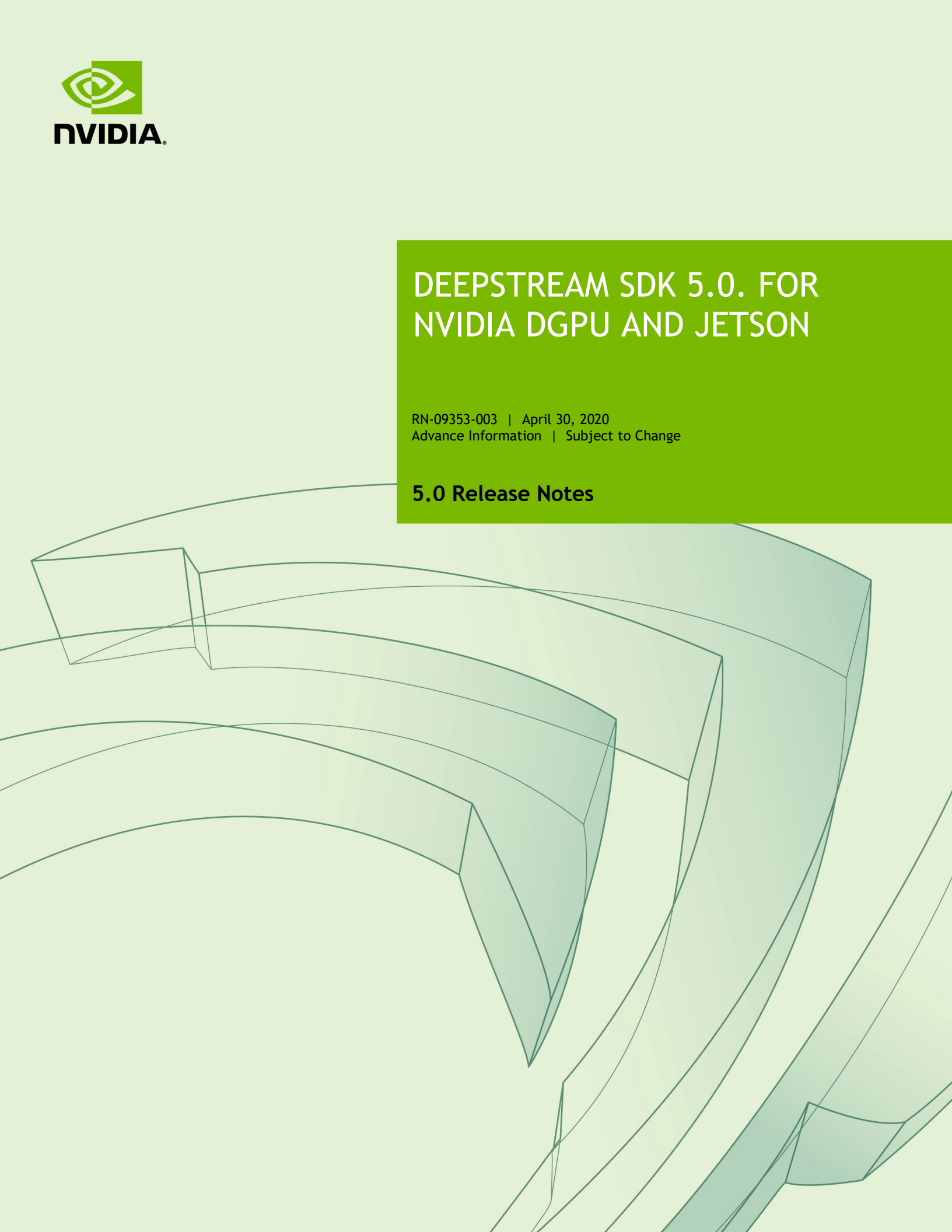

DEEPSTREAM SDK 5.0. FOR NVIDIA DGPU AND JETSON

RN-09353-003 | April 30, 2020
Advance Information | Subject to Change

5.0 Release Notes

TABLE OF CONTENTS

1.0 ABOUT THIS RELEASE

These release notes are for the NVIDIA® DeepStream SDK for NVIDIA® Tesla®, NVIDIA® Jetson AGX Xavier™, NVIDIA® Jetson Xavier™ NX, NVIDIA® Jetson Nano™, and NVIDIA® Jetson™ TX2 platforms.

1.1 WHAT'S NEW

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

- Support for Triton Inference Server
- On the fly Model updates
- Event based Smart Record
- Cloud to device messaging
- Improved NVDCF tracker
- Facility to attach encoded detected Image objects as meta data.
- Sample application which showcases use of opencv in dsexample plugin
- Python binding enhancements
    - Access to frame image data as NumPy array
    - Access to inference output tensor data
    - Additional sample applications
        - Probe for image data, then use OpenCV to annotate and save frames to file
        - Probe for inference output tensors to parse in Python
        - USB camera input
        - RTSP stream output
- Better time-stamp handling for live RTSP cameras
- DRC stream support for Jetson
- 10 Bit H264 and H265 stream support
- Misc. bug fixes and Improved stability

- Support for flowing Metadata attached before Gst-nvv4l2 decoder
- Gst-nvinfer plugin:
  - Support for TensorRT 7.0+:
  - Explicit Full Dimension Network Support
  - Non-maximum Suppression (NMS) for bounding box Clustering
  - On-the-fly model update (Engine/Plan file only)
  - Support for yolov3-ssp detector
- New plugins:
  - Gst-nvdsanalytics plugin for ROI detection, line crossing and direction detection
  - Gst-nvinferserver plugin for supporting Triton inference server using C++ client APIs (https://docs.nvidia.com/deeplearning/sdk/triton-inference-server-master-branch-guide/docs/index.html)
- New sample applications:
  - Analytics example: Demonstrates batched analytics like ROI filtering, Line crossing, direction detection and overcrowding
  - OpenCV example: Demonstrates the use of OpenCV in dsexample plugin
  - Image as Metadata example: Demonstrates how to attach encoded object image as meta data and save the images in jpeg format
  - Appsrc and Appsink example: Demonstrates AppSrc and AppSink usage for consuming and giving data from non-DS code respectively.

Note: **Existing DeepStream 4.0 Application can be used with DeepStream 5.0 Refer "Application Migration to DeepStream 5.0 from DeepStream 4.X" in *DeepStream 5.0 Plugin Manual*.**

1.2 CONTENTS OF THIS RELEASE

This release includes the following:

- The DeepStream SDK. Refer to *NVIDIA DeepStream SDK V5.0 Development Guide* for a detailed description of the contents of the DeepStream release package. The *Development 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 information about GStreamer plugins, metadata usage, Dockers, application and plugin migration to DeepStream 5.0, troubleshooting, and a FAQ, see the *DeepStream 5.0 Plugin Manual*.
- DeepStream SDK for dGPU and Jetson Software License Agreement (SLA).
- `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 Development Guide*
- *NVIDIA DeepStream SDK API Reference*
- *NVIDIA DeepStream SDK Plugin Manual*

1.4 DIFFERENCES WITH DEEPSTREAM 4.0

These are the major differences from DeepStream 4.0:

- The 360° camera use case is not supported. Docker container for the same to be updated later for DeepStream 5.0 specific changes
- Bounding box coordinates are now in float data type.

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, thereby requiring application restart.
- If the nvds log file, `ds.log`, has been deleted, then 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 as described in the nvds_logger section of the *DeepStream Plugin Manual*.
- On NVIDIA® Jetson AGX Xavier™, more than 50 instances of certain 1080p H.265 streams are not working due to limited memory for the decoder.
- On Jetson, 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 sometime output in DeepStream is not exactly same as observed in Transfer Learning Toolkit. This is due to input scaling algorithm differences.
- DRC support is Alpha quality.
- On the fly Model update only supports same type of Model with same Network parameters.
- Triton Inference Server is not supported on Jetson Nano, TX1 and TX2 in this release.

- On T4 servers the performance is degraded when used with Nvidia driver 430+ version. Refer section 3.4 for workaround.
- On Jetson there is performance degradation when NVDCF based tracker is used. This will be fixed in GA release.

3.0 NOTES

- Optical flow is only supported on dGPUs having Turing architecture and on NVIDIA® Jetson AGX Xavier™.
- NVIDIA® Jetson Nano™ and NVIDIA® Jetson™ TX2 support only FP16 and FP32 network precisions with NVIDIA® TensorRT™.
- Jetson AGX Xavier supports INT8, FP16 and FP32 network precisions with TensorRT.

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 in to your NGC account, go to Dashboard→ Setup→ Get API key to get your `nvcr.io` authentication details.

As an example, you can use the DeepStream 5.0 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 `deepstream5.0` package:

```
<deepstream-
5.0_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: $oauthtoken
   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:5.0-dp-20.04-devel
   docker pull nvcr.io/nvidia/deepstream:5.0-dp-20.04-samples
   docker pull nvcr.io/nvidia/deepstream:5.0-dp-20.04-iot
   docker pull nvcr.io/nvidia/deepstream:5.0-dp-20.04-base
   docker pull nvcr.io/nvidia/deepstream:5.0-dp-20.04-triton
   ```

   For Jetson dockers:

   ```
   docker pull nvcr.io/nvidia/deepstream-l4t:5.0-dp-20.04-samples
   docker pull nvcr.io/nvidia/deepstream-l4t:5.0-dp-20.04-iot
   docker pull nvcr.io/nvidia/deepstream-l4t:5.0-dp-20.04-base
   ```

   Container Create options:

   - For Jetson:

   ```
   {
       "HostConfig": {
               "Runtime": "nvidia"
       },
       "WorkingDir": "
   ```

```
/opt/nvidia/deepstream/deepstream-
5.0/sources/apps/sample_apps/deepstream-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream-5.0/bin/deepstream-test4-
app",
        "-i", "/opt/nvidia/deepstream/deepstream-5.0/
samples/streams/sample_720p.h264",
        "-p",
"/opt/nvidia/deepstream/deepstream-
5.0/lib/libnvds_azure_edge_proto.so",
        "--no-display",
        "-s",
        "1",
        "--topic",
        "mytopic"
    ]
}
```

- For X86:

```
{
    "HostConfig": {
        "Runtime": "nvidia"
    },
    "WorkingDir": "/opt/nvidia/deepstream/deepstream-
5.0/sources/apps/sample_apps/deepstream-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream-5.0/bin/deepstream-test4-
app",
        "-i", "/opt/nvidia/deepstream/deepstream-
5.0/samples/streams/sample_720p.h264",
        "-p",
        "/opt/nvidia/deepstream/deepstream-
5.0/lib/libnvds_azure_edge_proto.so",
        "--no-display",
        "-s",
        "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.

     ```
     {
       "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 `ds5`:

```
    {
        "routes": {
            "route": "FROM /messages/modules/ds5/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.

To correct this problem, you must recompile the test applications after replacing `nveglglessink` with `fakesink`. With `deepstream-test4`, you also have the option to specify `fakesink` by adding the `--no-display` command line switch.

3.3 INSTALLING DEEPSTREAM ON JETSON

1. Download the NVIDIA SDK Manager to install JetPack 4.4 Developer Preview (DP) and DeepStream SDK.
2. Select all the JetPack 4.4 components and DeepStreamSDK from the "Additional SDKs" section.

   Refer to the DeepStream Quick Start Guide for installation of updated NVIDIA V4L2 Gstreamer plugin.

Note:

- **NVIDIA Container Runtime" package shall be installed using JetPack 4.4 DP and is a pre-requisite for all DeepStream L4T docker containers.**
- **It is recommended to use SD card images for Jetson Nano instead of flashing through SDK Manager. The minimum recommended size for SD cards is 32 GB**

3.4 WORKAROUND FOR REDUCED PERFORMANCE ON NVIDIA 430+ DRIVER VERSION

To solve the T4 performance issue you will need to install 418 driver for nvcuvid decoder. CUDA 10.2 and CUDA 10.1 must coexist.

Steps to install Nvidia 418 driver on Ubuntu 18.04 machine:

1. Disable persistent mode

```
$sudo nvidia-smi -pm 0
```

2. kill X from different TTY (e.g., use Ctrl + Alt + F1) or remote shell (ssh)

```
$sudo service gdm (or gdm3 or lightdm)stop
```

3. Uninstall current driver using purge command

```
$sudo apt-get --purge remove "*nvidia*"
```

4. Install CUDA 10.1 and CUDA 10.2:

   Download CUDA 10.1 `.deb` or `.run` file from

   https://developer.nvidia.com/cuda-10.1-download-archive-update2?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804

   Download CUDA 10.2 .deb or .run file from

   https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804

   CUDA 10.1 would be used by nvcuvid decoder.

5. Install 418 driver

   Download from the following location:
   https://www.nvidia.in/Download/driverResults.aspx/158107/en-in

   ```
   $sudo ./NVIDIA-Linux-x86_64-418.126.02.run
   ```

6. Install TensorRT 7.0+ Debian package. Download the package from https://developer.nvidia.com/nvidia-tensorrt-7x-download
7. Install following packages for CUDA 10.2

   ```
   $sudo apt install cuda-samples-10-2 cuda-nvrtc-10-2 cuda-compat-10-2
   ```

8. Append path `/usr/local/cuda-10.2/compat/` in `/etc/ld.so.conf.d/cuda-10-2.conf` file
9. Run `$sudo ldconfig`
10. Run following commands to check CUDA 10.2 installation on 418 driver:

    ```
    cd /usr/local/cuda-10.2/samples/1_Utilities/deviceQuery
    sudo make
    ./deviceQuery
    ```

    `deviceQuery` must run successfully.

11. Install DeepStream 5.0 Package.

12. Run the DeepStream Application and observe improved performance.

```
deepstream-app -c /opt/nvidia/deepstream/deepstream-
5.0/samples/configs/deepstream-app/source30_1080p_dec_infer-
resnet_tiled_display_int8.txt
```

Note: **The steps above are required before you use the DeepStream 5.0 dockers.**

3.5 TRITON INFERENCE SERVER IN DEEPSTREAM

Triton inference server on dGPU is supported only via docker container `deepstream:5.0-dp-20.04-triton` for x86.

Refer to the *DeepStream 5.0 Plugin Manual* for more details about Triton inference server.

Triton Inference Sever Supports following frameworks:

Framework	Tesla	Jetson	Notes / Limitations
TensorRT	Yes	Yes	Supports TensorRT plan or engine file (.plan)
TensorFlow	Yes	Yes	Supports TensorRT optimization Supported model formats: *GraphDef* or *SavedModel* Other TF formats such as checkpoint variables or estimators not directly supported
ONNX	Yes	No	Supports TensorRT optimization
PyTorch	Yes	No	PyTorch model must be traced with an example input and saved as a TorchScript Module (.pt)
Caffe2	No	No	Caffe2 Netdef models not supported with DeepStream - Triton

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.

- TensorRT Optimization in Triton Inference Server for ONNX and Tensorflow: https://docs.nvidia.com/deeplearning/sdk/triton-inference-server-guide/docs/optimization.html#framework-specific-optimization

- TensorFlow with TensorRT: https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html
- Tensorflow Saved Model: https://www.tensorflow.org/guide/saved_model#the_savedmodel_format_on_disk

Notice

Trademarks