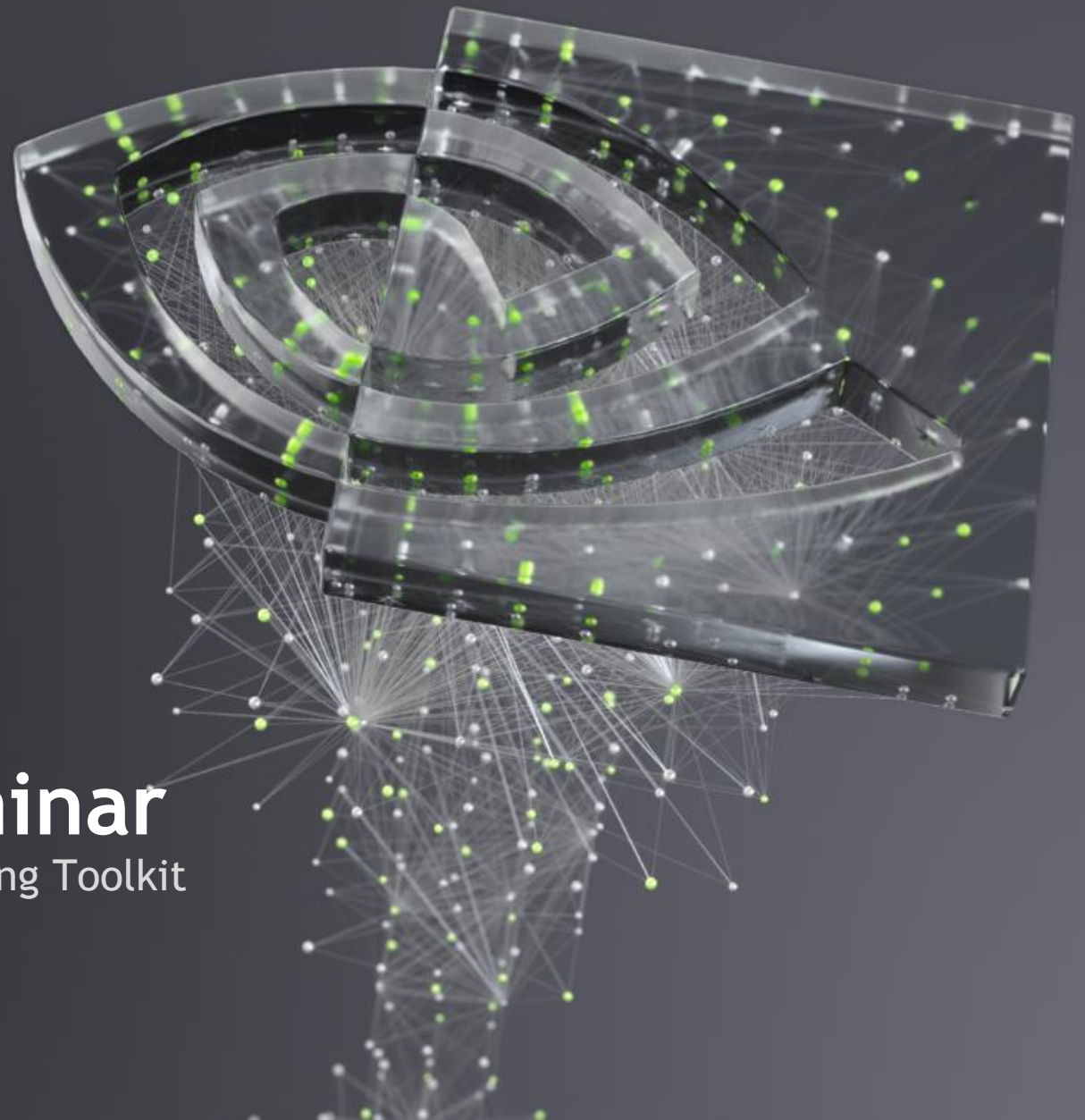


NVIDIA Hands-on Seminar

Deepstream SDK, TensorRT, Transfer Learning Toolkit

Jonghwan Lee, Deep Learning Data Scientist, NVIDIA

AGENDA

NVIDIA Inference Technology

TensorRT/ Deepstream SDK/ Transfer Learning Toolkit

NVIDIA TensorRT

High-performance deep learning inference platform

NVIDIA Deepstream SDK

AI-powered Intelligent Video Analytics framework

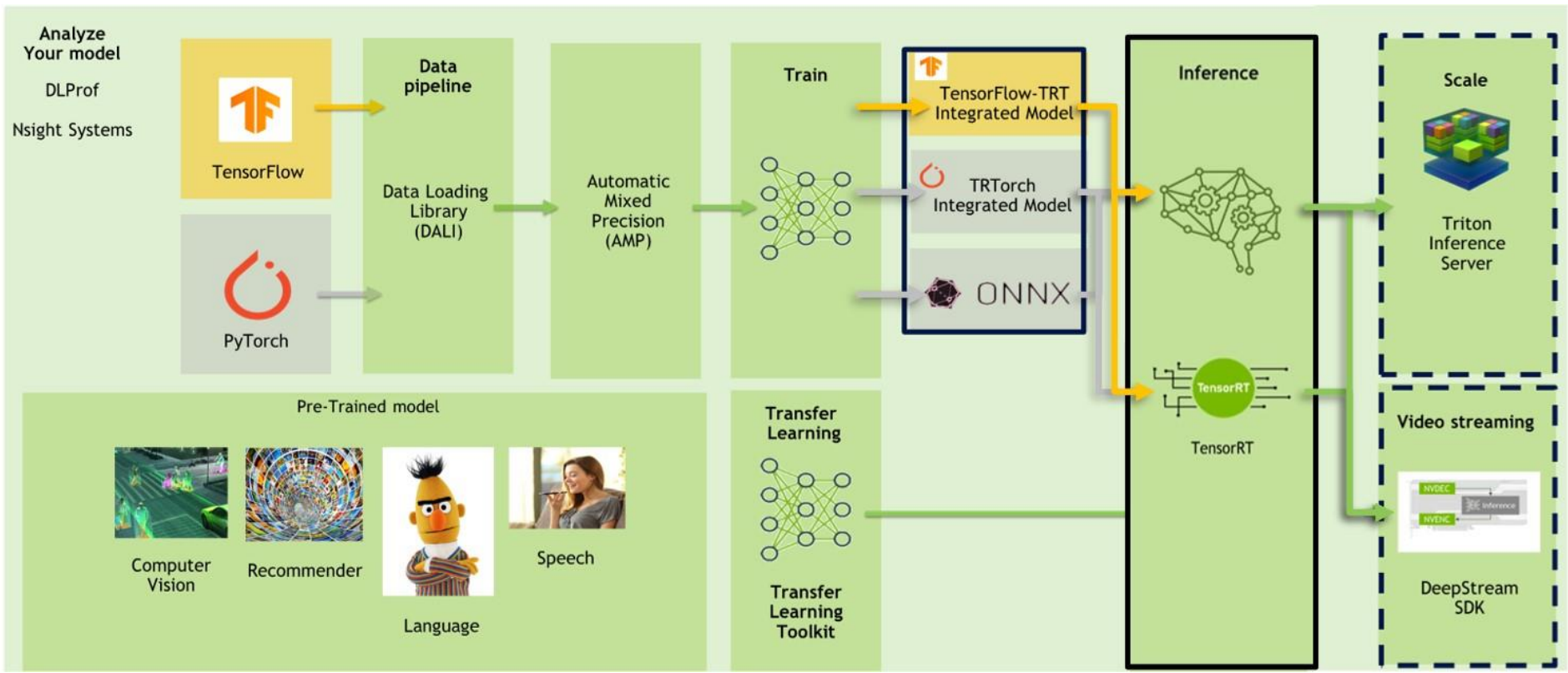
Hands-on with Jetson NX

Mainly Deepstream SDK, TensorRT and TLT


NVIDIA INFERENCE
TECHNOLOGY

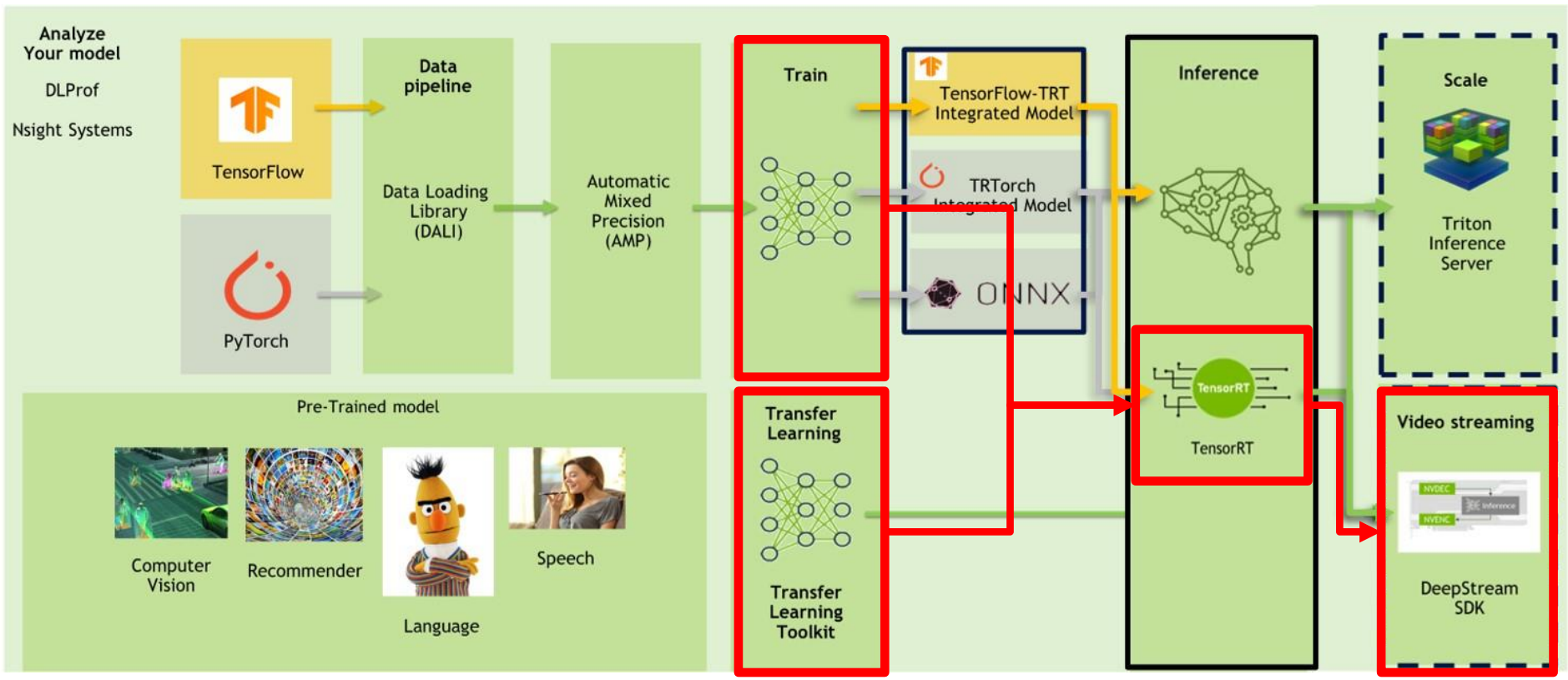
NVIDIA INFERENCE SOLUTIONS

Easily deploy AI, Tools and Libraries to accelerate your AI model and Service



NVIDIA INFERENCE SOLUTIONS

Easily deploy AI, Tools and Libraries to accelerate your AI model and Service

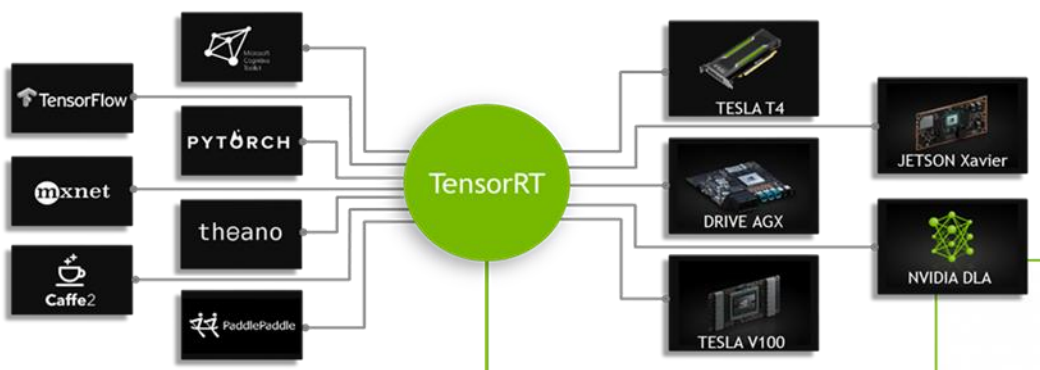


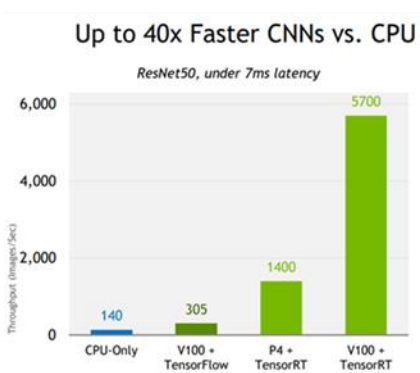
Today's focus!!!

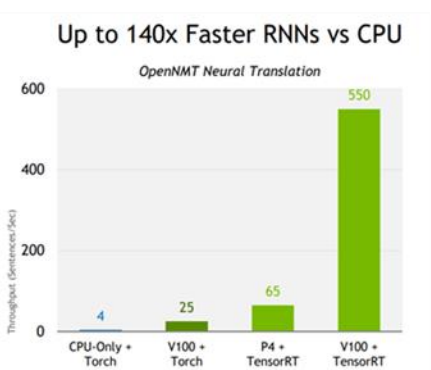
NVIDIA

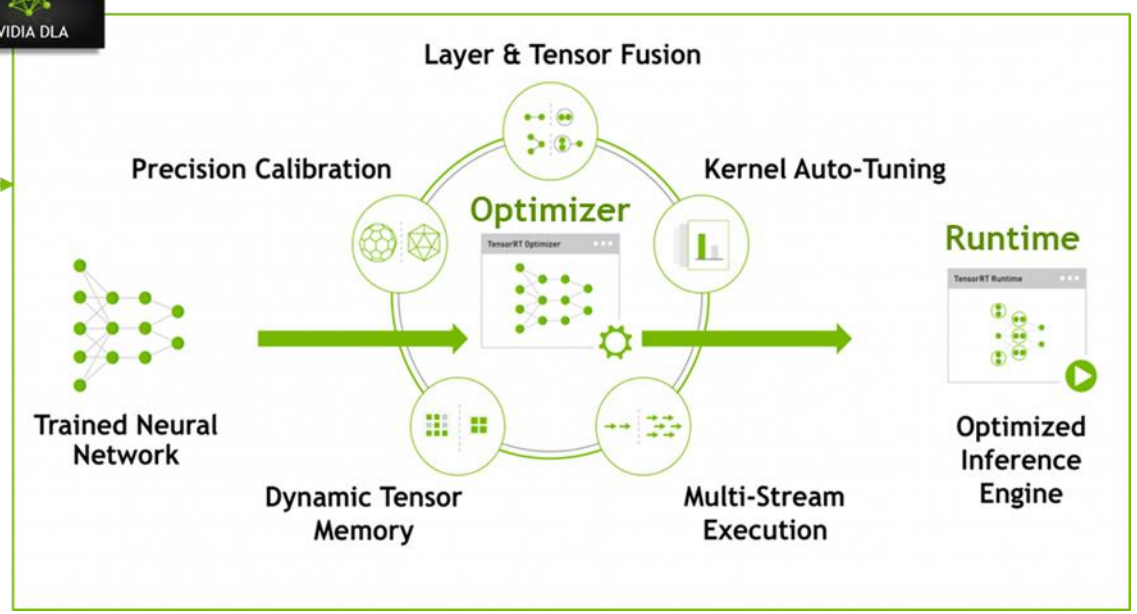
NVIDIA TensorRT

From Every Framework, Optimized For each Target platform





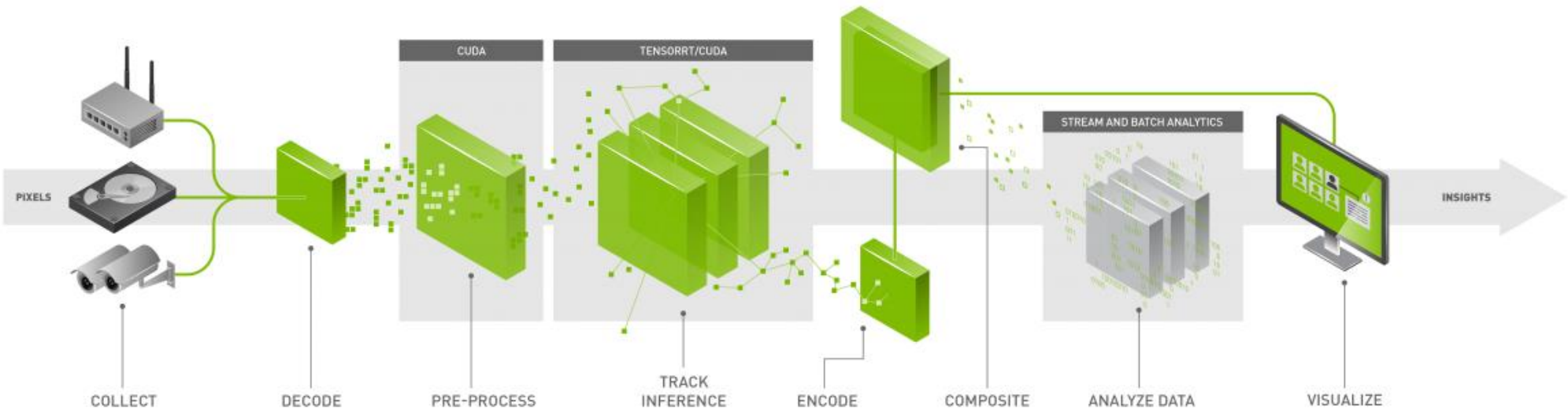




https://developer.nvidia.com/tensorrt

NVIDIA Deepstream SDK

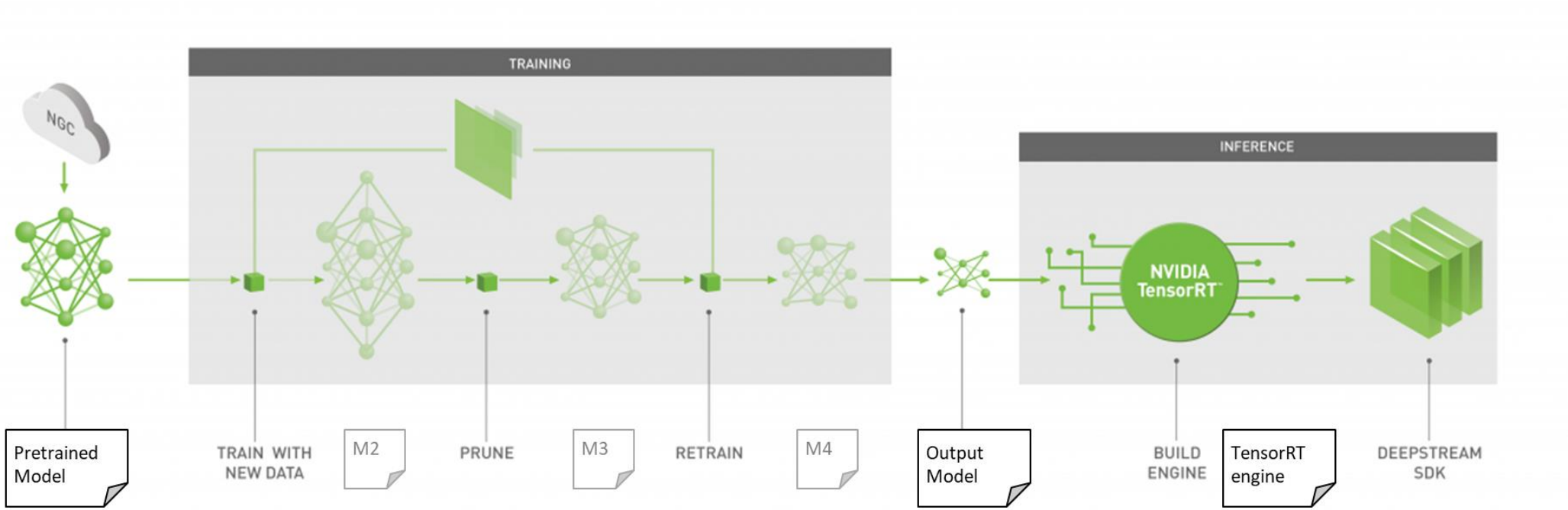
Analyze Data from Cameras, Sensors and IoT Gateways in Real-Time



https://developer.nvidia.com/deepstream-sdk

NVIDIA Transfer Learning Toolkit

Accelerates Throughput on Leading Industry Platforms



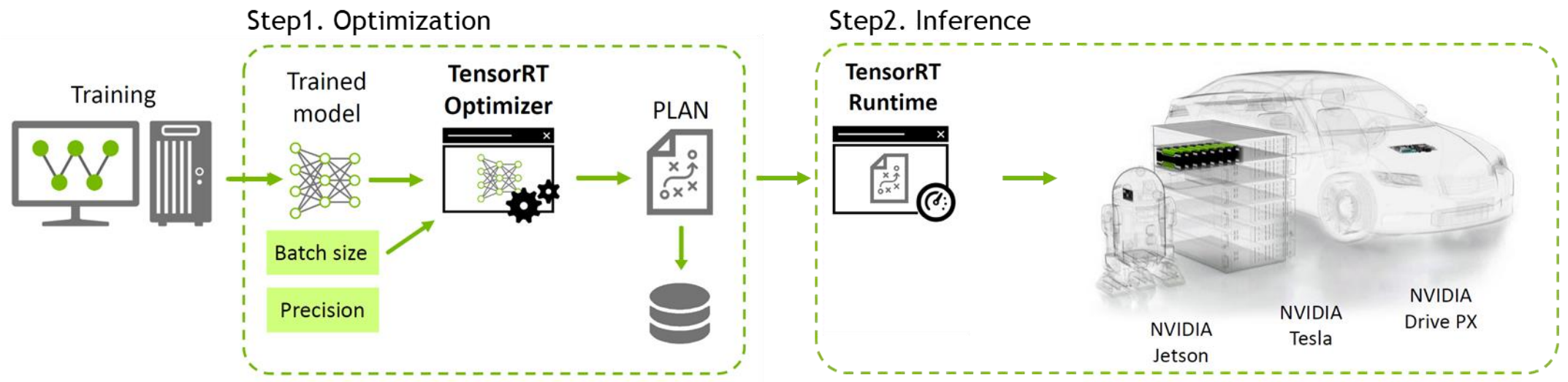
1. Download docker container -> 2. Pull Model -> 3. Train with your data -> 4. Prune -> 5. Retrain -> 6. Export

https://developer.nvidia.com/transfer-learning-toolkit


NVIDIA TensorRT

TENSORRT WORKFLOW

How to accelerate DNN inference with TensorRT



Import model → Optimize model → Serialize plan

Deserialize plan → Build Runtime engine → Deploy Runtime

NVIDIA TENSOR CORE

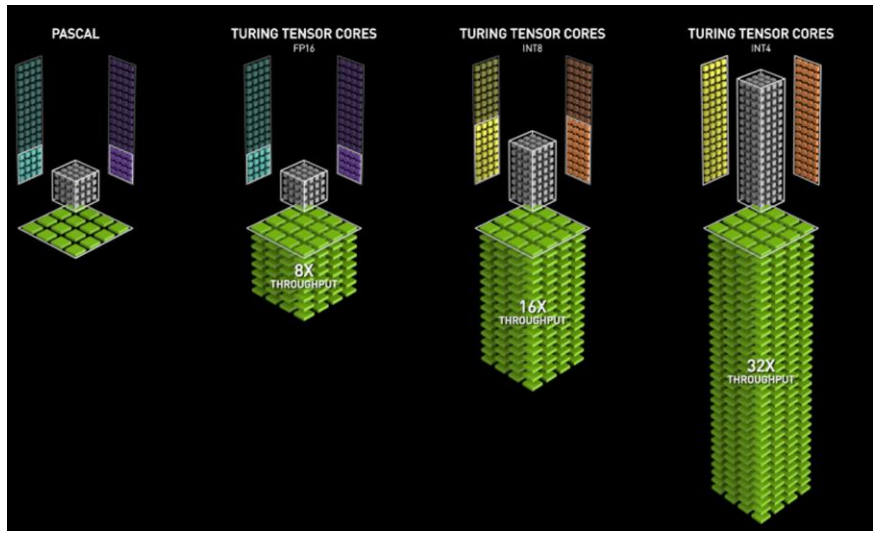
Mixed precision Matrix-Multiply-and-Add

Ampere has new 3rd TensorCore

- Available in Volta and Turing architecture GPUs
- 125 Tflops in FP16 vs 15.7 Tflops in FP32 (8x speed-up)
- Optimized 4x4x4 dot operation (GEMM)

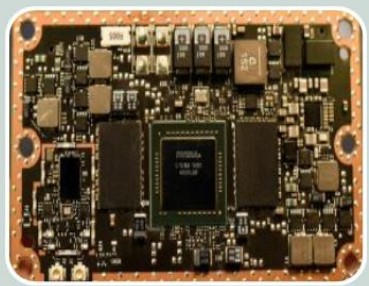
$$D = \underbrace{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}}_{\text{FP16}} \underbrace{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}}_{\text{FP16}} + \underbrace{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}_{\text{FP16 or FP32}}$$

FP16 or FP32

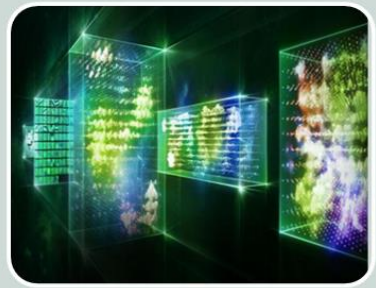


NVDLA

NVIDIA Deep Learning Accelerator



Developed as part of Xavier – NVIDIA's SOC for autonomous driving applications



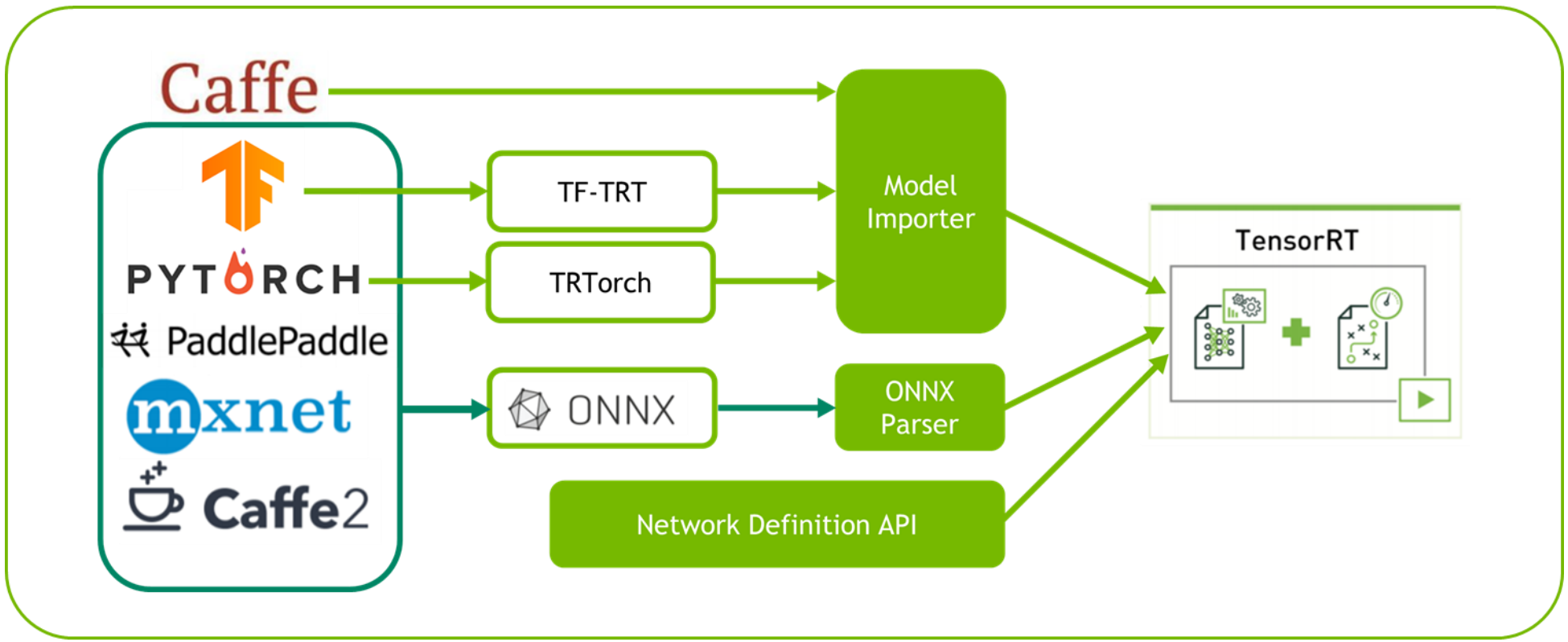
Optimized for Convolutional Neural Networks (CNNs), computer vision

Open source architecture and RTL release. NVDLA Compiler is now open source in the latest NVDLA v1.2.0-OC release

Complete solution with Verilog and C-model, Linux drivers, test benches, kernel and user-mode software and virtual platform

NVIDIA TENSORRT

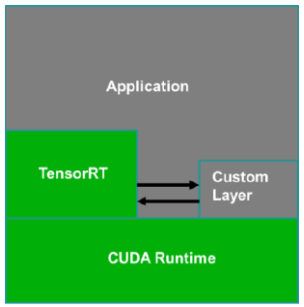
TensorRT Model import flow

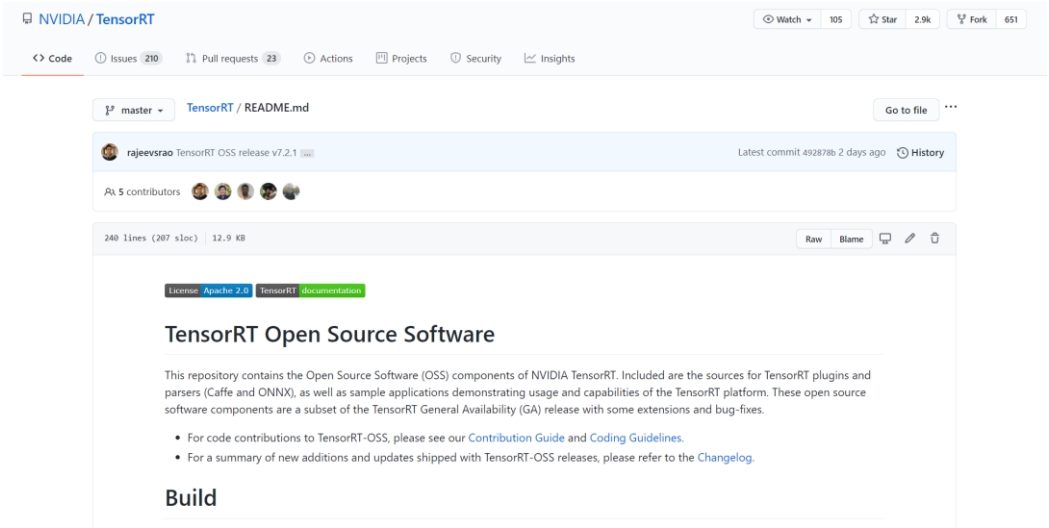


NVIDIA TENSORRT OSS

Supports custom plug-ins by CUDA

- Allow users to express and provide implementations of novel layers
  - TensorRT provides APIs and dedicated implementations for most common layers
  - Use the custom layer API for infrequent or more innovative layers or your own confident implementation
  - Register custom implementations via a callback mechanism
  - Can be used in conjunction with reduced precision optimization



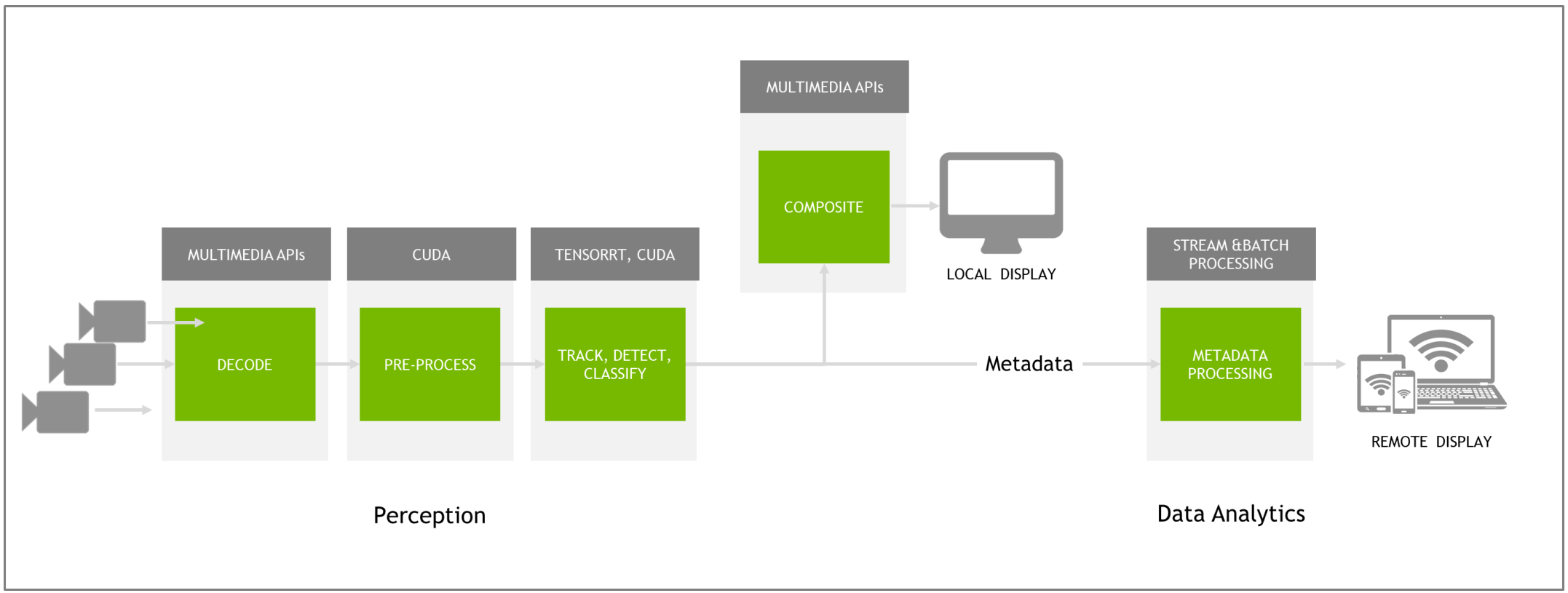


https://github.com/NVIDIA/TensorRT


NVIDIA Deepstream SDK

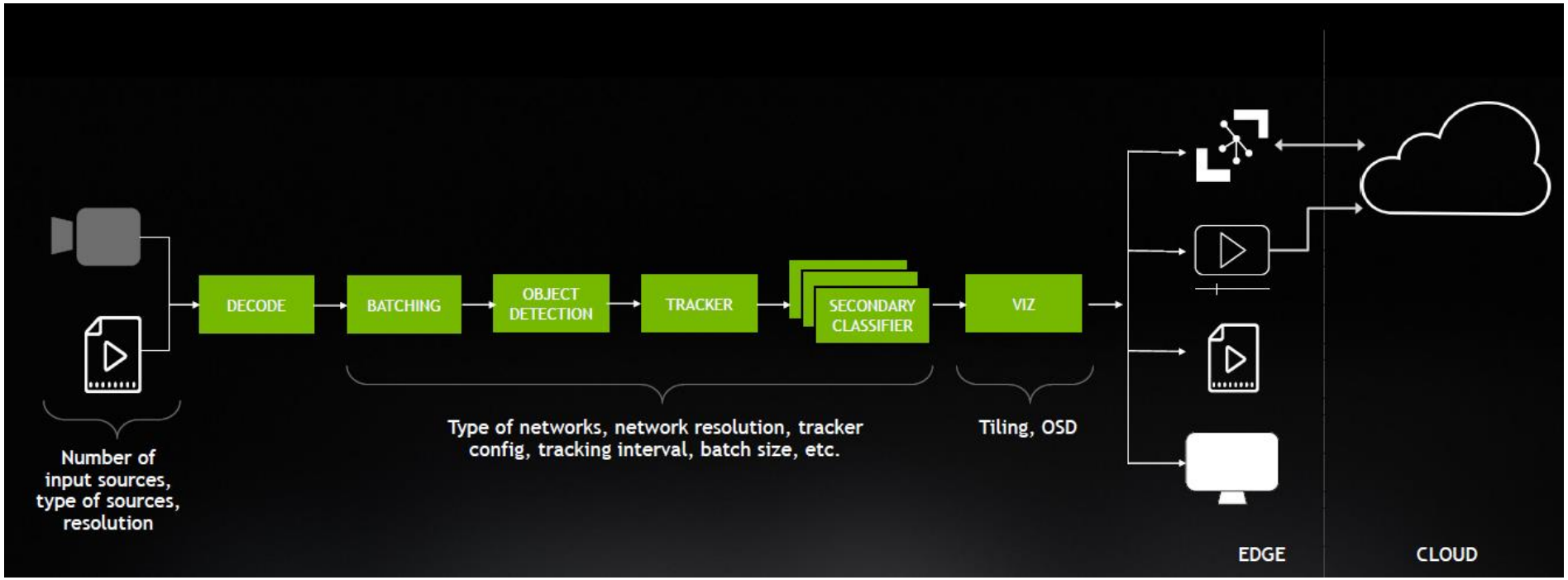
GENERAL INTELLIGENCE VIDEO ANALYSIS

Framework for analyzing video



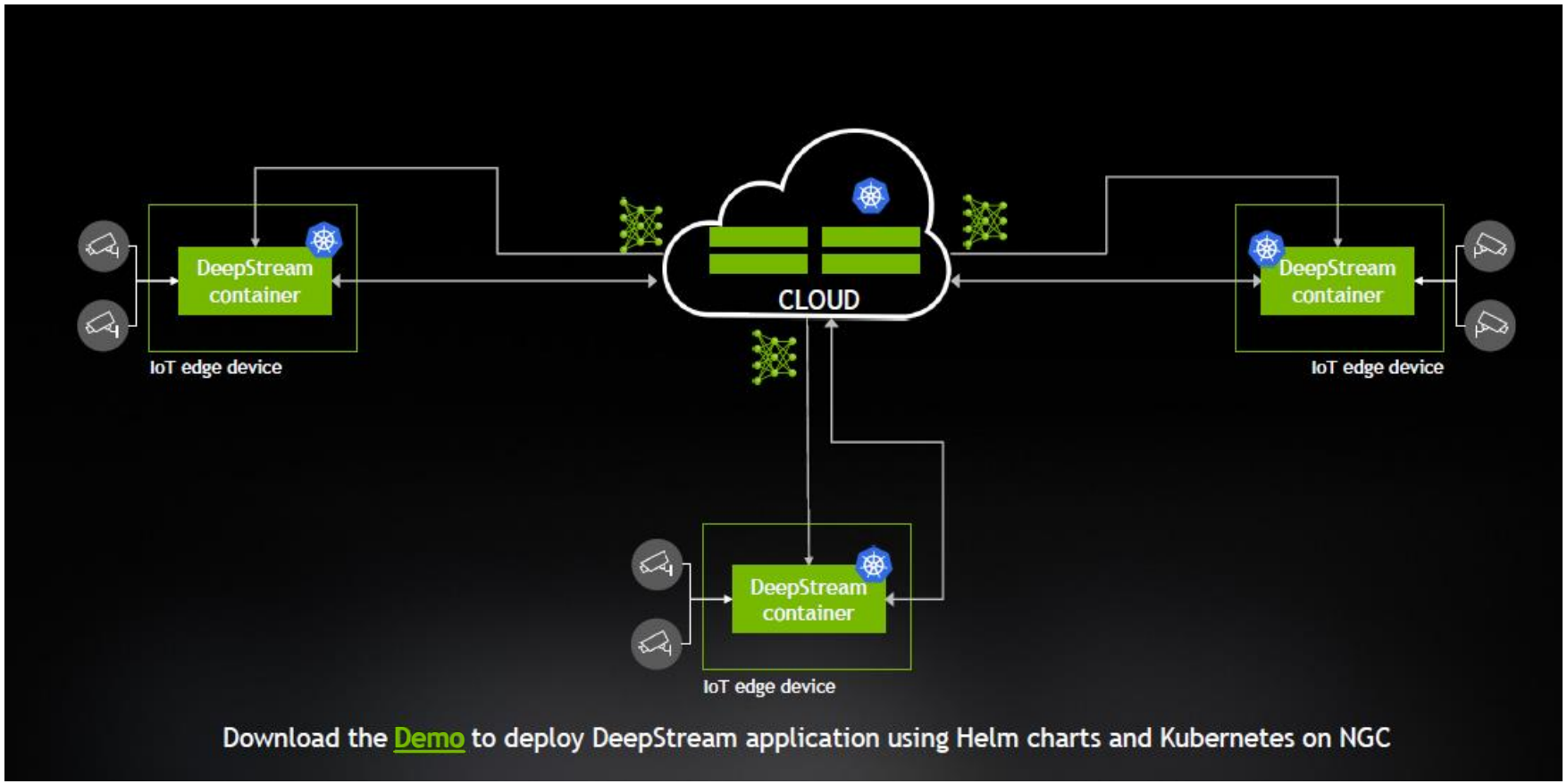
DEEPSTREAM SDK

End-to-End Deepstream Application – deepstream-test5



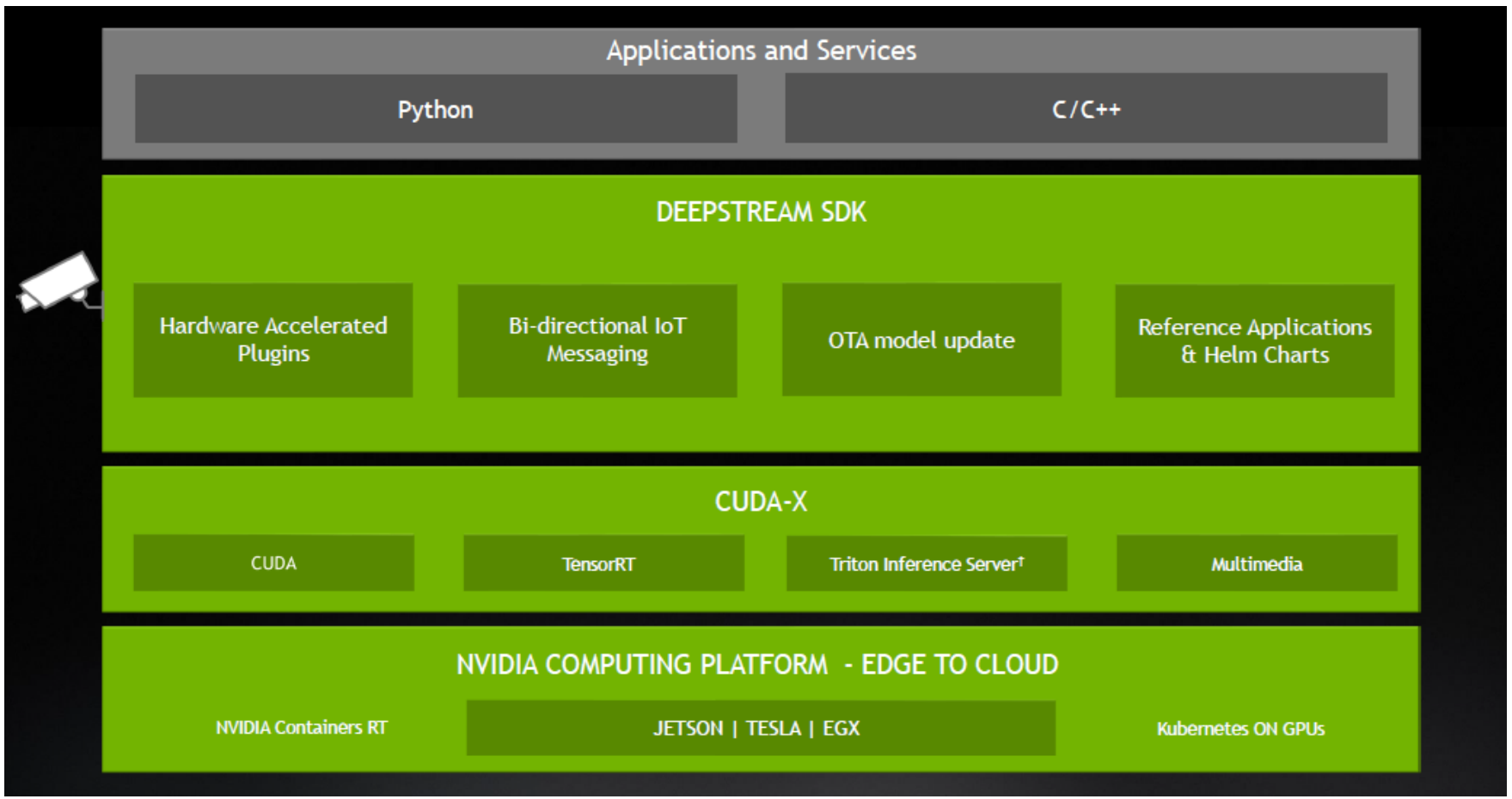
DEEPSTREAM SDK

At scale deployment



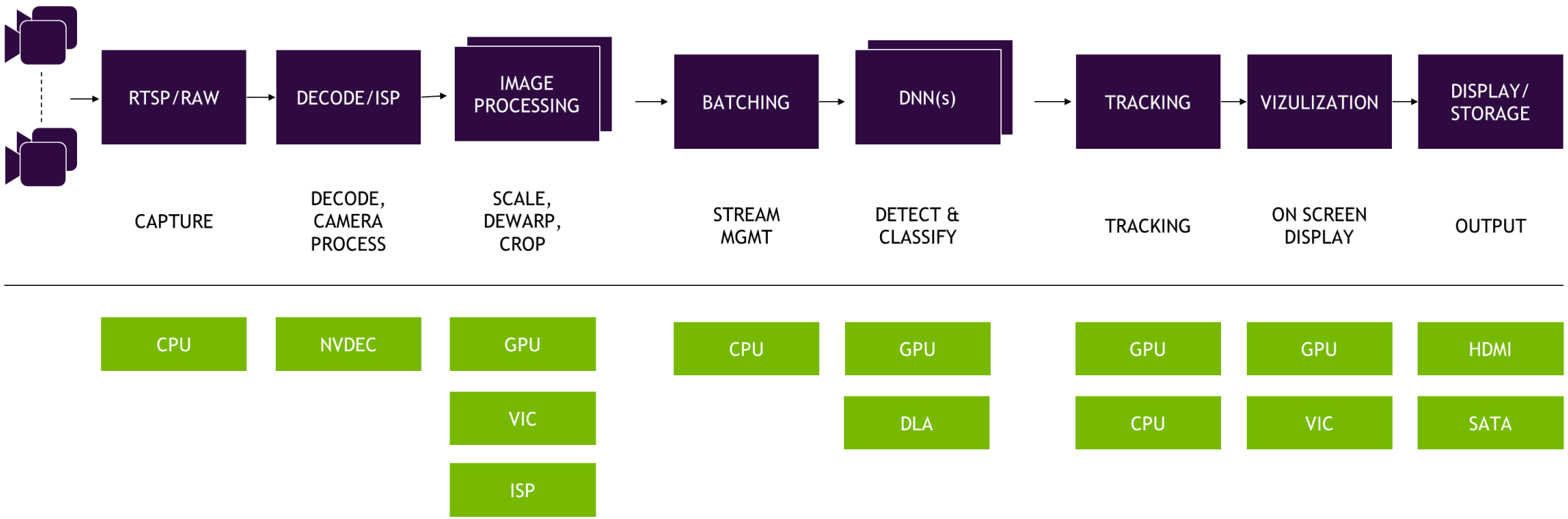
WHAT IS DEEPSTREAM?

AI-powered Intelligent Video Analytics framework



DEEPSTREAM GRAPH ARCHITECTURE

AI-powered Intelligent Video Analytics framework



CONCEPTS OF DEEPSTREAM SDK

Gstreamer Foundations

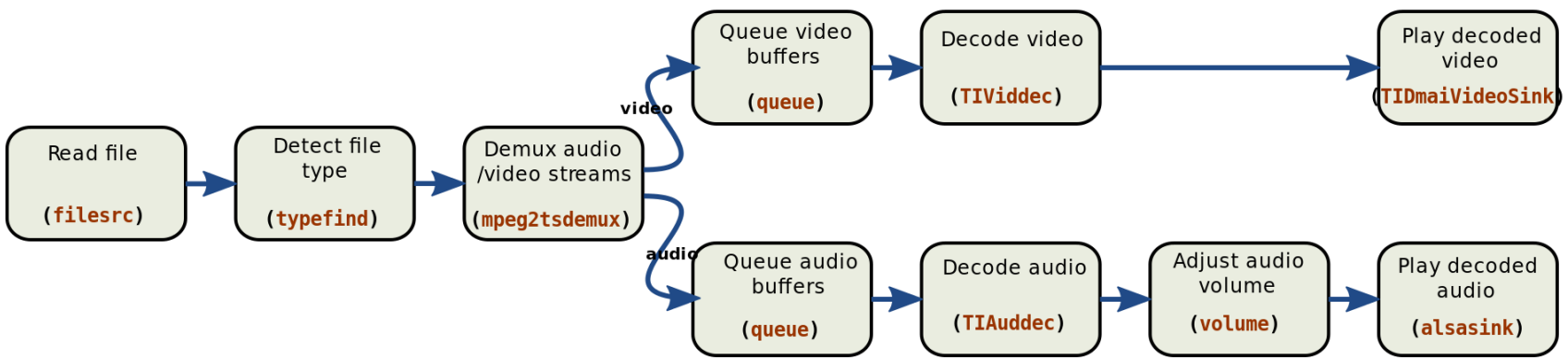
The DeepStream SDK is based on the open source GStreamer multimedia framework. There are a few key concepts in GStreamer that we need to touch on before getting started. These include Elements, Pads, Buffers, and Caps. We will be describing them at a high level, but encourage those who are interested in the details to read the GStreamer documentation to learn more.

CONCEPTS OF DEEPSTREAM SDK

Example of Gstreamer pipeline

The goals of GStreamer are to seprate the application (e.g. Video player, Video editor, etc.) from the streaming media complexity (e.g. hardware acceleration, remoteness)
GStreamer – streaming media
D-Bus – inter process communication
Use **gst-launch** command to create the GStreamer pipeline

```
gst-rtsp-server - v4l2src ! video/x-raw,width=1280,height=720 ! omxh264enc ! video/x-h264,profile=baseline ! h264parse config-interval=1 ! rtph264pay na
```



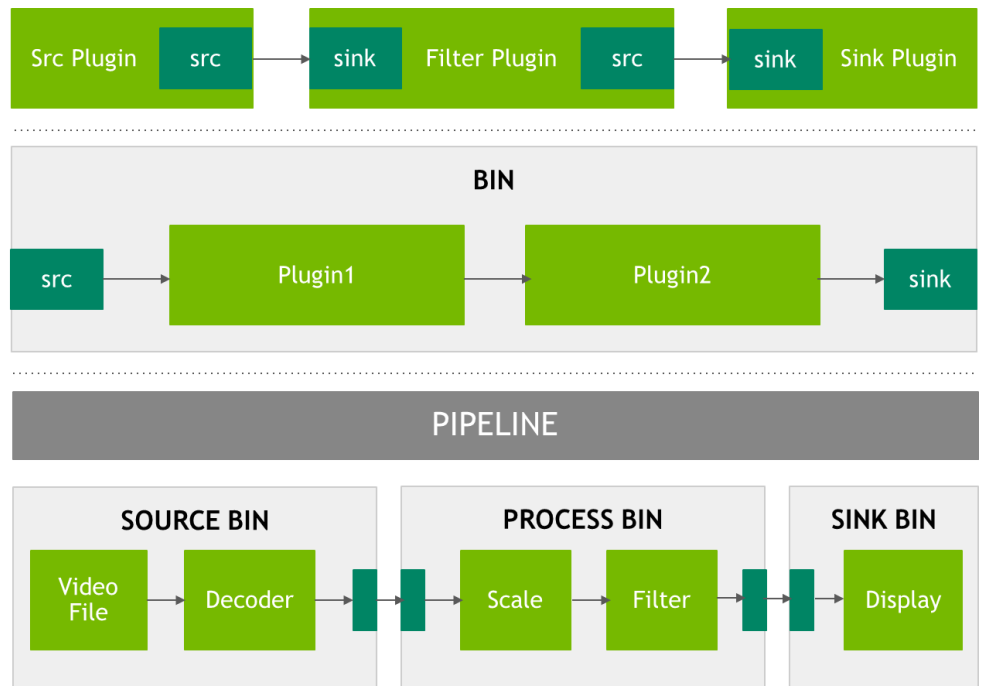

```
gst-launch filesrc location="video.ts" ! typefind ! mpeg2tsdemux name=demux \
      demux. ! 'video/x-h264' ! queue ! TIViddec ! TIDmaiVideoSink \
      demux. ! 'audio/mpeg'   ! queue ! TIAuddec ! volume volume=5 ! alsasink
```

https://en.wikipedia.org/wiki/GStreamer

CONCEPTS OF DEEPSTREAM SDK

Gstreamer pipeline

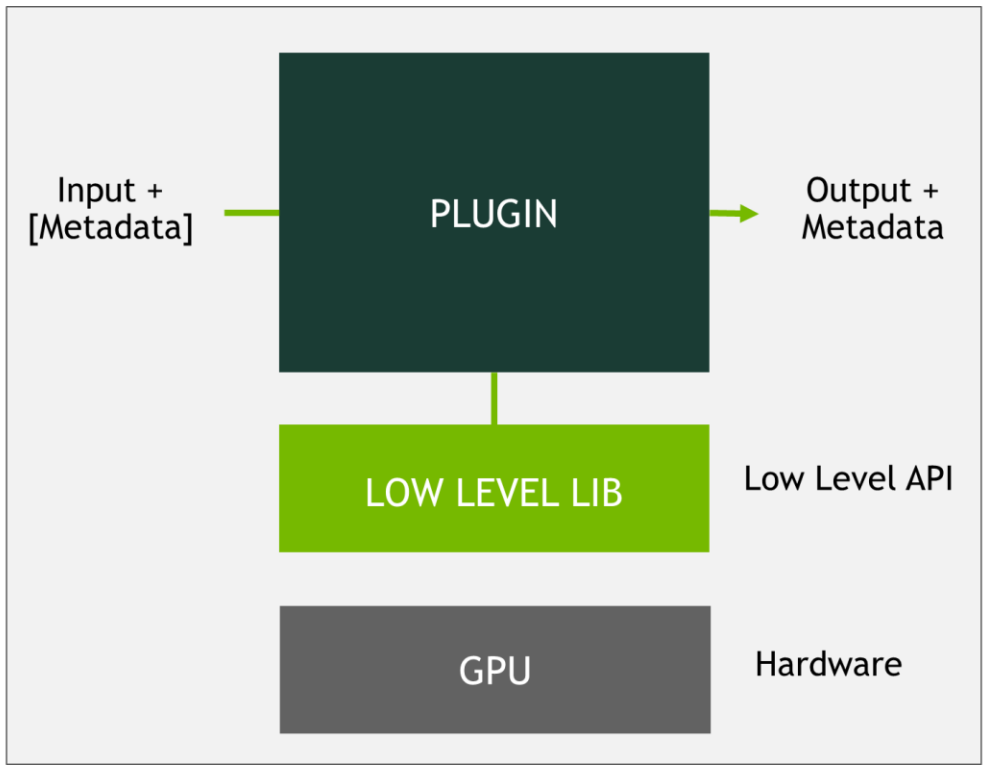
Level	Component	Function
1	PLUGINS	Basic building block connected through PADs
2	BINS	A container for a collection of plugins
3	PIPELINE	Top level bin providing a bus and managing the synchronization



CONCEPTS OF DEEPSTREAM SDK

Deepstream Building block

- A plugin model-based pipeline architecture
- Graph-based pipeline interface to allow high-level component interconnect
- Heterogenous processing on GPU and CPU
- Hides parallelization and synchronization under the hood
- Inherently multi-threaded



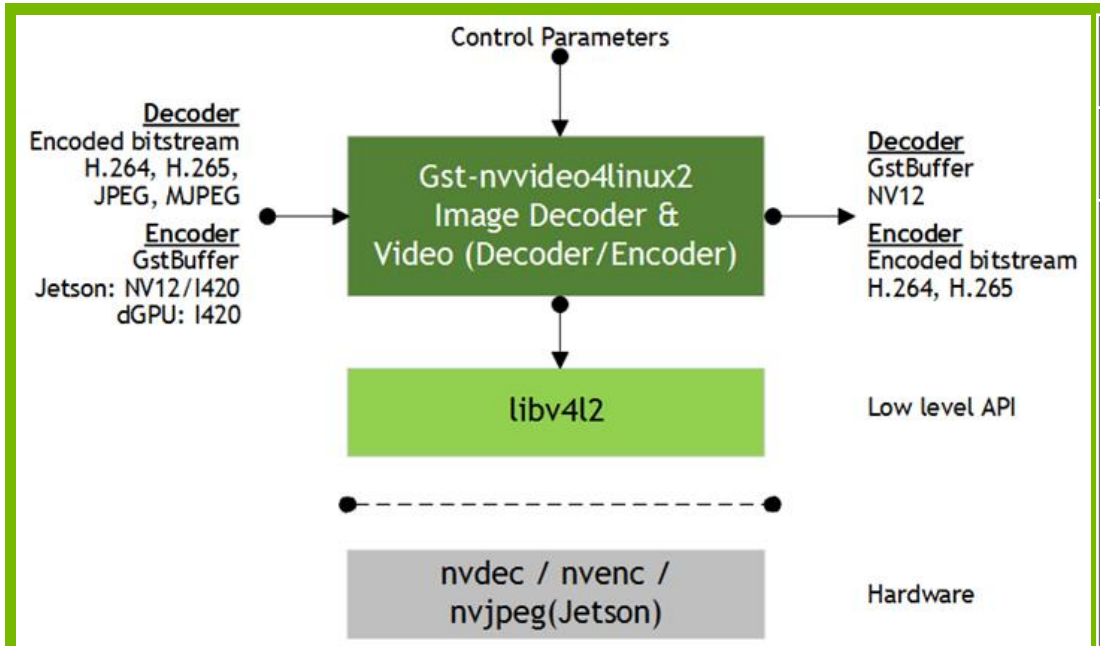
DEEPSTREAM SDK

Deepstream accelerated plugin

Plugin Name	Functionality
Gst-nvinfer	DL inference for detection, classification and segmentation with TRT engine
Gst-nvinferserver	DL inference for detection, classification and segmentation with any other native framework
Gst-nvtracker	Reference object trackers; KLT, IOU, NvDCF
Gst-nvstreammux	Stream aggregation, multiplexing and batching
Gst-nvstreamdemux	Demux batched frames into individual buffers
Gst-nvmultistreamtiler	Composites a 2D tile from batched buffers
Gst-nvdsosd	Draw boxes and text overlay
Gst-nvvideoconvert	Scaling, format conversion and rotation
Gst-nvdewarper	Dewarping for fish-eye degree cameras
Gst-nvof	Fast optical flow calculation on special HW accelerator for it
Gst-nvofvisual	Visualize input optical flow vector on frames
Gst-nvsegvisual	Visualize segmentation results on frames
Gst-nvvideo4linux2	Hardware accelerated decode and encode(NVDEC / NVENC)
Gst-nvjpegdec	Decoding JPEG images
Gst-nvmsgconv	Metadata generation
Gst-nvmsgbroker	Messaging to cloud
Gst-nvdsanalytics	ROI filtering, Overcrowding Detection, Direction Detection and Line crossing

DEEPSTREAM SDK

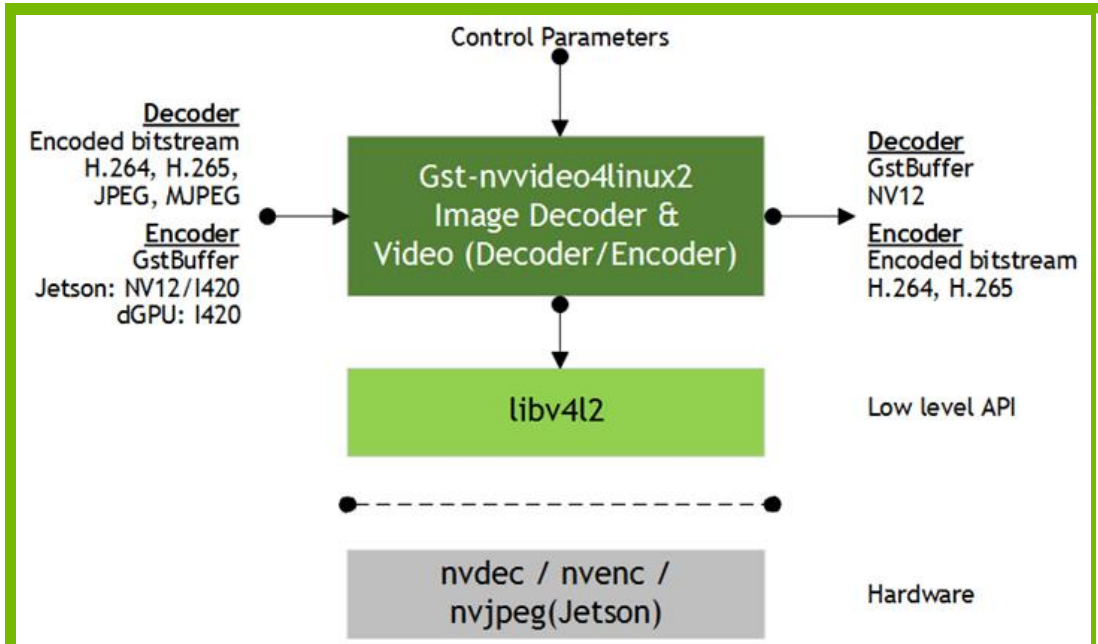
Deepstream accelerated plugin – Gst-nvvideo4linux2 (Decode)



Input	H.264, H.265, JPEG, MJPEG
Output	NV12, GstBuffer
Control Parameter	gpu-id num-extra-surfaces skip-frames cudadec-memtype drop-frame-interval

DEEPSTREAM SDK

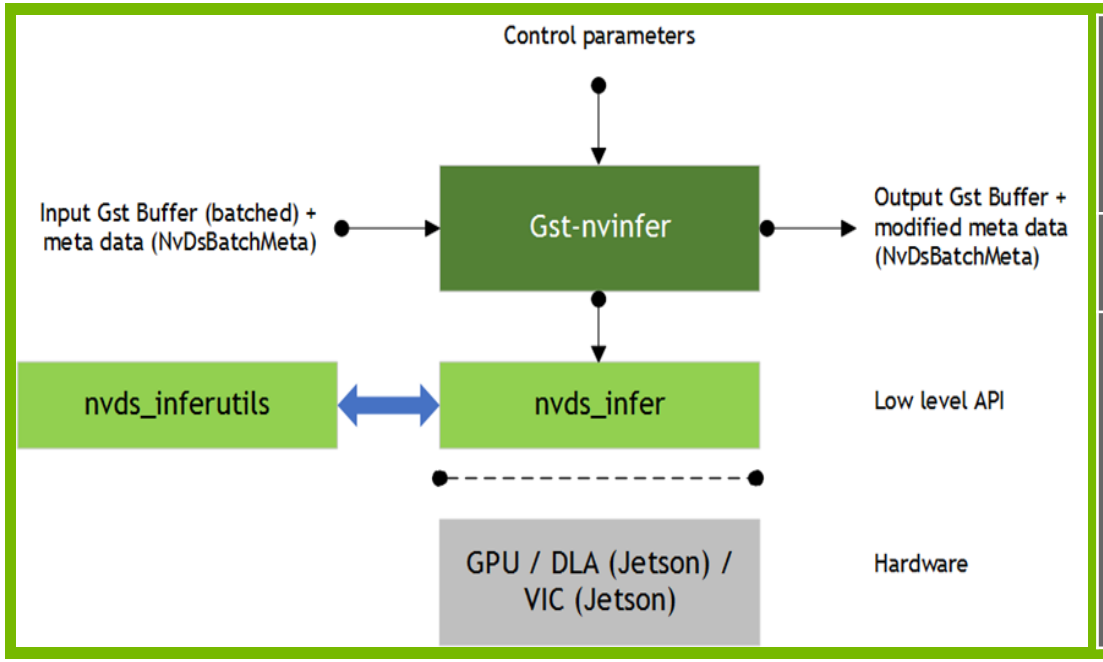
Deepstream accelerated plugin – Gst-nvvideo4linux2 (Encode)



Input	NV12, I420
Output	H.264, H265
Control Parameter	gpu-id profile bitrate control-rate Iframeinterval

DEEPSTREAM SDK

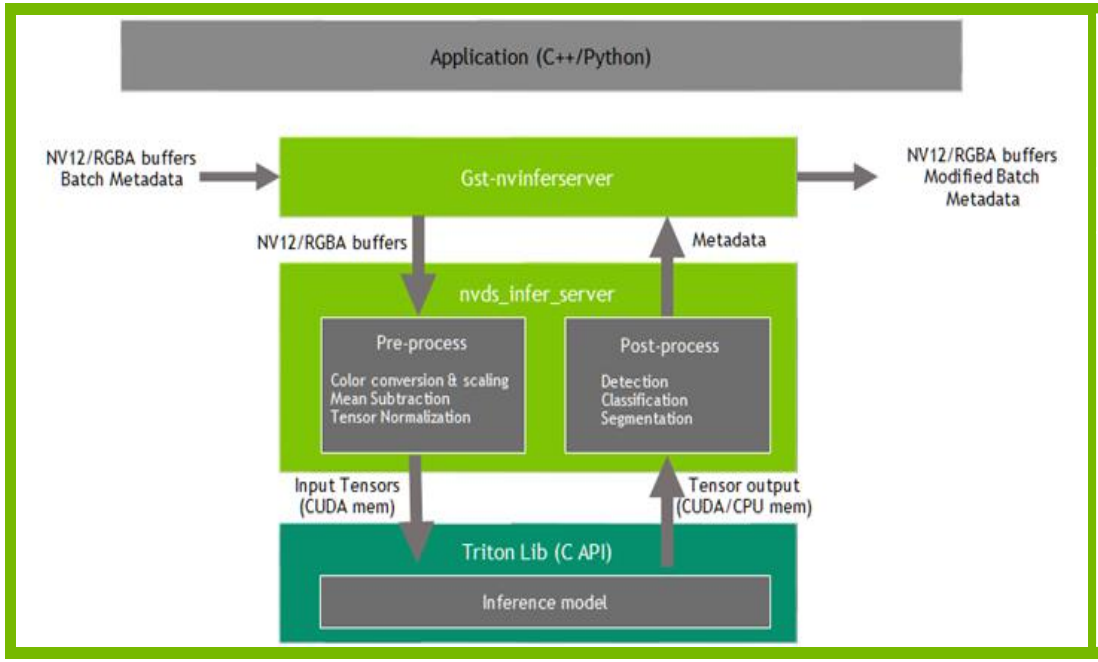
Deepstream accelerated plugin – Gst-nvinfer



Input	Gst Buffer, NvDsBatchMeta Caffe Model / Prototxt ONNX, UFF file TLT Encoded Model / Key, TRT engine
Output	Gst Buffer, NvDsObjectMeta
Control Parameter	Batch size Inference interval Attach inference tensor outputs as buffer metadata Attach instance mask output as in object metadata

DEEPSTREAM SDK

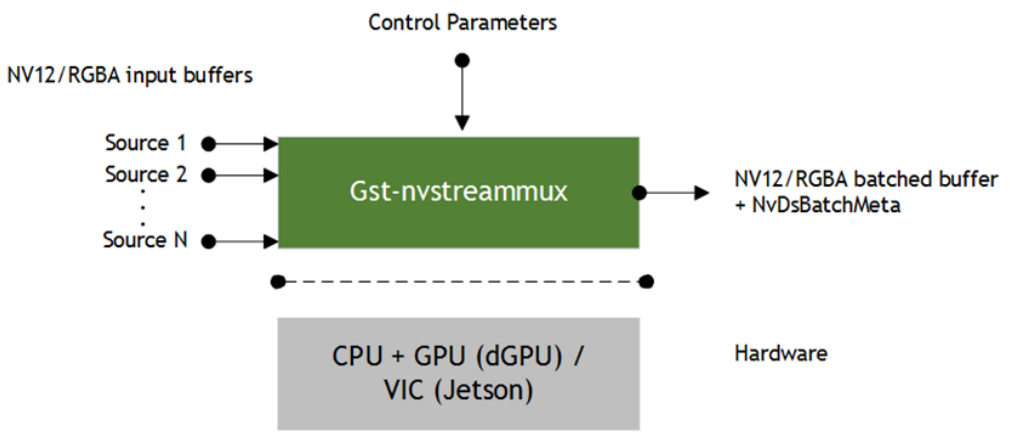
Deepstream accelerated plugin – Gst-nvinferserver



Input	Gst Buffer, NvDsBatchMeta Model repository directory path (model_repo.root)
Output	Gst Buffer, NvDsObjectMeta
Control Parameter	Batch size Process mode Unique id Inference on GIE id and operate on class ids [secondary mode only] Inference interval Raw output generated callback function

DEEPSTREAM SDK

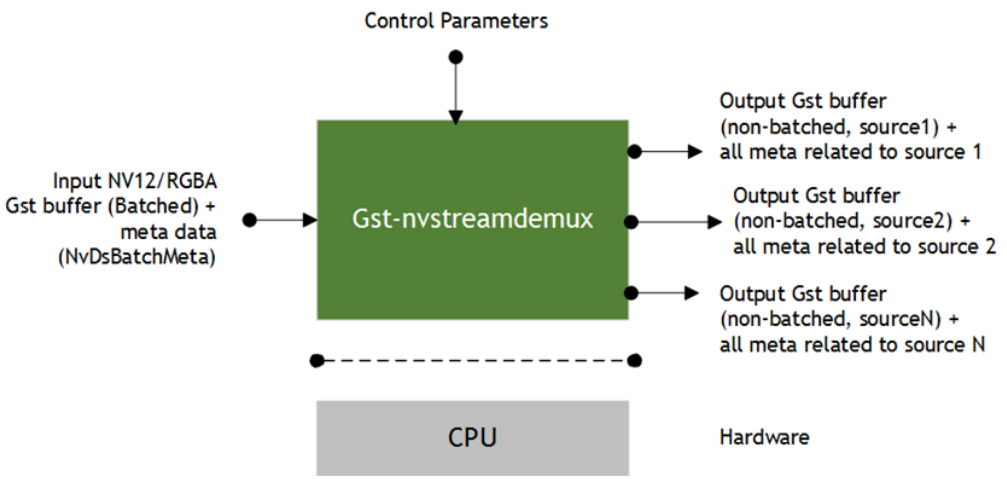
Deepstream accelerated plugin – Gst-nvstreammux



Input	NV12/RGBA buffers
Output	NV12/RGBA batched buffer GstNvBatchMeta
Control Parameter	batch-size, batched-push-timeout Width, height, enable-padding gpu-id, live-source, nvbuf-memory-type num-surfaces-per-frame buffer-pool-size attach-sys-ts

DEEPSTREAM SDK

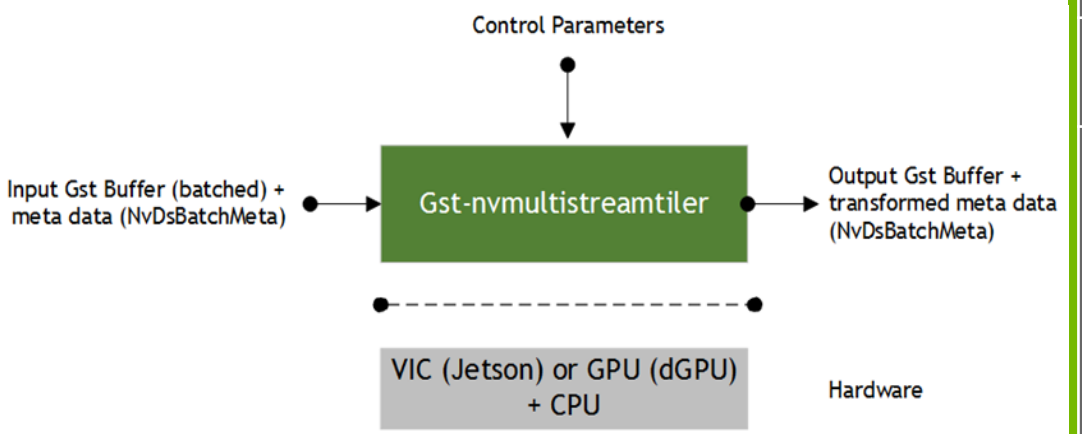
Deepstream accelerated plugin – Gst-nvstreamdemux

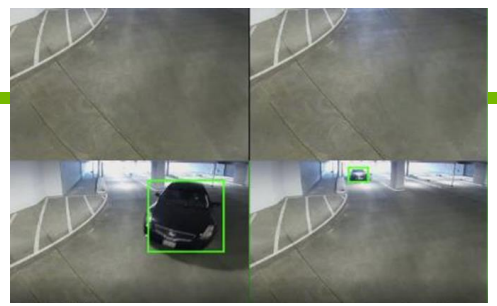


Input	Gst Buffer(batched), NvDsBatchMeta
Output	Gst Buffer (non-batched, single source) Meta related to each Gst Buffer source
Control Parameter	N/A

DEEPSTREAM SDK

Deepstream accelerated plugin – Gst-nvmultistreamtiler

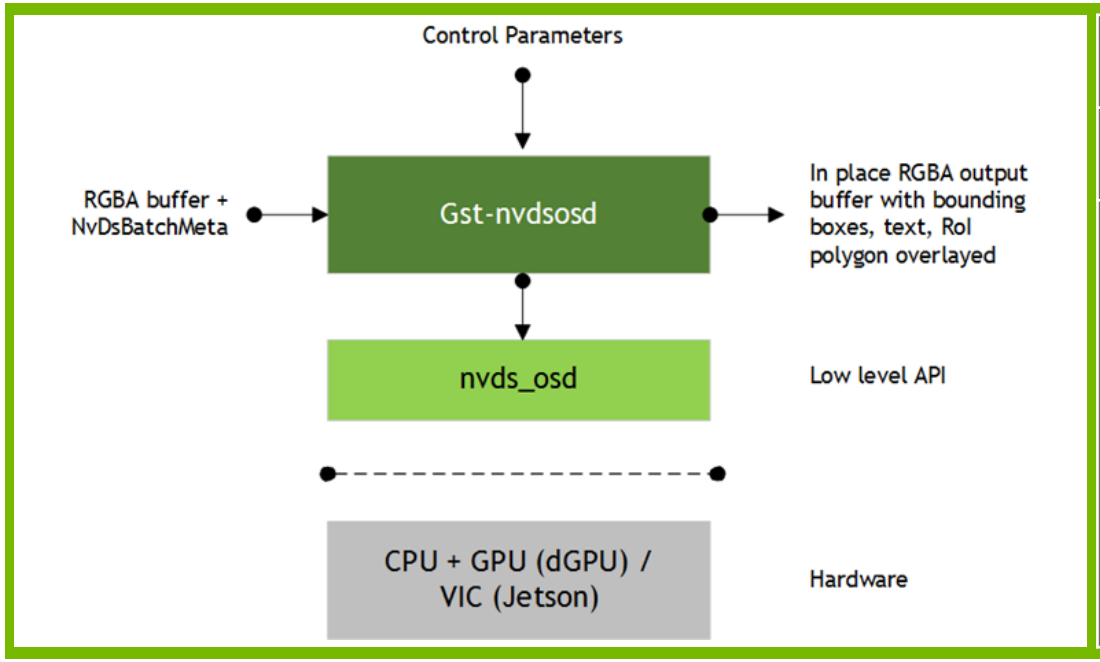




Input	Gst Buffer batched buffer NvDsBatchMeta
Output	Gst Buffer with composited input frames Transformed NvDsBatchMeta
Control Parameter	rows, columns width, height gpu-id (dGPU only) show-source nvbuf-memory-type custom-tile-config

DEEPSTREAM SDK

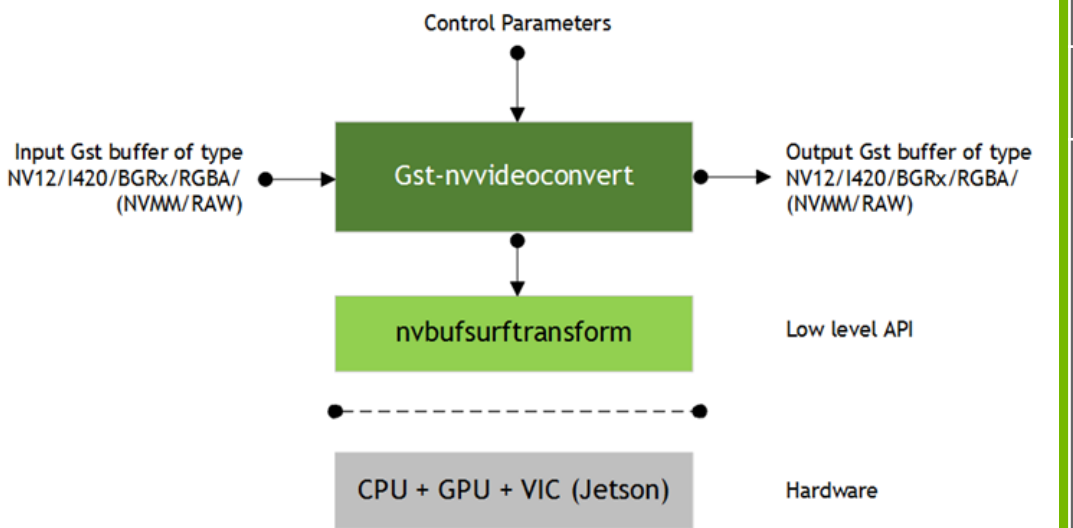
Deepstream accelerated plugin – Gst-nvdsosd



Input	RGBA buffer, NvDsBatchMeta
Output	RGBA buffer modified in place to overlay
Control Parameter	gpu-id, display-clock, display-text clock-font, clock-font-size x-clock-offset, y-clock-offset clock-color, process-mode hw-blend-color-attr, display-bbox, display-mask

DEEPSTREAM SDK

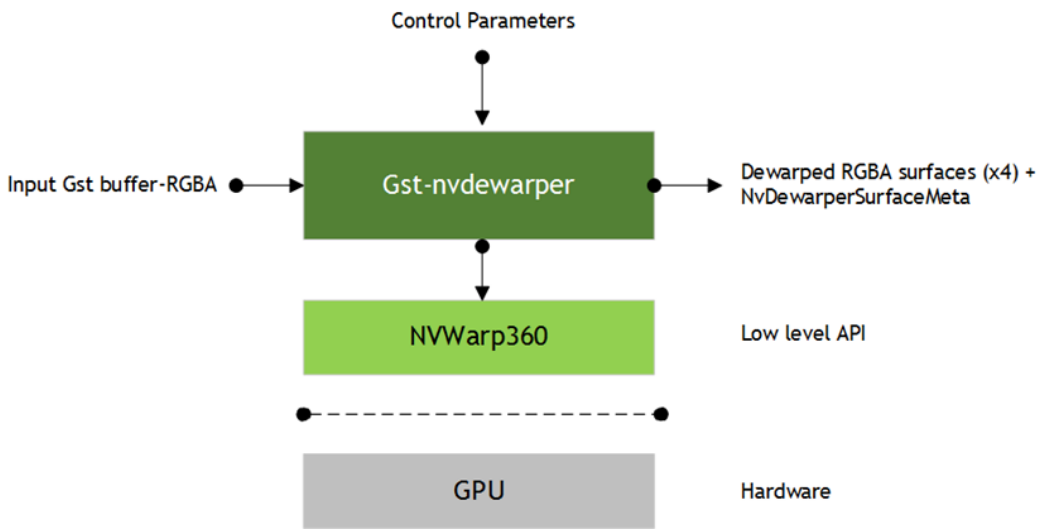
Deepstream accelerated plugin – Gst-nvvideoconvert



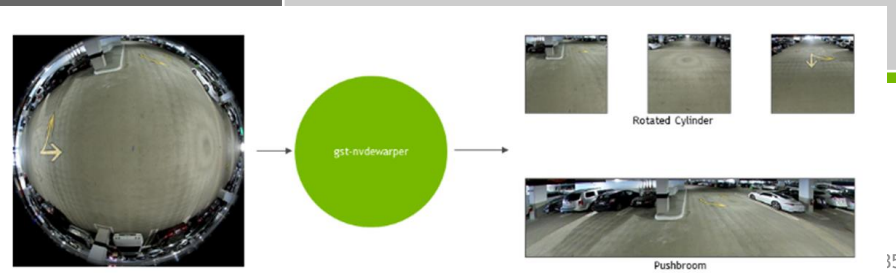
Input	Gst Buffer batched, NvDsBatchMeta
Output	Gst Buffer, NvDsBatchMeta
Control Parameter	gpu-id, nvbuf-memory-type src-crop, dst-crop interpolation-method compute-hw

DEEPSTREAM SDK

Deepstream accelerated plugin – Gst-nvdewarper

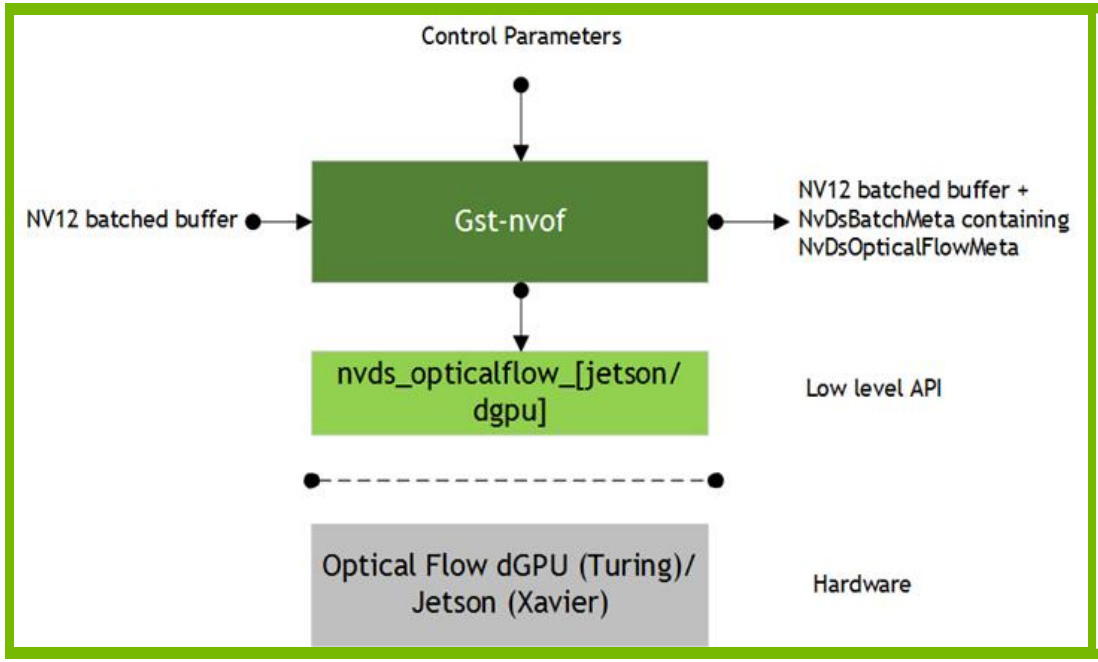


Input	RGBA buffer
Output	Dewarped RGBA surfaces NvDewarperSurfaceMeta
Control Parameter	gpu-id config-file



DEEPSTREAM SDK

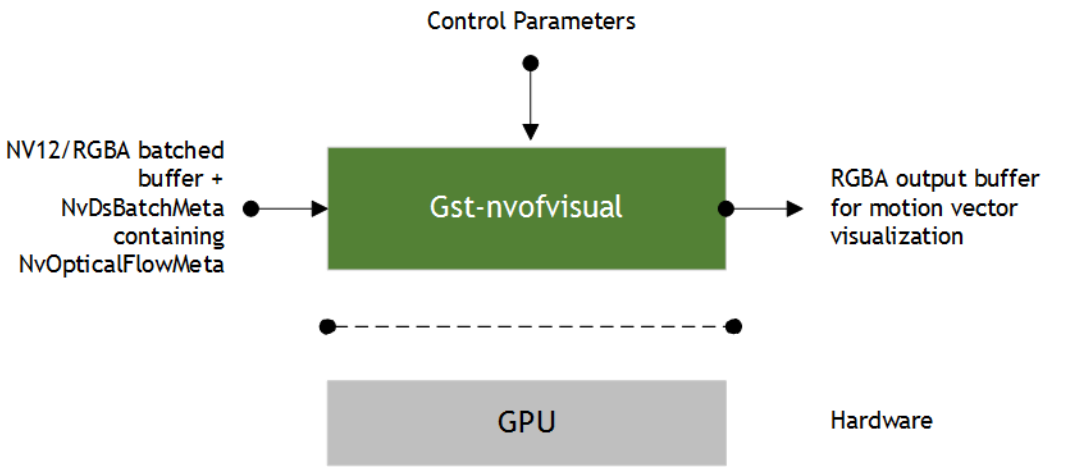
Deepstream accelerated plugin – Gst-nvof



Input	NV12, I420
Output	H.264, H265
Control Parameter	gpu-id profile bitrate control-rate Iframeinterval

DEEPSTREAM SDK

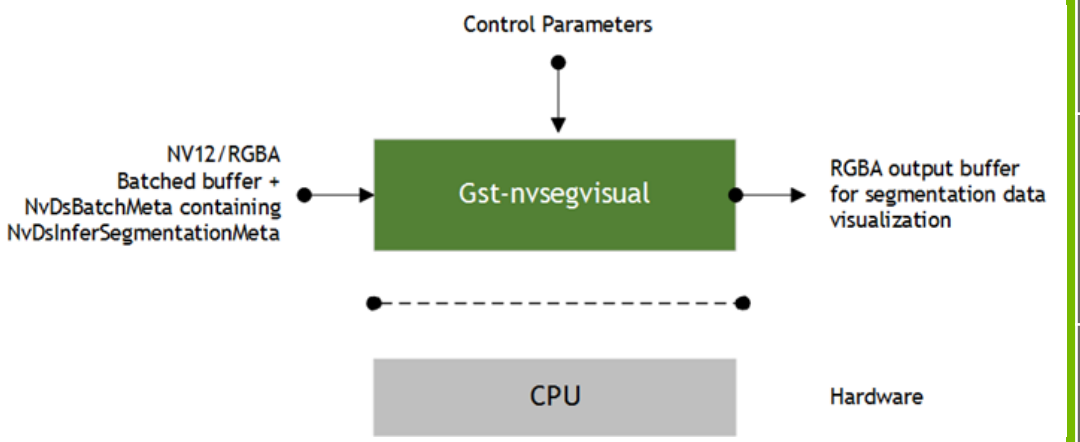
Deepstream accelerated plugin – Gst-nvofvisual



Input	NV12, RGBA (batched) Buffer, NVDsOpticalFlowMeta
Output	RGBA (batched) Buffer
Control Parameter	gpu-id

DEEPSTREAM SDK

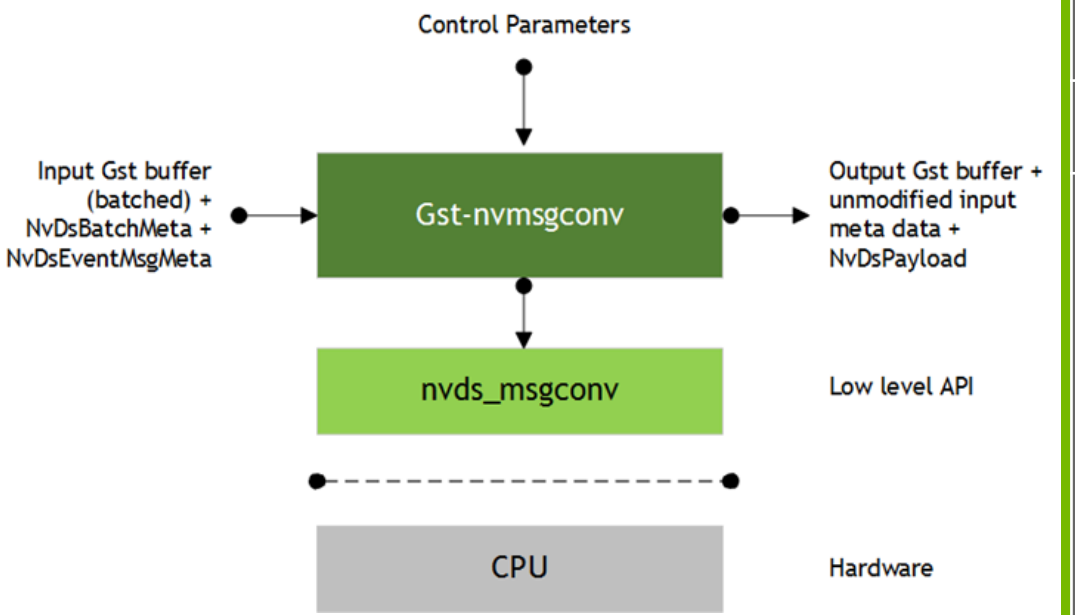
Deepstream accelerated plugin – Gst-nvsegvisual



Input	NV12, RGBA buffer, NvDsInferSegmentationMeta
Output	Segmented RGBA buffer
Control Parameter	gpu-id width height

DEEPSTREAM SDK

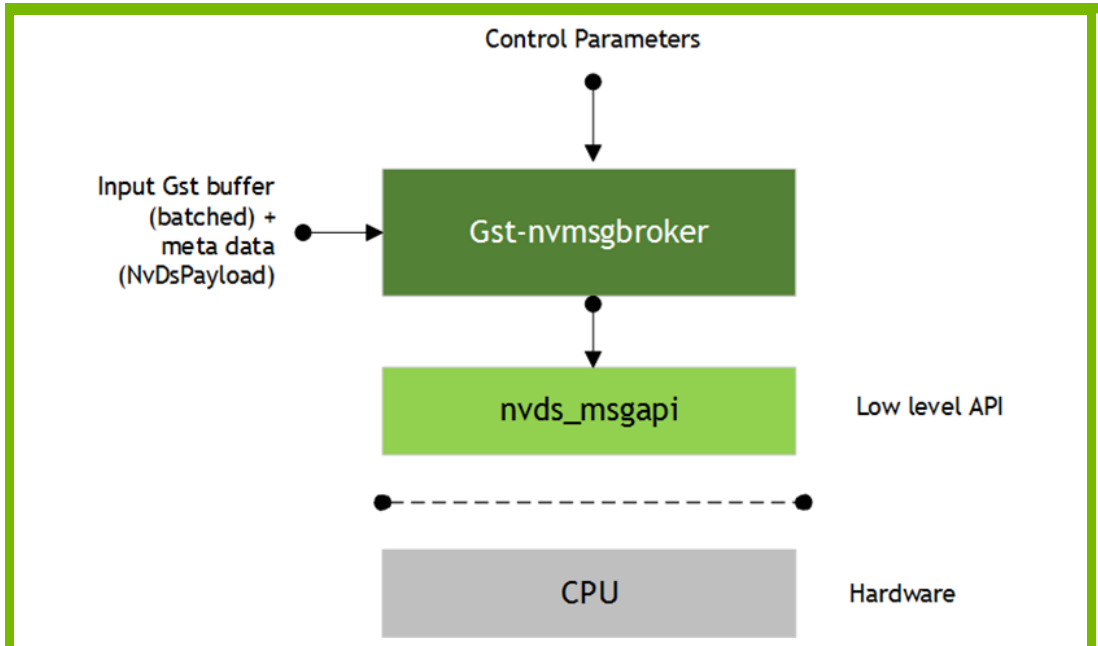
Deepstream accelerated plugin – Gst-nvmsgconv



Input	Gst Buffer, NvDsEventMsgMeta
Output	Gst Buffer, NvDsPayload metadata
Control Parameter	config msg2p-lib payload-type comp-id

DEEPSTREAM SDK

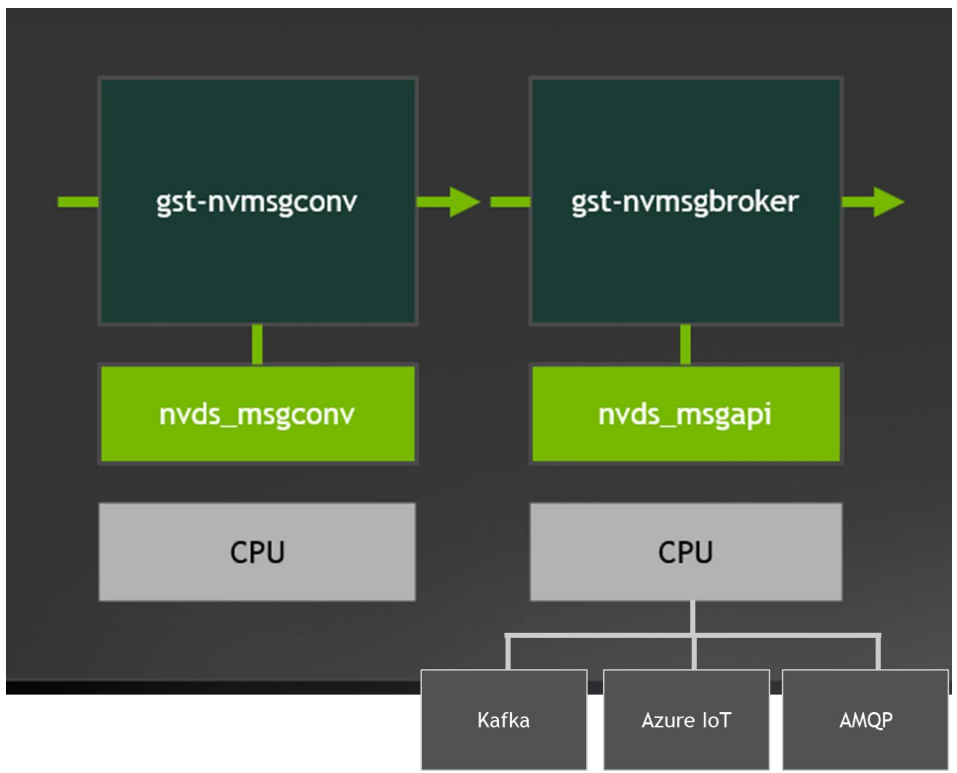
Deepstream accelerated plugin – Gst-nvmsgbroker



Input	Gst Buffer, NvDsPayload
Output	N/A
Control Parameter	Config conn-str proto-lib comp-id Topic

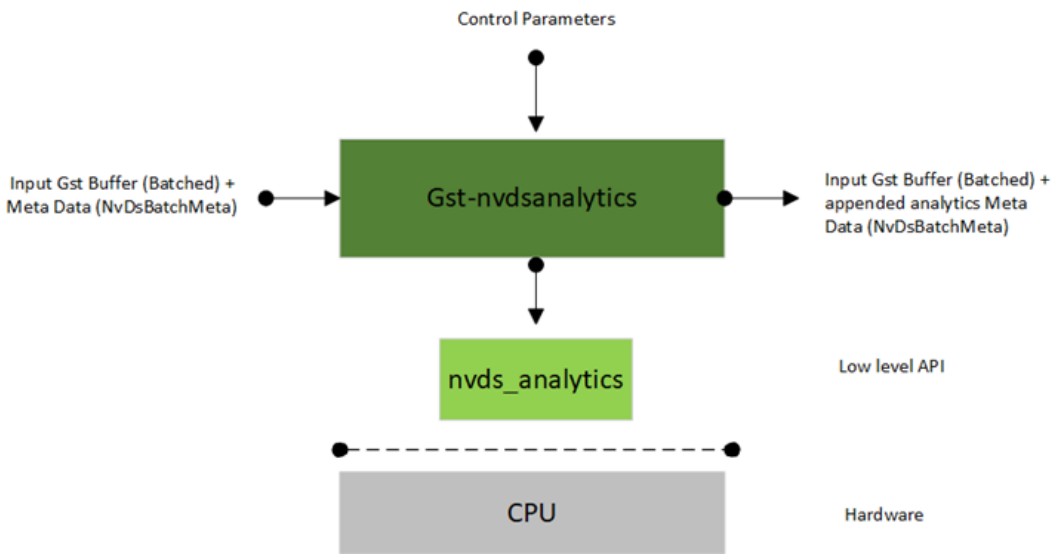
DEEPSTREAM SDK

Deepstream accelerated plugin – Gst-nvmsgconv & Gst-nvmsgbroker



DEEPSTREAM SDK

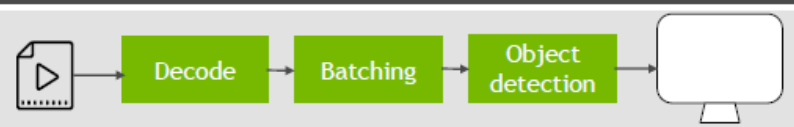
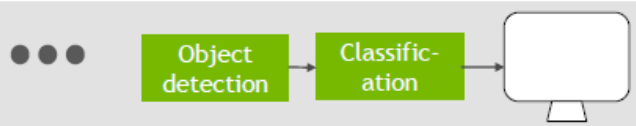
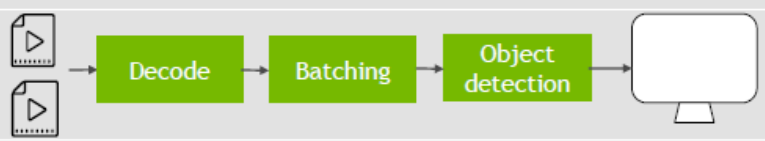
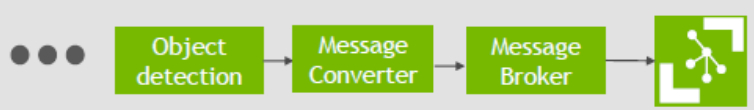
Deepstream accelerated plugin – Gst-nvdsanalytics



Input	NV12, RGBA buffer, NVDsBatchMeta
Output	MV12, RGBA buffer, NvBatchMeta
Control Parameter	enable config-file

DEEPSTREAM SDK

Get started applications (available both C and Python)

Name	Function	
deepstream-test1	DeepStream Hello world. Single video from file to on screen display with bounding box	
deepstream-test2	Builds on test1 and adds secondary object classification on detected objects	
deepstream-test3	Builds on test1 and adds multiple video inputs	
deepstream-test4	Builds on test1 and adds connections to IoT services thru the nvmsgbroker plugin	

Native-C apps: *sources/apps/sample_apps/*
Python apps : *https://github.com/NVIDIA-AI-IOT/deepstream_python_apps*

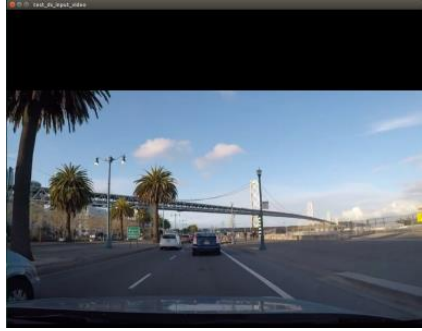

HANDS-ON
FOR DEEPSTREAM

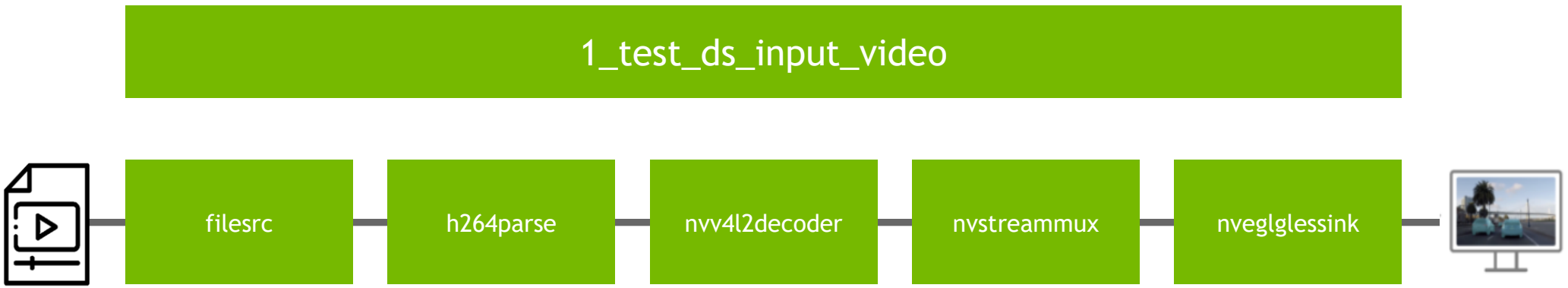
DEEPSTREAM SDK HANDS-ON

Setting up for the Hands-on

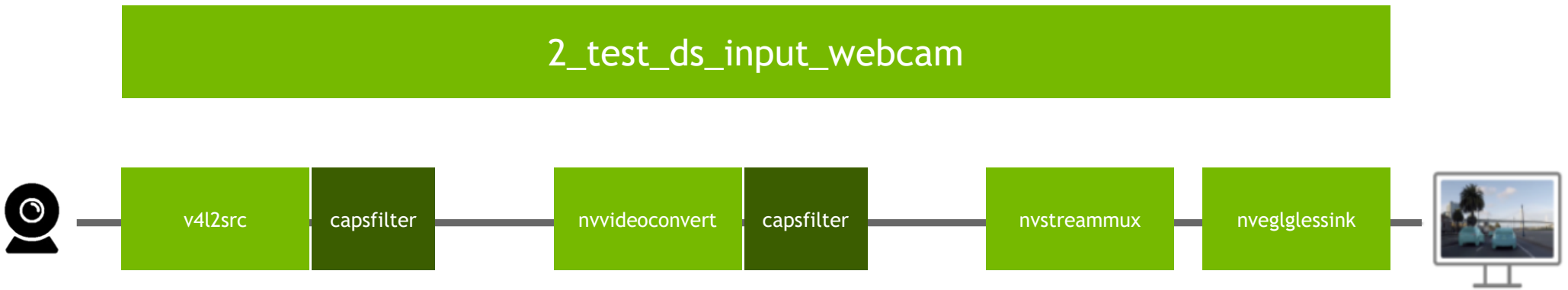
- Prerequisites
  - **Host machine**
    - SSH terminal
    - VLC player
    - (For TLT exercise) GPU machine is required & docker runtime environment
    - $ docker run --gpus all -it -v $(pwd):/workspace –w /usr/local/src -p 8888:8888 cycoslee/nv-deepstreamsdk-handson:tlt_host_210127
  - **Jetson NX (based on Jetpack 4.4.1)**
    - Docker container pull
    - $ sudo docker run -ti --runtime=nvidia --rm --net=host -e DISPLAY=$DISPLAY -w /opt/nvidia/deepstream/deepstream-5.0 --device /dev/video0 -v /tmp/.X11-unix/:/tmp/.X11-unix -v $(pwd):/workspace cycoslee/nv-deepstreamsdk-handson:ds_nx_210127
  - **Assets**
    - IP camera(supports RTSP)
    - USB camera(webcam)

HOW TO CONTROL INPUTS

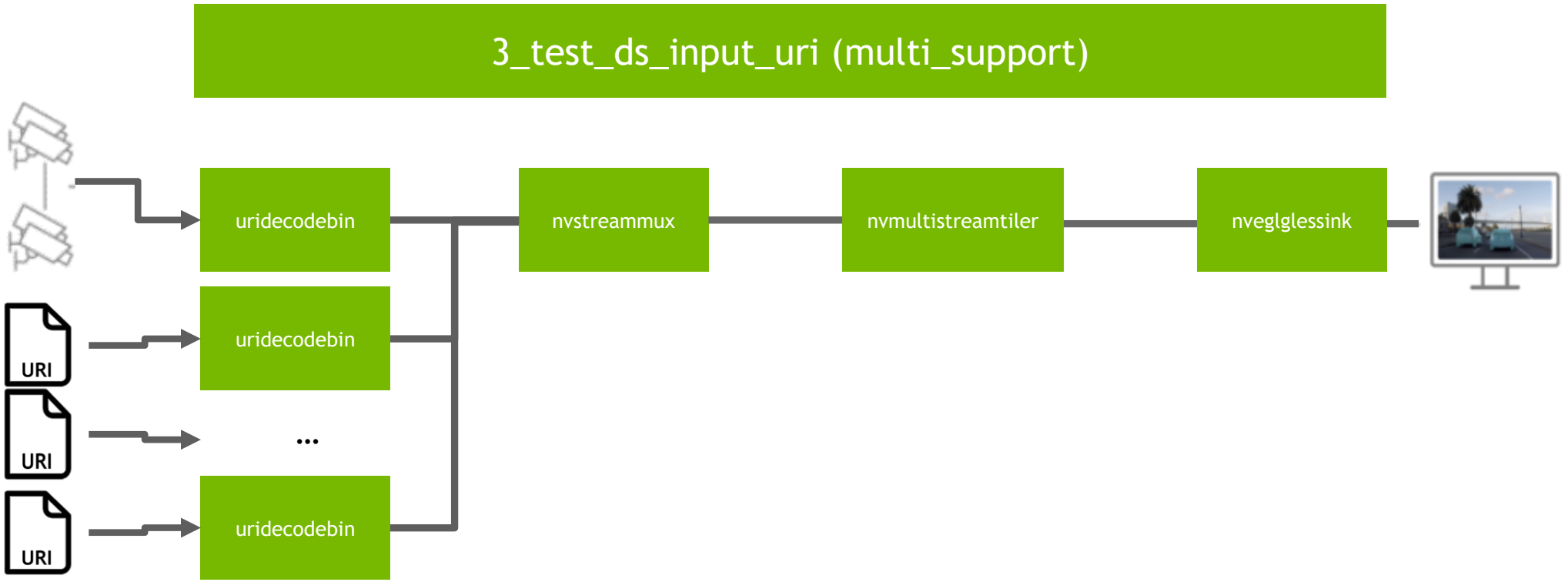




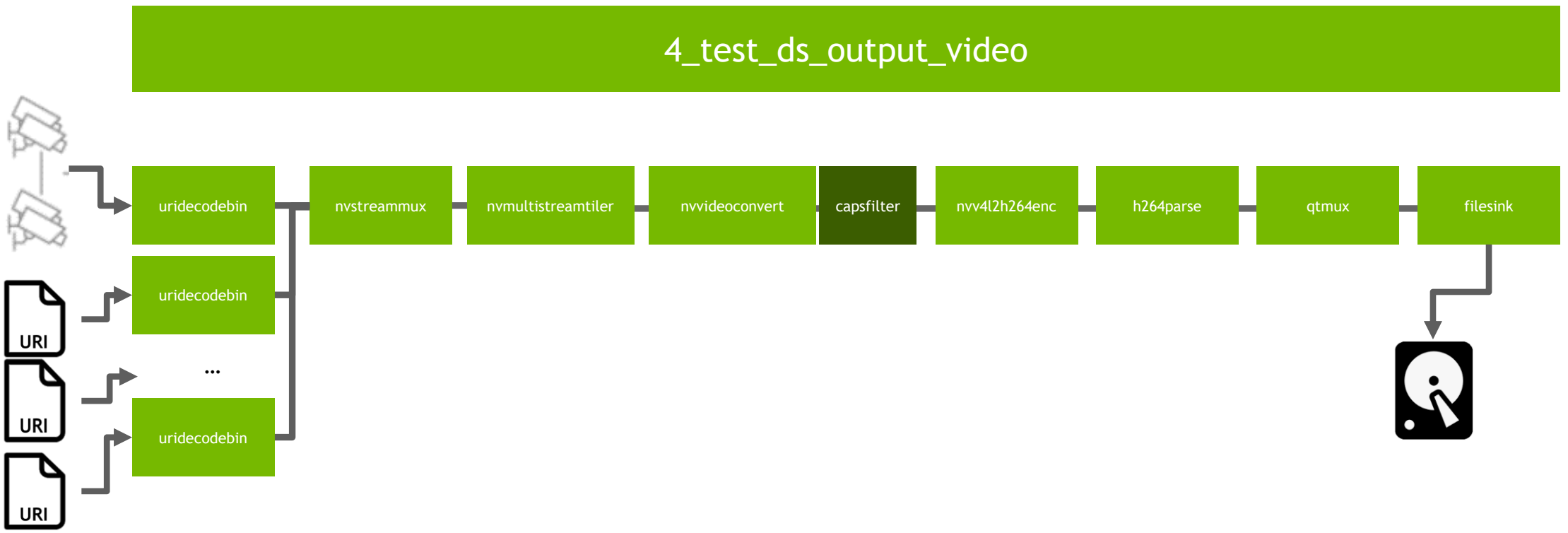
HOW TO CONTROL INPUTS



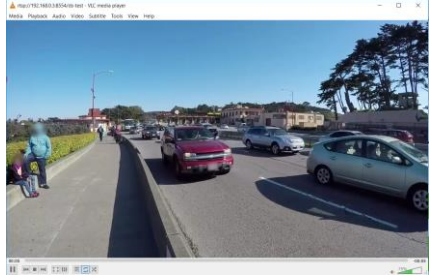
HOW TO CONTROL INPUTS

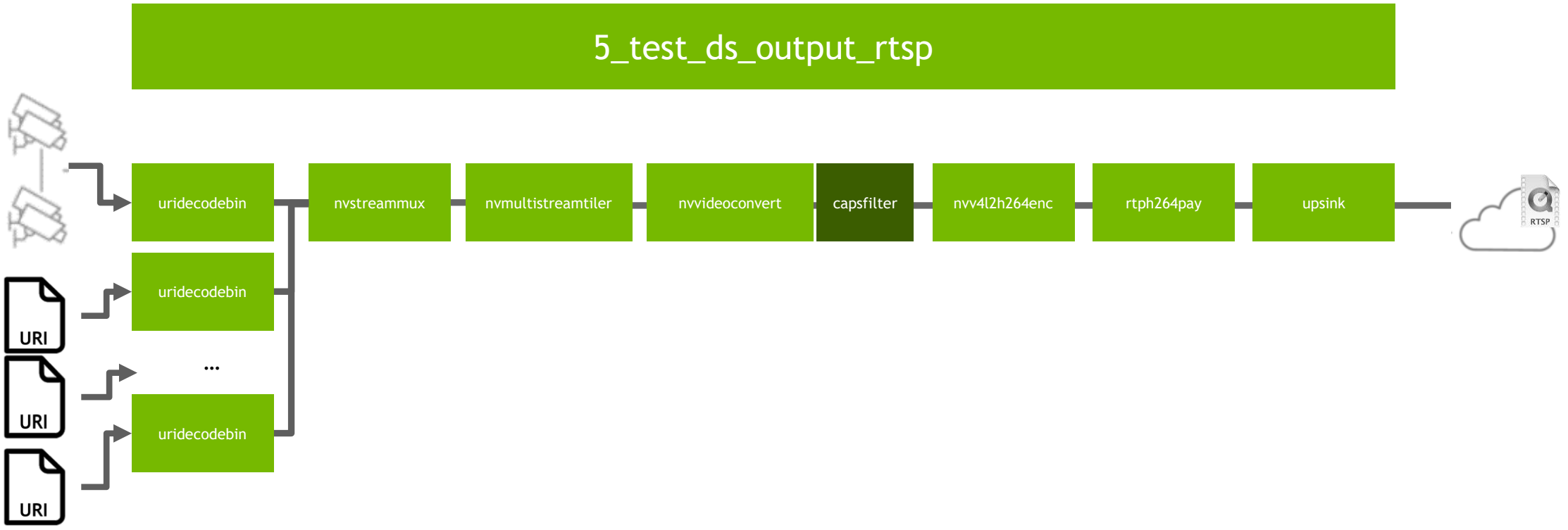


HOW TO CONTROL OUTPUTS

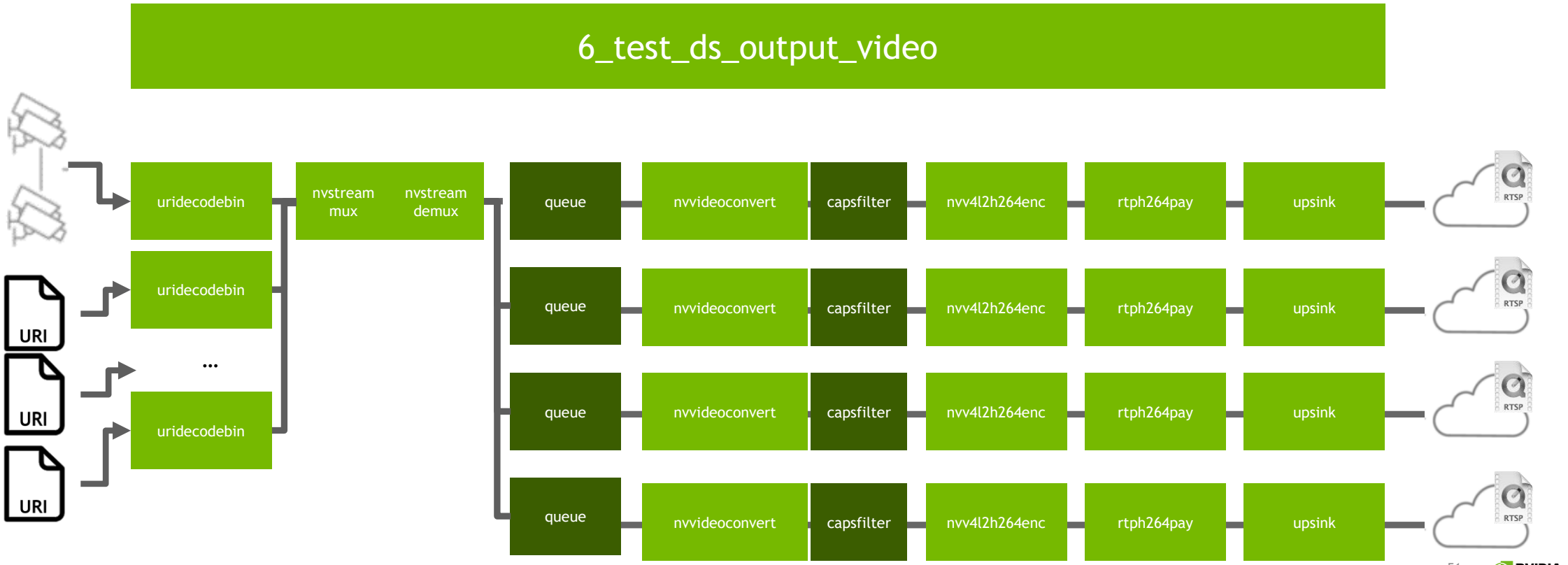


HOW TO CONTROL OUTPUTS





HOW TO CONTROL OUTPUTS



```
------------------------

                Model Name         FPS
0             inception_v4  317.924917
1  super_resolution_bsd500  152.118646
2        unet-segmentation  148.245192
3          yolov3-tiny-416  556.410331
4         ResNet50_224x224  866.609455
5         ssd-mobilenet-v1  887.422532
```

HOW TO DEPLOY DL

How to use TensorRT easily by using trtexec

7_test_trt_benchmark

Benchmarks Targeted for Jetson Xavier NX (Using GPU+2DLA)

The script will run following Benchmarks:

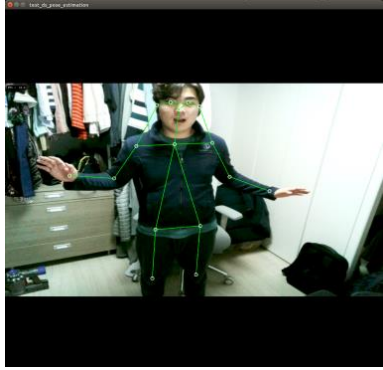
- Names : Input Image Resolution
- Inception V4 : 299x299
- ResNet-50 : 224x224
- OpenPose : 256x456
- VGG-19 : 224x224
- YOLO-V3 : 608x608
- Super Resolution : 481x321
- Unet : 256x256

For benchmark results on all NVIDIA Jetson Products; please have a look at NVIDIA jetson_benchmark webpage

Following scripts are included:

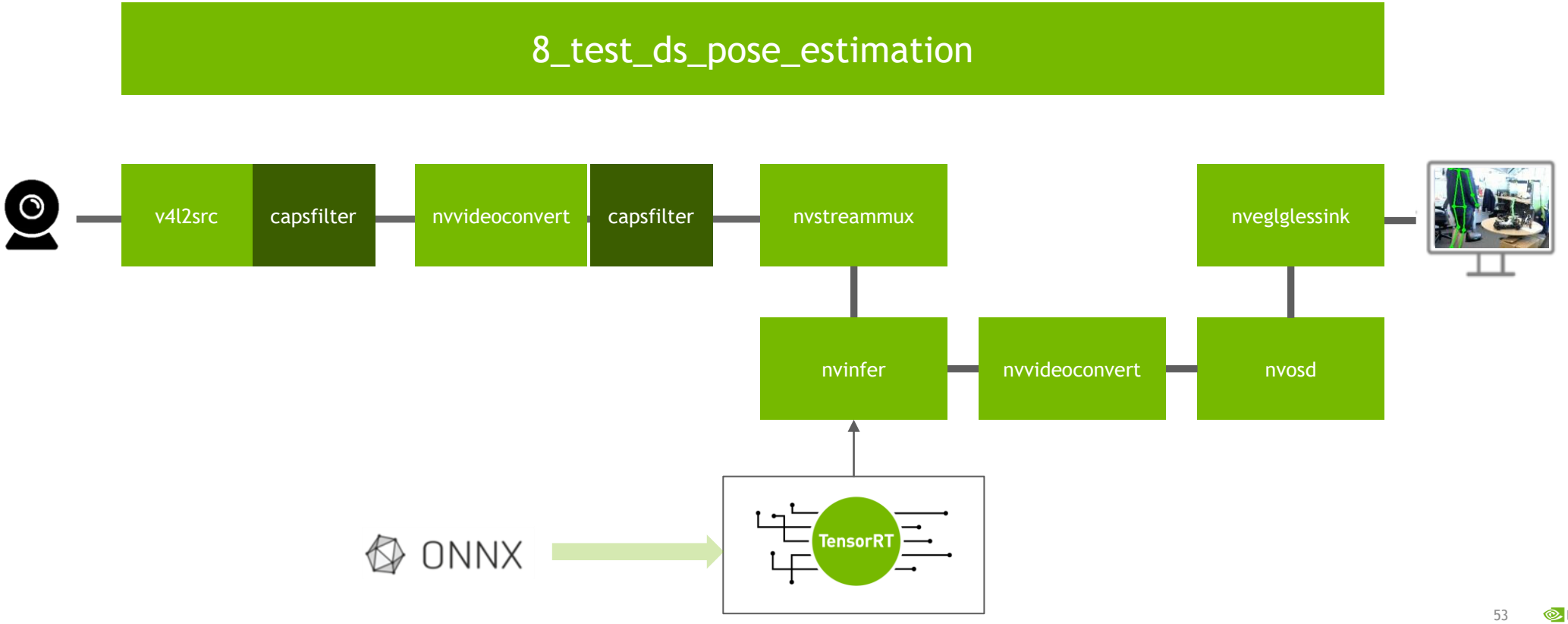
1. Install Requirements for running benchmark script (install_requirements.sh)
2. CSV files containing parameters (benchmark_csv folder)
3. Download Model (utils/download_models.py)

Model Name	FPS
inception_v4	311.73
vgg19_N2	66.43
super_resolution_bsd500	150.46
unet-segmentation	145.42
pose_estimation	237.1
yolov3-tiny-416	546.69
ResNet50_224x224	824.02
ssd-mobilenet-v1	887.6



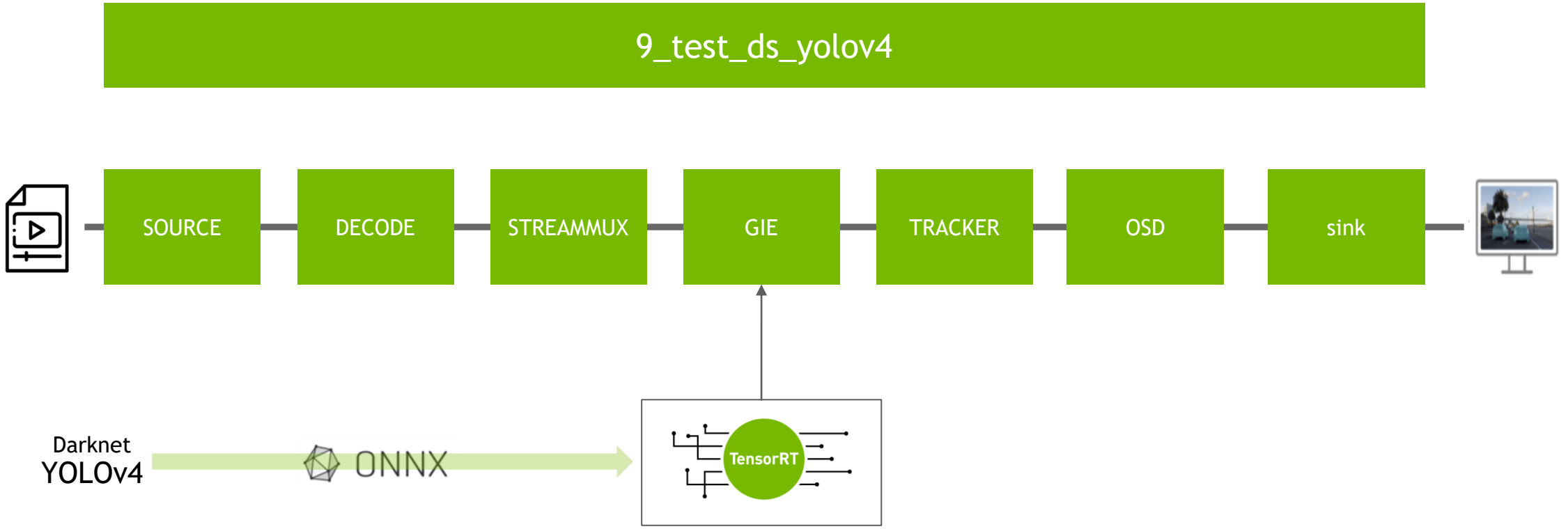
HOW TO DEPLOY DL

By using TensorRT



HOW TO DEPLOY DL

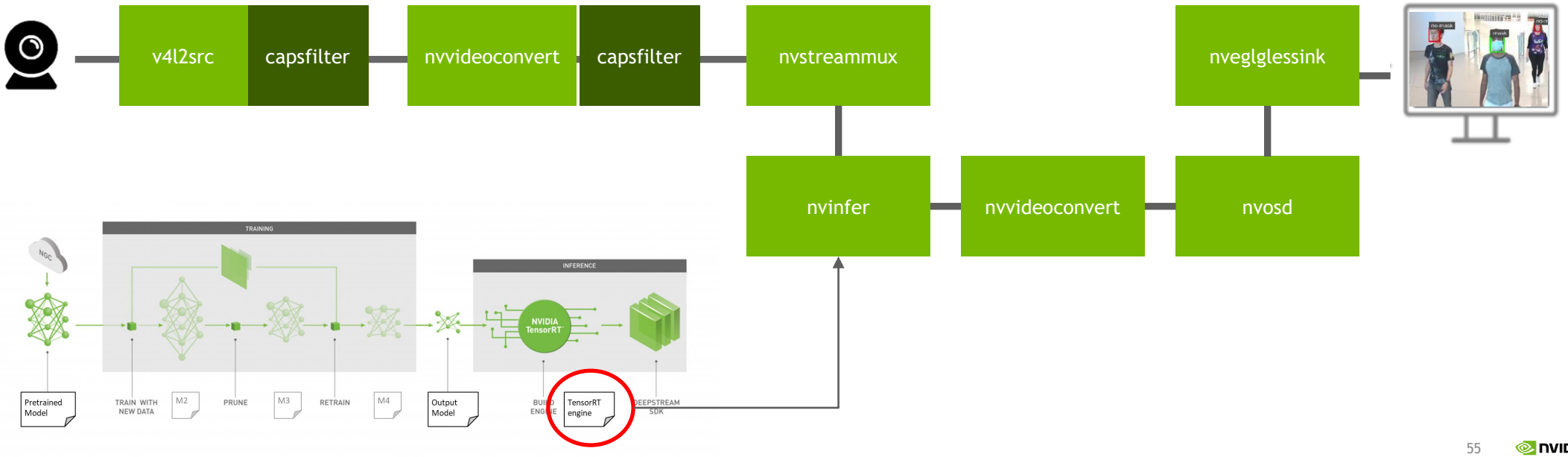
By using TensorRT



HOW TO DEPLOY DL

By using Transfer Learning Toolkit

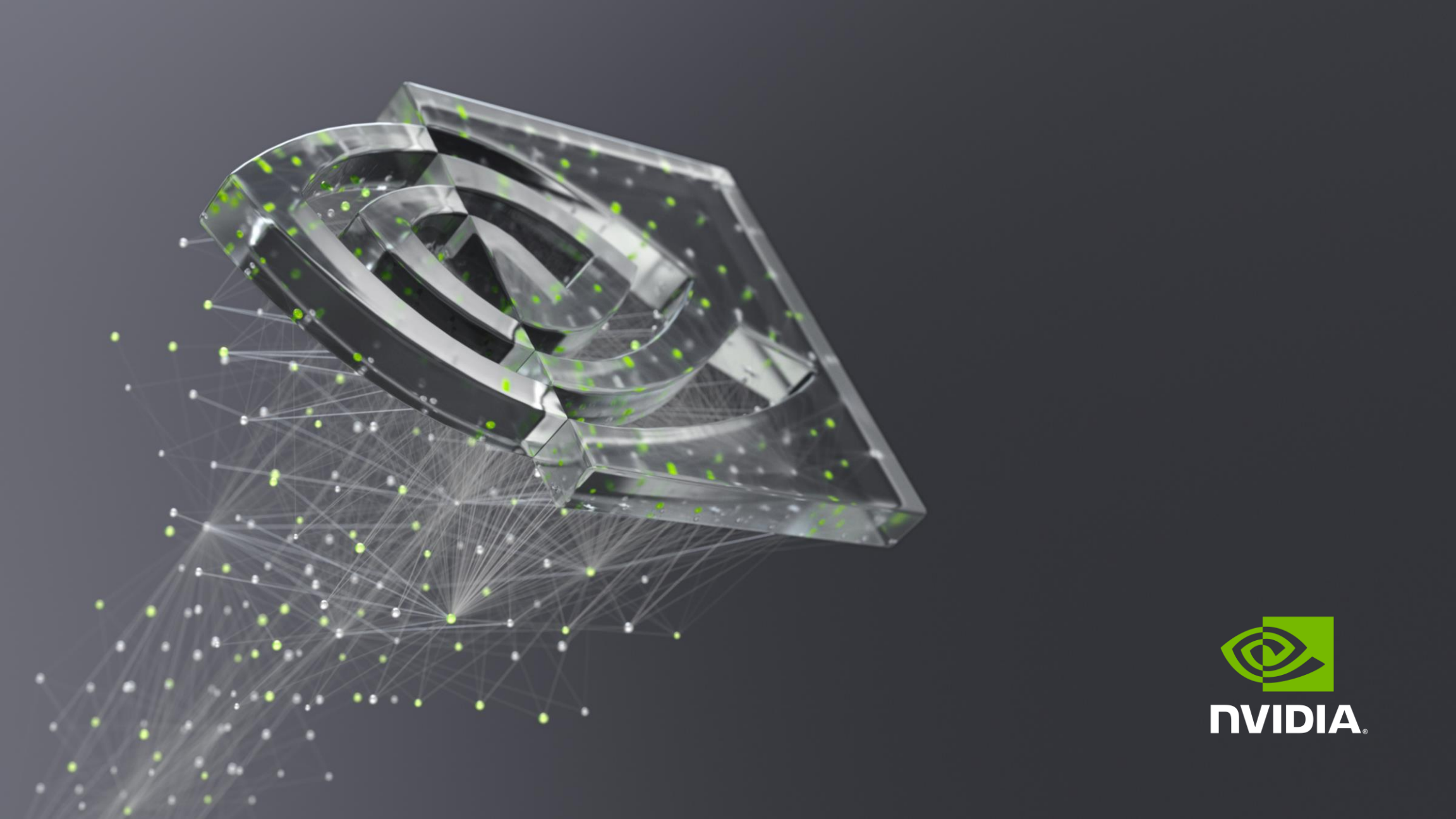
10_test_ds_facemask



NVIDIA

Q&A

You can send an E-MAIL me (jonghwanl@nvidia.com)
if you have any questions for this


NVIDIA