GPU TECHNOLOGY CONFERENCE

September 13, 2016 | Beijing

NVIDIA AI
BRAIN OF SELF DRIVING AND HD MAPPING

September 13, 2016

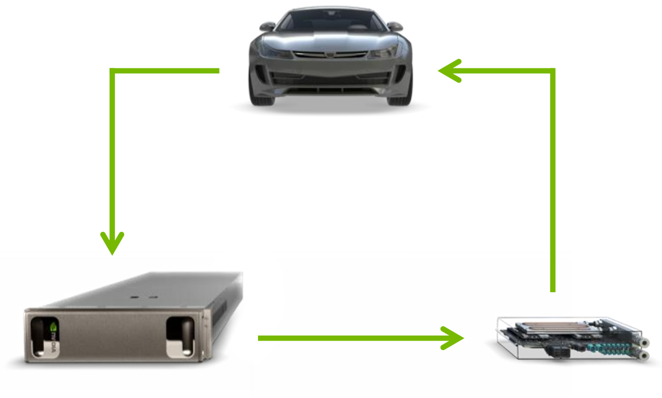
PRESENTED BY

NVIDIA.

AI FOR AUTONOMOUS DRIVING

Caffe
CNTK
KALDI
TensorFlow
theano
torch

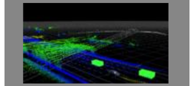
Training on DGX-1



NVIDIA DGX-1

NVIDIA DRIVE PX 2

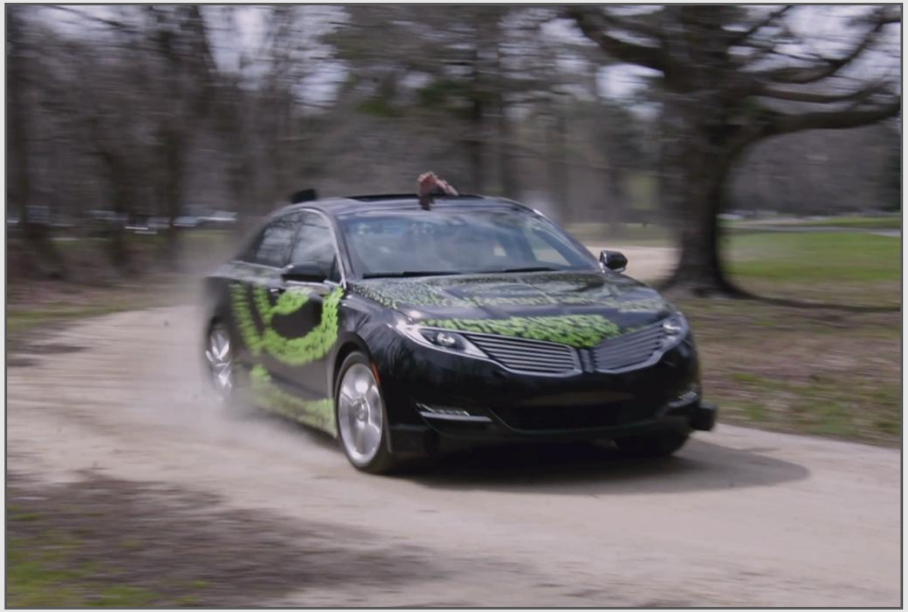
MAPPING



LOCALIZATION

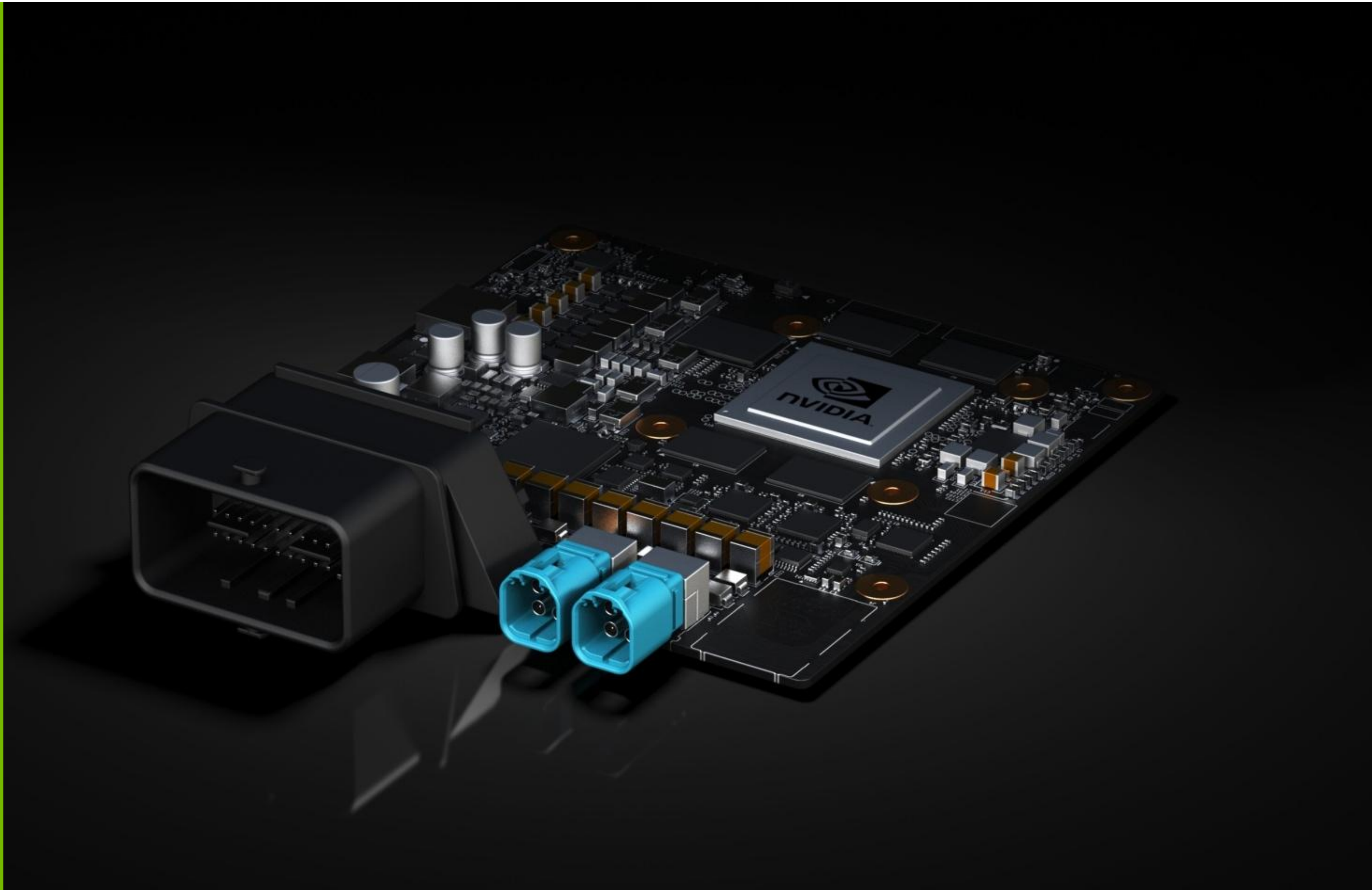
DRIVENET

Driving with DriveWorks



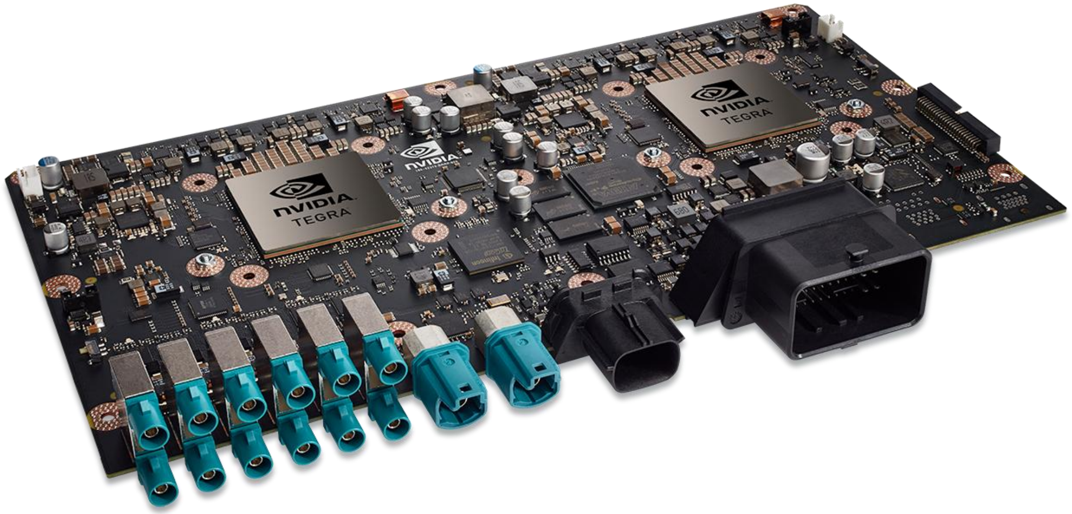
NVIDIA DRIVE PX 2 FOR AUTOCRUISE

- Tegra Parker SoC
  - 1.3 TFLOPS GPU
  - 6 CPU Cores
  - Integrated ISP
- 8 GB LPDDR4
- 64 GB eMMC
- 64 MB Boot ROM
- Automotive IO
- Connect & fuse data from cameras, LIDAR, radar, ultrasonic sensors
- Includes DriveWorks software & SDK
- 125 x 125 mm
- 10 W

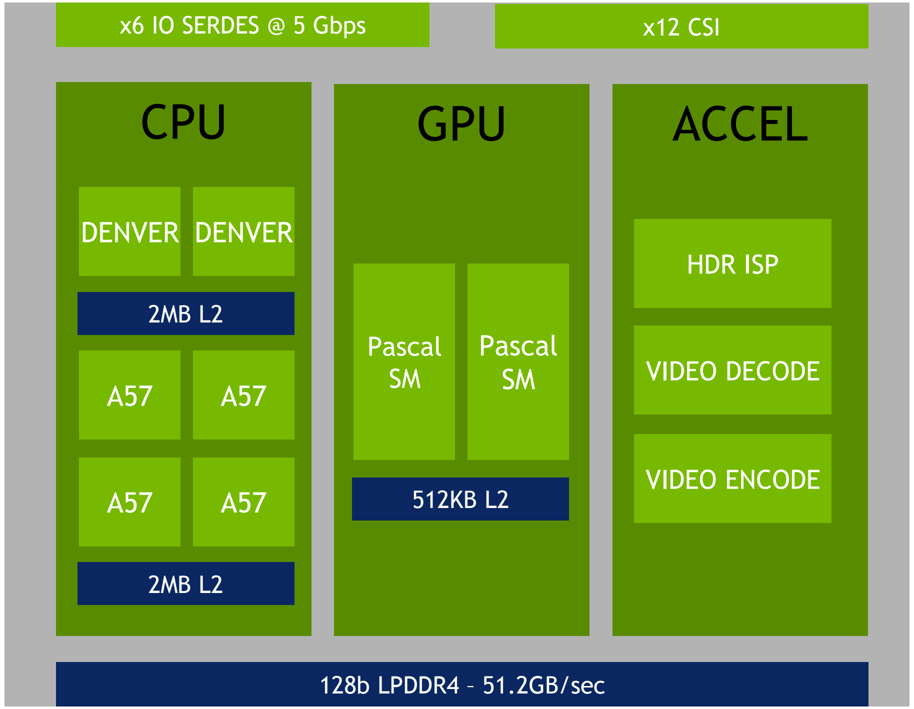


NVIDIA DRIVE PX 2 FOR AUTOCHAUFFEUR

- Processing Power
  - 2x Tegra Parker SoC
  - 2x Pascal dGPU
  - 8 TFLOPS
  - 24 DNN TOPs
- Connect & fuse data from up to 12 cameras, LIDAR, radar, ultrasonic sensors
- Includes DriveWorks software & SDK
- Platform for AI, part of deep learning system
- Available now



A PEEK INSIDE DRIVE PX 2
PARKER SOC



PERFORMANCE
Best in Class GPU & CPU
1.3 TFLOPS processing
1.5Gpix/s Native HDR ISP
Highest Memory BW and Efficiency
Lowest Sustained Power Consumption

ASIL-B SAFETY ARCHITECTURE
Integrated Safety Engine
Lock-Step R5 Cluster
Memory Error Correction

AUTOMOTIVE INTEGRATION
CAN and Ethernet AVB I/O
Up to 12 Camera Inputs
x6 IO SERDES up to 5Gbps

NVIDIA AI SELF-DRIVING CAR PLATFORM

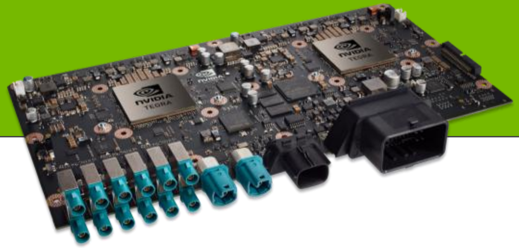
MAPPING AI

PERCEPTION AI

LOCALIZATION AI/CV

DRIVING AI

NVIDIA DRIVEWORKS OS



CLOUD MAP

+

AI ALGORITHMS

+

AI SUPERCOMPUTER

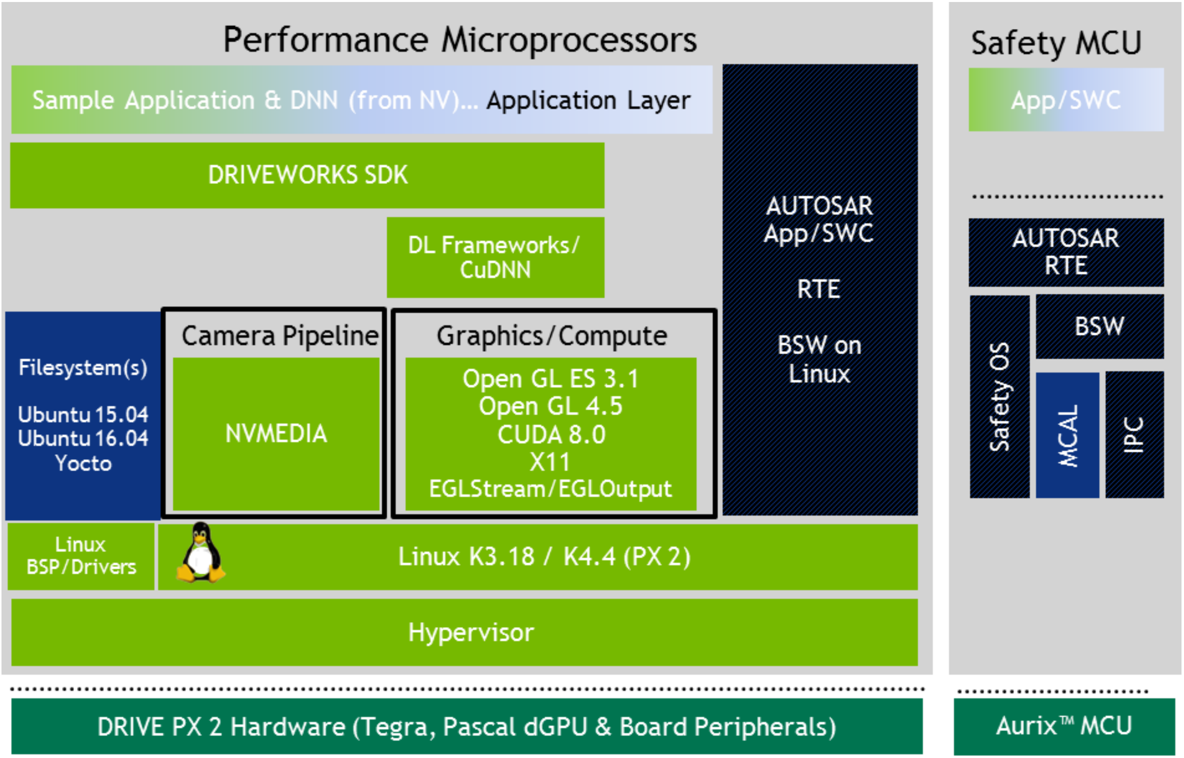
SOFTWARE

A full stack of rich software components

NVIDIA Vibrante Linux & Comprehensive BSP

Rich Middleware

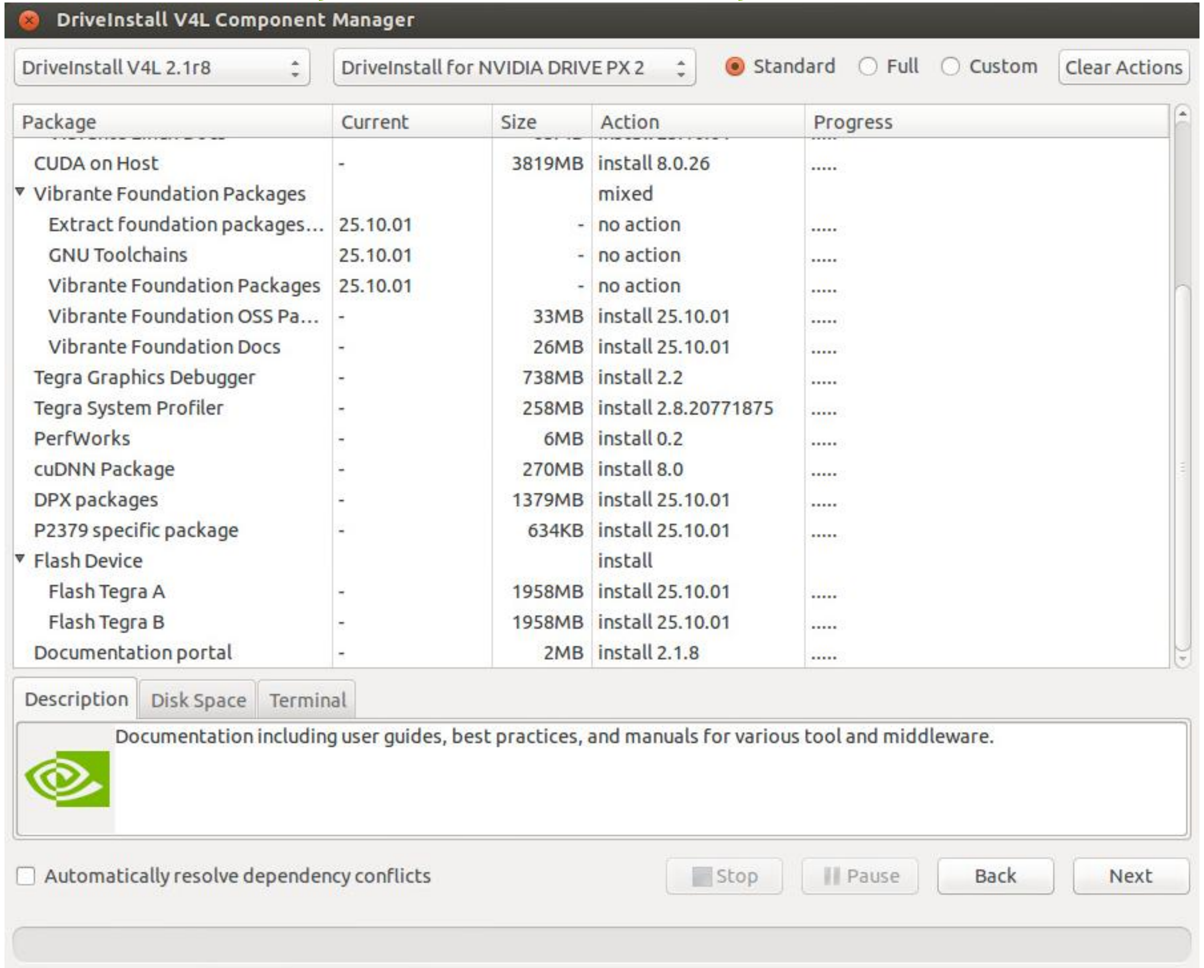
SDK, Samples and more



T1/OEM SW · OS/3rd SW/HW · Elektrobit · NVIDIA Licensed SW · Drive PX Hardware

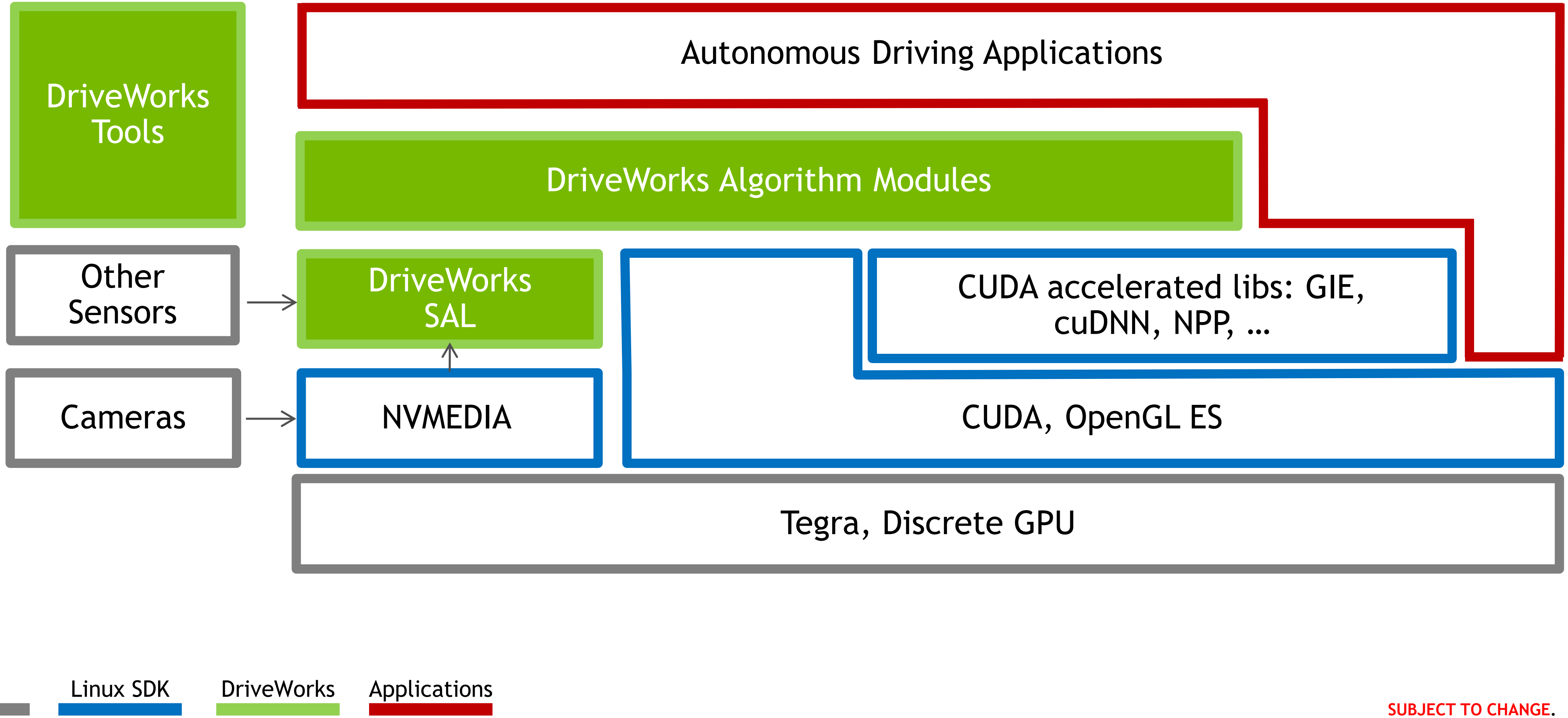
DRIVEINSTALL

An easy tool to flash your board



DRIVEWORKS SDK

SW Stack



SUBJECT TO CHANGE.

DRIVEWORKS TOOLS

CALIBRATION AND SENSOR REGISTRATION

Set of tools to calibrate sensors, and runtime module to perform online calibration

Features

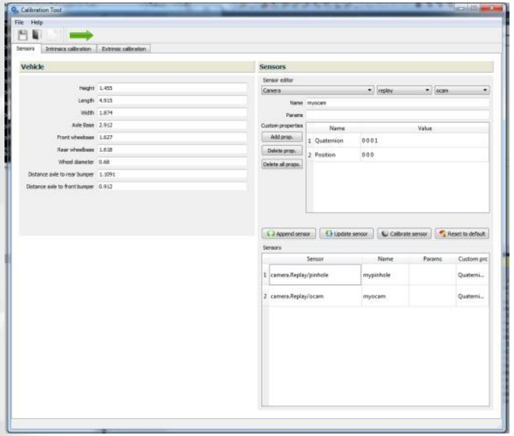
- Factory calibration tool (goal: zero stop calibration)
- Camera Intrinsic calibration – OCAM/Pinhole model. Pattern
- Camera Extrinsic calibration
    - Two cameras
    - 4-camera setup (surroundview config)
- Lidar to camera extrinsic calibration

- Online calibration
    - Recalibration of extrinsics only, with possible extension recalibrate intrinsics as well
    - Optimized bundle adjustment for automotive configurations

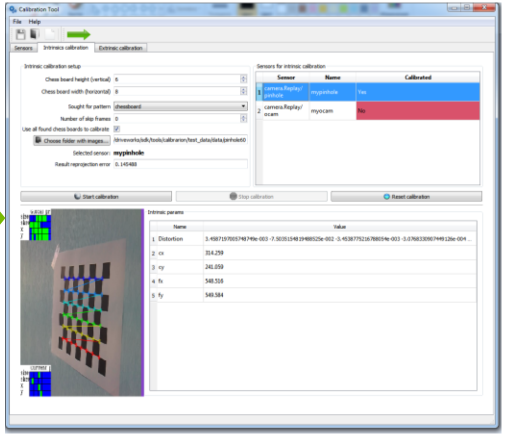
Modules
- Productized tools
- Patterns and tool for intrinsic calibration
- Patterns and library for extrinsic calibration
- Libraries for on rig calibration

CALIBRATION AND SENSOR REGISTRATION

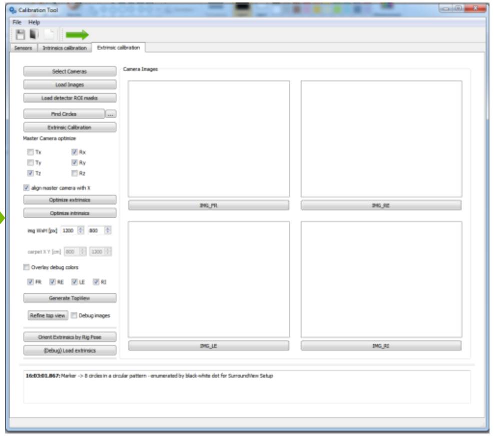
- Rig defines sensors and also rough location estimates
- Camera Intrinsics: OCAM and OpenCV Pinhole parameters
- Camera Extrinsics: 4 SurroundView or relative between 2 cameras
- Lidar Extrinsic: relative to a camera that sees the pattern



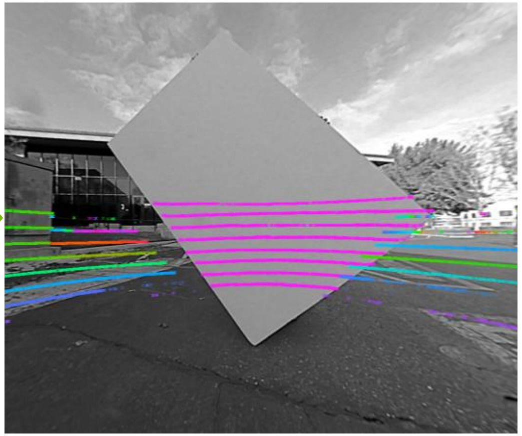
Rig Configuration



Camera Intrinsics



Camera Extrinsics



Lidar Extrinsics

TRACE CAPTURING AND REPLAY

Same platform and SW as both development and deployment

- Tuned performance to avoid glitches during capturing and recording
- Optimized for Load balancing threads and cores, memory and IO

Unique time synchronization protocol (PTP Aurix)

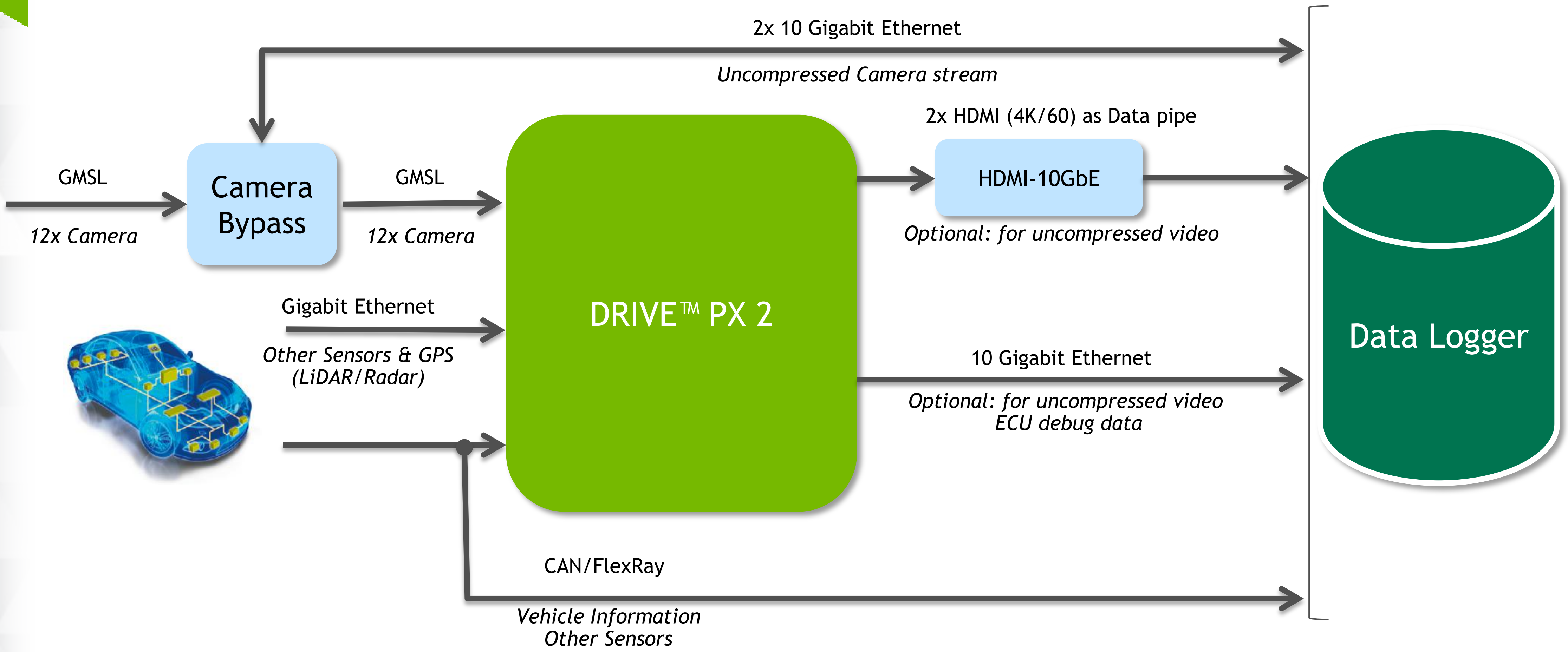
Single man operation:

- Support to launch multi-sensor recording at one key press
- Coordinated play/pause/stop for all sensors

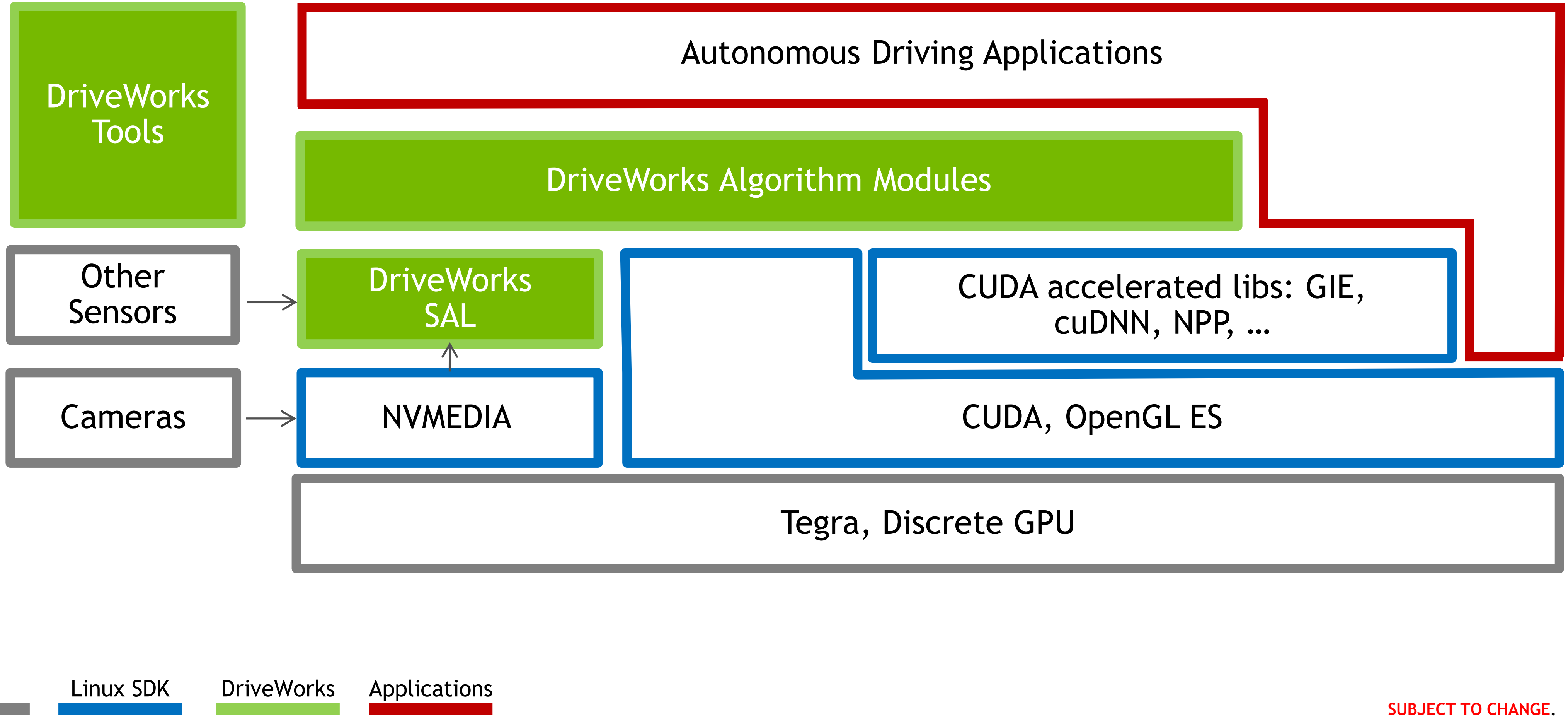
For future versions include

- Synchronization between multiple processes (different Tegras)
- Built-in calibration capabilities

DATA LOGGING

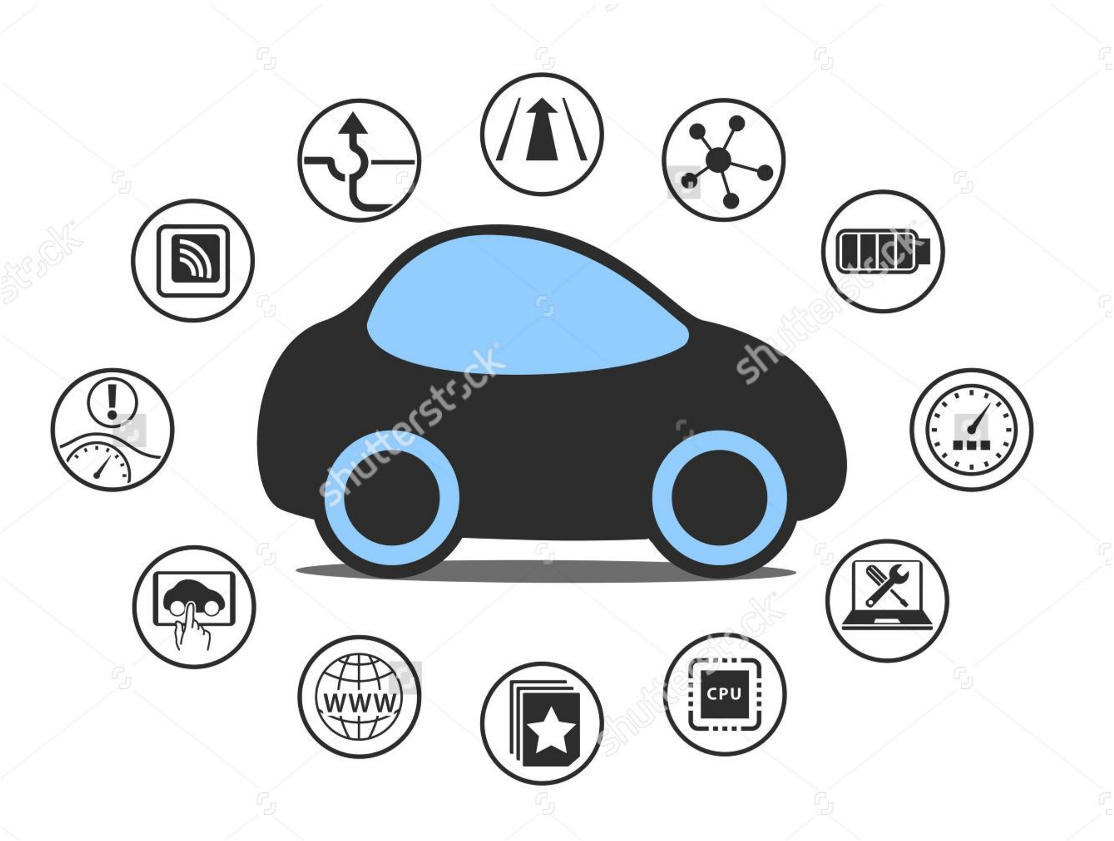


DRIVEWORKS SDK
SW Stack



SUBJECT TO CHANGE.

DRIVEWORKS SAL



SENSOR ABSTRACTION LAYER (SAL)

Goals

- Provide a common and simple unified interface to the sensors
- Provide both HW sensor abstraction as well as virtual sensors (for replay)
- Provide raw sensor serialization (for recording)
- Deal with platform and SW particularities
  - API/Processor Conversion/transfer: CUDA, GL, NvMedia, CPU
  - Exploit additional SoC engines: H264/H265 codec, VIC

COMMON SENSOR API

dwSAL_createSensor

Prepare sensor for data delivery: power up, establish connection, open socket, allocation FIFOs, etc...

dwSAL_startSensor

Start recording into sensor FIFO

Data consumption

Minimum Latency

dwSAL_stopSensor

Stop recording into sensor FIFO, drain FIFO.

dwSAL_releaseSensor

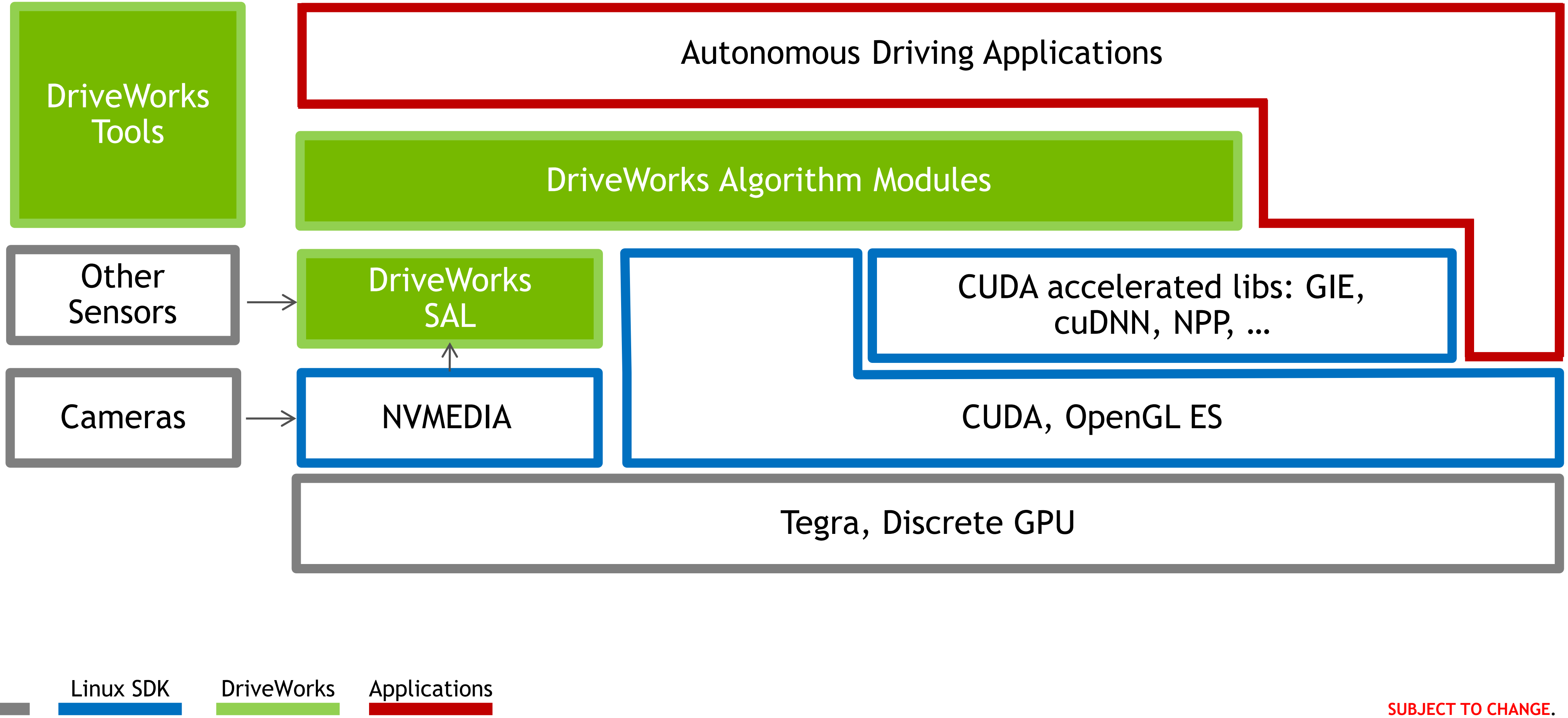
Shutdown sensor, release resources

SCHEDULING

- Current paradigm is non-blocking functions and blocking with timeout
- Defined by EGL, CUDA and NvMedia paradigms and capabilities
- Goal is event-driven and non-blocking data-flow model to be light-weight and efficient
  - Be able to schedule work ahead to hide latencies on triggering work for all our HW engines
  - Use as little threads as necessary to increase runtime determinism of the system

DRIVEWORKS SDK

SW Stack



SUBJECT TO CHANGE.

DRIVEWORKS SDK MODULES

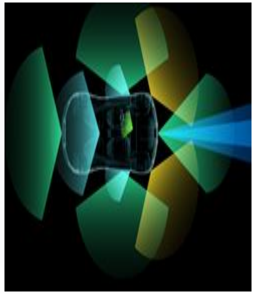
DRIVEWORKS SDK

Overview

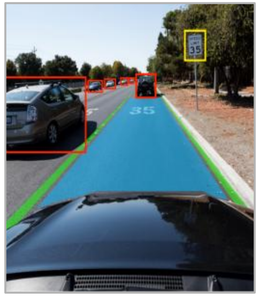
	DETECTION	LOCALIZATION	DRIVING	VISUALIZATION
DRIVEWORKS SDK	Detection/Classification	Map Localization	Vehicle Control	Streaming to cluster
	Sensor Fusion	HD-Map Interfacing	Scene understanding	ADAS rendering
	Segmentation	Egomotion (SFM, Visual Odometry)	Path Planning solvers	Debug Rendering
System SW	V4L/V4Q, CUDA , cuDNN, NPP, OpenGL, …			
Hardware	Tegra , dGPU			
Sensors	Camera, LIDAR, Radar, GPS, Ultrasound, Odometry, Maps			

SUBJECT TO CHANGE.

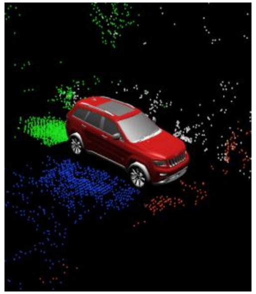
DRIVEWORKS ALGORITHM MODULES



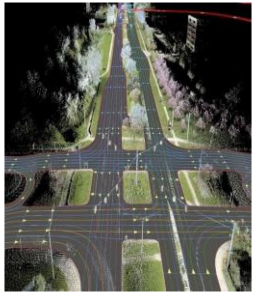
Sensor Fusion



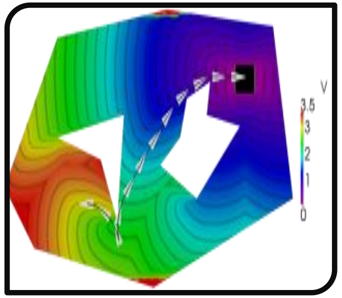
Detection



Locali-zation

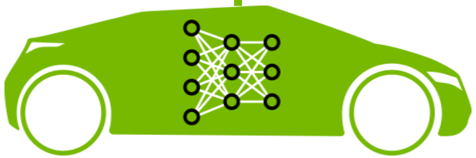


HD Maps

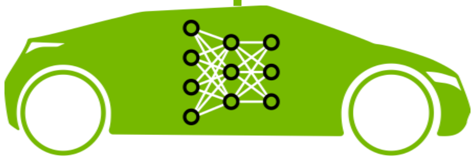


Planning

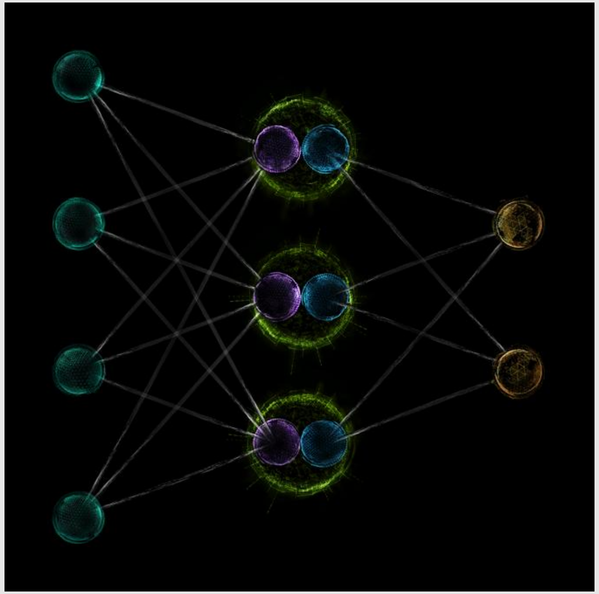
Driving



NVIDIA DRIVENET



TENSOR RT INFERENCE ENGINE

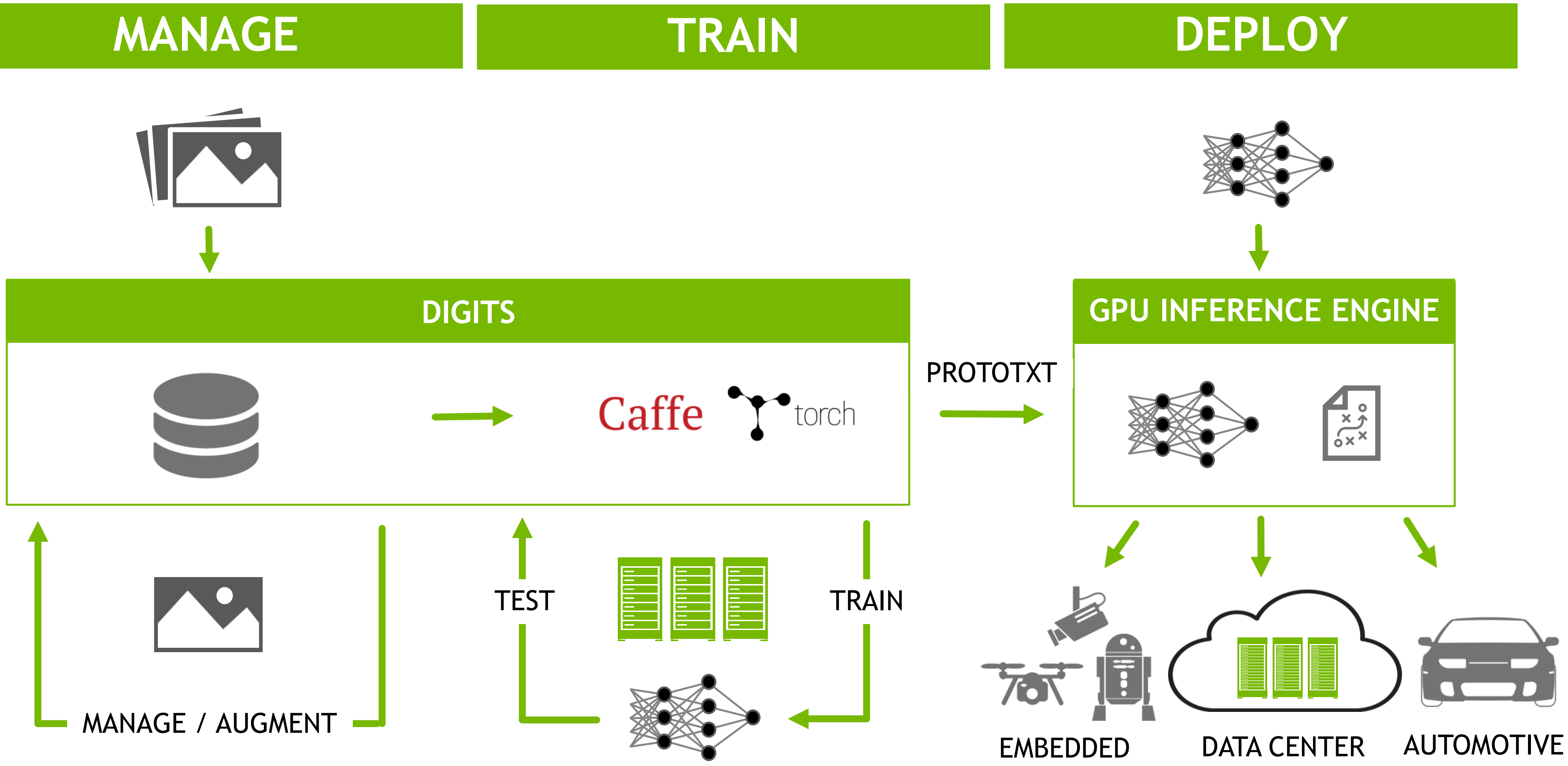


FP 32 / FP16 / INT8 | Vertical & Horizontal Fusion | Pruning

VGG, GoogLeNet, ResNet, AlexNet & Custom Layers

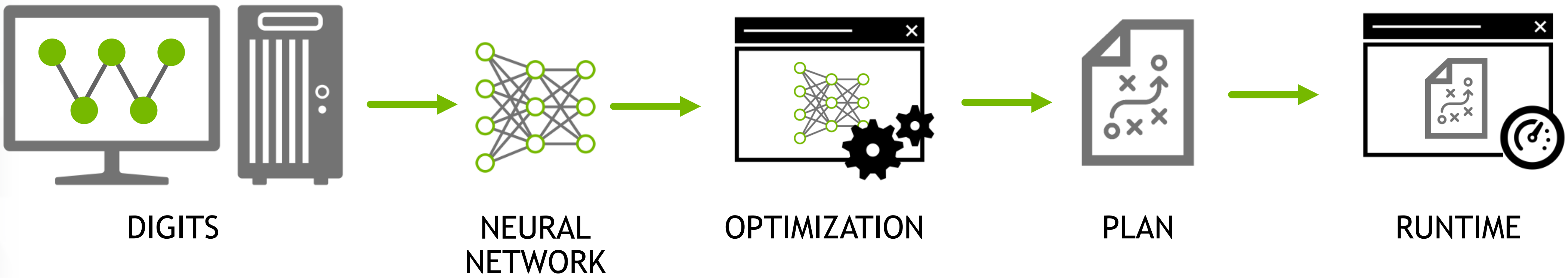
Available on DRIVE PX 2 Today

A COMPLETE DEEP LEARNING PLATFORM



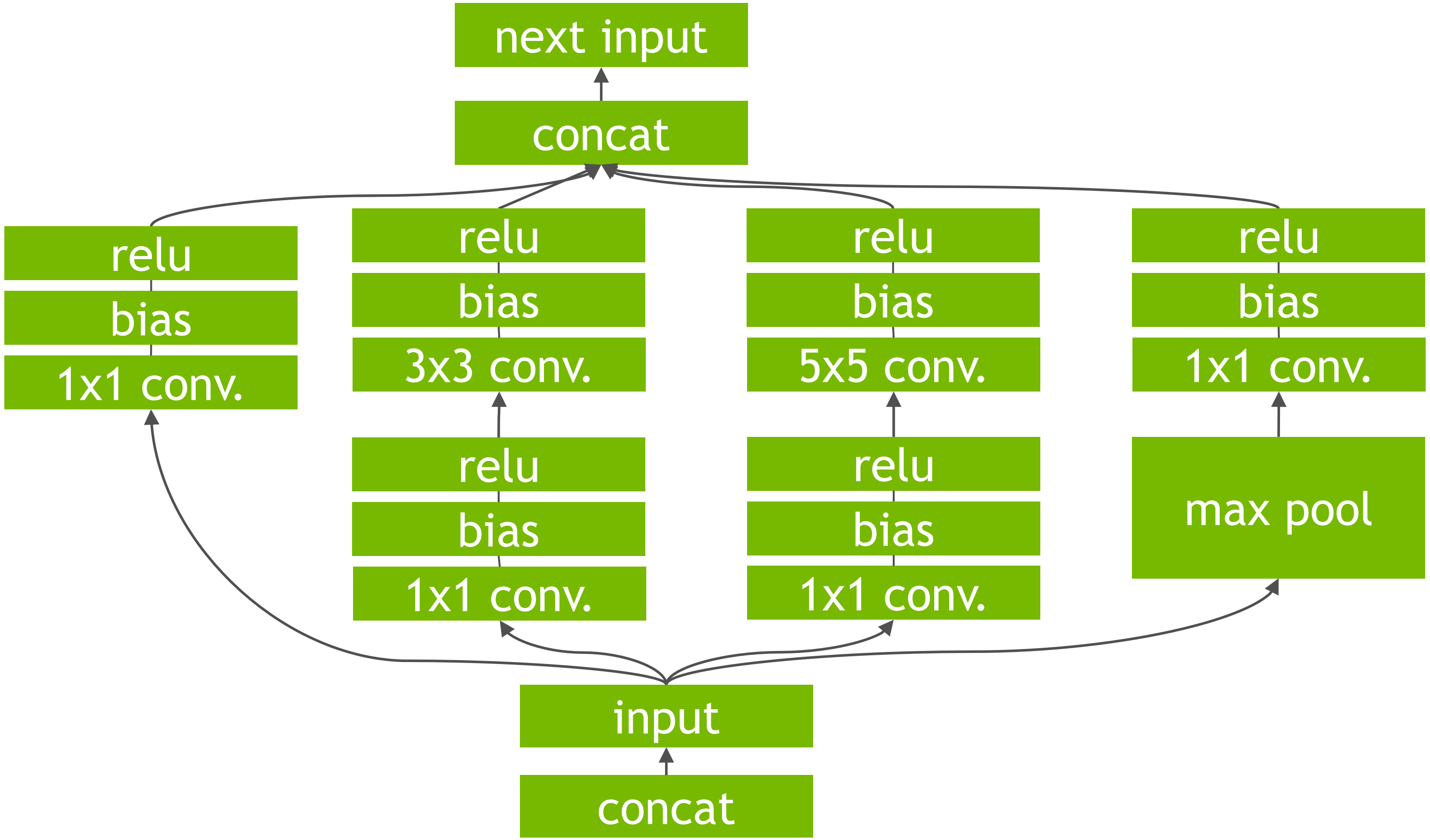
TENSOR RT INFERENCE ENGINE

Workflow



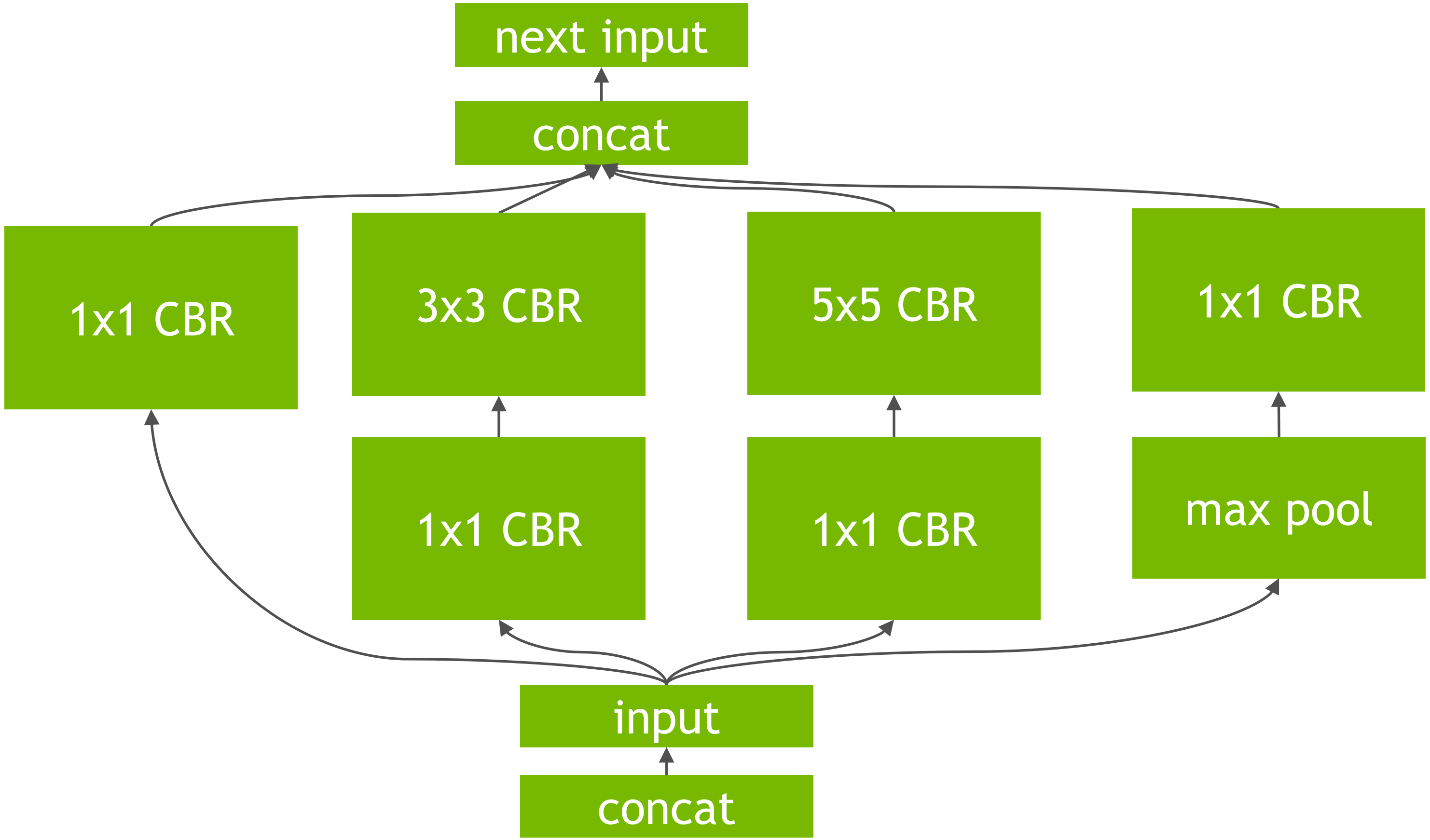
GRAPH OPTIMIZATION

Unoptimized network



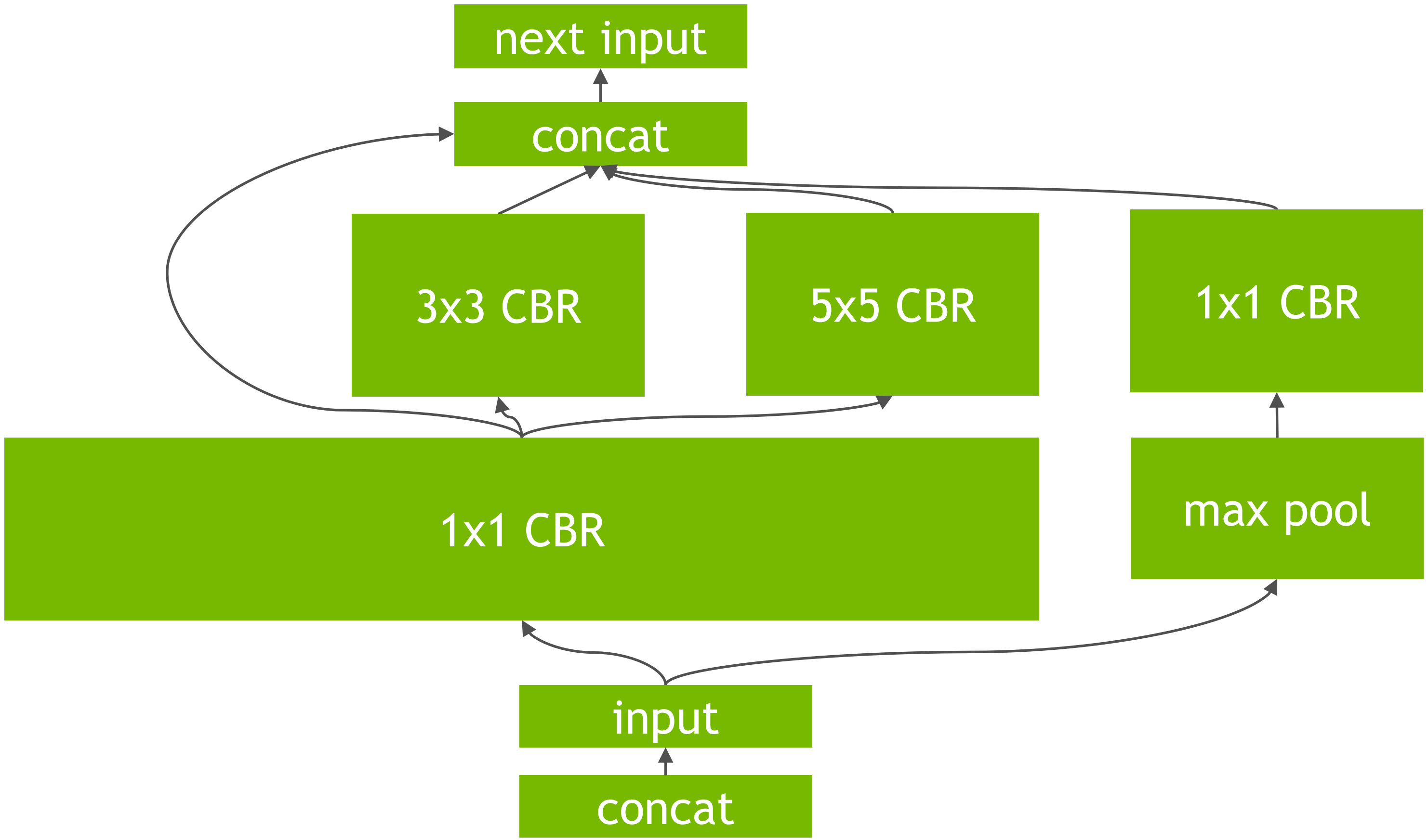
GRAPH OPTIMIZATION

Vertical fusion



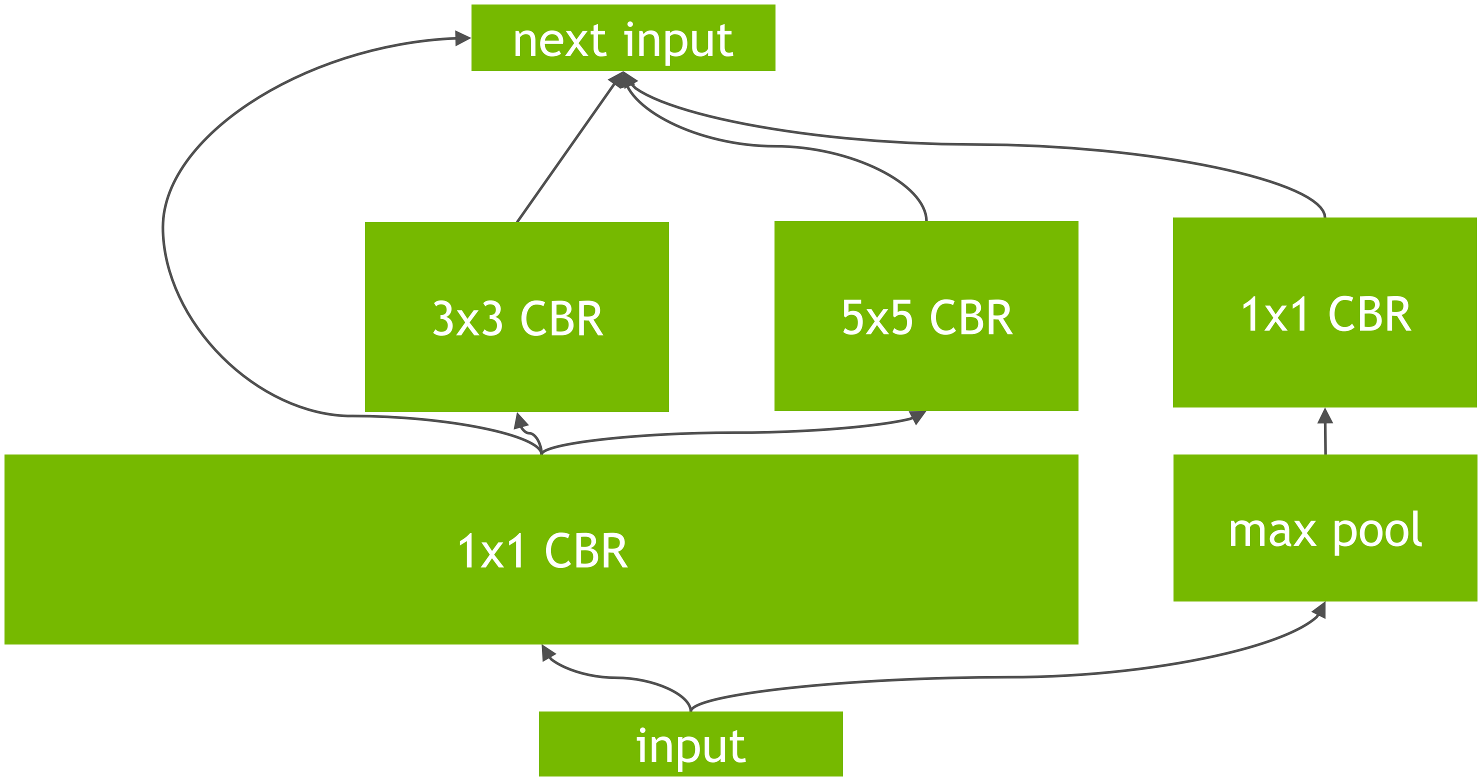
GRAPH OPTIMIZATION

Horizontal fusion



GRAPH OPTIMIZATION

Concat elision



BUILD

Importing a Caffe Model

```
// create a builder object
Ibuilder* builder = createInferBuilder(gLogger);

// create the network definition
INetworkDefinition* network = infer->createNetwork();

// populate the network definition and map
CaffeParser* parser = new CaffeParser;
IBlobNameToTensor *blobToTensor = parser->parse("net.prototxt",
        "net.weights", *network, DataType::kFLOAT); // or kHALF

// tell GIE which tensors are network outputs
for (auto& s : outputs)
    network->markOutput(*blobNameToTensor->find(s.c_str()));
```

BUILD

Engine Creation

```
// Specify the maximum batch size, scratch size, internal format
builder->setMaxBatchSize(maxBatchSize);
builder->setMaxWorkspaceSize(1 << 20);
builder->setHalf2Mode(true);

// create the engine, serialize to storage (C++ ostream)
ICudaEngine* builtEngine = builder->buildCudaEngine(*network);
builtEngine->serialize(storage);

// deserialize the engine from storage
IRuntime* runtime = createInferRuntime(gLogger);
ICudaEngine* engine = runtime->deserializeCudaEngine(storage);
```

RUNTIME

Running the Engine

```
// create an execution context for each engine instance
// Network weights are shared between contexts
IExecutionContext* context = engine->createExecutionContext();

// add GIE kernels to the given cuda stream
cudaEvent_t reuseInput;
context->enqueue(batchSize, buffers, stream, reuseInput);

<…>

// wait on the execution stream
cudaStreamSynchronize(stream);
```

RUNTIME

Caffe-free operation

- GIE is currently split into two libraries: libnvcaffeparser and libnvinfer
  - libnvinfer currently includes the builder and the runtime
- It’s possible to build/run networks without Caffe parser via C++ API
- Sample C++ API calls:
  - ITensor* in = network->addInput(“input”, DataType::kFloat, Dims4{…});
  - IPoolingLayer* pool = network->addPooling(in, PoolingType::kMAX, …);
  - pool->setStride(Dims2{2,2});
  - …