지능형 비디오 분석을 위한 NVIDIA DEEPSTREAM SDK 및 TRANSFER LEARNING TOOLKIT 소개

Gwangsoo Hong, Solution Architect, ghong@nvidia.com

AGENDA

- DeepStream for IVA
  - Realtime Streaming Video Analytics
  - Framework for Analyzing Video
  - Understand the Basics: DeepStream SDK
  - Plugins for GPU Acceleration
  - Build with DeepStream: Example Applications
- Transfer Learning Toolkit for IVA
- Getting Started Resources
- Demo

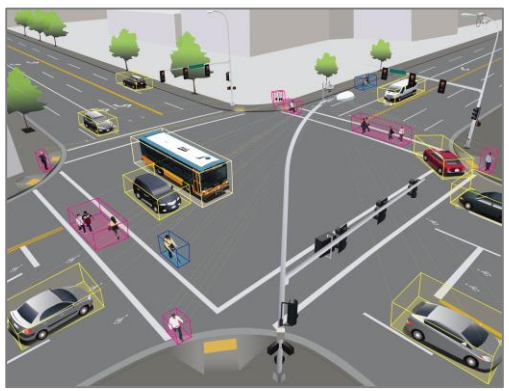
REALTIME STREAMING VIDEO ANALYTICS

REALTIME STREAMING VIDEO ANALYTICS FROM EDGE TO CLOUD

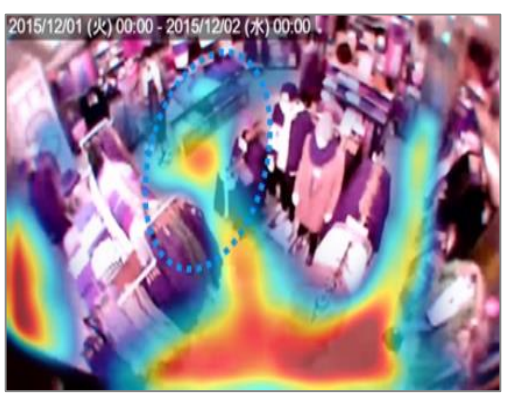
Access Control

Managing operations

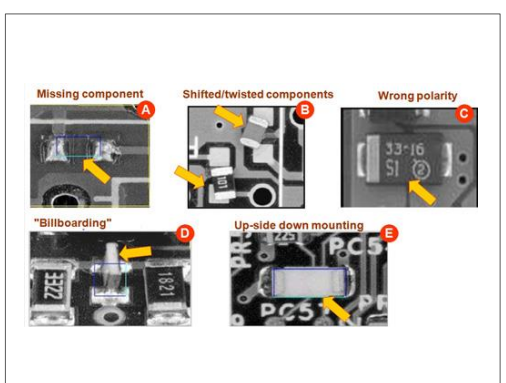
Parking Management



Traffic Engineering



Retail Analytics



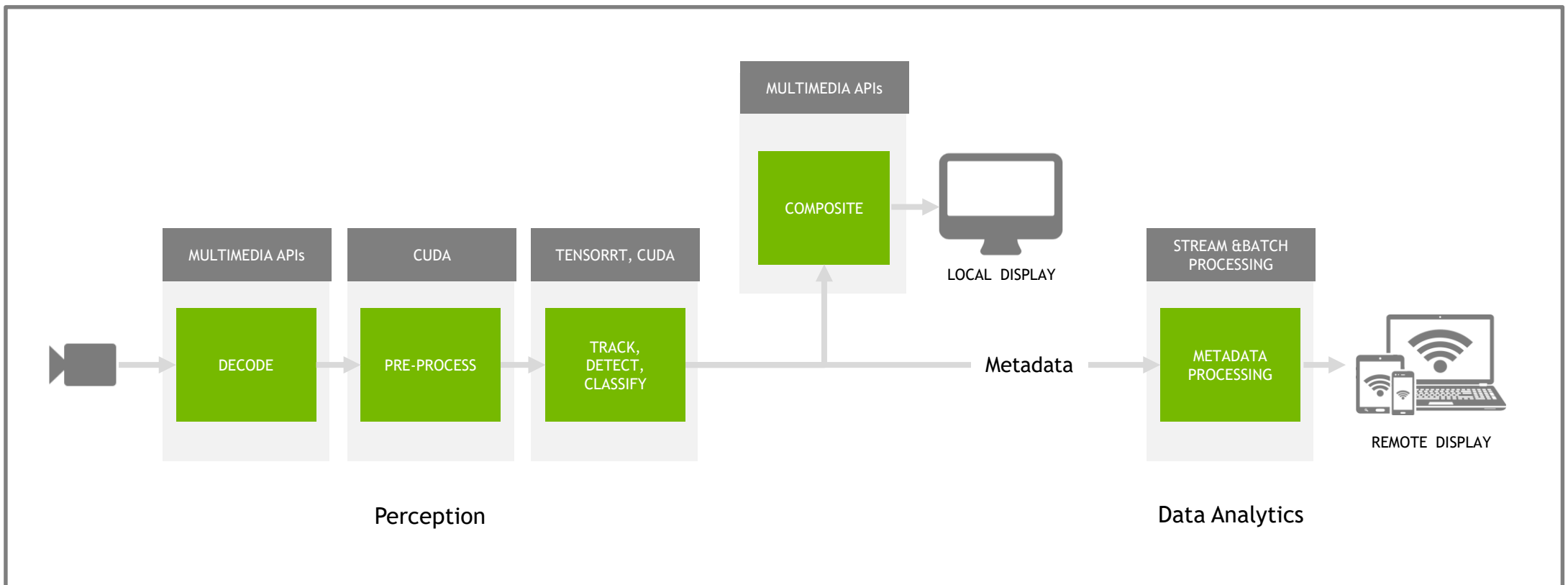
Optical Inspection

Managing Logistics

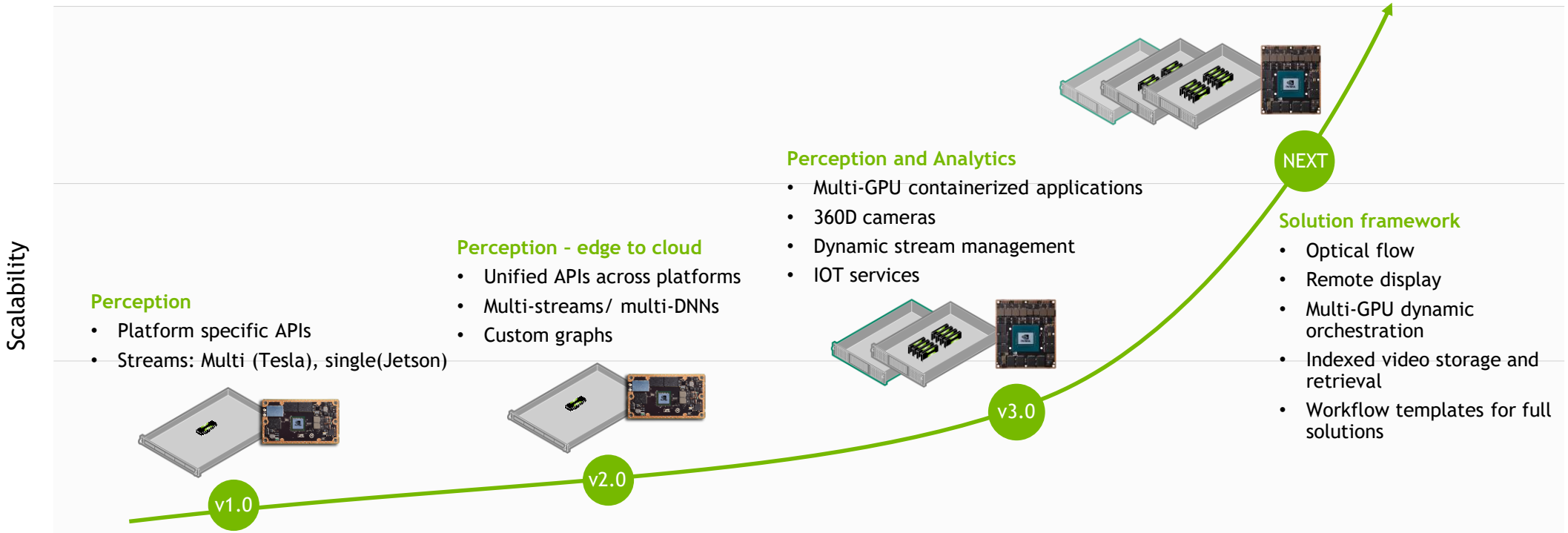
Content Filtering

FRAMEWORK FOR ANALYZING VIDEO

FRAMEWORK FOR ANALYZING VIDEO

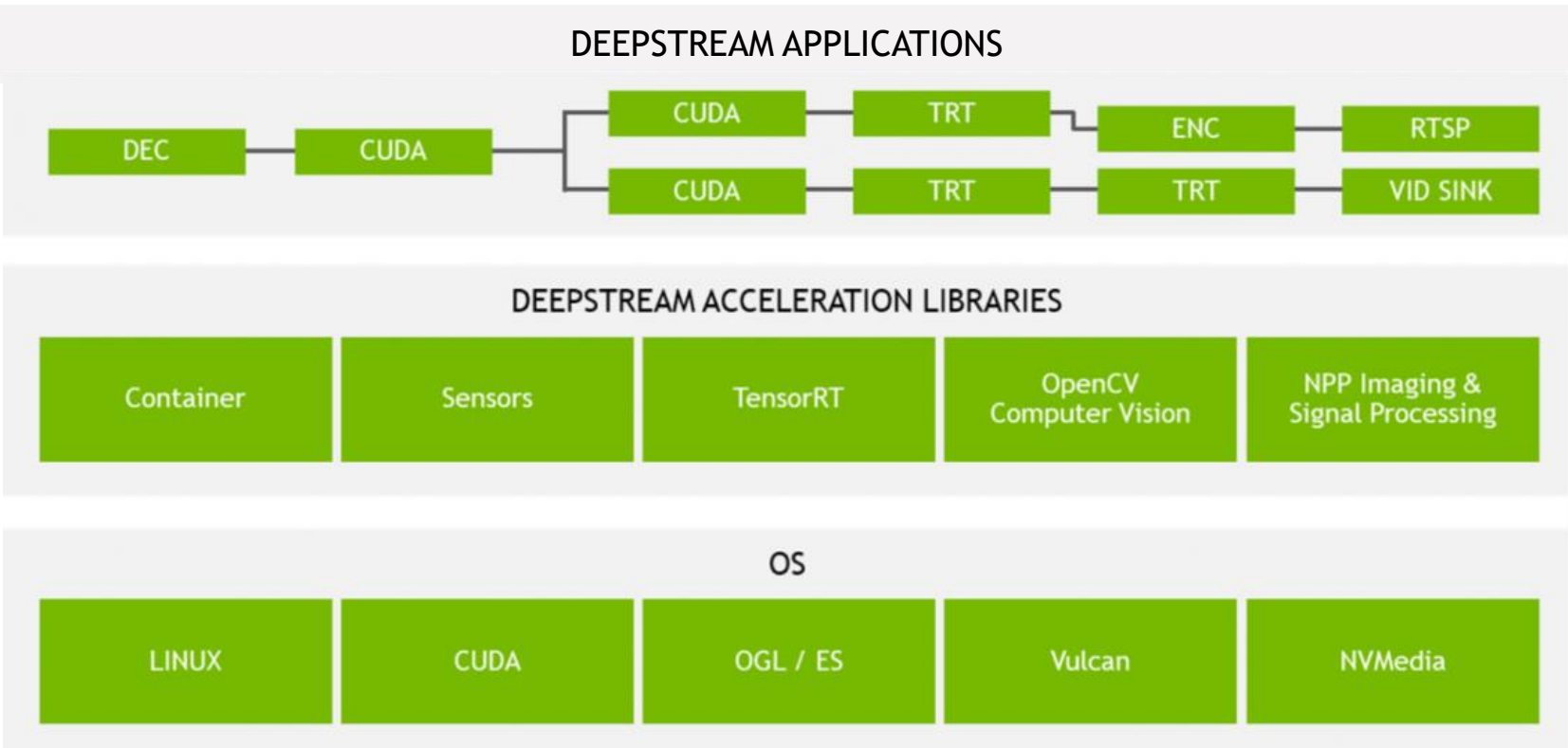


DEEPSTREAM FOR AI APPLICATION PERFORMANCE AND SCALE



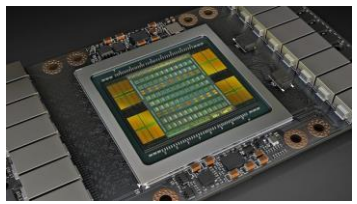
UNDERSTAND THE BASICS: DEEPSTREAM SDK

DEEPSTREAM SDK



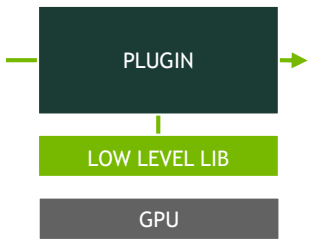
WHAT'S NEW IN DEEPSTREAM 3.0

LATEST PLATFORMS - TESLA T4, Jetson XAVIER



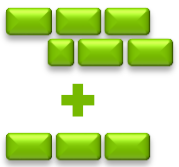
TensorRT 5, CUDA 10

NEW PLUGINS



Increased capability and throughput

DYNAMIC STREAM MANAGEMENT



Add, remove, modify streams on the fly

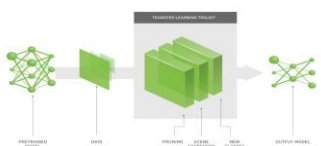
CONNECT EDGE TO CLOUD

Stream and Batch Analytics on Metadata

EASY TO SCALE AND MANAGE

Deploy in Docker Containers

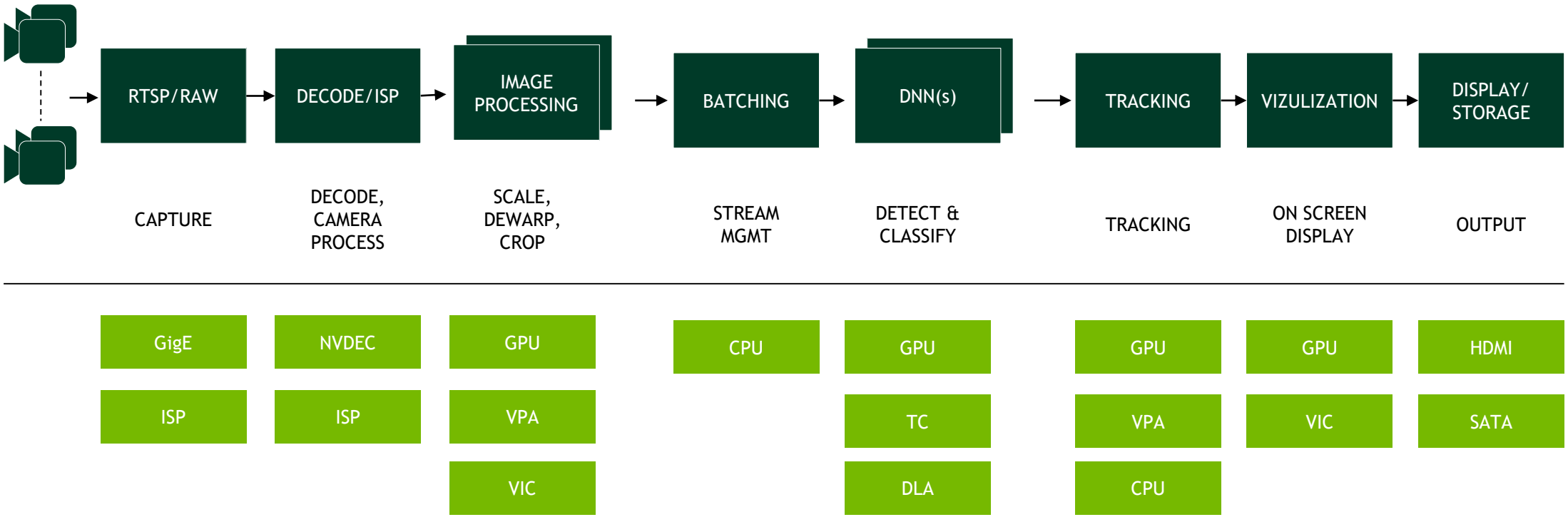
HIGH EFFICIENCY AND THROUGHPUT WITH Transfer Learning Toolkit



Transfer Learning Toolkit model files are plug-n-play

Dynamic stream management and containers available on Tesla only. Jetson support coming soon

DEEPSTREAM STREAMING ARCHITECTURE

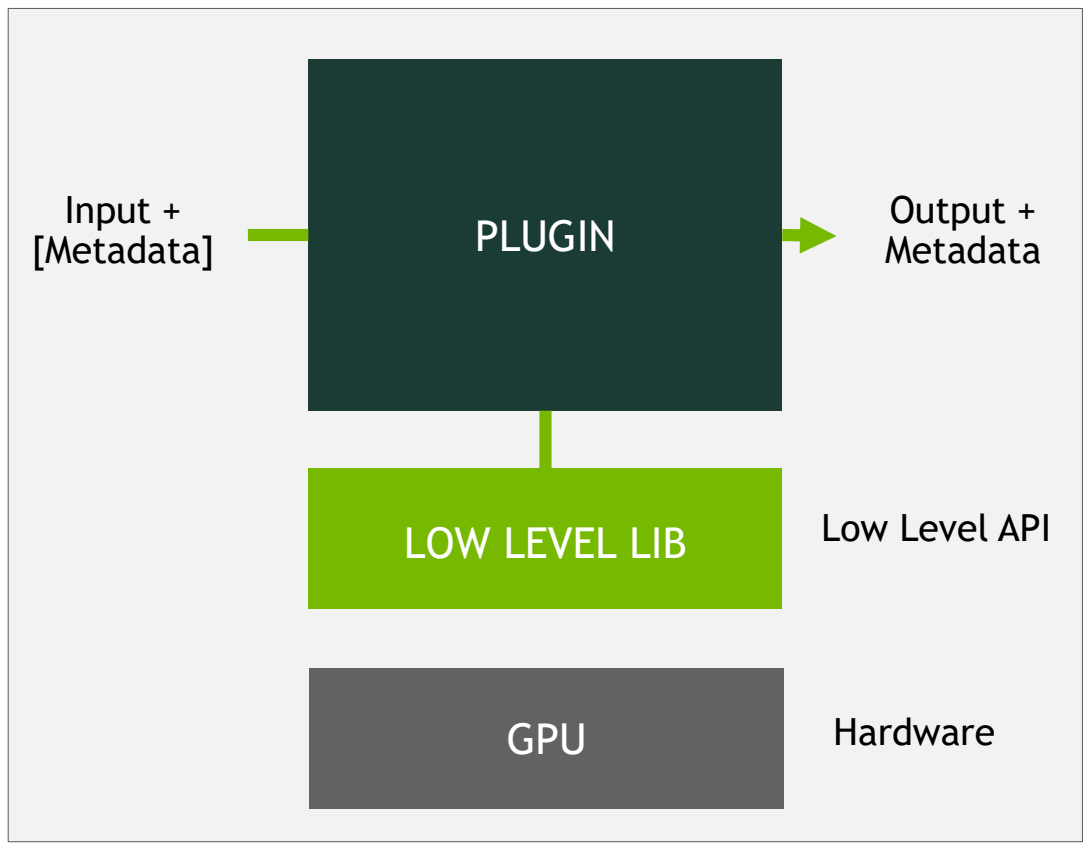


GSTREAMER FOUNDATIONS

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.

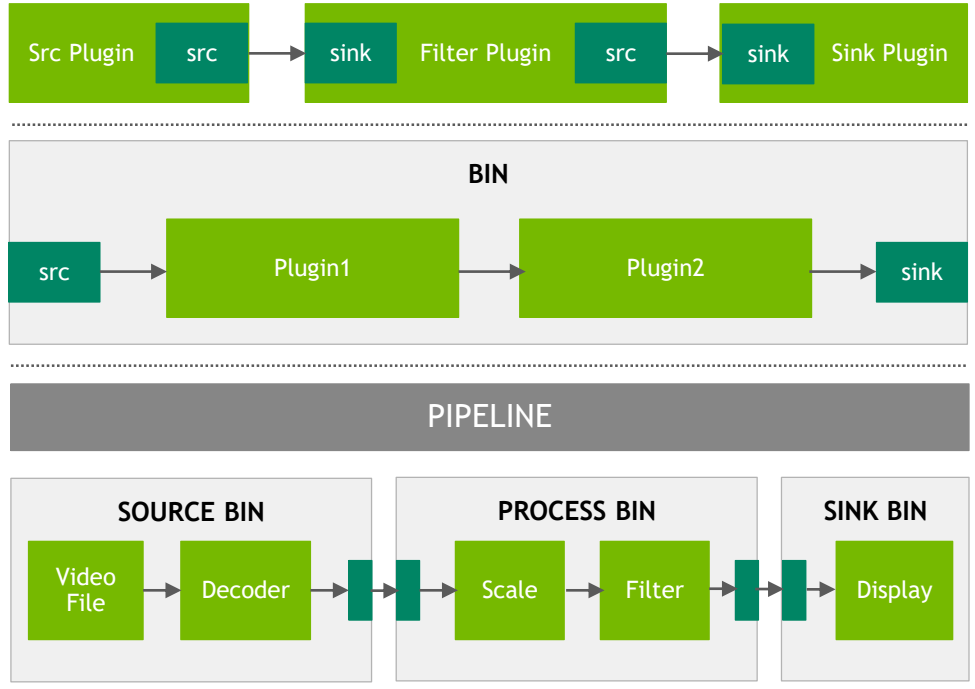
DEEPSTREAM BUILDING BLOCK

- 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

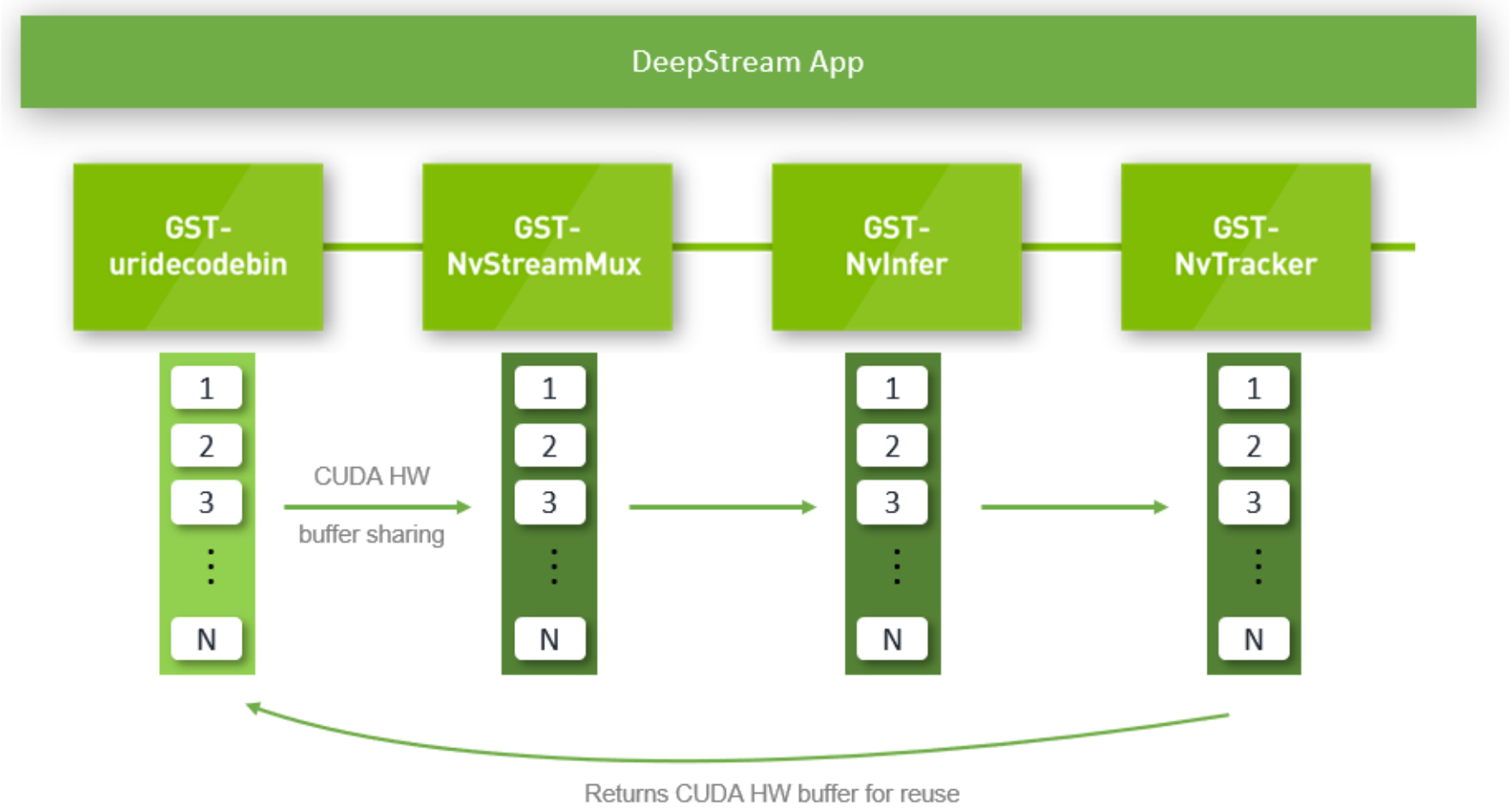


GSTREAMER

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

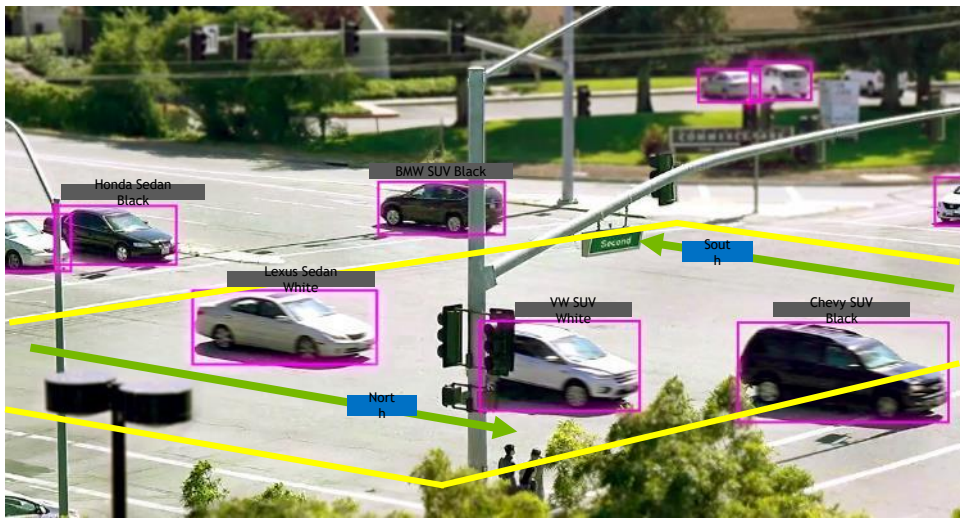


MEMORY MANAGEMENT IN DEEPSTREAM



METADATA IN DEEPSTREAM

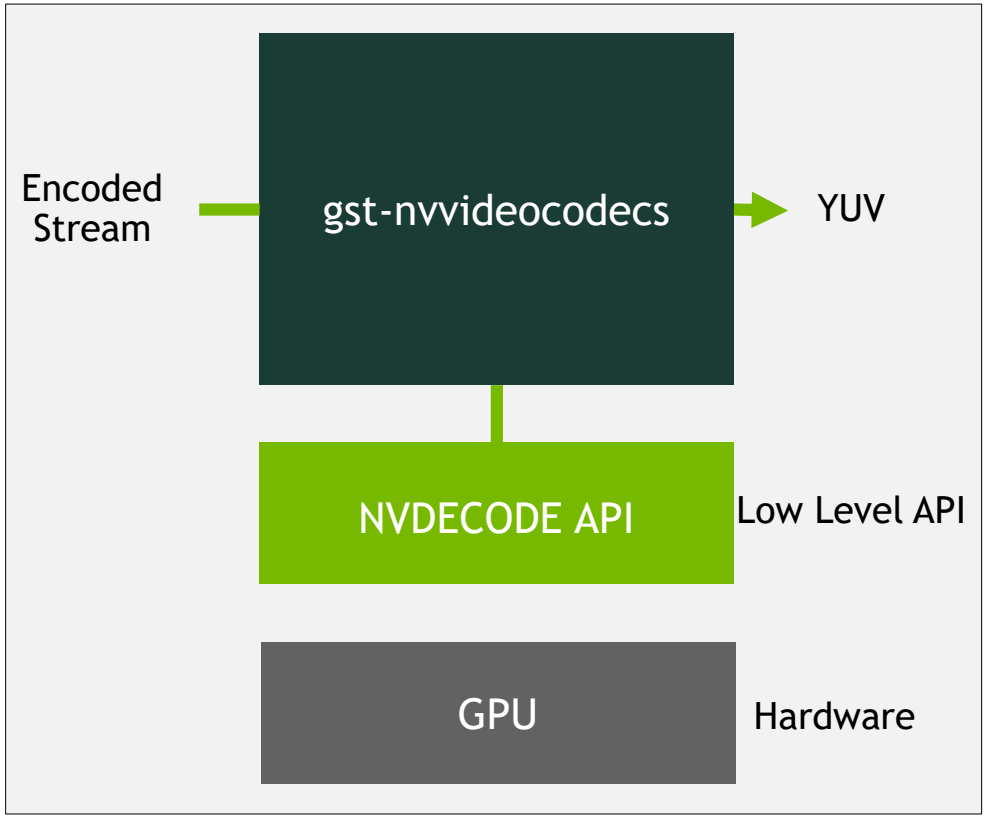
- Metadata is generated by plugins in the graph
- Plugins can progressively populate generated metadata
- Metadata generated at every stage of the graph can be used for further processing
- Metadata examples
  - Type of object detected
  - ROI coordinates
  - Object classification
  - Unique ID
  - Source and GPU ID
  - Rendering information and many more



PLUGINS FOR GPU ACCELERATION

DECODER PLUGIN

gst-nvvideocodecs



Input	H.264, H.265, VP8, VP9, MPEG2/4
Output	NV12
Parameters	Bit rate control, i-frame decoding

DECODER PLUGIN

NVDEC and NVENC

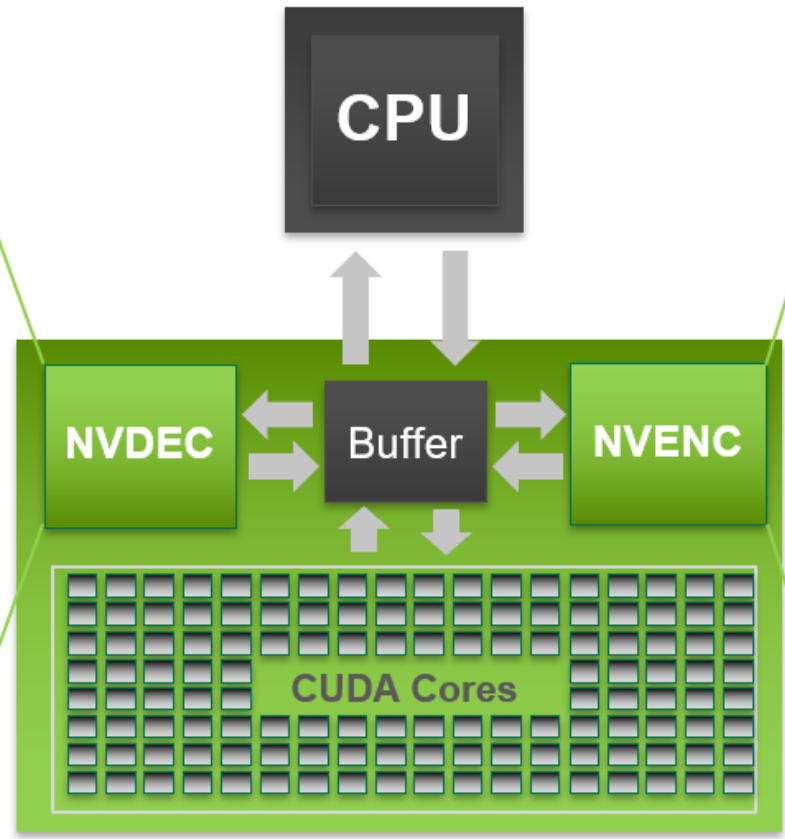
Decode HW*

Formats:
- MPEG-2
- VC1
- VP8
- VP9
- H.264
- H.265
- Lossless

Bit depth:
- 8 bit
- 10 bit
- 12 bit

Color**
- YUV 4:2:0
- YUV 4:4:4

Resolution
- Up to 8K***



Encode HW*

Formats:
- H.264
- H.265
- Lossless

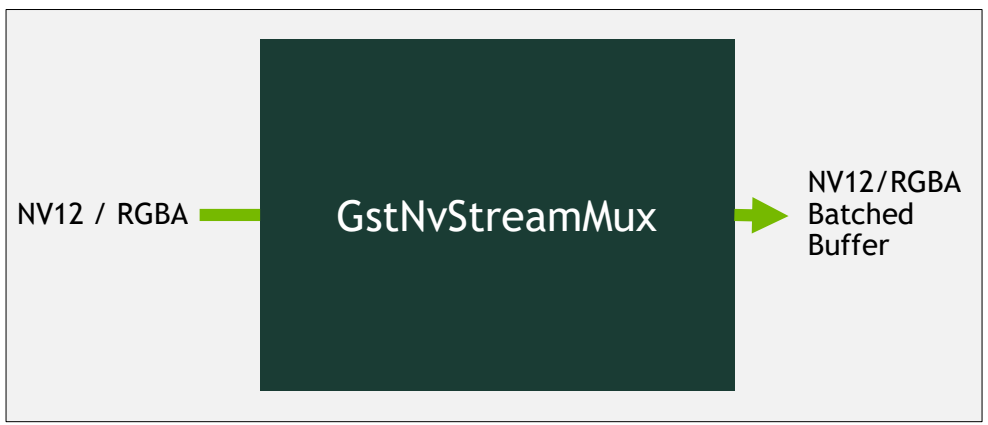
Bit depth:
- 8 bit
- 10 bit

Color**
- YUV 4:4:4
- YUV 4:2:0

Resolution
- Up to 8K***

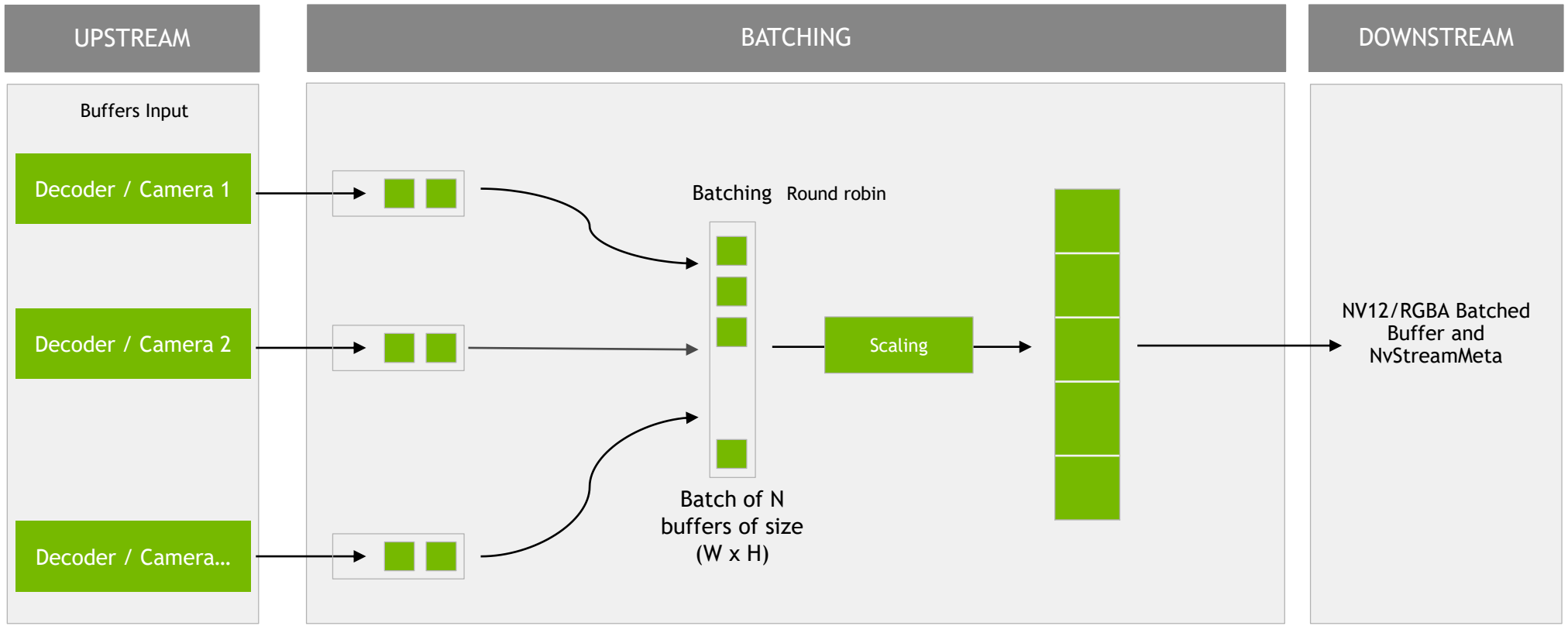
BATCHING

GstNvStreamMux

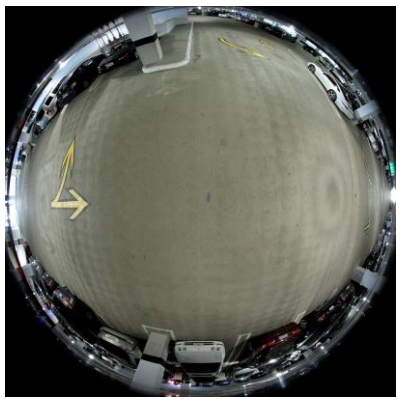


Input	NV12 / RGBA streams
Output	NV12/RGBA Batched Buffer Metadata containing information about input frames : original timestamps, frame numbers
Parameters	Batch size (int) - Number of buffers in a batch Batch timeout (int) - Time in microseconds to wait to form a batch Width, Height (int) – Scaling factor for source frames Frame padding (int) – Maintain source aspect ratio by padding with black bands

BATCHING - GSTNVSTREAMMUX



ENABLING 360D CAMERA PROCESSING



NVWARP360 SDK

Equirectangular

Perspective

Cylindrical

Rotated cylinder

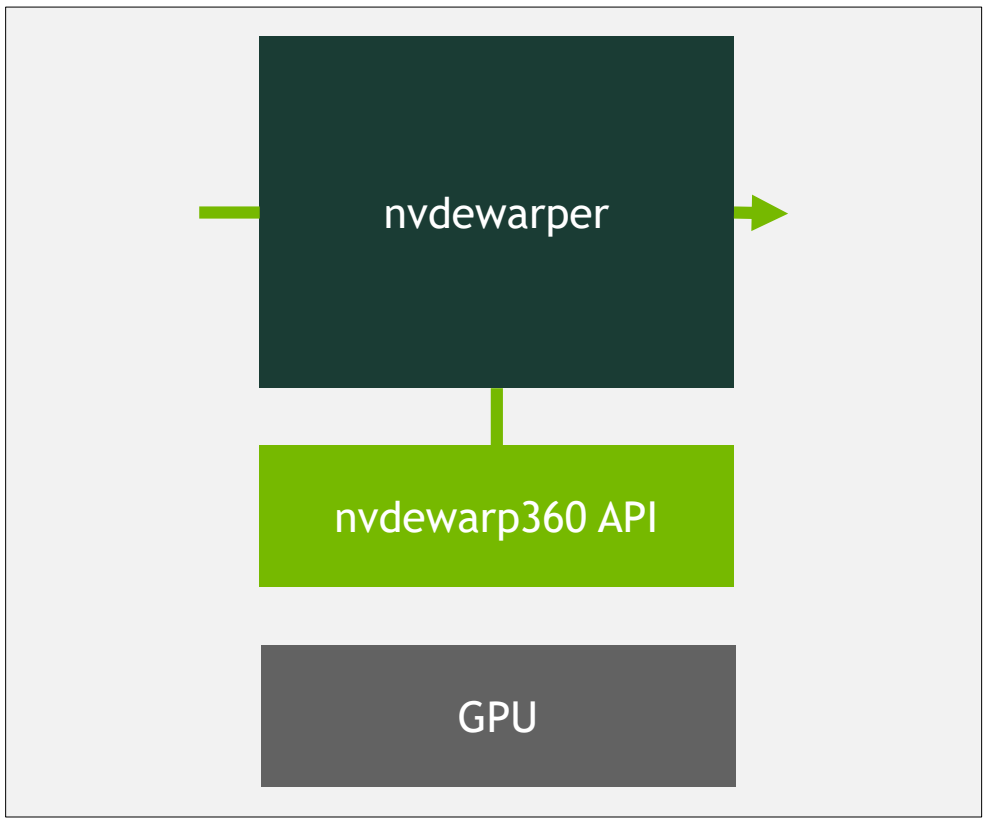
Panini

Pushbroom

Tesla only

SEAMLESS PLUG-AND-PLAY IN DEEPSTREAM

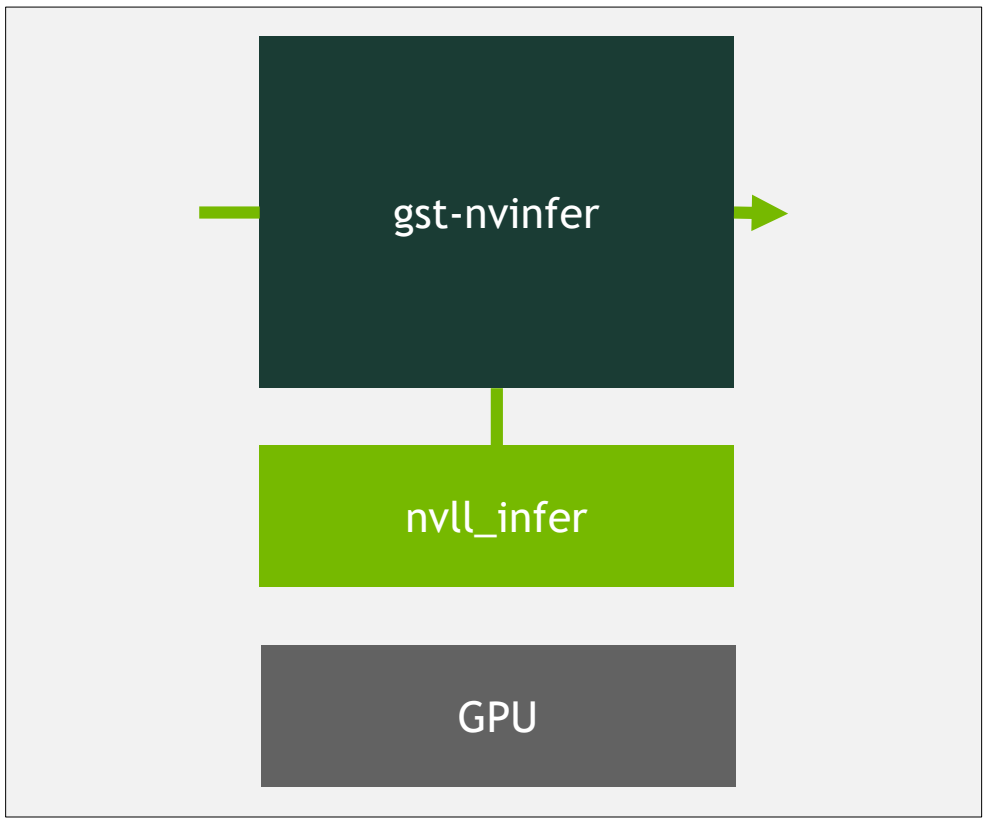
nvdewarper



Input	360D RGBA frame
Output	RGBA buffer for each surface/projections. Projections: Pushbroom, Rotated cylinder, Perspective, Equirectangular, Panini, Cylindrical
Parameters	Number of Dewarping surface per frame, each surface width and height, projection type for each surface, dewarping angles like top, bottom, yaw, roll etc.

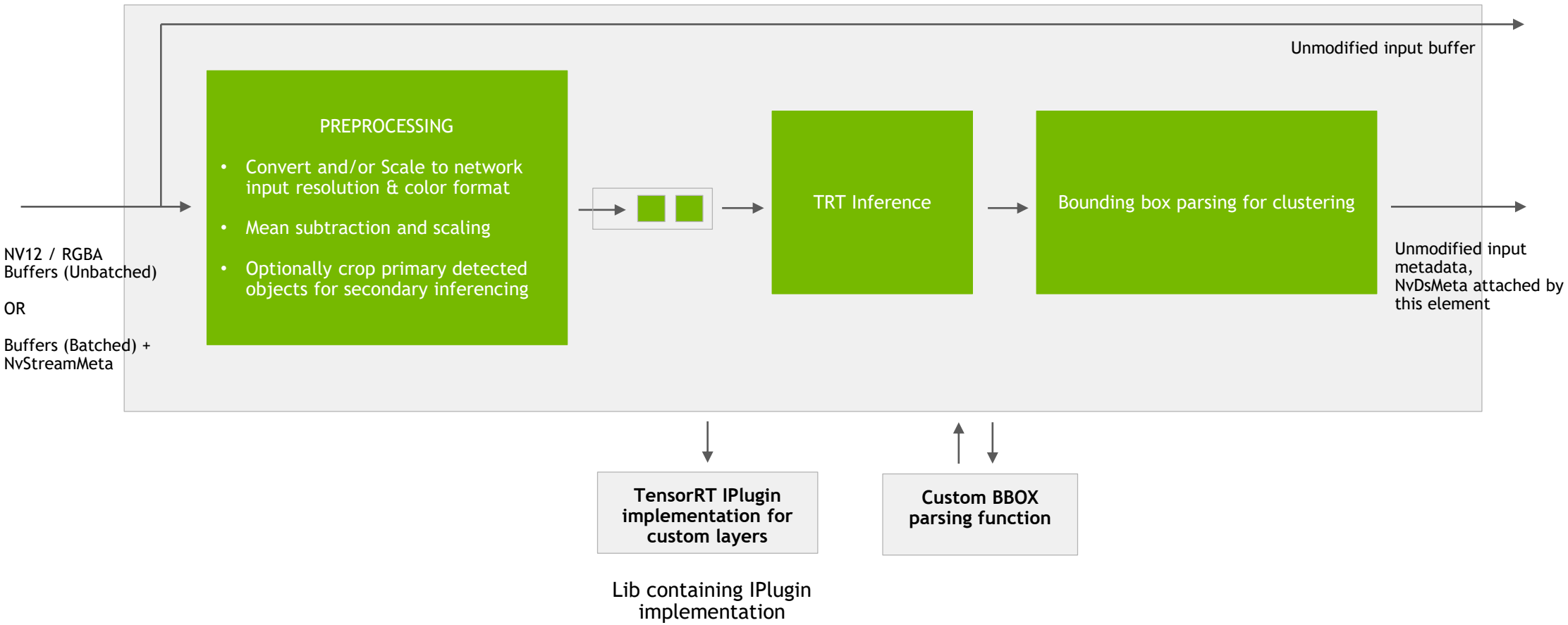
AN ALL NEW INFERENCE PLUGIN

gst-nvinfer



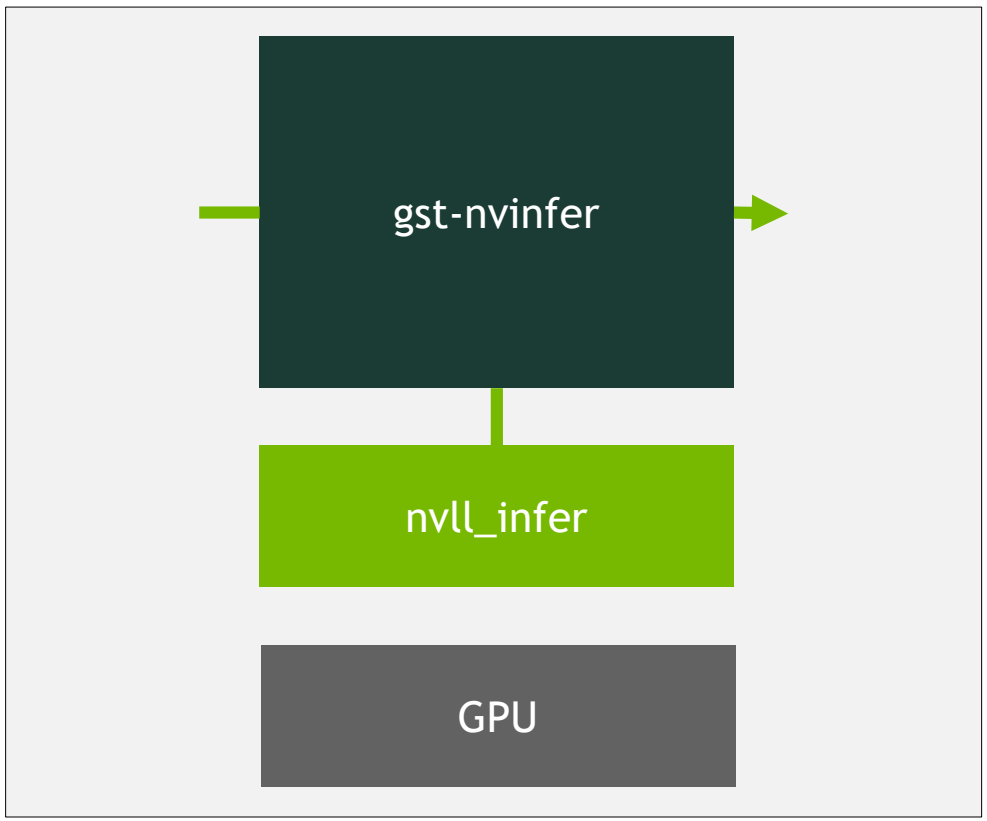
Input	NV12/RGBA buffer, Model files – Caffe Model and Caffe Prototxt, ONNX, UFF file, TRT plan files
Output	Original metadata + NvDsMeta attached by this element (class, bbox, etc)
Parameters	Batch size, inference interval, clustering params, class threshold, Bbox color, width & height of Bbox to filterout some boxes for downstream component etc.

GSTNVINFER


Unmodified input buffer
PREPROCESSING
Convert and/or Scale to network input resolution & color format
Mean subtraction and scaling
Optionally crop primary detected objects for secondary inferencing
TRT Inference
Bounding box parsing for clustering
NV12 / RGBA Buffers (Unbatched)
OR
Buffers (Batched) + NvStreamMeta
Unmodified input metadata, NvDsMeta attached by this element
TensorRT IPlugin implementation for custom layers
Custom BBOX parsing function
Lib containing IPlugin implementation

AN ALL NEW INFERENCE PLUGIN

gst-nvinfer



Input	NV12/RGBA buffer, Model files – Caffe Model and Caffe Prototxt, ONNX, UFF file, TRT plan files
Output	Original metadata + NvDsMeta attached by this element (class, bbox, etc)
Parameters	Batch size, inference interval, clustering params, class threshold, Bbox color, width & height of Bbox to filterout some boxes for downstream component etc.

GSTNVINFER

configuration file

- Support variable model files
  - Caffe model “model-file” “proto-file”
  - ONNX model “onnx-file”
  - UFF model “uff-file”
  - TRT plan model “model-engine-file”

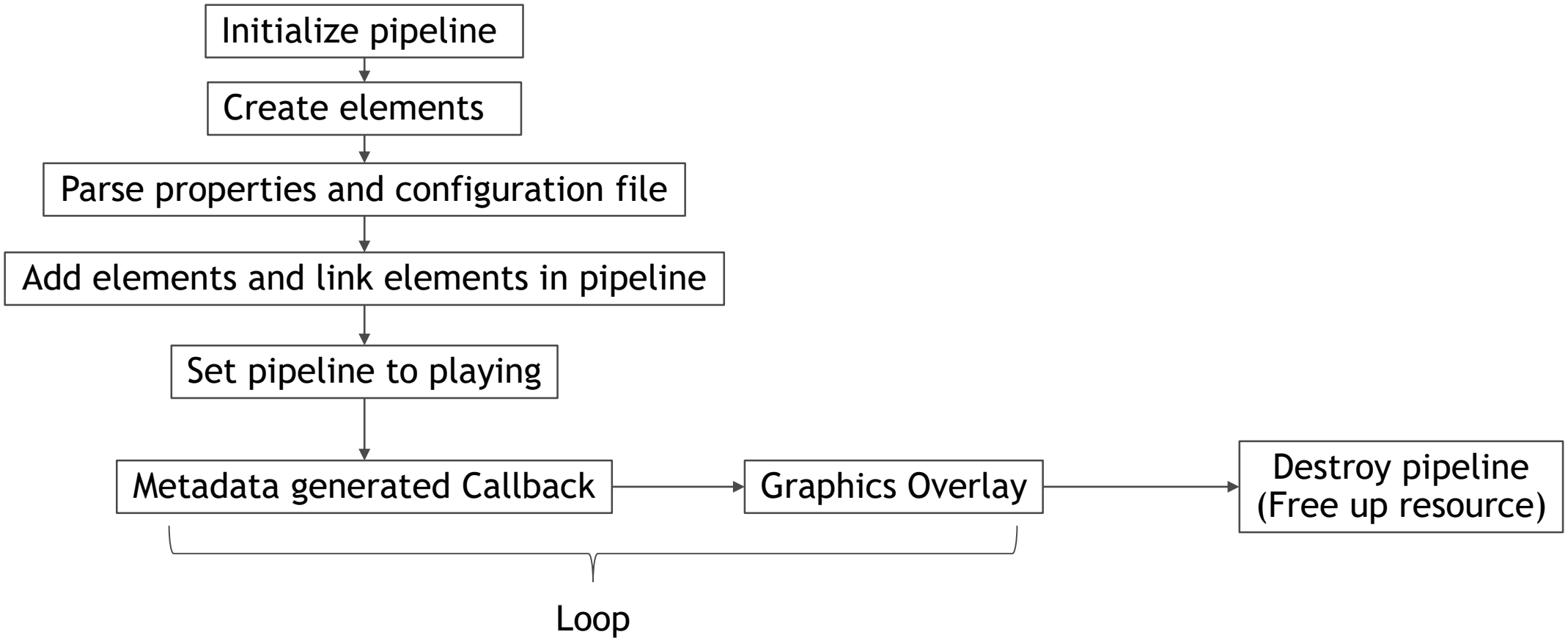
```
[property]
gpu-id=0
net-scale-factor=0.0039215697906911373
model-file=../../../samples/models/Primary_Detector/resnet10.caffemodel
proto-file=../../../samples/models/Primary_Detector/resnet10.prototxt
labelfile-path=../../../samples/models/Primary_Detector/labels.txt
int8-calib-file=../../../samples/models/Primary_Detector/cal_trt4.bin
batch-size=1
network-mode=1
num-detected-classes=4
interval=0
gie-unique-id=1
parse-func=4
output-blob-names=conv2d_bbox;conv2d_cov/Sigmoid

[class-attrs-all]
threshold=0.2
eps=0.2
group-threshold=1
```

PLUGINS IN DEEPSTREAM SDK

Plugin Name	Functionality
gst-nvvideocodecs	Accelerated video decoders
gst-nvstreammux	Stream aggregator - muxer and batching
gst-nvinfer	TensorRT based inference for detection & classification
gst-nvtracker	Reference KLT tracker implementation
gst-nvosd	On-Screen Display API to draw boxes and text overlay
gst-tiler	Renders frames from multi-source into 2D grid array
gst-eglglessink	Accelerated X11 / EGL based renderer plugin
gst-nvvidconv	Scaling, format conversion, rotation
Gst-nvdewarp	Dewarping for 360 Degree camera input
Gst-nvmsgconv	Meta data generation
Gst-nvmsgbroker	Messaging to Cloud

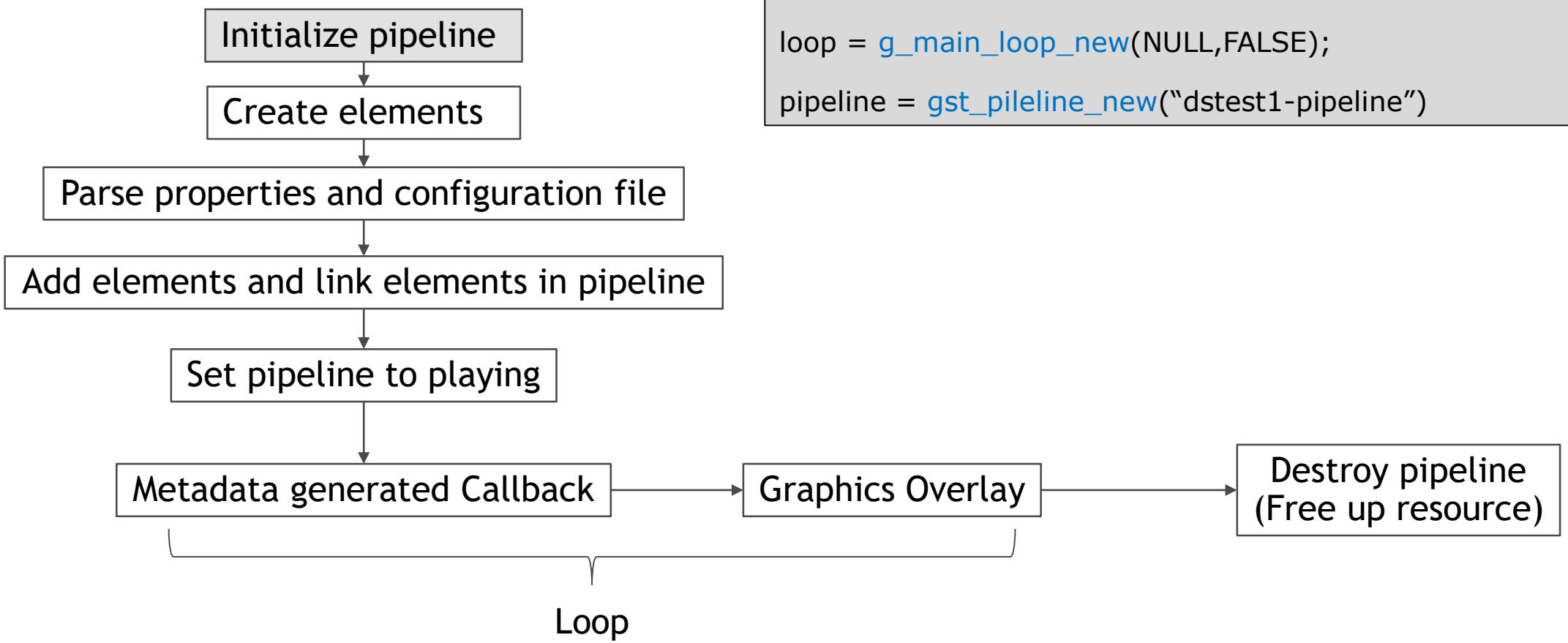
BUILD WITH DEEPSTREAM: EXAMPLE APPLICATIONS

APPLICATION FLOW



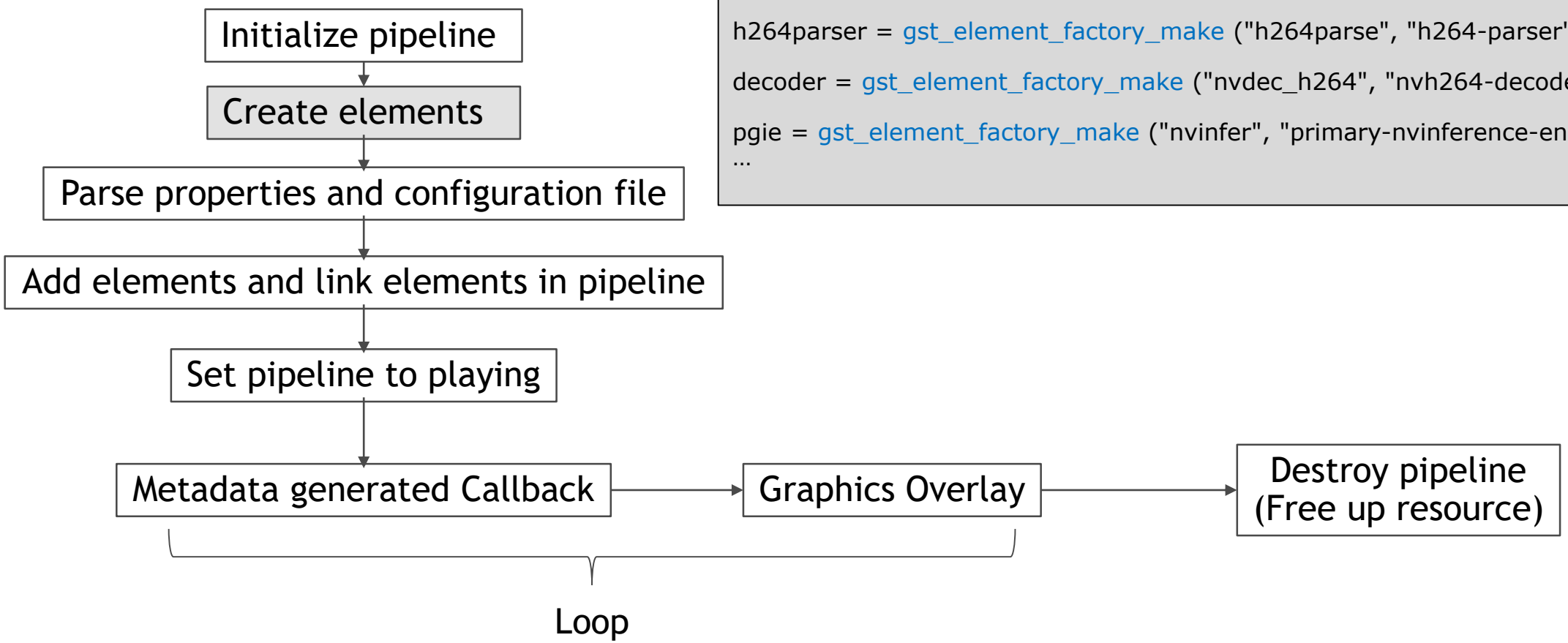
APPLICATION FLOW

```
gst_init(&argc,&argv);

loop = g_main_loop_new(NULL,FALSE);

pipeline = gst_pileline_new("dstest1-pipeline")
```



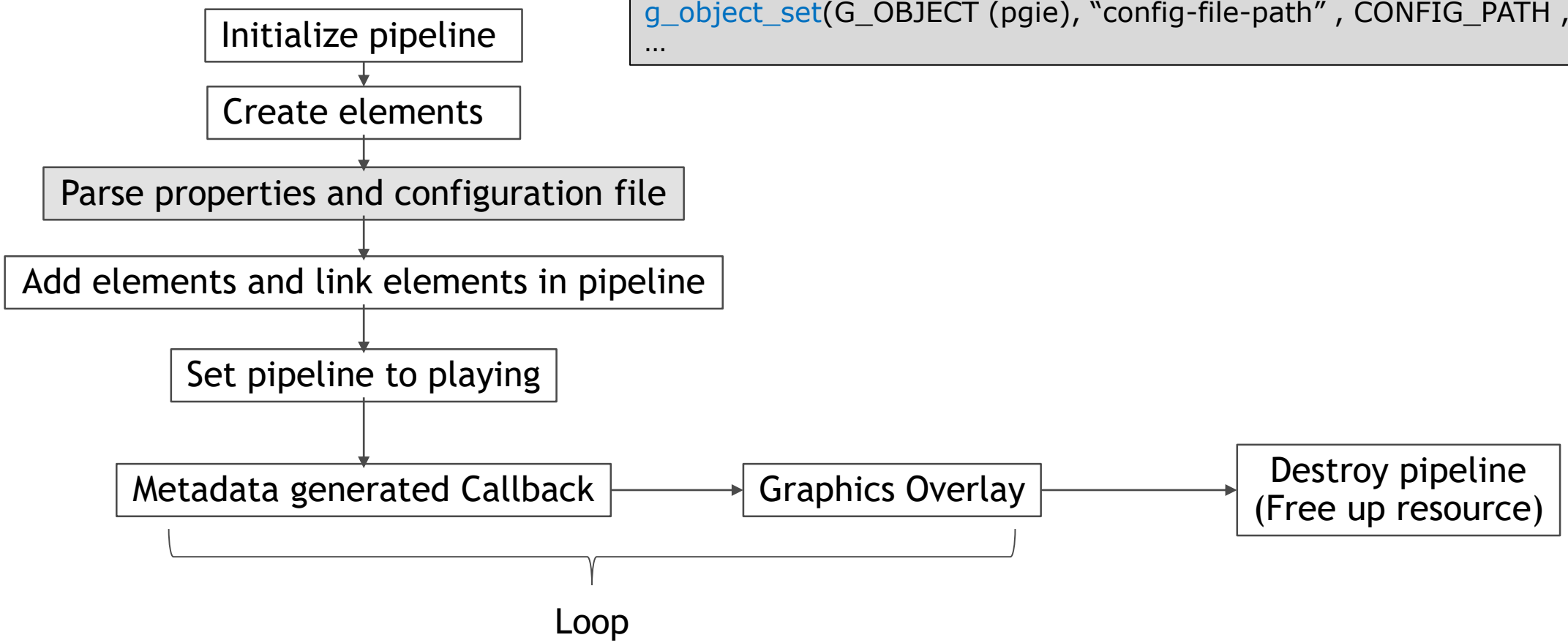
APPLICATION FLOW

```
source = gst_element_factory_make(“filesrc”, “file-source”)

h264parser = gst_element_factory_make ("h264parse", "h264-parser");

decoder = gst_element_factory_make ("nvdec_h264", "nvh264-decoder");

pgie = gst_element_factory_make ("nvinfer", "primary-nvinference-engine");
…
```

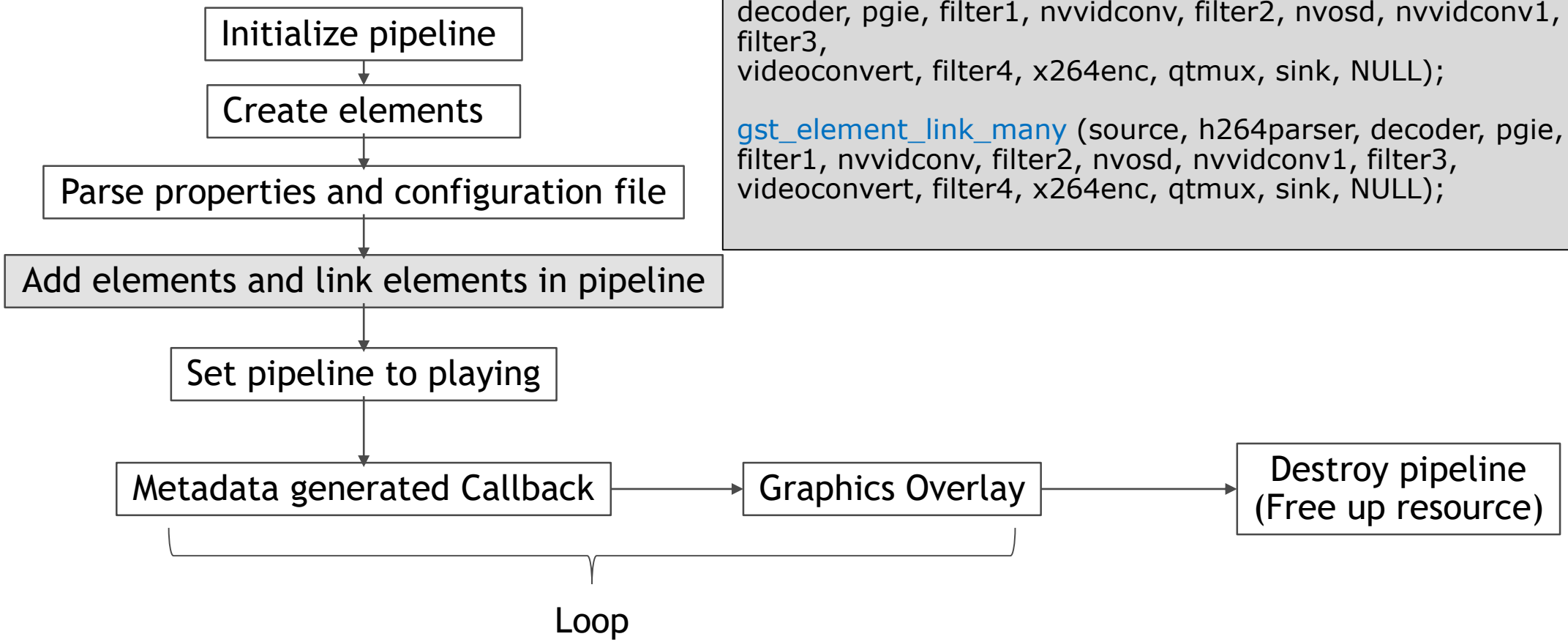


APPLICATION FLOW

```
g_object_set(G_OBJECT (source) , "location", VIDEO_PATH, NULL)
g_object_set(G_OBJECT (pgie), "config-file-path" , CONFIG_PATH ,NULL)
...
```



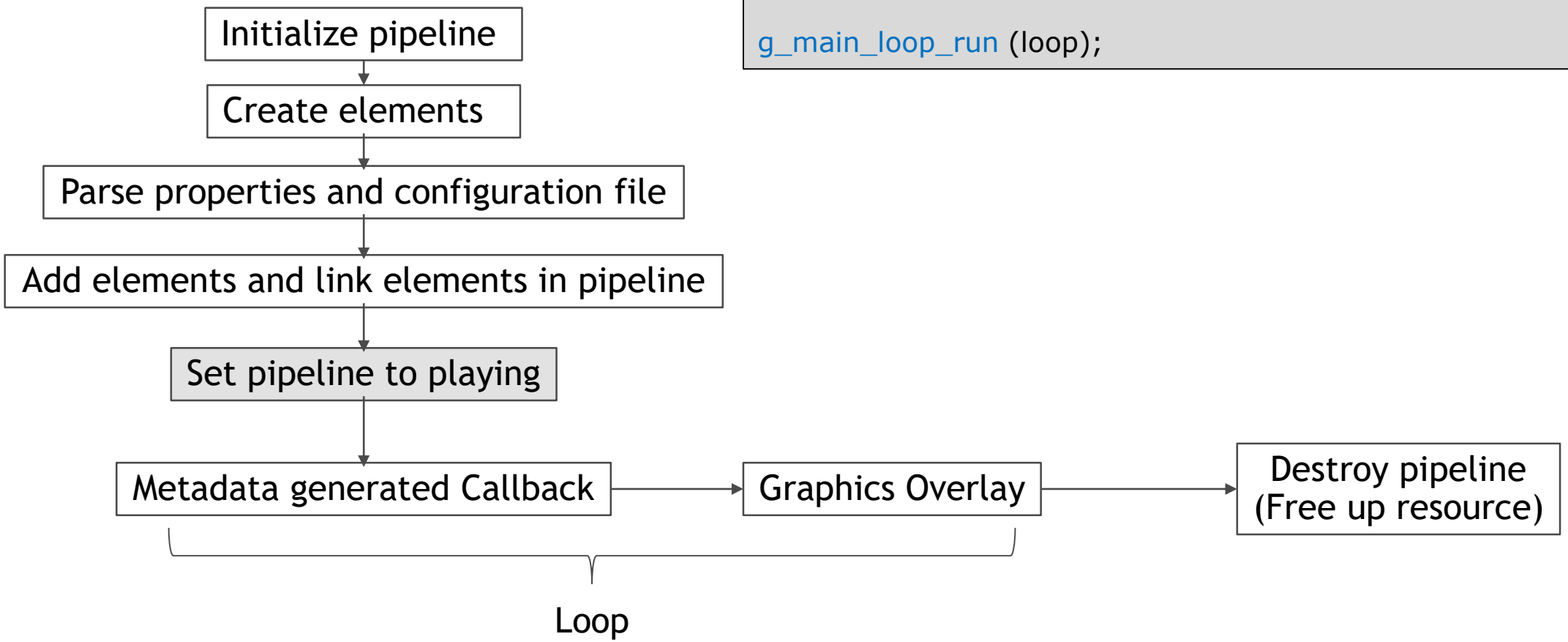
APPLICATION FLOW




```
gst_bin_add_many (GST_BIN (pipeline), source, h264parser,
decoder, pgie, filter1, nvvidconv, filter2, nvosd, nvvidconv1,
filter3,
videoconvert, filter4, x264enc, qtmux, sink, NULL);

gst_element_link_many (source, h264parser, decoder, pgie,
filter1, nvvidconv, filter2, nvosd, nvvidconv1, filter3,
videoconvert, filter4, x264enc, qtmux, sink, NULL);
```

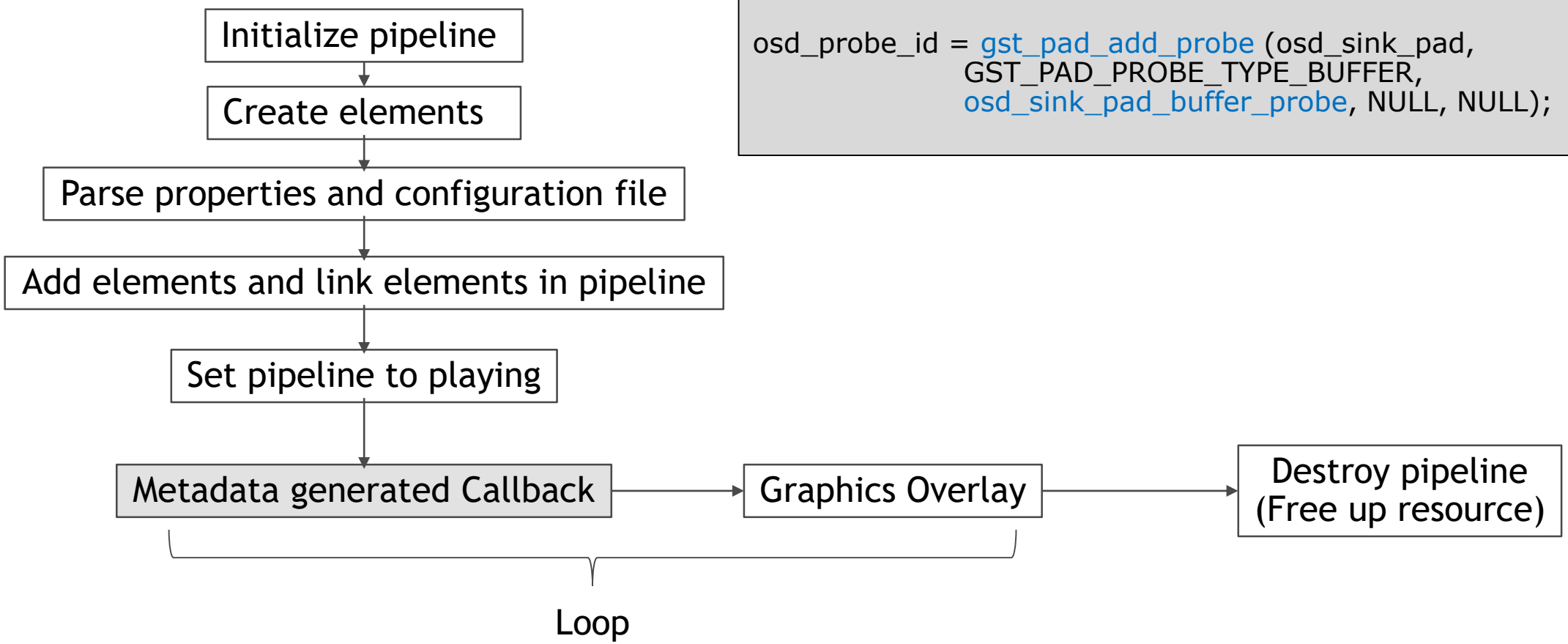
APPLICATION FLOW

```
gst_element_set_state (pipeline, GST_STATE_PLAYING);

g_main_loop_run (loop);
```



APPLICATION FLOW

```
osd_sink_pad = gst_element_get_static_pad (nvosd, "sink");

osd_probe_id = gst_pad_add_probe (osd_sink_pad,
                    GST_PAD_PROBE_TYPE_BUFFER,
                    osd_sink_pad_buffer_probe, NULL, NULL);
```



BUILDING PIPELINES AT THE COMMAND LINE

gst-launch-1.0

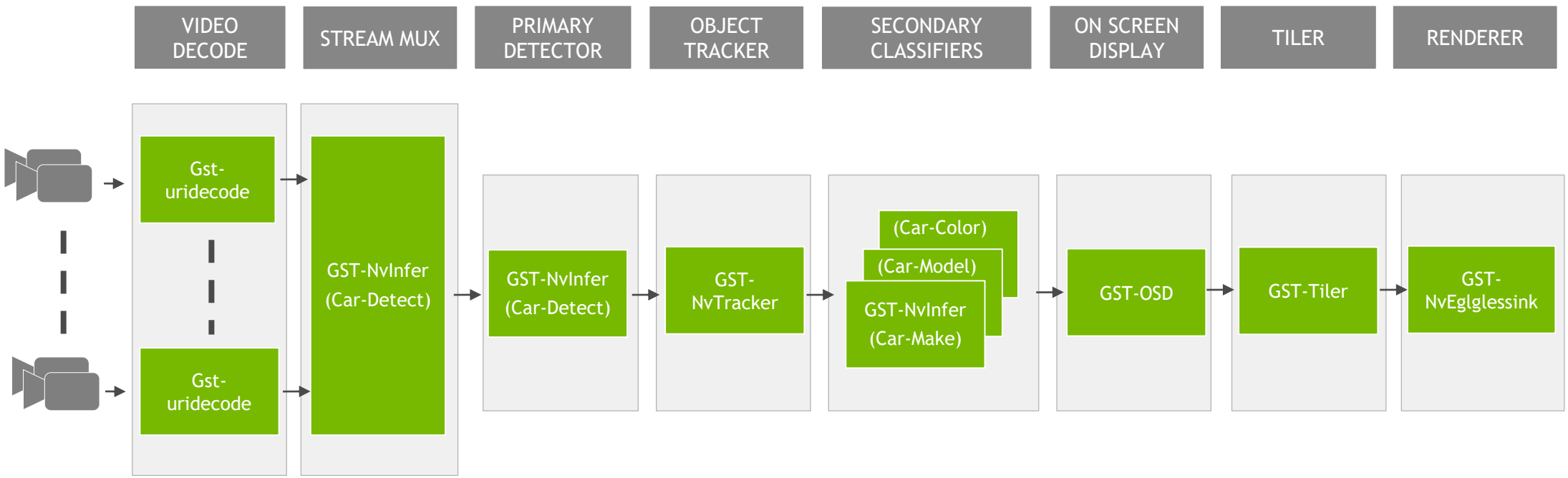
- Gstreamer provides powerful tool like gst-launch to create trial experimental graphs
  - builds and runs the pipeline using {gst-launch-1.0 [OPTIONS] PIPELINE-DESCRIPTION}
  - 1) File stream with Primary object detection and OnScrean Display

```
$gst-launch-1.0 uridecodebin uri=file:///home/nvidia/video.mp4 ! nvinfer <primary-infer-properties> !
queue ! nvosd <osd-properties> ! nveglglessink
```

  - 2) RTSP stream with primary object detection + tracking + secondary classification labels + OnScreen Display

```
$gst-launch-1.0 uridecodebin uri=rtsp://10.24.1.1/video0 ! nvinfer <primary-infer-properties> ! queue !
nvtracker ! queue ! nvinfer >secondary-infer-properties> ! queue ! nvosd <osd-properties> ! nveglglessink
```

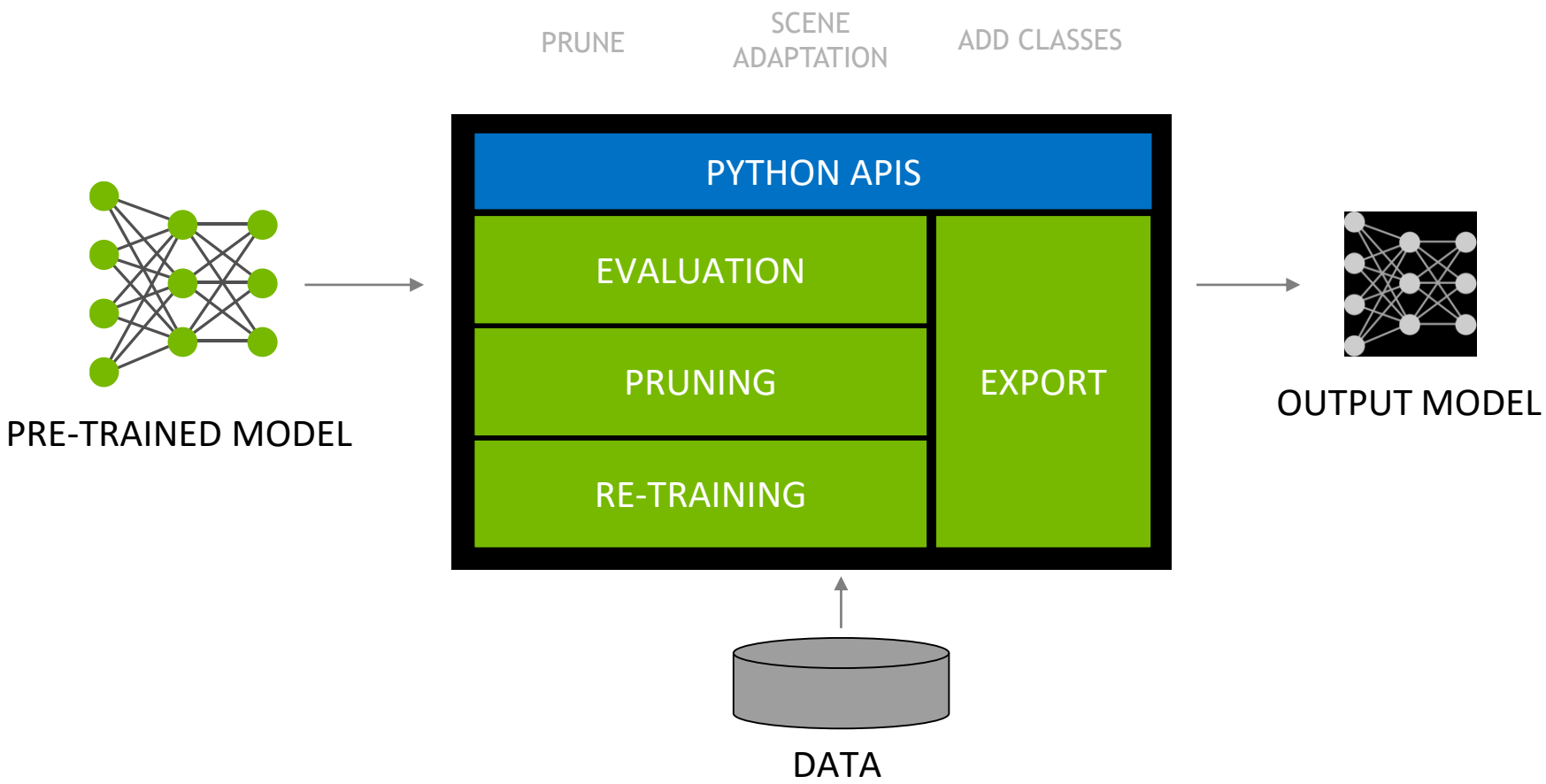
MULTI-STREAM REFERENCE APPLICATION



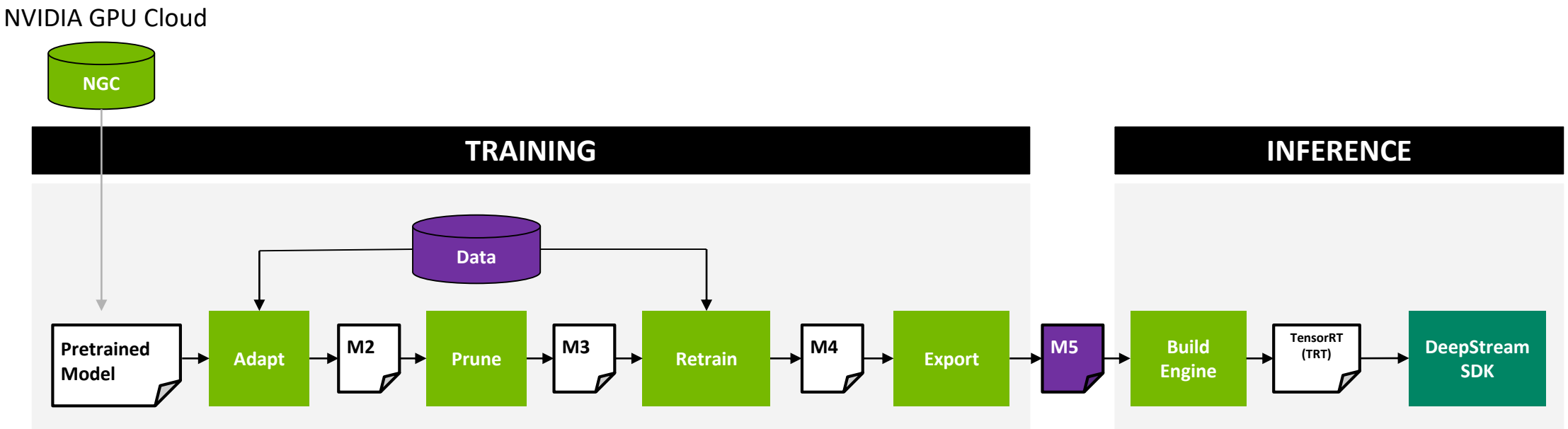
REFERENCE APPLICATION VIDEO

TRANSFER LEARNING TOOLKIT FOR IVA

TRANSFER LEARNING TOOLKIT



TRANSFER LEARNING WORKFLOW



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

AVAILABLE MODELS

- Image classification
  - ResNet 18
  - ResNet 50
  - VGG 16
  - VGG 19
  - AlexNet
  - GoogLeNet
- Object Detection
  - ResNet50
  - VGG 16
  - GoogLeNet

STEP 1: DOWNLOAD MODEL

```
$tlt-pull --list_models --k $API_KEY

$tlt-pull --list_version --model_name $MODEL_NAME –k $API_KEY

$tlt-pull --model_name $MODEL_NAME --version $VERSION --k $API_KEY \
--dir ./path/to/save/model
```

```
Downloaded 42.16 MB in 4s, Download speed: 10.51 MB/s
--------------------------------------------------
Transfer id: tlt_iva_object_detection_resnet18_v1 Download status: Completed.
Downloaded local path: /tmp/tmpDwl1ST/tlt_iva_object_detection_resnet18_v1
Total files downloaded: 2
Total downloaded size: 42.16 MB
Started at: 2019-06-24 13:36:29.284327
Completed at: 2019-06-24 13:36:33.294006
Duration taken: 4s seconds
--------------------------------------------------
Finished downloading tlt_iva_object_detection_resnet18
```

STEP 2: DATASET CONVERT

```
$tlt-dataset-convert [-h] --d DATASET_EXPORT_SPEC \
--o OUTPUT_FILENAME [-f VALIDATION_FOLD] [-v]
```

- Input size: 3 x W x H, where W>=480, H>= 272, and W and H are multiples of 16
  - if using pretrained weights, the input size should be 3 x 1248 x 384
- Image format: JPG, JPEG, PNG
- Label format: KITTI detection

STEP 3: TRAINING

Model configuration and modify labels

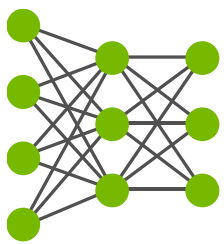
```
random_seed: 42
model_config {
  template: "resnet"
  num_layers: 18
  use_pooling: False
  use_batch_norm: True
  dropout_rate: 0.0
  training_precision: {
    backend_floatx: FLOAT32
  }
  objective_set: {
    cov {}
    bbox {
      scale: 35.0
      offset: 0.5
    }
  }
}
```

```
training_config {
  batch_size_per_gpu: 32
  num_epochs: 20
  learning_rate {
    soft_start_annealing_schedule {
      min_learning_rate: 5e-6
      max_learning_rate: 5e-4
      soft_start: 0.1
      annealing: 0.7
    }
  }
…
```

```
item {
    name: "/m/01g317"
    id: 1
    display_name: "person"
}
item {
    name: "/m/0199g"
    id: 2
    display_name: "bicycle"
}

item {
    name: "/m/0k4j"
    id: 3
    display_name: "motocycle"
}

…
```

ADDING A NEW CLASS

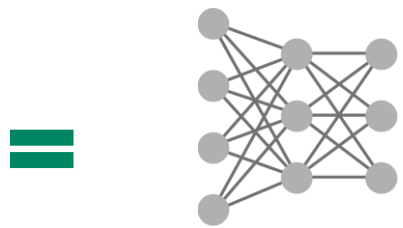
Easy to edit models to add new class



Pre-trained 6 class GoogleNet classifying sedan,
suv, truck, van, coupe, large vehicles

Add Emergency vehicle as another class simply by editing the spec file

Train, Prune and Retrain new model using new spec file



New network with 7 classes classifying sedan,
suv, truck, van, coupe, large vehicles, emergency vehicles

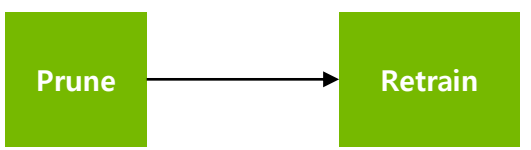
```
$tlt-train classification -e new_spec -r results -m model_file -n model_name

$tlt-train detection –e new_spec –r results –m model_file –n model_name
```

STEP 3: STARTING THE TRAINING PROCESS

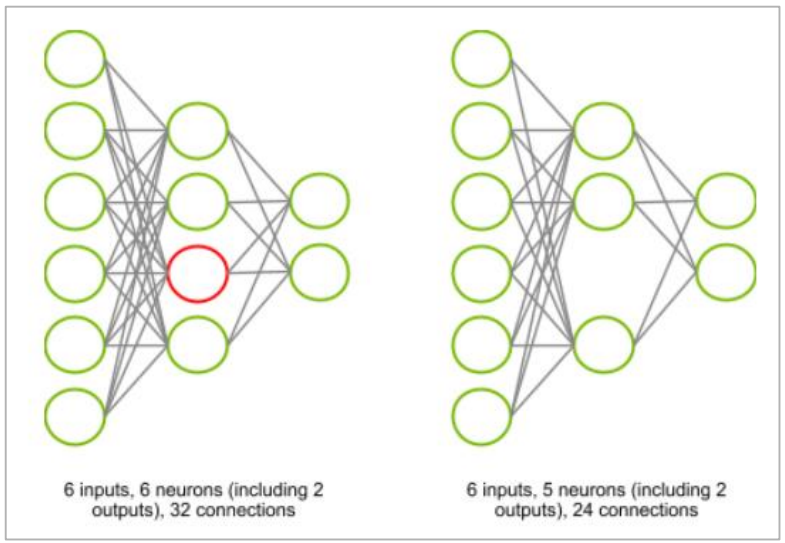
```
INFO:tensorflow:loss = 0.06663409, epoch = 0.0, step = 0
INFO:tensorflow:Saving checkpoints for step-1.
INFO:tensorflow:loss = 0.06506028, epoch = 0.0029585798816568047, step = 1 (41.655 sec)
INFO:tensorflow:loss = 0.063239686, epoch = 0.008875739644970414, step = 3 (13.557 sec)
INFO:tensorflow:loss = 0.049856987, epoch = 0.05917159763313609, step = 20 (5.510 sec)
INFO:tensorflow:global_step/sec: 0.423988
INFO:tensorflow:loss = 0.033803187, epoch = 0.10059171597633136, step = 34 (17.995 sec)
INFO:tensorflow:loss = 0.019279372, epoch = 0.15088757396449703, step = 51 (5.580 sec)
INFO:tensorflow:global_step/sec: 3.06484
INFO:tensorflow:loss = 0.011884072, epoch = 0.20118343195266272, step = 68 (5.502 sec)
INFO:tensorflow:loss = 0.008009276, epoch = 0.2514792899408284, step = 85 (5.614 sec)
INFO:tensorflow:global_step/sec: 3.00939
INFO:tensorflow:loss = 0.0070827967, epoch = 0.29881656804733725, step = 101 (5.356 sec)
INFO:tensorflow:loss = 0.006678746, epoch = 0.34615384615384615, step = 117 (5.457 sec)
INFO:tensorflow:global_step/sec: 2.96071
INFO:tensorflow:loss = 0.0059694266, epoch = 0.39349112426035504, step = 133 (5.362 sec)
INFO:tensorflow:loss = 0.0062045157, epoch = 0.4408284023668639, step = 149 (5.447 sec)
INFO:tensorflow:loss = 0.005166124, epoch = 0.4881656804733728, step = 165 (5.329 sec)
INFO:tensorflow:global_step/sec: 2.97297
INFO:tensorflow:loss = 0.005466017, epoch = 0.5355029585798816, step = 181 (5.447 sec)
INFO:tensorflow:loss = 0.004121407, epoch = 0.5857988165680473, step = 198 (5.576 sec)
INFO:tensorflow:global_step/sec: 2.99357
```

PRUNING

1. Reduce model size and increase throughput
2. Incrementally retrain model after pruning to recover accuracy

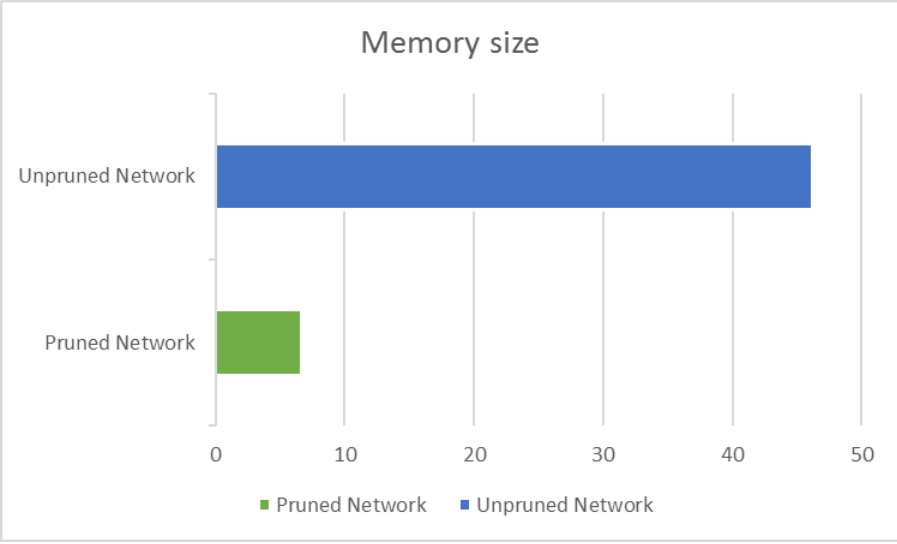


tlt-prune *tlt-train*

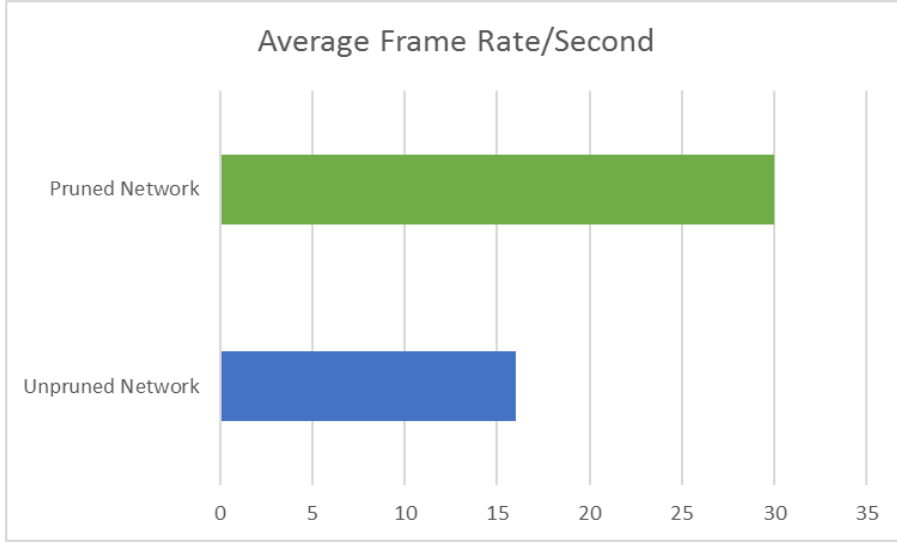


PRUNING EXAMPLE

6.5x reduction in model size



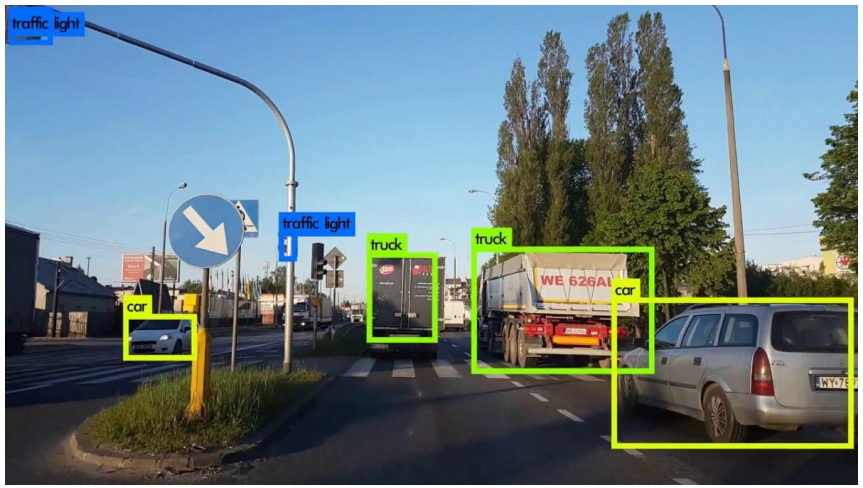
2x increase in throughput



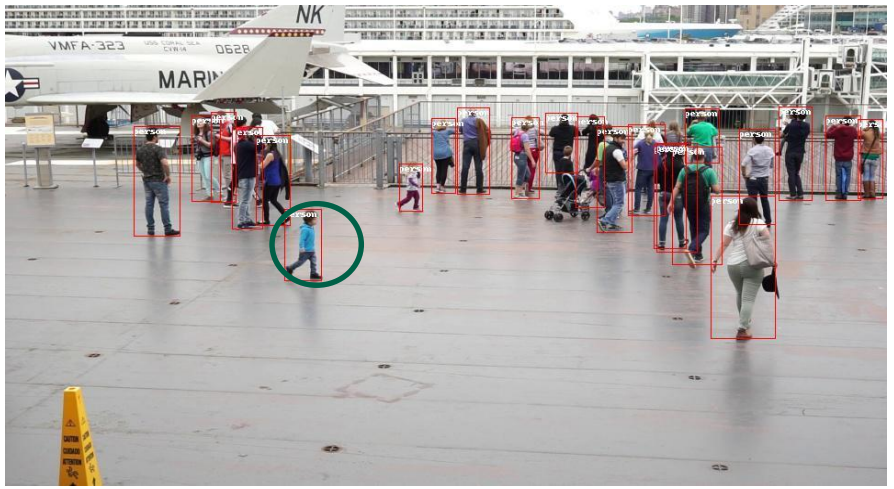
Network - ResNet18 4-class (Car, Person, Bicycle, Roadsign)
Memory size - 46.2 MB to 6.7 MB
FPS - 16fps to 30 fps

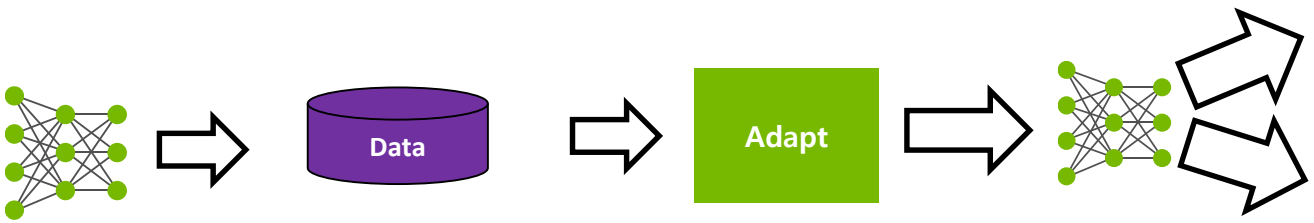
SCENE ADAPTATION

Camera location vantage point



Person with blue shirt





Same network adapting to different angles and vantage points

Same network adapting to new data

Train with new data from another vantage point, camera location, or added attribute

GETTING STARTED RESOURCES

NVIDIA NGC CONTAINERS

DeepStream

Publisher	Built By	Latest Tag	Modified	Size
NVIDIA	NVIDIA	3.0-18.11	March 19, 2019	1.18 GB

Description

DeepStream SDK delivers a complete toolkit for real-time situational awareness through intelligent video analytics (IVA). With hardware-accelerated building blocks, the application framework allows you to focus on building core deep learning networks and IP rather than designing end-to-end solutions from scratch.

Labels

Inference | Smart Cities | Toolkit

Pull Command

```
docker pull nvcr.io/nvidia/deepstream:3.0-18.11
```

https://ngc.nvidia.com/catalog/containers/nvidia:deepstream

Transfer Learning Toolkit for Video Streaming Analytics

Publisher	Built By	Latest Tag	Modified	Size
NVIDIA	NVIDIA	v0.3_py2	April 16, 2019	1.92 GB

Description

NVIDIA's Transfer Learning Toolkit is a python-based SDK that allows developers looking into faster implementation of industry specific Deep Learning solutions to leverage optimized, ready-to-use, pretrained models built in-house by NVIDIA. These pre-trained models accelerate the developer's deep learning training process and reduce higher costs associated with large scale data collection, labeling, and training models from scratch. This is an Early Access version, apply on the website for documentation and model access.

Labels

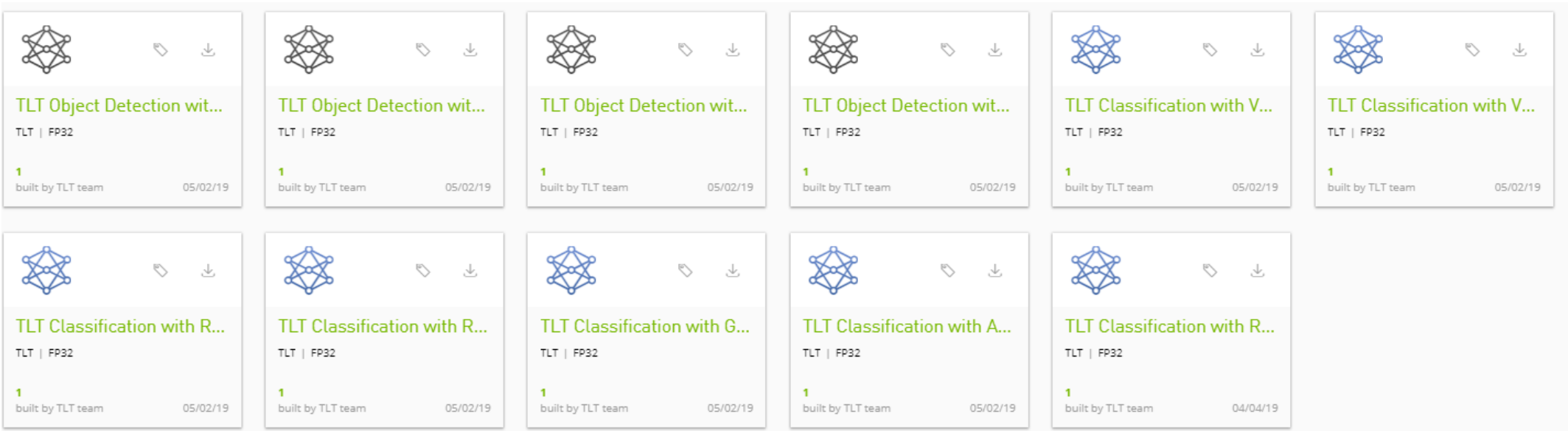
SDK | Smart Cities | Toolkit | Training

Pull Command

```
docker pull nvcr.io/nvidia/tlt-streamanalytics:v0.3_py2
```

https://ngc.nvidia.com/catalog/containers/nvidia:tlt-streamanalytics

NVIDIA NGC TLT PRETRAINED MODEL



https://ngc.nvidia.com/catalog/models

ONLINE RESOURCES FOR DEEPSTREAM

- Gstreamer Plugin and Application Development Guide
  - https://gstreamer.freedesktop.org/documentation/
- NVIDIA DeepStream SDK
  - Product Page
- Blogs
  - Breaking the Boundaries of Intelligent Video Analytics with DeepStream SDK 3.0
  - Multi-Camera Large-Scale Intelligent Video
  - Using Calibration to Translate Video Data to the Real World
- Webinar:
  - Streamline Deep Learning for Video Analytics with DeepStream SDK 2.0

ONLINE RESOURCES FOR TLT

- NVIDIA Transfer Learning Toolkit SDK
  - Product Page
- Blogs
  - Accelerating Intelligent Video Analytics with Transfer Learning Toolkit
  - Pruning Models with NVIDIA Transfer Learning Toolkit

LEARN MORE DURING AI CONFERENCE

- 15:40 ~ 16:20 Track1
  - GPU를 활용한 Image Augmentation 가속화 방안 – DALI by NVIDIA 한재근 과장
- 16:30 ~ 17:10 Track3
  - GPU Profiling 기법을 통한 Deep Learning 성능 최적화 기법 소개 by NVIDIA 홍광수 과장
- 17:20 ~ 18:00 Track2
  - 효율적인 Deep Learning 서비스 구축을 위한 핵심 애플리케이션 - NVIDIA TensorRT Inference Server by NVIDIA 정소영 상무
- NVIDIA DEMO BOOTH
  - Metropolis – Smart traffic/ Transportation


NVIDIA