DEEPSTREAM SDK 6.2 FOR NVIDIA DGPU/X86 AND JETSON

RN-09353-003 | February 02, 2023
Advance Information | Subject to Change

6.2 Release Notes
# TABLE OF CONTENTS

## 1.0 ABOUT THIS RELEASE

1.1 What's New ................................................................. 3
1.1.1 DS 6.2 ............................................................................. 3
1.1.2 DS 6.1.1 (Previous release) ........................................... 5
1.1.3 Graph Composer 2.5......................................................... 6
1.2 Contents of this Release..................................................... 7
1.3 Documentation in this Release ............................................. 7
1.4 Differences With Deepstream 6.1 ........................................ 7

## 2.0 LIMITATIONS

## 3.0 NOTES

3.1 Applications May Be Deployed in a Docker Container ................. 12
3.2 Sample Applications Malfunction if Docker Environment Cannot Support Display ............. 15
3.3 Installing Deepstream On Jetson ......................................... 16
3.4 Triton Inference Server In Deepstream.................................... 16
1.0 ABOUT THIS RELEASE

These release notes are for the NVIDIA® DeepStream SDK for NVIDIA® Tesla®, NVIDIA® Ampere®, NVIDIA® Hopper®, NVIDIA® Ada Lovelace®, NVIDIA® Jetson AGX Xavier™, NVIDIA® Jetson Xavier™ NX, NVIDIA® Jetson AGX Orin™, and NVIDIA® Jetson Orin™ NX.

1.1 WHAT’S NEW

The following new features are supported in this DeepStream SDK release:

1.1.1 DS 6.2

- Supports Triton 22.09 for x86/dGPU, Triton 23.01 for Jetson and Rivermax v1.20.
- Jetson package based on JP 5.1 GA (r35.2.1 BSP).
- DeepSORT tracker support.
- REST API support to control DeepStream pipeline on-the-fly (Alpha, x86 only).
- LIDAR support (Alpha).
- ASR and TTS support on Jetson.
- Enable Pre-Processing plugin with SGIE.
- Dewarper enhancements to support additional projections.
- Support Google protobuf encoding and decoding message to message brokers (only Kafka).
- Nvdsxfer plugin implementation (NVLink based, x86 only).
- Enhancements in new Gst-nvstreammux plugin. New nvstreammux can be enabled by exporting USE_NEW_NVSTREAMMUX=yes. For more information, see the “Gst-nvstreammux” section in the NVIDIA DeepStream SDK Developer Guide 6.2 Release.
- Performance optimizations.
- Improved NVDCF tracker.
- GPU based drawing for text, line, circles, arrow using OSD plugin (alpha).
NVIDIA TAO toolkit (previously called NVIDIA Transfer Learning Toolkit) Models from [https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps](https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps) (branch: release/tao4.0_ds6.2ga) integrated into SDK.


Added support for ReIdentificationNet Model and Retail Object Recognition Model.

New deepstream-mdx-perception-app application for embedding vector from re-identification network, to identify objects captured in different scenes.

nvdsudpsink plugin optimizations for supporting Mellanox NIC for transmission.

Improved stability.

depthstream-dewarper-app now supports following new projections: Fisheye to Perspective, Fisheye to Fisheye, Fisheye to Cylindrical, Fisheye to Equirectangular, Fisheye to Panini, Perspective to Equirectangular, Perspective to Panini, Equirectangular to Cylindrical, Equirectangular to Equirectangular, Equirectangular to Fisheye, Equirectangular to Panini, Equirectangular to Perspective, Equirectangular to PushBroom, Equirectangular to Stereographic, Equirectangular to Vertical Radial Cylindrical.

New plugins/bins:

- **Gst-nvdsxfer** plugin transfers data over nvlink across multiple GPU under single process.
- **gst-nvmultiurisrcbin**: The bin to integrate nvurisrcbin, and nvstreammux into a single GstBin. The bin can be configured to act as REST API server.

New sample applications:

- DeepStream Can Orientation Sample App: Demonstrates can orientation detection with CV-based VPI template matching algorithm. VPI template matching is implemented with DeepStream video template plugin.
- Sample app to demonstrate loading CUDA memory from appsrsrc in the pipeline.

Python bindings and samples updates:

- New samples:
2. Deepstream-imagedata-multistream-cupy: demonstrating GPU buffer access for decoded images via CuPy.
3. Deepstream-custom-binding-test: demonstrating use of custom user metadata. This is a sample for the new custom user metadata bindings guide.
   - New guides for adding custom bindings including custom user metadata.

Opensource plugins:
- Triton plugin Gst-nvinferserver.
- Dewarper plugin Gst-nvdewarper.

1.1.2 DS 6.1.1 (Previous release)
- Supports Triton 22.07 and Rivermax v1.11.5.
- Jetson package based on JP 5.0.2 GA.
- Enhancements in new Gst-nvinferserver plugin to support CUDA shared memory (on x86/dGPU) for input tensors in gRPC mode.
- Supports YoloV3 post-processing on CUDA.
- DeepSORT tracker support (Alpha).
- Cloud to Device support for AMQP.
- Enhance nvinferserver to work with Preprocess plugin.
- Enhancements in new Gst-nvstreammux plugin. New nvstreammux can be enabled by exporting USE_NEW_NVSTREAMMUX=yes. For more information, see the “Gst-nvstreammux” section in the NVIDIA DeepStream SDK Developer Guide 6.1.1 Release.
- Performance optimizations.
- Improved NVDCF tracker.
- NVIDIA TAO toolkit (previously called NVIDIA Transfer Learning Toolkit) Models from https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps (branch: release/tao3.0_ds6.1.1ga) integrated into SDK.
- Improved stability.
- New Sample TritonOnnxYolo added to run Triton inference with dynamic-sized output tensors even with zero bytes using Onnx YoloV3 models. Inside the sample, A DS-Triton(gst-nvinferserver) custom-lib implementation shows users how to do multi-input tensors preprocessing and mixed-batch tensors postprocessing.
- Python bindings and samples updates:
• New sample application deepstream-demux-muti-in-multi-out added to demonstrate demuxing multi-stream batch into separate output streams.
• Updated Jupyter notebook deepstream_test_4.ipynb.
• Minor bindings fixes.


1.1.3 Graph Composer 2.5

► Graph Execution Engine
  • Graph runtime to execute graphs implemented based on Graph Specification
  • Supported on Ubuntu 20.04 x86_64 and NVIDIA Jetson

► Graph composer tools
  • Composer with new UI
    • x86 only (Ubuntu 20.04)
    • User friendly editor view
    • Registry list in view
    • Graph edit with various options
    • Graph open/save
    • Property editor
    • Graph Launcher
    • Container Builder Launcher
    • Registry options
    • Extension Generator
    • Subgraph support

► Registry CLI
  • Local and NVIDIA Cloud repository
  • Version management based on Semantic versioning
  • Command Line Interface tool
  • Graph install for graph deploy
  • Container Builder CLI

• WebRTC and OV streaming
1.2 CONTENTS OF THIS RELEASE

This release includes the following:

- The DeepStream SDK. Refer to NVIDIA DeepStream SDK Developer Guide 6.2 Release for a detailed description of the contents of the DeepStream release package. The Developer Guide also contains other information to help you get started with DeepStream, including information about system software and hardware requirements and external software dependencies that you must install before you use the SDK.

  - For detailed troubleshooting information and frequently asked questions, see the “DeepStream Troubleshooting and FAQ Guide” section in the NVIDIA DeepStream SDK Developer Guide 6.2 Release.

- Graph Composer 2.5 and DeepStream reference graphs for dGPU and Jetson.
- LICENSE.txt contains the license terms of third-party libraries used.

1.3 DOCUMENTATION IN THIS RELEASE

This release contains the following documentation.

- NVIDIA DeepStream SDK Developer Guide 6.2 Release
- NVIDIA DeepStream SDK API Reference
- NVIDIA DeepStream Python API Reference

1.4 DIFFERENCES WITH DEEPSTREAM 6.1

gstreamer1.0-libav, libav, OSS encoder, decoder plugins (x264/x265) and audioparsers packages are removed in DeepStream dockers. You may install these packages based on your requirement (gstreamer1.0-plugins-good/ gstreamer1.0-plugins-bad/ gstreamer1.0-plugins-ugly). While running DeepStream applications inside dockers, you may see the following warnings:

WARNING from src_elem: No decoder available for type 'audio/mpeg, mpegversion=(int)4, framed=(boolean)true, stream-format=(string)raw, level=(string)2, base-profile=(string)lc, profile=(string)lc, codec_data=(buffer)119056e500, rate=(int)48000, channels=(int)2'.

Debug info: gsturidecodebin.c(920): unknown_type_cb ():
To avoid such warnings, install `gstreamer1.0-libav` and `gstreamer1.0-plugins-good` inside docker.

Specifically, for deepstream-nmos, deepstream-avsync-app and python based deepstream-imagedata-multistream app you would need to install `gstreamer1.0-libav` and `gstreamer1.0-plugins-good`.

`Gst-nveglglessink` plugin is deprecated. Use `Gst-nv3dsink` plugin for Jetson instead.
2.0 LIMITATIONS

This section provides details about issues discovered during development and QA but not resolved in this release.

- With V4L2 codecs only MAX 1024 (decode + encode) instances are provided. The maximum number of instances can be increased by doing changes in open-source code.
- `detected-min-w` and `detected-min-h` must be set to values larger than 32 in the primary inference configuration file (`config_infer_primary.txt`) for `gst-dsexample` on Jetson.
- The Kafka protocol adapter sometimes does not automatically reconnect when the Kafka Broker to which it is connected goes down and comes back up. This requires the application to restart.
- If the `nvds` log file `ds.log` has been deleted, to restart logging you must delete the file `/run/rsyslogd.pid` within the container before reenabling logging by running the `setup_nvds_logger.sh` script. This is described in the “nvds_logger: Logging Framework” sub-section in the “Gst-nvmsgbroker” section in the NVIDIA DeepStream Developer Guide 6.2 Release.
- Running a DeepStream application over SSH (via putty) with X11 forwarding does not work.
- DeepStream currently expects model network width to be a multiple of 4 and network height to be a multiple of 2.
- Triton Inference Server implementation in DeepStream currently supports a single GPU. The models need to be configured to use a single GPU.
- For some models output in DeepStream is not exactly same as observed in TAO Toolkit. This is due to input scaling algorithm differences.
- Dynamic resolution change support is Alpha quality.
- On the fly Model update only supports same type of Model with same Network parameters.
- DeepStream cannot be installed on the current 16 GB Xavier NX production modules since Jetpack software takes the entire 16 GB emmc memory space. We recommend using Xavier NX developer kits with 32 GB SD card.
- Rivermax SDK is not part of DeepStream. So, the following warning is observed (gst-plugin-scanner:33257):

```
GStreamer-WARNING **: 11:38:46.882: Failed to load plugin '/usr/lib/x86_64-linux-gnu/gstreamer-1.0/deepstream/libnvdsgst_udp.so': librivermax.so.0: cannot open shared object file: No such file or directory
```

You can ignore this warning safely.

- Sample graphs containing NvDsMultiSrcInput component result in segmentation fault when an error occurs during graph/pipeline initialization.
- When using Composer WebSocket streaming, sometimes error like "Error while sending buffer: invalid state" is seen, or the window becomes unresponsive. Refreshing the browser page might fix it.
- Composer WebRTC Streaming is supported only on RTX GPUs.
- On jetson, when the screen is idle, fps is lowered for DeepStream applications. this behavior is by design to save power. However, if user does not want screen idle then refer to the FAQ for WAR.
- RDMA functionality only supported on x86 and that too in x86 devel docker for now.
- You cannot build the DeepStream out of the box on jetson dockers except its Triton variant.
- Optical flow plugin not supported on NVIDIA® Jetson AGX Orin™ and NVIDIA® Jetson Orin™ NX.
- There can be performance drop from TensorRT to Triton for some models (5 to 15%).
- To generate the YOLOV3, YOLOV4 and YOLOV4-tiny model engines, the precision of some layers should be specified as FP32 for TensorRT 8.4.1.5 limitations. The solution is updated in [https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps](https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps).
- NVRM: XID errors seen sometimes when running 225+ streams on Ampere and Hopper.
- NVRM: XID errors seen on some setups with gst-dsexample and transfer learning sample apps.
- Sometimes during deepstream-testsr app execution, assertion "GStreamer-CRITICAL **: 12:55:35.006: gst_pad_link_full: assertion 'GST_IS_PAD sinkpad)' failed" is seen which can be safely ignored.
- Gst-nvdsasr plugin and deepstream-avsync-app is not supported on Hopper GPU.
- While running deepstream-image-decode-app assertion being seen which can be safely ignored.
- ASR and TTS plugins are not supported on NVIDIA Hopper since RIVA container for the same not available.
- deepstream-server app is not supported with new nvstreammux plugin.
TAO point-pillar model works only in FP32 mode.

REST API support for few components (decoder, preprocessor, nvinfer along with stream addition deletion support) with limited configuration options. However, user can extend the functionality with steps mentioned in SDK documentation.

Critical error (masked_scan_uint32_peek: assertion '(guint64) offset + size <= reader->size - reader->byte' failed) observed while running python segmentation application but it can be ignored safely.

While running two instances of nveglglessink component on Jetson you would see error like “NvVicCompose Failed”. In such case user can use nv3dsink component instead nveglglessink.

On NVIDIA® Jetson AGX Orin™ rarely “SyncPoint wait for Profiling” error is observed.

On Jetson dockers while running DeepStream applications the error “modprobe: FATAL: Module nvidia not found...” is seen but can be safely ignored.

With Basler camera, on Jetson, images with width only multiple of 4 supported.

DLA inference performance drop is observed for Peoplenet, TrafficCamNet, DashCamNet, FRCNN, RetinaNet, Bodypose3D, Action recognition 2D & 3D models.
3.0 NOTES

- Optical flow is supported only on dGPUs having Turing architecture (onwards) and on NVIDIA® Jetson AGX Xavier™ and NVIDIA® Jetson Xavier™ NX.
- REST API commands only work after the video shows up on the host screen.
- The REST API server application deepstream-server-app should be used with dsserver_config.yml config file. dsserver_pgie_config.yml should not be used as this is inference config file.
- On vGPU, only cuda device memory NVBUF_MEM_CUDA_DEVICE supported.
- Jetson AGX Orin would be useful compared to Jetson AGX Xavier for the cases where DeepStream performance on Jetson AGX Xavier is GPU bound.

**Note:**
- OpenCV is deprecated by default. But you can enable OpenCV in plugins such as nvinfer (nvdsinfer) and dsexample (gst-dsexample) by setting WITH_OPENCV=1 in the Makefile of these components. Refer to the component README for more instructions.
- When using docker make sure libopencv-dev package is installed inside docker if the Application requires it.

3.1 APPLICATIONS MAY BE DEPLOYED IN A DOCKER CONTAINER

Applications built with DeepStream can be deployed using a Docker container, available on NGC (https://ngc.nvidia.com/). Sign up for an NVIDIA GPU Cloud account and look for DeepStream containers to get started.
After you sign into your NGC account, navigate to Dashboard → Setup → Get API key to get your nvcr.io authentication details.

As an example, you can use the DeepStream 6.2. docker containers on NGC and run the deepstream-test4-app sample application as an Azure edge runtime module on your edge device.

The following procedure deploys deepstream-test4-app:

- Using a sample video stream (sample_720p.h264)
- Sending messages with minimal schema
- Running with display disabled
- Using message topic mytopic (message topic may not be empty)

Set up and install Azure IoT Edge on your system with the instructions provided in the Azure module client README file in the deepstream-6.2 package:

<deepstream-6.2_package>/sources/libs/azure_protocol_adaptor/module_client/README

**Note:** For the Jetson platform, omit installation of the Moby packages. Moby is currently incompatible with NVIDIA Container Runtime.

See the Azure documentation for information about prerequisites for creating an Azure edge device on the Azure portal:

https://docs.microsoft.com/en-us/azure/iot-edge/how-to-deploy-modules-portal#prerequisites

**To deploy deepstream-test4-app as an Azure IoT edge runtime module**

1. On the Azure portal, click the IoT edge device you have created and click Set Modules.
2. Enter these settings:

   **Container Registry Settings:**
   
   Name: NGC
   Address: nvcr.io
   User name: $oauth_token
   Password: use the password or API key from your NGC account

   **Deployment modules:**
   Add a new module with the name ds.

   **Image URI:**
   
   - For x86 dockers:
docker pull nvcr.io/nvidia/deepstream:6.2-devel
docker pull nvcr.io/nvidia/deepstream:6.2-samples
docker pull nvcr.io/nvidia/deepstream:6.2-iot
docker pull nvcr.io/nvidia/deepstream:6.2-base
docker pull nvcr.io/nvidia/deepstream:6.2-triton

- For Jetson dockers:
  
  docker pull nvcr.io/nvidia/deepstream-l4t:6.2-base
  docker pull nvcr.io/nvidia/deepstream-l4t:6.2-iot
  docker pull nvcr.io/nvidia/deepstream-l4t:6.2-samples
  docker pull nvcr.io/nvidia/deepstream-l4t:6.2-triton

Container Create options:

- For Jetson:

  
  
  ```json
  {
    "HostConfig": {
      "Runtime": "nvidia"
      "WorkingDir": "/opt/nvidia/deepstream/deepstream/sources/apps/sample_apps/deepstream-test4",
      "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream/bin/deepstream-test4-app",
        "-i", "/opt/nvidia/deepstream/deepstream/samples/streams/sample_720p.h264",
        "-p",
        "/opt/nvidia/deepstream/deepstream/lib/libnvds_azure_edge_proto.so",
        "--no-display",
        "-g",
        "-1",
        "--topic",
        "mytopic"
      ]
  }
  ```

- For X86:

  
  ```json
  {
    "HostConfig": {
      "Runtime": "nvidia"
    },
    "WorkingDir": "/opt/nvidia/deepstream/deepstream/sources/apps/sample_apps/deepstream-test4",
    "ENTRYPOINT": [
      "/opt/nvidia/deepstream/deepstream/bin/deepstream-test4-app",
      "-i", "/opt/nvidia/deepstream/deepstream/samples/streams/sample_720p.h264",
      "-p",
      "/opt/nvidia/deepstream/deepstream/lib/libnvds_azure_edge_proto.so",
      "--no-display",
      "-g",
      "-1",
      "--topic",
      "mytopic"
    ]
  }
  ```
3. Specify route options for the module:

- Option 1: Use a default route where every message from every module is sent upstream.

```json
{
    "routes": {
        "route": "FROM /messages/* INTO $upstream"
    }
}
```

- Option 2: Specific routes where messages sent upstream can be filtered based on topic name. For example, in the sample test programs, topic name mytopic is used for the module name ds:

```json
{
    "routes": {
        "route": "FROM /messages/modules/ds/outputs/mytopic INTO $upstream"
    }
}
```

3.2 SAMPLE APPLICATIONS MALFUNCTION IF DOCKER ENVIRONMENT CANNOT SUPPORT DISPLAY

If the Docker environment cannot support display, the sample applications `deepstream-test1`, `deepstream-test2`, `deepstream-test3`, and `deepstream-test4` do not work as expected.

**Workaround:**

To correct this problem, you must recompile the test applications after replacing `nveglglessink` on x86 and `nv3dsink` on Jetson with `fakesink`. With `deepstream-test4`, you also have the option to specify `fakesink` by adding the `--no-display` command line switch.
Alternatively virtual display can be used. For more information refer to “How to visualize the output if the display is not attached to the system” section in “Quick Start Guide” section of NVIDIA DeepStream Developer Guide 6.2 Release.

### 3.3 Installing DeepStream on Jetson

1. Download the NVIDIA SDK Manager to install JetPack 5.1 GA.
2. Select all the JetPack 5.1 components except DeepStreamSDK from the “Additional SDKs” section.

Refer to the “Quick Start Guide” section in NVIDIA DeepStream Developer Guide 6.2 Release to update additional BSP libraries if available. Continue with the DeepStream installation instructions after the BSP update.

**Note:** NVIDIA Container Runtime package shall be installed using JetPack 5.1 GA and is a pre-requisite for all DeepStream L4T docker containers.

### 3.4 Triton Inference Server in DeepStream

Triton inference server (version 22.09) on dGPU is supported only via docker `deepstream:6.2-triton` for x86. On Jetson we support that with or without docker.

Refer to the NVIDIA DeepStream Development Guide 6.2 Release for more details about Triton inference server.

Triton inference server Supports following frameworks:

<table>
<thead>
<tr>
<th>Framework</th>
<th>Tesla</th>
<th>Jetson</th>
<th>Notes / Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorRT</td>
<td>Yes</td>
<td>Yes</td>
<td>Supports TensorRT plan or engine file (.plan)</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Yes</td>
<td>Yes</td>
<td>Supports TensorRT optimization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Supported model formats: GraphDef or SavedModel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other TF formats such as checkpoint variables or estimators not directly supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Supports both Tensorflow 1.x and Tensorflow 2.x. Triton defaults to use Tensorflow 1.x. If users need to run Tensorflow 2.x models, need to update plugin config with:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>infer_config{ backend {</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trt_is { model_repo{</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>backend_configs {</td>
</tr>
</tbody>
</table>

DeepStream SDK 6.2 for NVIDIA dGPU/X86 and Jetson

RN-09353-003 | 16
For more information refer to the following links:

- Triton inference server model repository: https://docs.nvidia.com/deeplearning/sdk/triton-inference-server-guide/docs/model_repository.html
  
  Also contains more information on the supported frameworks.


- TensorFlow saved model: https://www.tensorflow.org/guide/saved_model#the_savedmodel_format_on_disk
Notice

THE INFORMATION IN THIS DOCUMENT AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION REFERENCED IN THIS DOCUMENT IS PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the product described in this document shall be limited in accordance with the NVIDIA terms and conditions of sale for the product. THE NVIDIA PRODUCT DESCRIBED IN THIS DOCUMENT IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE, AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE (INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

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

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

Trademarks

NVIDIA, the NVIDIA logo, TensorRT, NVIDIA Ampere, NVIDIA Hopper and NVIDIA Tesla are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright © 2023 NVIDIA CORPORATION & AFFILIATES. All rights reserved.