadData(&mModules[i],
↪cubins[i])]: 200
```

The root cause is that certain operations generate PIL, which is not supported on the safe CUDA runtime, as reported by `slog2info`:

```
CUDA: PIL is not supported, recompile with -Xptxas -pic=false, status : [200]
```

How does it impact the customer?

Certain models may fail at inference time on QNX Safety. This is known to affect models that use trigonometric functions such as `sin` and `cos`.

Workaround

Use the safety proxy on QNX Standard.

Expected fix

A fix is planned for a future TensorRT release.

Affected platforms

Standard, SDK, QNX Safety only

Chapter 7. Known Limitations

This section describes known limitations in NVIDIA TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5. These are features or behaviors that are not supported or have restrictions in the current release. Unlike known issues, these limitations are expected behaviors that may be addressed in future releases.

7.1. Datatypes

7.1.1. FP8 (E4M3)

Network input and output tensors cannot use FP8 in this release; conversion must happen at engine-internal layer boundaries (for example, through quantize/dequantize layers). For the full safety-scope datatype tables, see the **Default Constraints** table under *Layer Limitations in GPU Safety Restricted Mode* in the *NVIDIA TensorRT 11.0.1 Safety Production Guide for DriveOS*.

7.1.2. NVFP4 (E2M1)

Network input and output tensors cannot use `NVFP4` in this release; conversion must happen at engine-internal layer boundaries (for example, through quantize/dequantize layers). For the full safety-scope datatype tables and the per-layer `NVFP4` support scope, see the **Default Constraints** table under *Layer Limitations in GPU Safety Restricted Mode* in the *NVIDIA TensorRT 11.0.1 Safety Production Guide for DriveOS*.

`NVFP4` is not supported for plugin field types.

7.2. Hardware Accelerators

7.2.1. DLA (Deep Learning Accelerator)

The Deep Learning Accelerator (DLA) is not supported in NVIDIA TensorRT 11.0.1 for NVIDIA DriveOS 7.2.5. Workloads that previously used DLA on TensorRT 10.x must either re-target to GPU-only execution (`EngineCapability::kSTANDARD` or `EngineCapability::kSAFETY`) or remain on the TensorRT 10.x branch until DLA is reintroduced in a future release. For migration guidance, see **DLA Considerations** in *Migrating TensorRT from 10.x to 11.x on NVIDIA DriveOS* in the *NVIDIA TensorRT 11.0.1 Migration Guide for DriveOS*.

Chapter 8. TensorRT Release Properties

This section provides detailed information about the NVIDIA TensorRT 11.0.1 release properties, including supported software versions, hardware compatibility, and platform-specific requirements for NVIDIA DriveOS 7.2.5.

8.1. Release Properties

The following table describes the release properties and software versions for each supported platform.

Table 8.1: TensorRT Release Properties

Property	Linux x86-64	Linux AArch64	QNX AArch64 Safety	QNX AArch64 Standard
CUDA version	13.3	13.3	13.3	13.3
Python API	Yes	Yes	No	No
C++ API NvOnnxParser	Yes	Yes	No	Yes

Note

Serialized engines are not generally portable across 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 11.0.1 Safety Production Guide for DriveOS 7.2.5.

8.2. Hardware and Precision Support

The following table lists NVIDIA hardware and the precision modes that each hardware supports.

Runtime Support:

- **Standard runtime**: TensorRT supports SM 7.5 and higher
- **Proxy runtime**: TensorRT supports hardware with a capability of 10.0 or 11.0
- **Safety runtime**: TensorRT supports hardware with a capability of 11.0

For more information, refer to the FAQ section in the NVIDIA TensorRT 11.0.1 Developer Guide for DriveOS 7.2.5.

Table 8.2: Hardware and Precision Support for TensorRT

SM	Example Device	TF32	FP32	FP16	FP8	BF16	NVFP	INT8	FP16 Tensor Cores	INT8 Tensor Cores
11.0	NVIDIA DRIVE AGX Thor (TensorRT Safety, Safety Proxy)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10.0	B200 (TensorRT Standard, Safety Proxy)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

8.3. Software Versions per Platform

The following table lists the compiler and Python versions required for each supported platform.

Table 8.3: Software Versions per Platform for TensorRT

Platform	Compiler	Python
Ubuntu 24.04 AArch64	gcc 13.2.0	3.12
Ubuntu 24.04 x86-64	gcc 13.2.0	3.12
QNX SDP 8.0.4	gcc 12.2.0	N/A

8.4. Tested Compatibility

NVIDIA TensorRT 11.0.1 has been tested with the following software versions:

- CUDA 13.3
- PyTorch >= 2.4.0
- ONNX 1.20.0 and opset 25

Chapter 9. Packages and Components

This section describes the NVIDIA TensorRT packages available for different NVIDIA DriveOS platforms in the TensorRT 11.0.1 release. Each package is tailored to the specific platform and includes the components needed for that environment.

9.1. Available Packages

9.1.1. DriveOS x86

Package Type: Standard+Safety Proxy

Included Components:

- Builder
- Standard runtime
- Proxy runtime
- Parsers
- Python bindings
- Sample code
- Standard and safety headers
- Documentation

Capabilities:

- Build engines for standard runtime and safety proxy runtime
- Build standard engines restricted to operations that the safety and proxy runtimes support
- Include safety headers for compatibility with future NVIDIA DriveOS 7.2 releases

9.1.2. DriveOS Linux Standard

Package Type: Standard

Included Components:

- Builder
- Standard runtime

- Parsers
- Python bindings
- Sample code
- Standard and safety headers
- Documentation

Capabilities:

- Build engines for standard runtime
- Build standard engines restricted to operations supported by safety and proxy runtimes
- Includes safety headers for compatibility with future NVIDIA DriveOS 7.2 releases

9.1.3. DriveOS QNX Standard

Package Type: Standard+Safety Proxy

Included Components:

- Builder
- Standard runtime
- Proxy runtime
- Parsers
- Sample code
- Standard and safety headers
- Documentation

Capabilities:

- Build engines for standard runtime and safety proxy runtime
- Build standard engines restricted to operations supported by safety and proxy runtimes
- Includes safety headers for compatibility with future NVIDIA DriveOS 7.2 releases

9.1.4. DriveOS QNX Safety

Package Type: Safety

Included Components:

- Safety runtime
- Safety headers only
- API documentation specific to the safety runtime

Capabilities:

- Load and execute safety engines (engine capability kSAFETY)
- Available only on QNX safety platforms

9.2. Runtime Components

The following runtime components are available across different packages:

Standard Runtime

A library that allows applications to load serialized engine plans and perform inference. Supports unrestricted engines and engines with operations limited to the safety scope.

Proxy Runtime

A version of the safety runtime for platforms that are not safety-certified. Available on:

- NVIDIA DRIVE OS QNX SDK
- NVIDIA DRIVE OS QNX PDK

The proxy runtime is part of the development flow for safety but is not certified itself. It only supports engines with engine capability kSAFETY (safe engines).

Safety Runtime

A library that allows applications to load serialized engine plans and perform inference. Available only on QNX safety platforms. The safety runtime only supports engines with engine capability kSAFETY (safe engines). For more information, refer to *DriveOS QNX Safety*.

Safety Headers

Headers that allow applications to compile against both the proxy runtime and the safety runtime. Included in packages that support safety-related development. For more information, refer to *DriveOS QNX Safety*.

Chapter 10. Safe Engine Version Compatibility

This section describes the version compatibility requirements for safe engines used with the NVIDIA TensorRT safety runtime.

10.1. Version Compatibility Requirements

The NVIDIA TensorRT 11.0.1 safety runtime only supports engines built with TensorRT 11.0.1. Engines built with earlier versions of TensorRT are not compatible and must be rebuilt.

Important

When upgrading to TensorRT 11.0.1, you must rebuild all safe engines using the new version. Existing engines from previous TensorRT versions will not work with the 11.0.1 safety runtime.

Copyright