Applying Artificial Intelligence to Product Development

Arvind Jayaraman, Senior Application Engineering

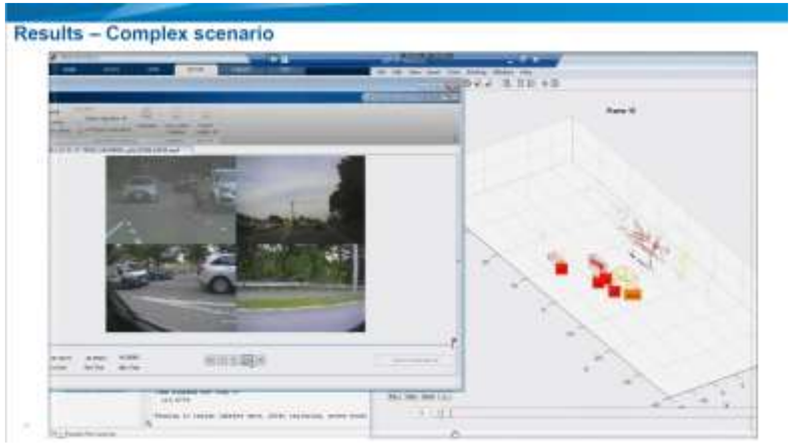
Diverse Set of Automotive Customers use MATLAB for AI

Caterpillar

Cloud Based Data Labeling

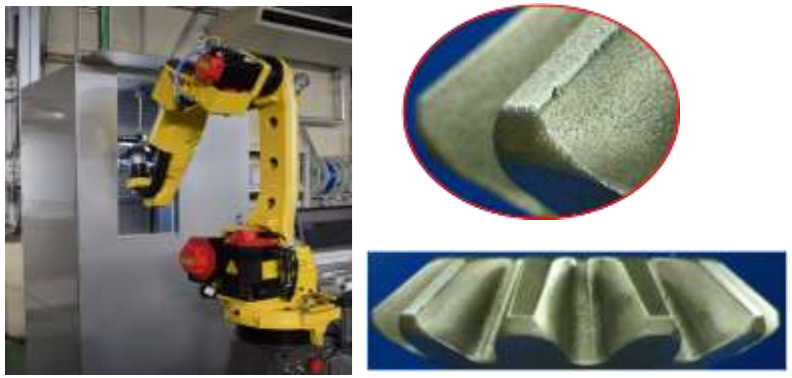
Alpine

Ground Detection



Veoneer

Radar Sensor Verification



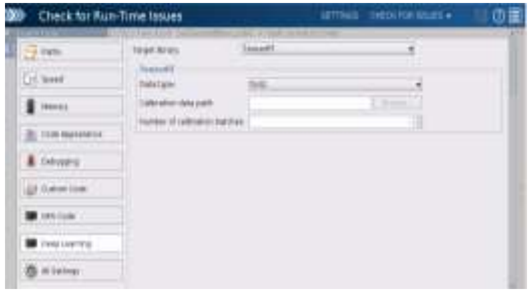
Musashi Seimitsu

Automotive Part Defect Detection

Outline

Ground Truth Labeling

Network Design and Training



CUDA and TensorRT Code Generation

Jetson Xavier and DRIVE Xavier Targeting

Key Takeaways

Platform Productivity: Workflow automation, ease of use

Framework Interoperability: ONNX, Keras-TensorFlow, Caffe

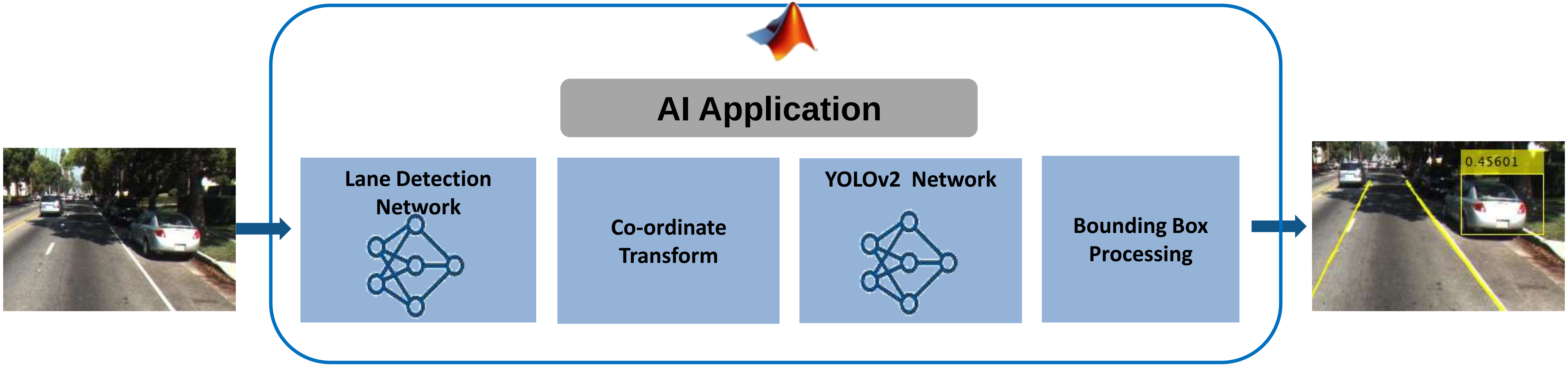
Key Takeaways

Optimized CUDA and TensorRT code generation

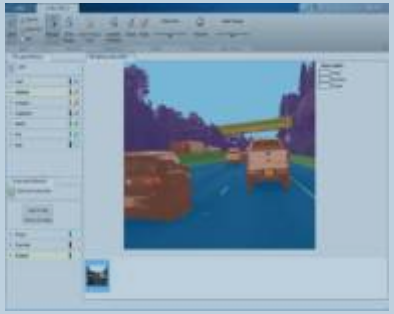
Jetson Xavier and DRIVE Xavier targeting

Processor-in-loop(PIL) testing and system integration

Example Used in Today’s Talk

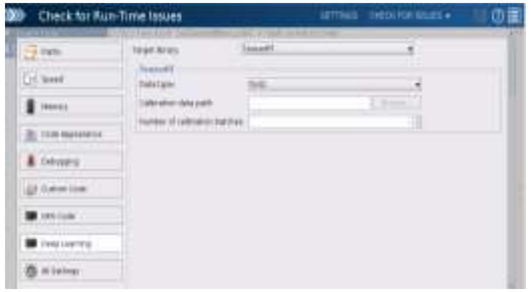


Outline



Ground Truth Labeling

Network Design and Training



CUDA and TensorRT Code Generation

Jetson Xavier and DRIVE Xavier Targeting

Unlabeled Training Data

Ground Truth Labeling

```
>> gTruth

gTruth =

  groundTruth with properties:

          DataSource: [1×1 groundTruthDataSource]
    LabelDefinitions: [4×3 table]
           LabelData: [250×4 timetable]

>> gTruth.LabelData

ans =

  250×4 timetable

        Time              Car          LaneMarker     Sunny    Shadow
    ____________    _____________    ____________    _____    ______

    0 sec           [2×4 double]     {2×1 cell}      true     false
    0.033333 sec    [2×4 double]     {2×1 cell}      true     false
    0.066667 sec    []               []              false    false
```

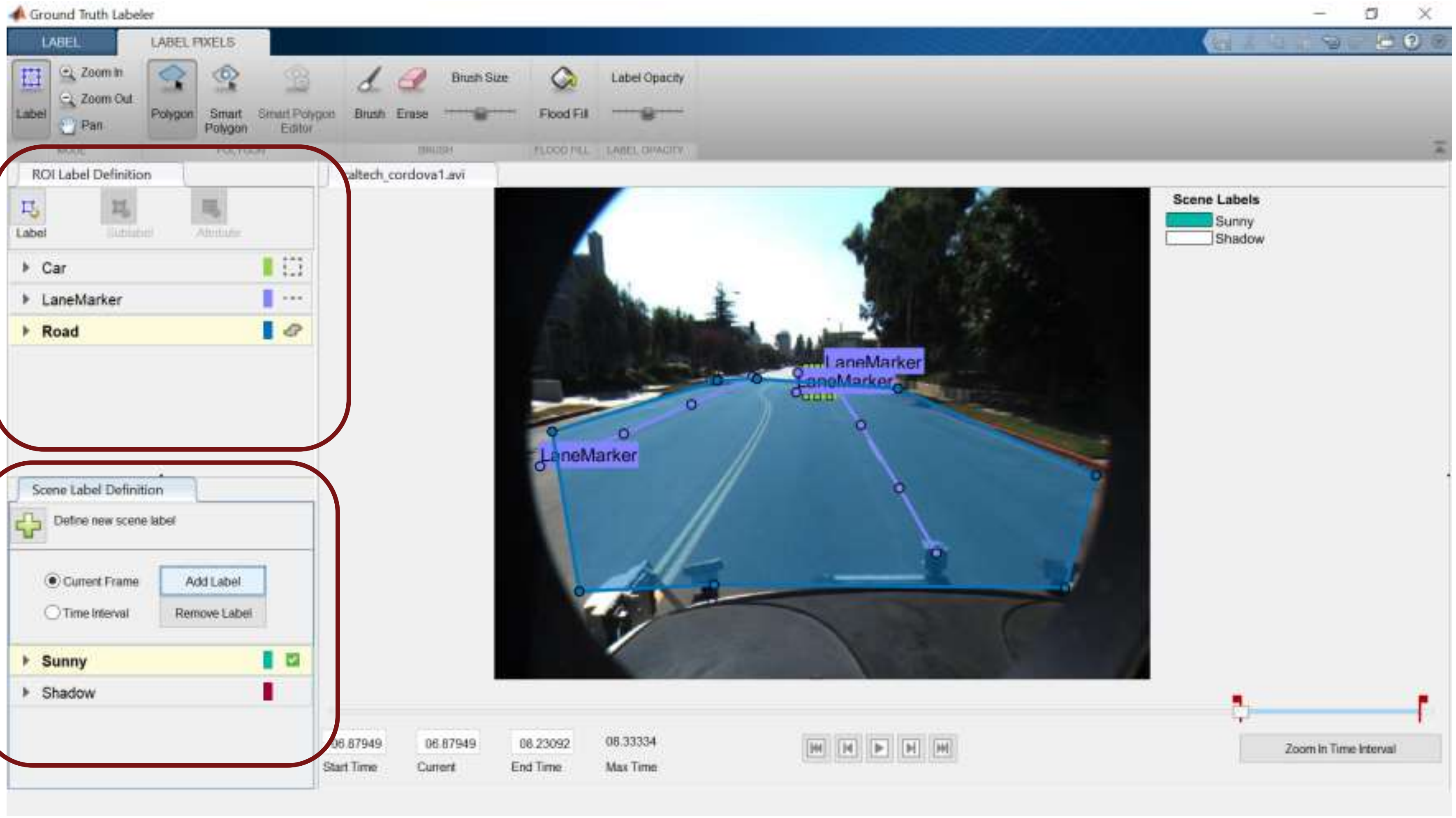
Labels for Training

Interactive Tools for Ground Truth Labeling

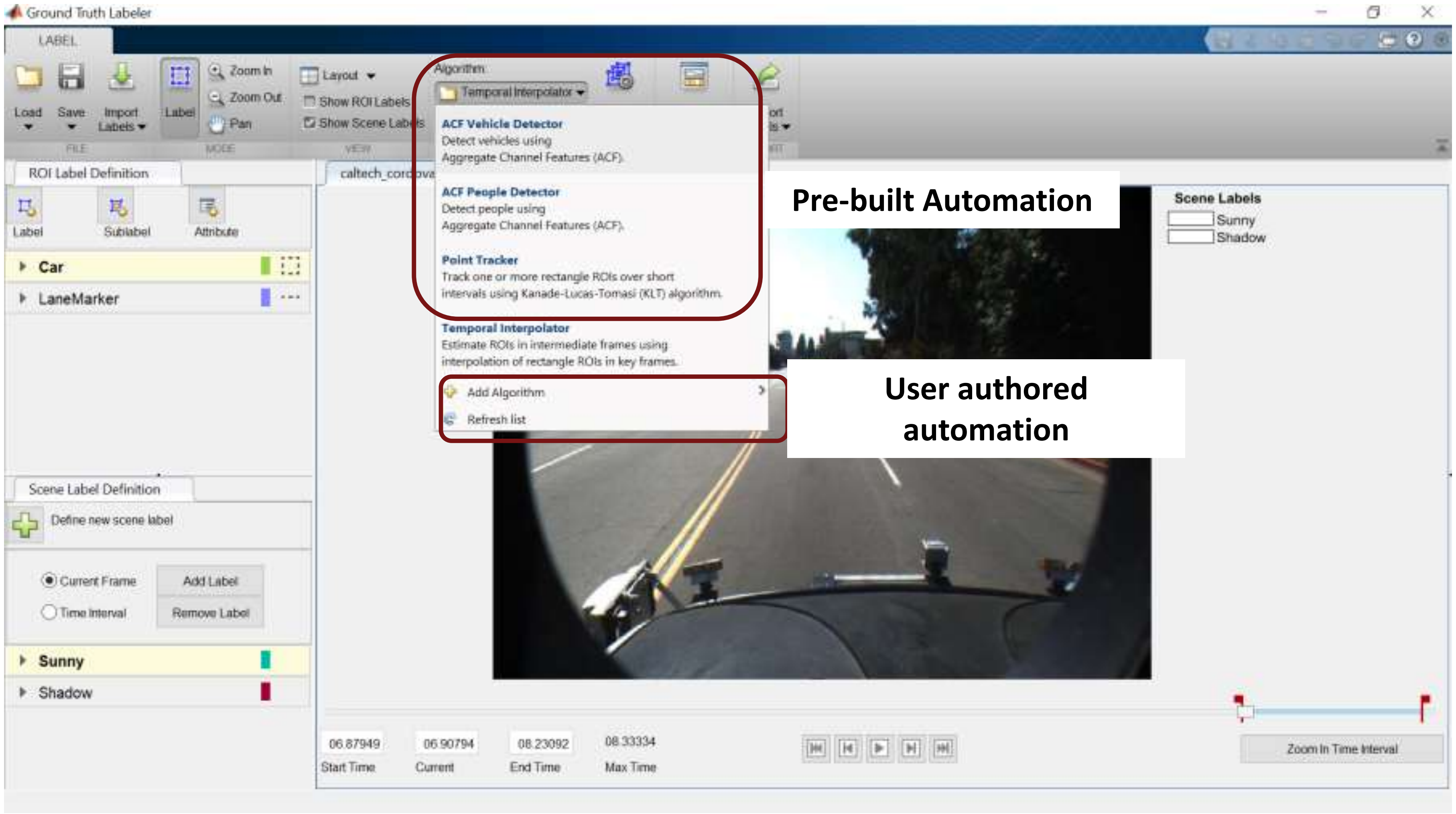
ROI Labels

- Bound boxes
- Pixel labels
- Poly-lines

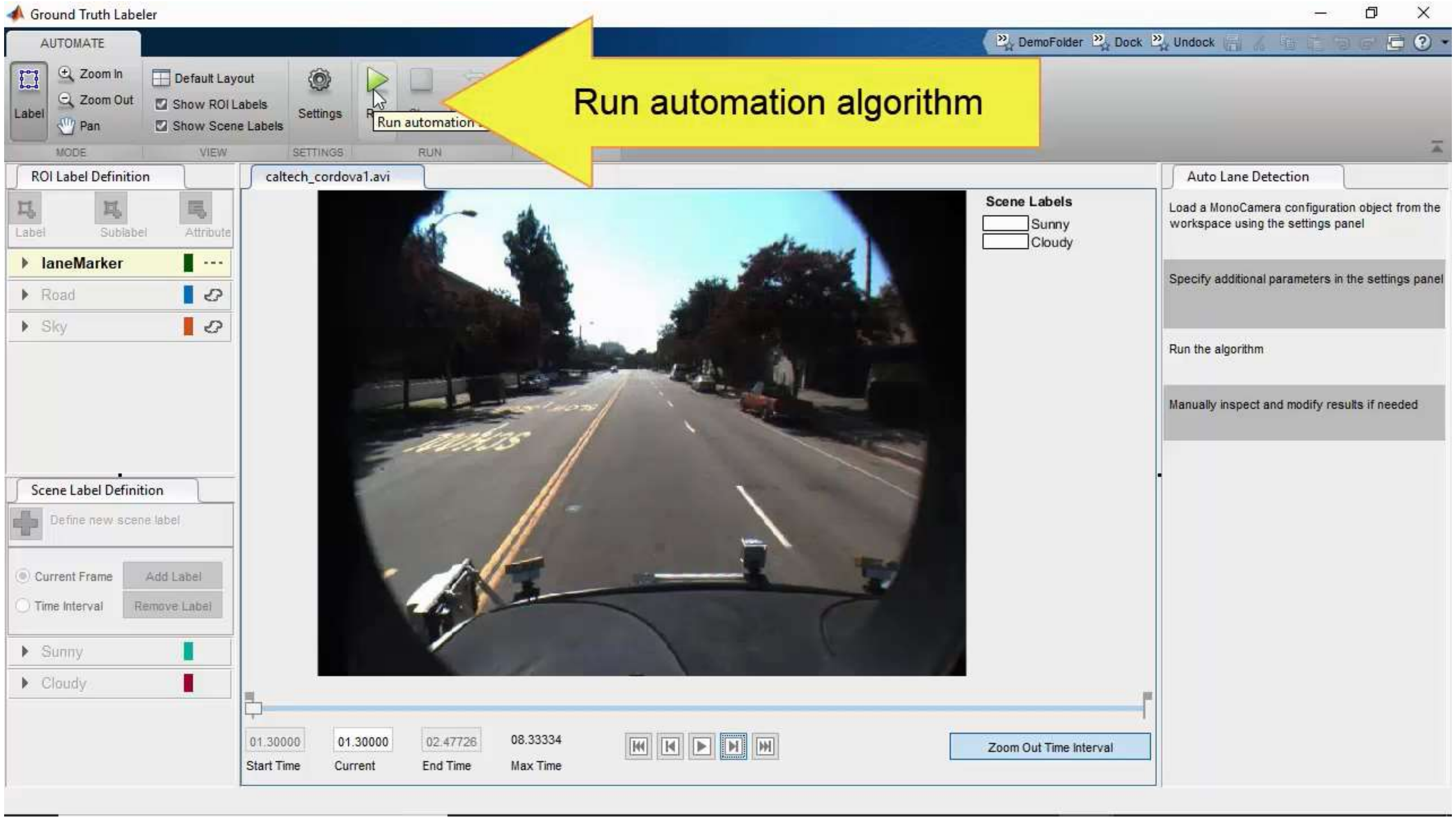
Scene Labels



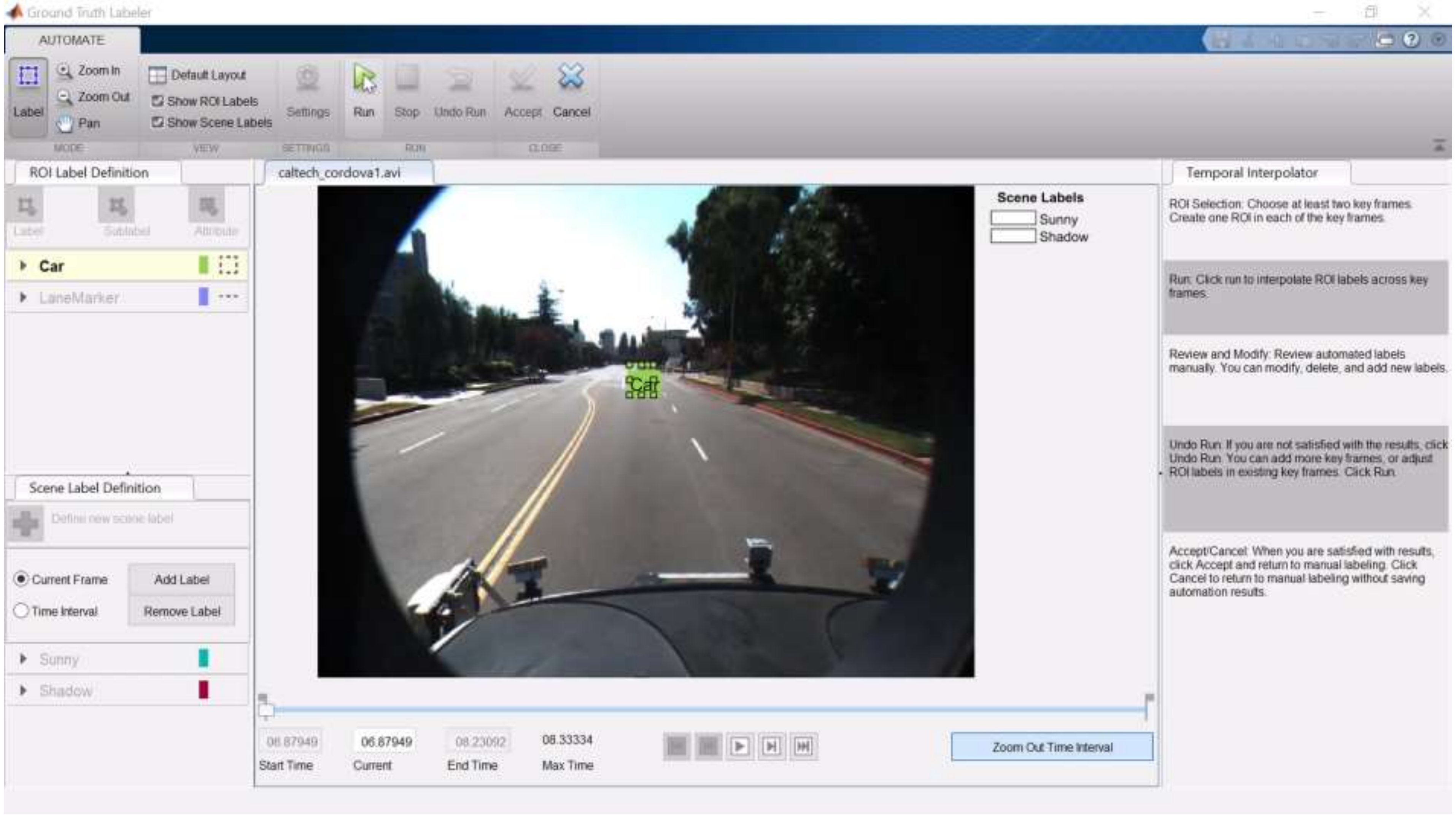
Automate Ground Truth Labeling



Automating Labeling of Lane Markers



Automate Labeling of Bounding Boxes for Vehicles



Export Labeled Data for Training

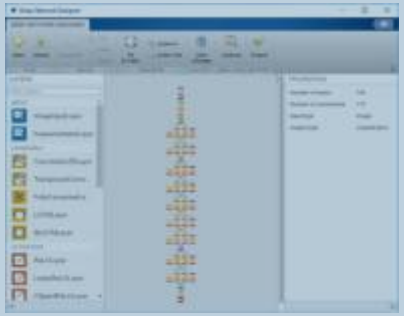

```
>> gTruth

gTruth =

  groundTruth with properties:

          DataSource: [1×1 groundTruthDataSource]
    LabelDefinitions: [4×3 table]
           LabelData: [250×4 timetable]

>> gTruth.LabelData

ans =

  250×4 timetable

       Time             Car           LaneMarker     Sunny    Shadow
    ____________    ____________    ____________    _____    ______

    0 sec           [2×4 double]    {2×1 cell}      true     false
    0.033333 sec    [2×4 double]    {2×1 cell}      true     false
    0.066667 sec    []              []              false    false
```

Bounding Boxes Labels

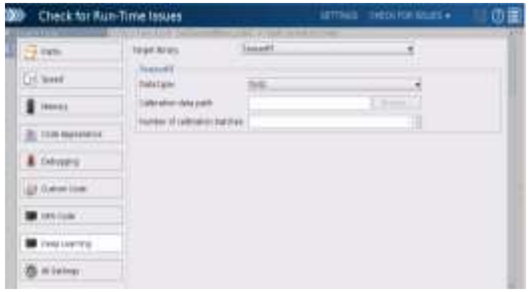
Polyline Labels

Outline

Ground Truth Labeling



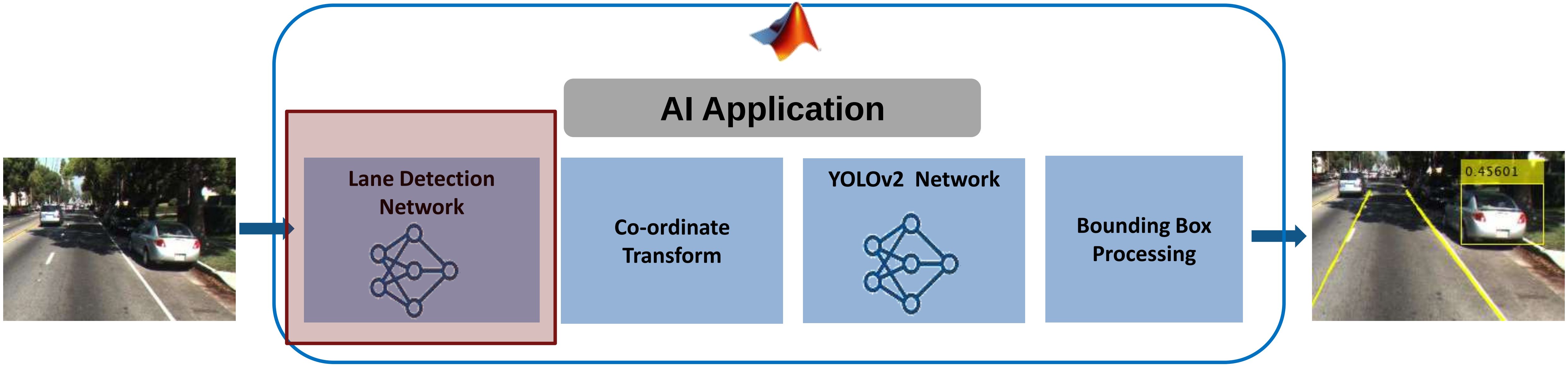
Network Design and Training



CUDA and TensorRT Code Generation

Jetson Xavier and DRIVE Xavier Targeting

Example Used in Today's Talk



Lane Detection Algorithm

Pretrained Network
(E.g. AlexNet)

Modify Network for Lane Detection

Coefficients of parabola

regressionOutputs =

1225×6 table

leftLane_a	leftLane_b	leftLane_c	rightLane_a	rightLane_b	rightLane_c
3.5482e-05	0.0060327	1.7599	-0.00015691	0.030256	-2.0559
-3.9519e-05	0.014116	1.662	-0.00097636	0.02979	-2.0749
-6.778e-07	-0.00063188	1.776	-7.0963e-05	0.0024721	-1.9428
-0.00023646	0.0088324	1.8188	-0.00050391	-0.0015166	-1.973
-0.00055867	0.012996	1.8074	-8.6643e-05	0.00098652	-1.935

Transform to Image Coordinates

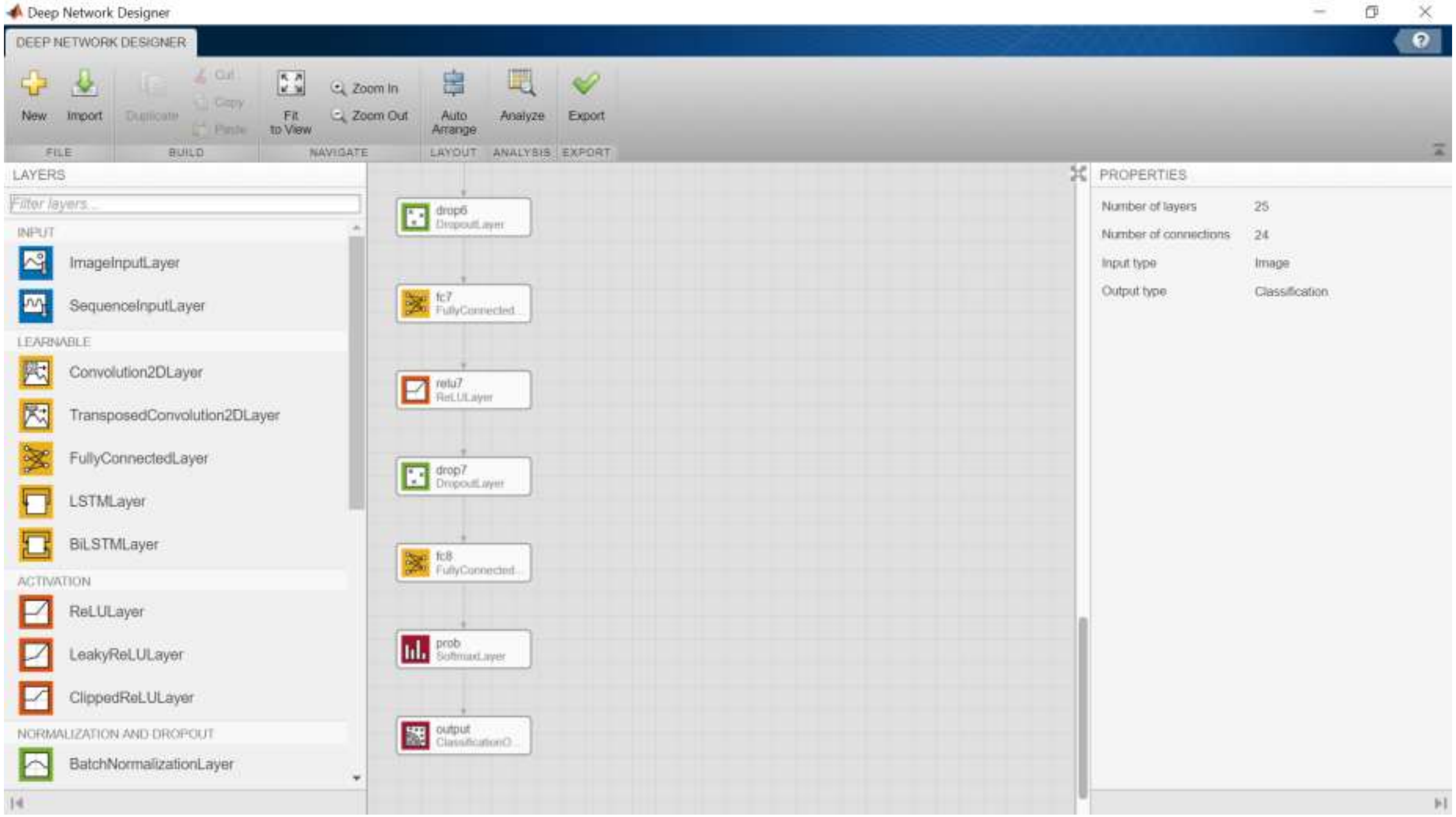
Lane Detection: Load Pretrained Network

Lane Detection Network

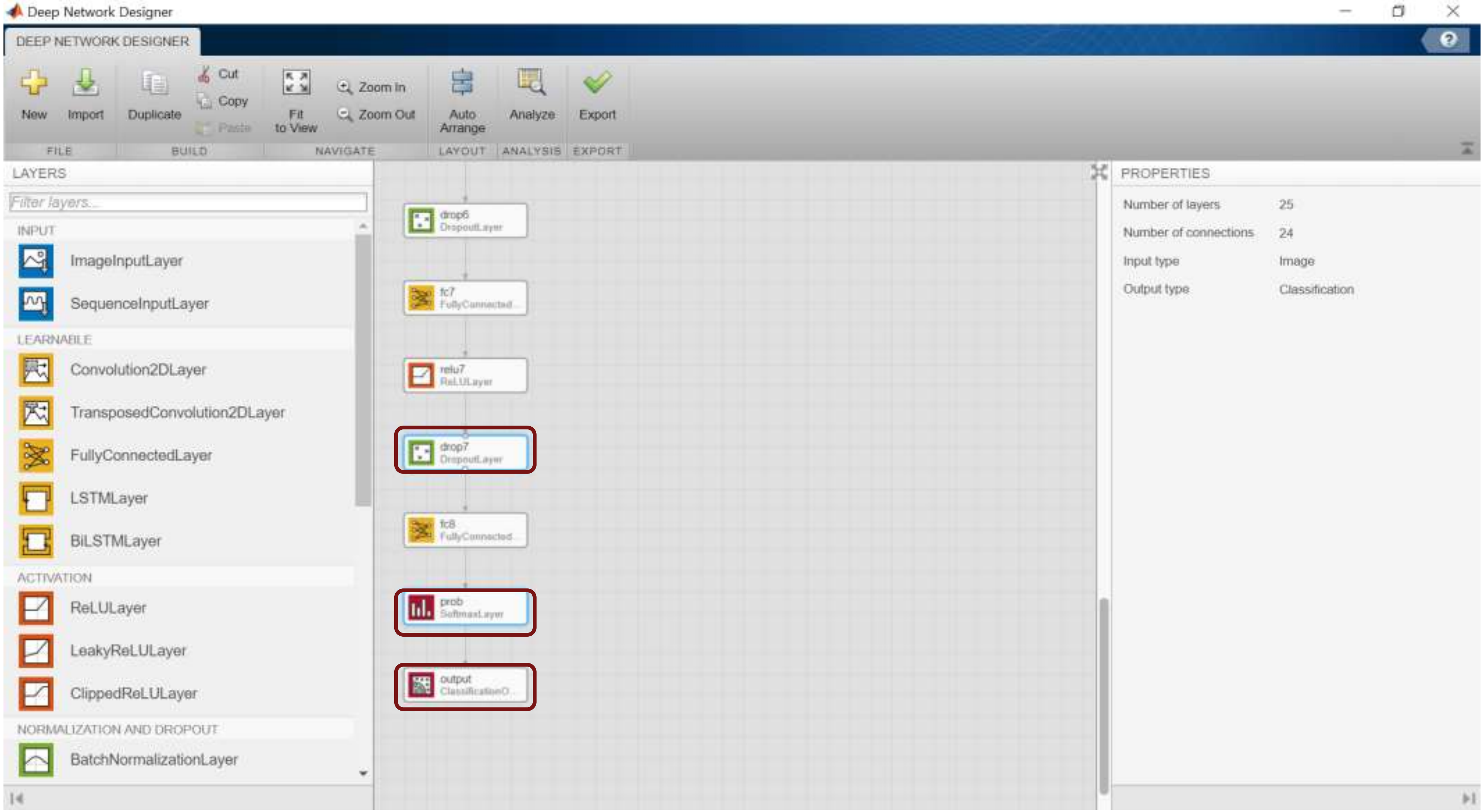
- Regression CNN for lane parameters
- MATLAB code to transform to image co-ordinates

```
>> net = alexnet
>> deepNetworkDesigner
```

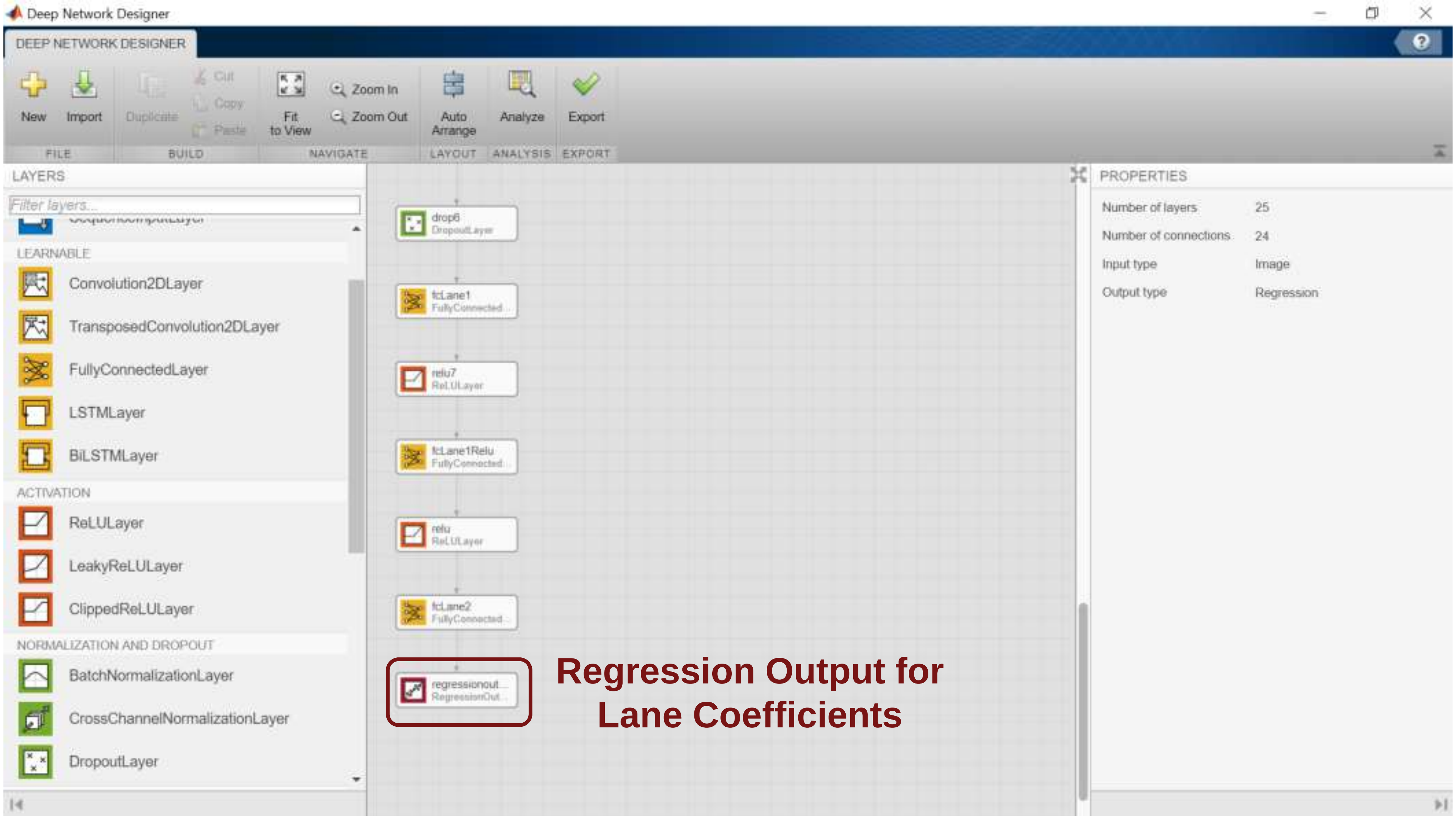
View Network in Deep Network Designer App



Remove Layers from AlexNet



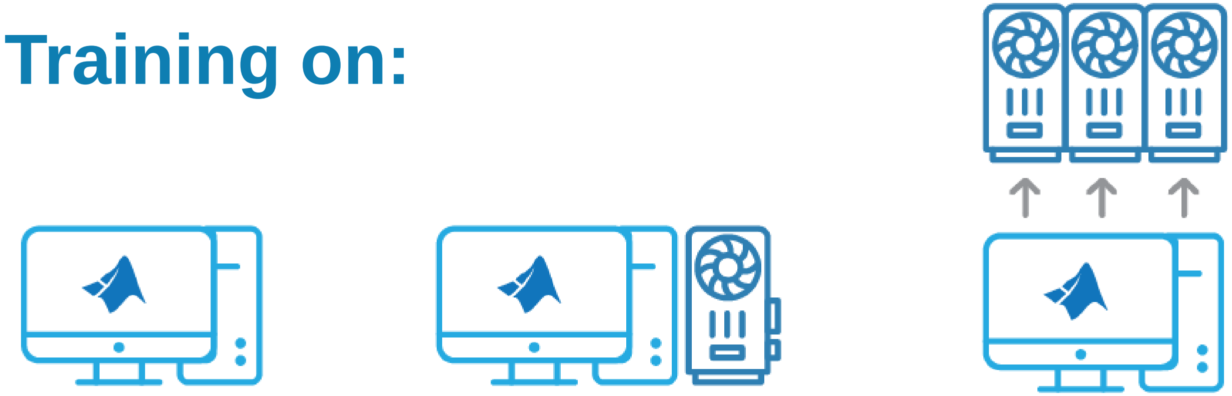
Add Regression Output for Lane Parameters



Regression Output for Lane Coefficients

Transparently Scale Compute for Training

Specify Training on:



'CPU' 'gpu' 'multi-gpu'

Works on Windows
(no additional setup)

Quickly change training hardware

```
opts = trainingOptions('sgdm', ...
    chs', 100, ...
    'MiniBatchSize', 250, ...
    'InitialLearnRate', 0.00005, ...
    'ExecutionEnvironment', 'auto');
```

NVIDIA NGC & DGX Supports MATLAB for Deep Learning

- GPU-accelerated MATLAB Docker container for deep learning
  - Leverage multiple GPUs on NVIDIA DGX Systems and in the Cloud
    - Cloud providers include: AWS, Azure, Google, Oracle, and Alibaba

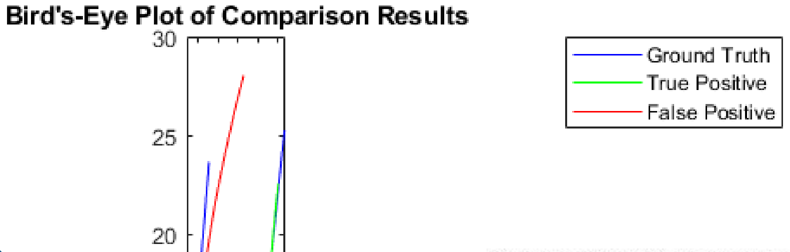
- NVIDIA DGX System / Station
  - Interconnects 4/8/16 Volta GPUs in one box

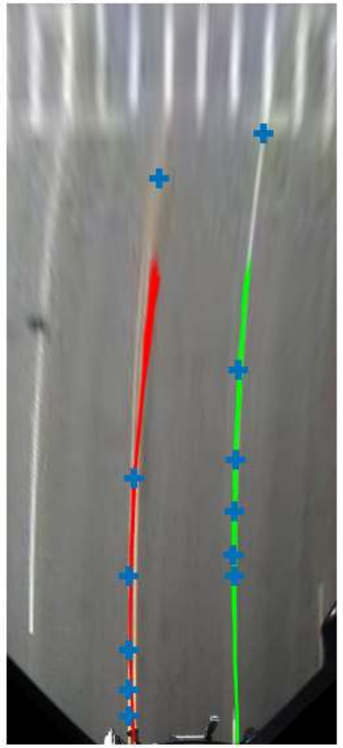
- Containers available for R2018a and R2018b
  - New Docker container with every major release (a/b)

- Download MATLAB container from NGC Registry
  - https://ngc.nvidia.com/registry/partners-matlab

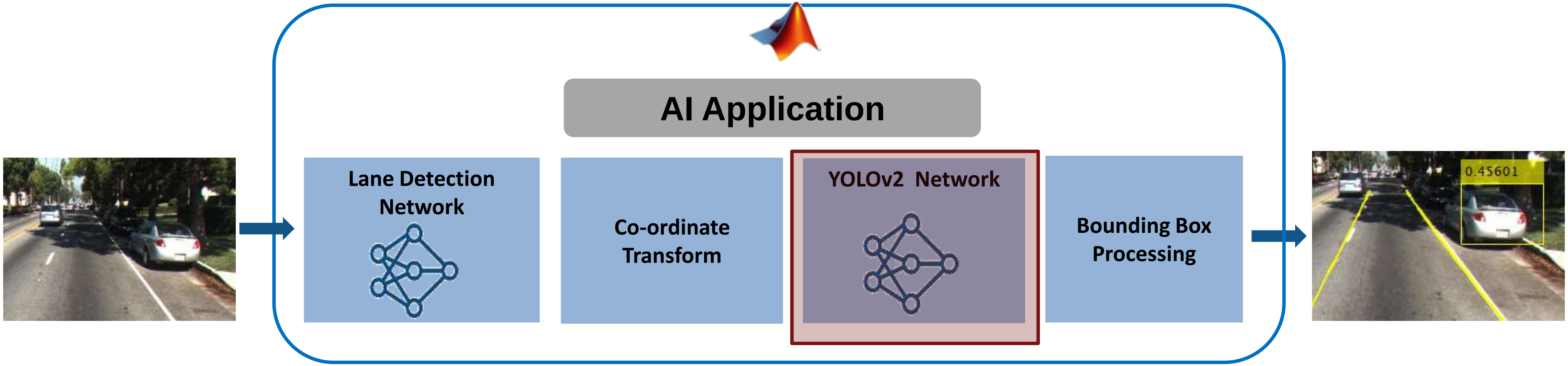
Evaluate Lane Boundary Detections vs. Ground Truth

`evaluateLaneBoundaries`

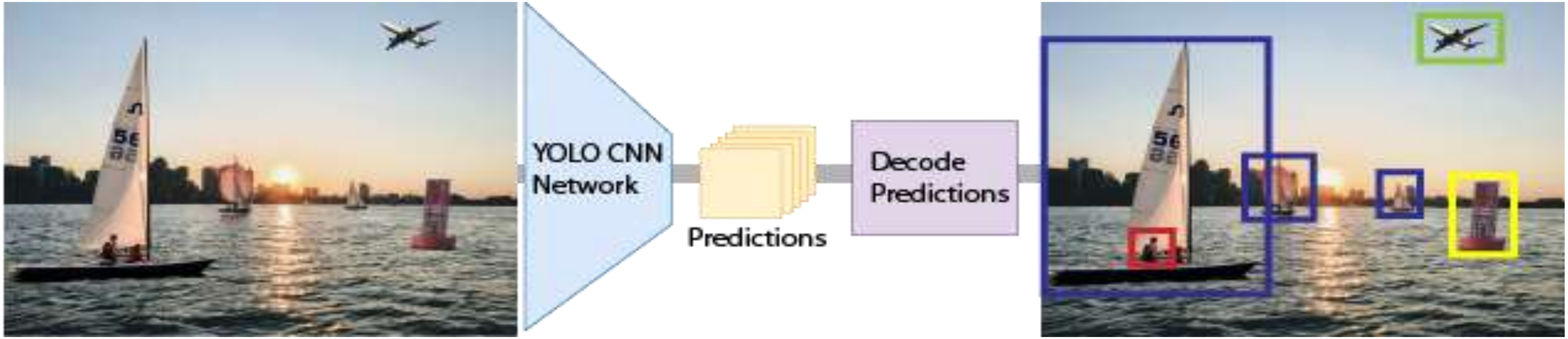




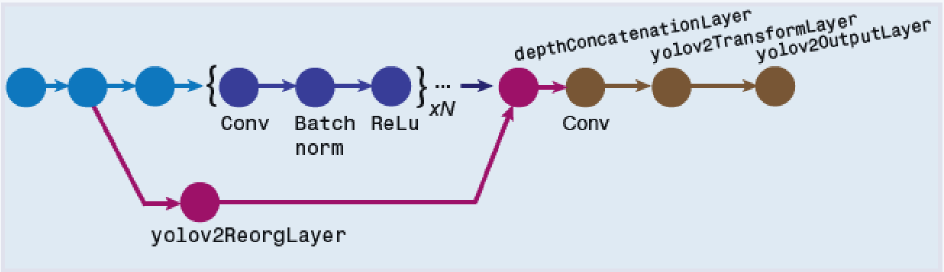
Example Used in Today’s Talk



YOLO v2 Object Detection



Pretrained Network Feature Extractor
(E.g. ResNet 50)



Detection Subnetwork

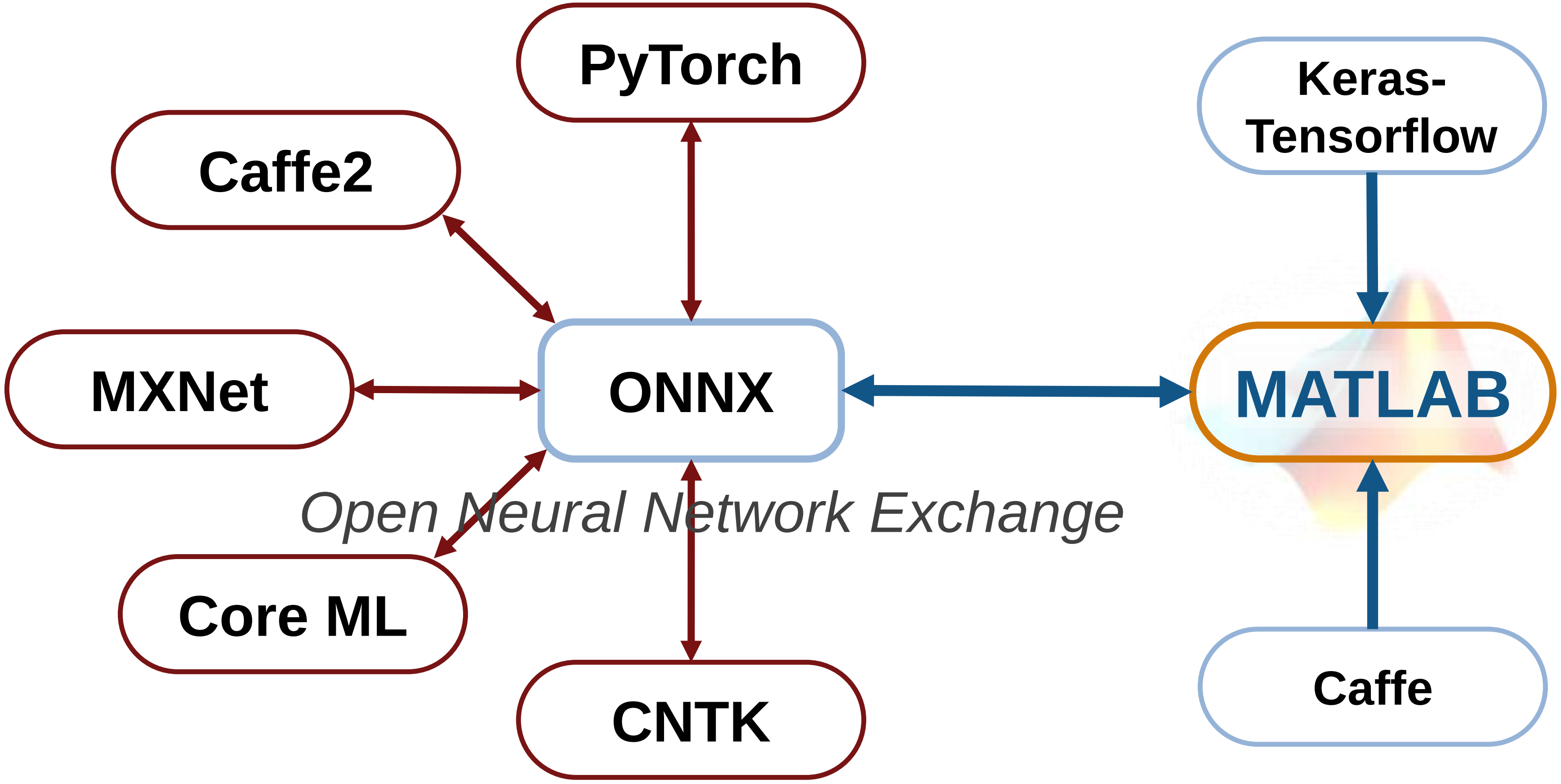
YOLO CNN Network

Two anchor boxes
Class: airplane
Class: sailboat

Filter by class scores, perform non-max suppression and intersection over union

Decode Predictions

Model Exchange with MATLAB



Import Pretrained Network in ONNX Format

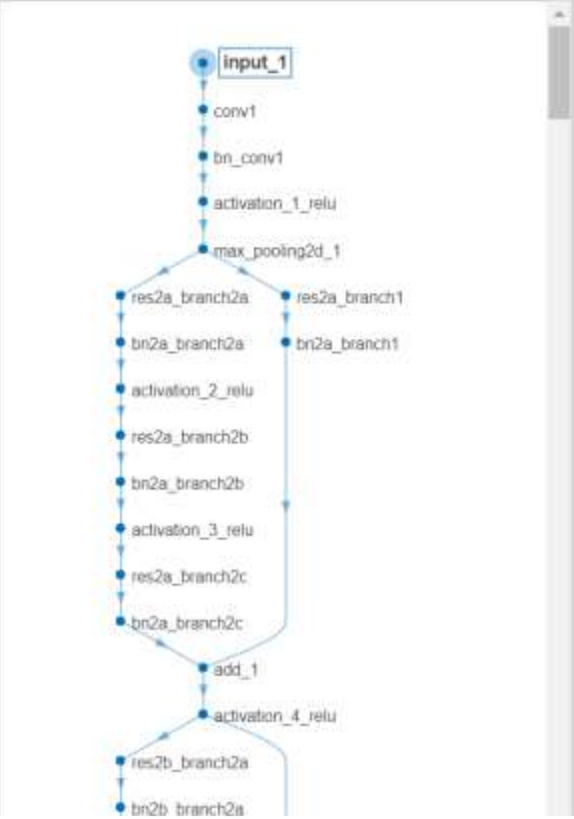
```
load resnetClassNames.mat
net = importONNXNetwork('resnet50.onnx', ...
                        'OutputLayerType', 'classification', ...
                        'ClassNames', classnames);
analyzeNetwork(net)
```

Import Pretrained Network in ONNX Format

resnet50

Analysis date: 09-Jan-2019 09:39:08

177 layers | 0 warnings | 0 errors



ANALYSIS RESULT

	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	input_1 224x224x3 images with 'zerocenter' normalization	Image Input	224×224×3	-
2	conv1 64 7x7x3 convolutions with stride [2 2] and padding [3 3 3 3]	Convolution	112×112×64	Weights 7×7×3×64 Bias 1×1×64
3	bn_conv1 Batch normalization with 64 channels	Batch Normalization	112×112×64	Offset 1×1×64 Scale 1×1×64
4	activation_1_relu ReLU	ReLU	112×112×64	-
5	max_pooling2d_1 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	55×55×64	-
6	res2a_branch2a 64 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	55×55×64	Weights 1×1×64×64 Bias 1×1×64
7	bn2a_branch2a Batch normalization with 64 channels	Batch Normalization	55×55×64	Offset 1×1×64 Scale 1×1×64
8	activation_2_relu ReLU	ReLU	55×55×64	-
9	res2a_branch2b 64 3x3x64 convolutions with stride [1 1] and padding 'same'	Convolution	55×55×64	Weights 3×3×64×64 Bias 1×1×64
10	bn2a_branch2b Batch normalization with 64 channels	Batch Normalization	55×55×64	Offset 1×1×64 Scale 1×1×64
11	activation_3_relu ReLU	ReLU	55×55×64	-
12	res2a_branch2c 256 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	55×55×256	Weights 1×1×64×256 Bias 1×1×256
13	res2a_branch1 256 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	55×55×256	Weights 1×1×64×256 Bias 1×1×256
14	bn2a_branch2c	Batch Normalization	55×55×256	Offset 1×1×256

Modify Network

```
lgraph = layerGraph(net);
lgraph = removeLayers(lgraph,'Input_input_1');
lgraph = removeLayers(lgraph,'fc1000_Flatten1');
lgraph = connectLayers(lgraph,'avg_pool','fc1000');

avgImgBias = -1*(lgraph.Layers(1).Bias);

%Create new input layer and incorporate average image bias
larray = imageInputLayer([224 224 3],...
    'Name','input',...
    'AverageImage',avgImgBias);

lgraph = replaceLayer(lgraph,'input_1_Sub',larray);

netModified = assembleNetwork(lgraph);

save('resnet50_model.mat','netModified');
```

Removing the 2 ResNet-50 layers

`imageInputLayer` replaces the input and subtraction layer

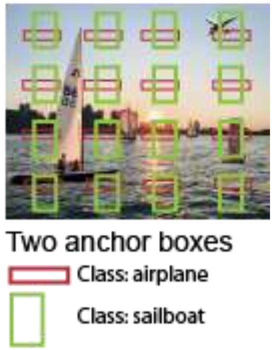
Save MAT file for code gen

YOLOv2 Detection Network

- **`yolov2Layers`:** Create network architecture

```
>> lgraph = yolov2Layers(imageSize, numClasses, anchorBoxes, network, featureLayer)
```

Number of Classes



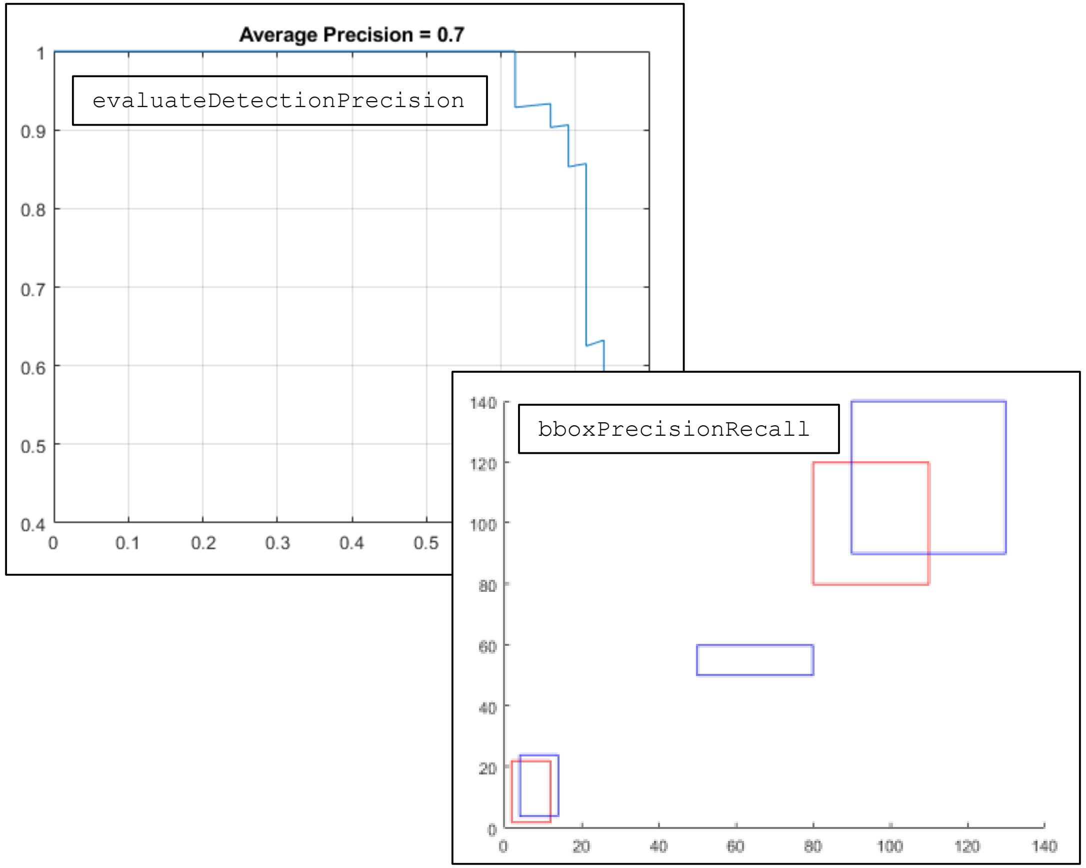
Pretrained Feature Extractor

```
>> detector = trainYOLOv2ObjectDetector(trainingData,lgraph,options)
```

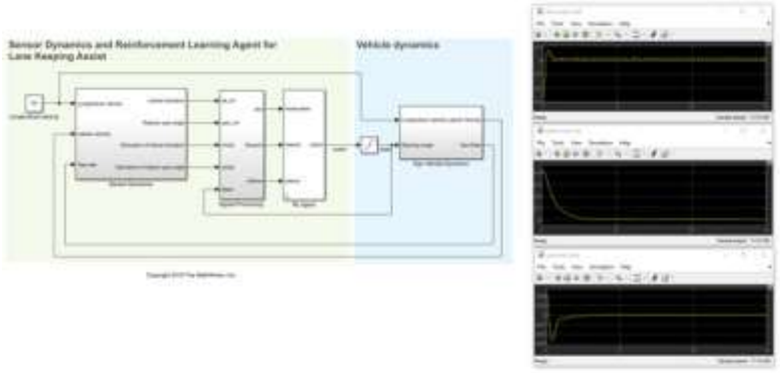
Evaluate Performance of Trained Network

- **Set of functions** to evaluate trained network performance
  - `evaluateDetectionMissRate`
  - **`evaluateDetectionPrecision`**
  - `bboxPrecisionRecall`
  - `bboxOverlapRatio`

```
>> [ap,recall,precision] =
evaluateDetectionPrecision(results,vehicles(:,2));
```

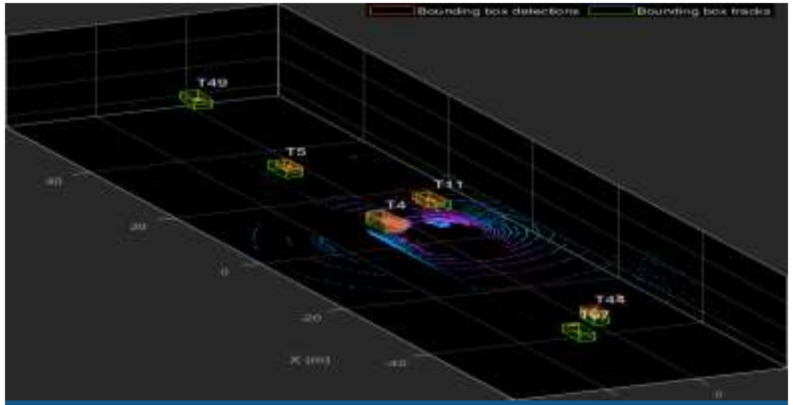


Example Applications using MATLAB for AI Development



Lane Keeping Assist using Reinforcement Learning

Occupancy Grid Creation using Deep Learning

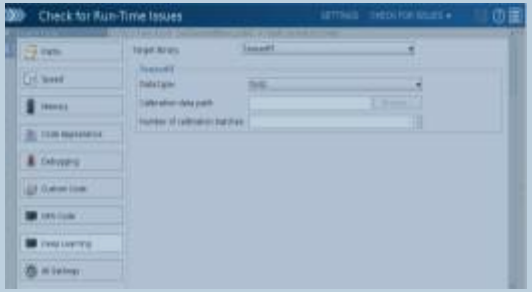


Lidar Segmentation with Deep Learning

Outline

Ground Truth Labeling

Network Design and Training



CUDA and TensorRT Code Generation

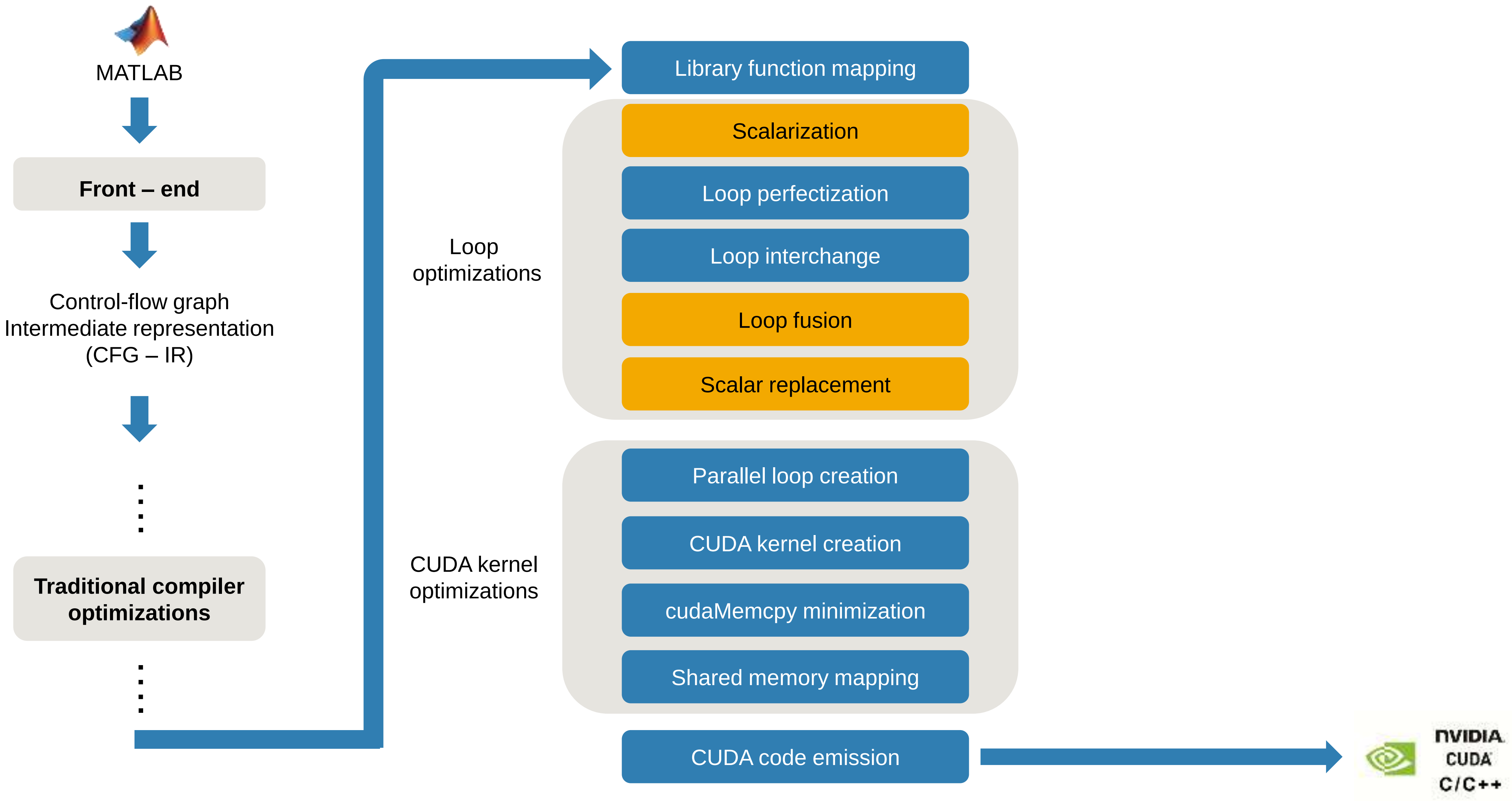
Jetson Xavier and DRIVE Xavier Targeting

Key Takeaways

Platform Productivity: Workflow automation, ease of use

Framework Interoperability: ONNX, Keras-TensorFlow, Caffe

GPU Coder runs a host of compiler transforms to generate CUDA



Example Used in Today's Talk

AI Application

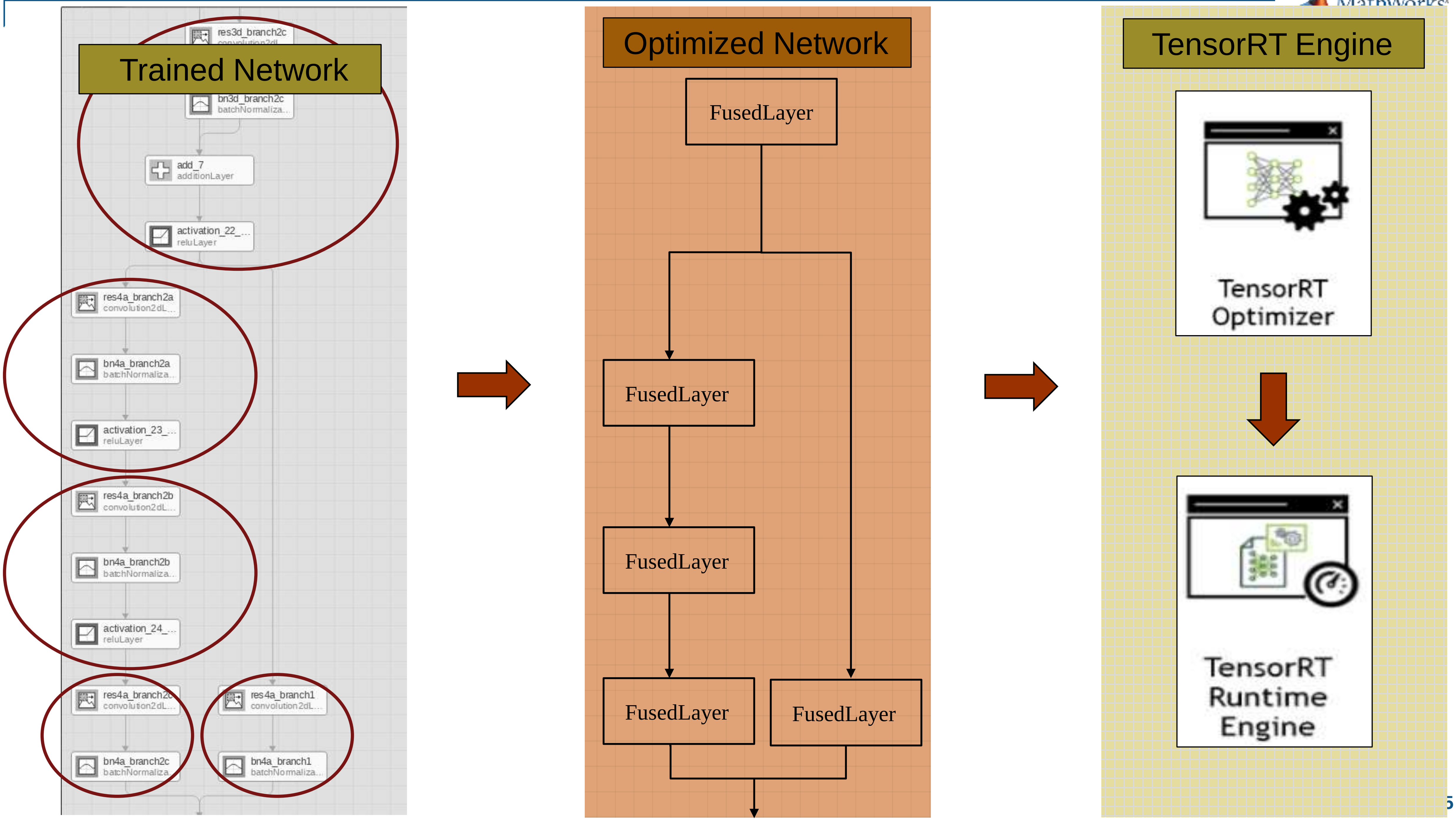
Lane Detection Network

Co-ordinate Transform

YOLOv2 Network

Bounding Box Processing

Optimized TensorRT Code for Models


Trained Network
res3d_branch2c
bn3d_branch2c
batchNormaliza...
add_7
additionLayer
activation_22_...
reluLayer
res4a_branch2a
convolution2dL...
bn4a_branch2a
batchNormaliza...
activation_23_...
reluLayer
res4a_branch2b
convolution2dL...
bn4a_branch2b
batchNormaliza...
activation_24_...
reluLayer
res4a_branch2c
convolution2dL...
res4a_branch1
convolution2dL...
bn4a_branch2c
batchNormaliza...
bn4a_branch1
batchNormaliza...
Optimized Network
FusedLayer
FusedLayer
FusedLayer
FusedLayer
FusedLayer
TensorRT Engine
TensorRT Optimizer
TensorRT Runtime Engine

MATLAB R2019b - prerelease use

HOME PLOTS APPS EDITOR PUBLISH VIEW

/ mathworks devel sandbox jshankar GTC2019 demofolder demo_files

Current Folder

- codegen
- caltech_washington1.avi
- lane_and_vehicleDetection.m
- lane_yolo.m
- LaneDetectionNet.mat
- VehicleDetectorNet.mat

Editor - /mathworks/devel/sandbox/jshankar/GTC2019/demofolder/demo_files/lane_yolo.m

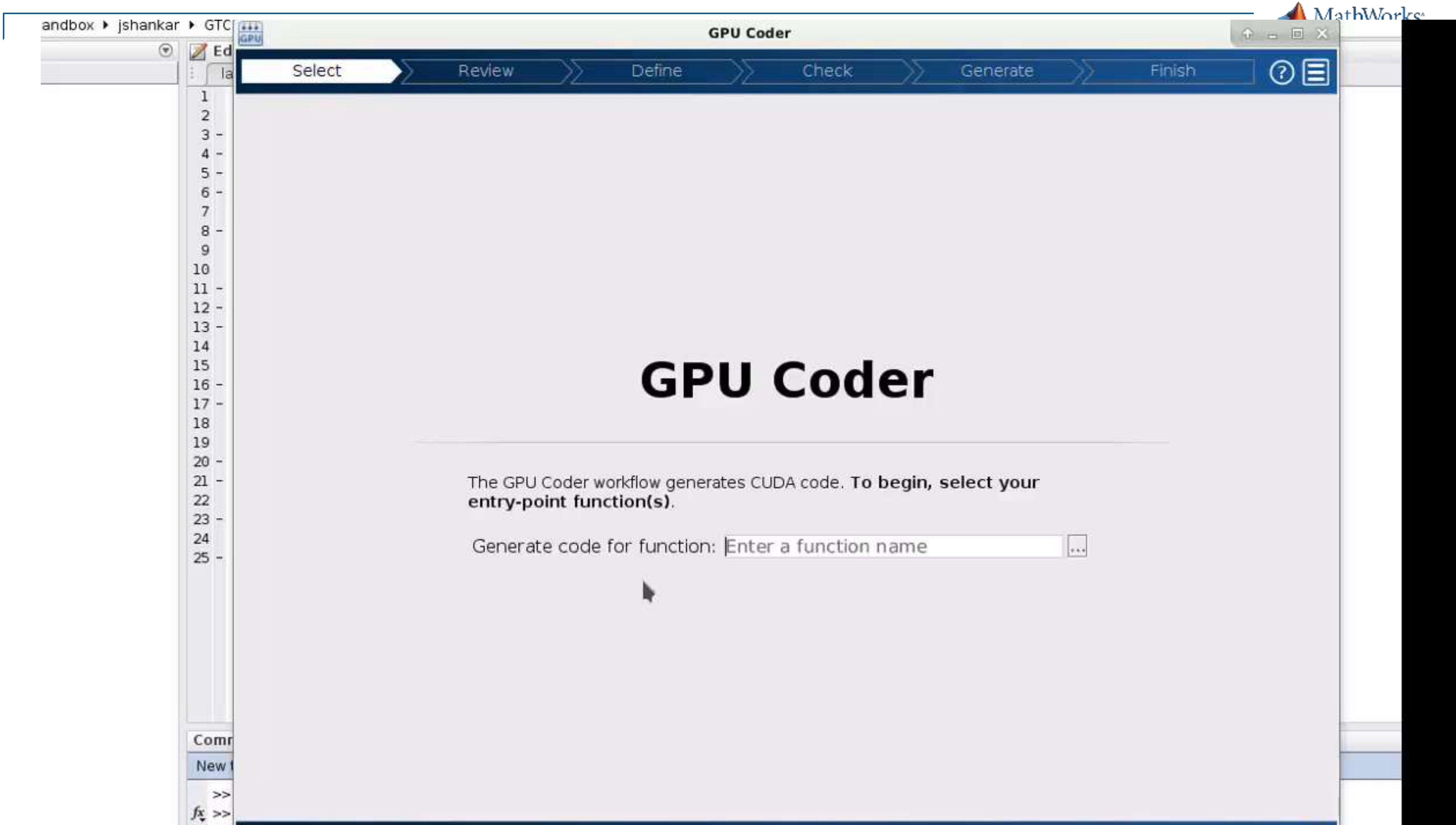
lane_yolo.m | lane_and_vehicleDetection.m

```matlab
function Out = lane_yolo(In)
% The regression network is trained to detect parameters of lane parabola
% The outputs are unnormalized and converted to left and right  lane points
% in image coordinates.
% The camera coordinates are described by the caltech mono camera model.

%#codegen

frame = imresize(In, [227,227]);

persistent lanenet;
if isempty(lanenet)
    lanenet = coder.loadDeepLearningNetwork('LaneDetectionNet.mat', 'lanenet');
end

lanecoeffsNetworkOutput = lanenet.predict(frame);

% Recover original coeffs by reversing the normalization steps
laneCoeffMeans = [-.0002 , .0002, 1.4740, -.0002, .0045, -1.3787];
laneCoeffStds = [.0030, .0766, .6313, .0026, .0736, .9846];
params = lanecoeffsNetworkOutput .* laneCoeffStds + laneCoeffMeans;

% should be more than 0.5 for it to be a lane
isRightLaneFound = abs(params(6)) > 0.5;
isLeftLaneFound =  abs(params(3)) > 0.5;

vehicleXPoints = 3:30; %meters, ahead of the sensor
ltPts = coder.nullcopy(zeros(28,2,'single'));
rtPts = coder.nullcopy(zeros(28,2,'single'));

% map vehicle to image coordinates
if isRightLaneFound && isLeftLaneFound
```

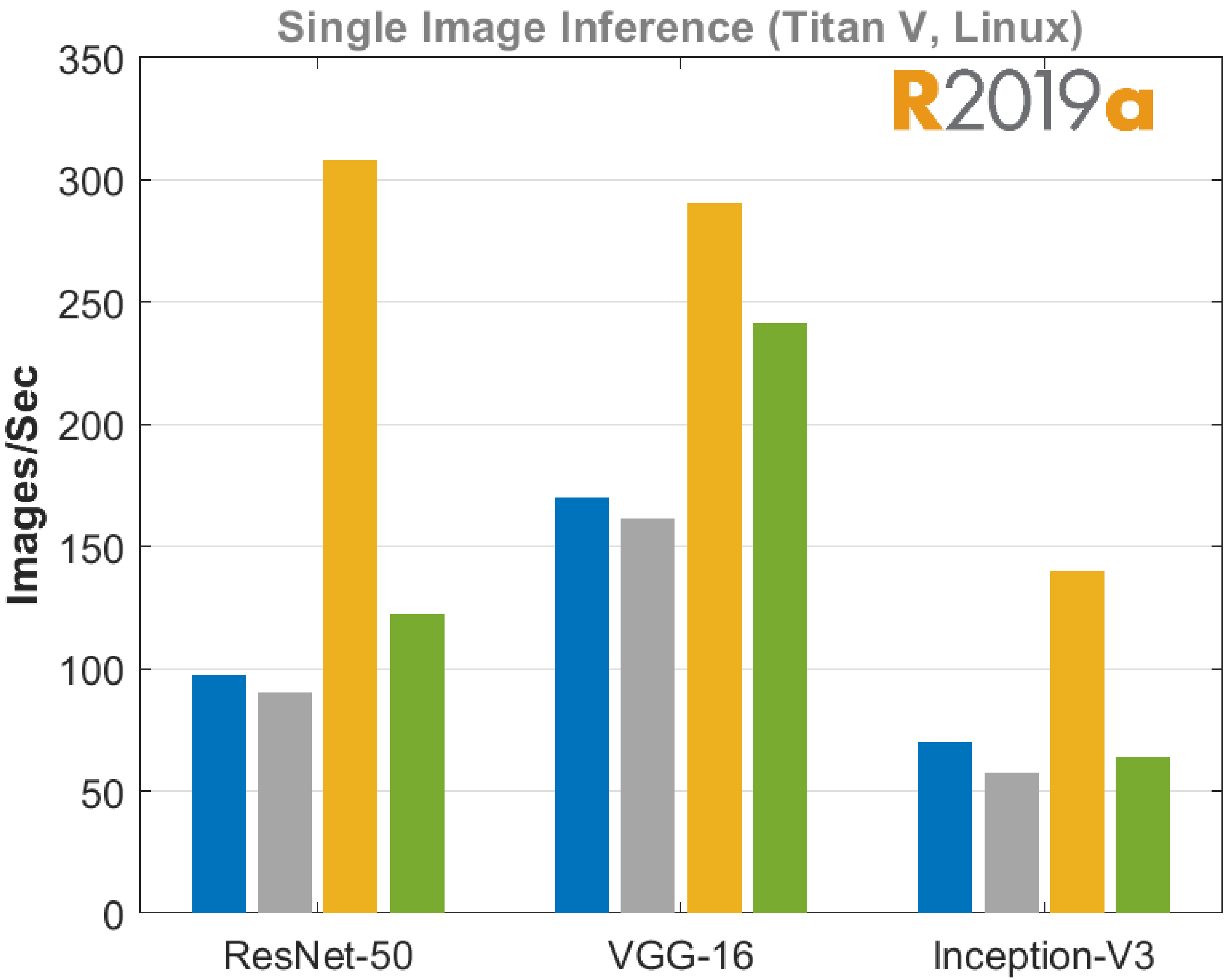
Command Window

New to MATLAB? See resources for Getting Started.

```
>>
```


andbox ▸ jshankar ▸ GTC
GPU Coder
Select
Review
Define
Check
Generate
Finish
GPU Coder
The GPU Coder workflow generates CUDA code. **To begin, select your entry-point function(s)**.
Generate code for function: Enter a function name
MathWorks

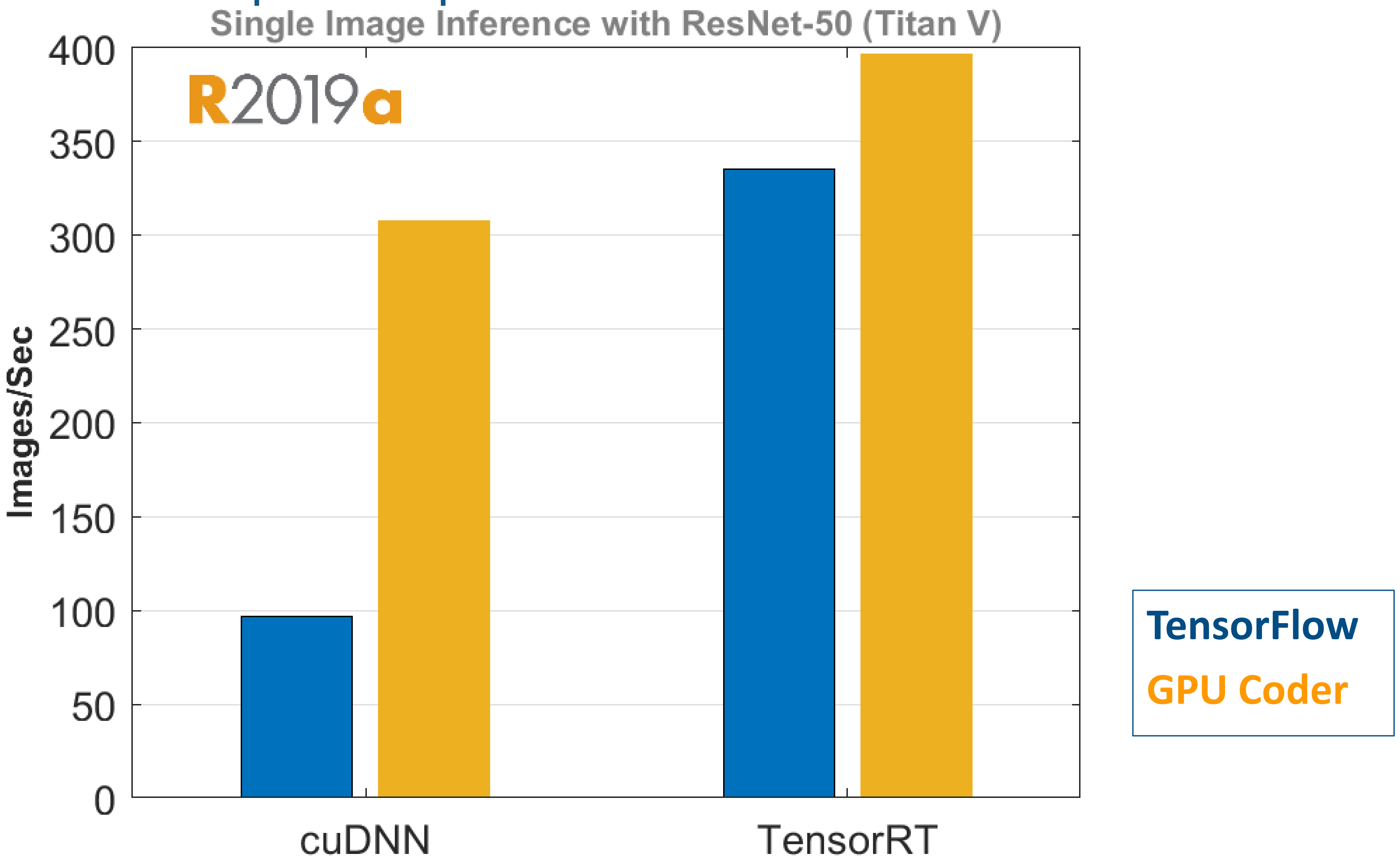
With GPU Coder, MATLAB is fast



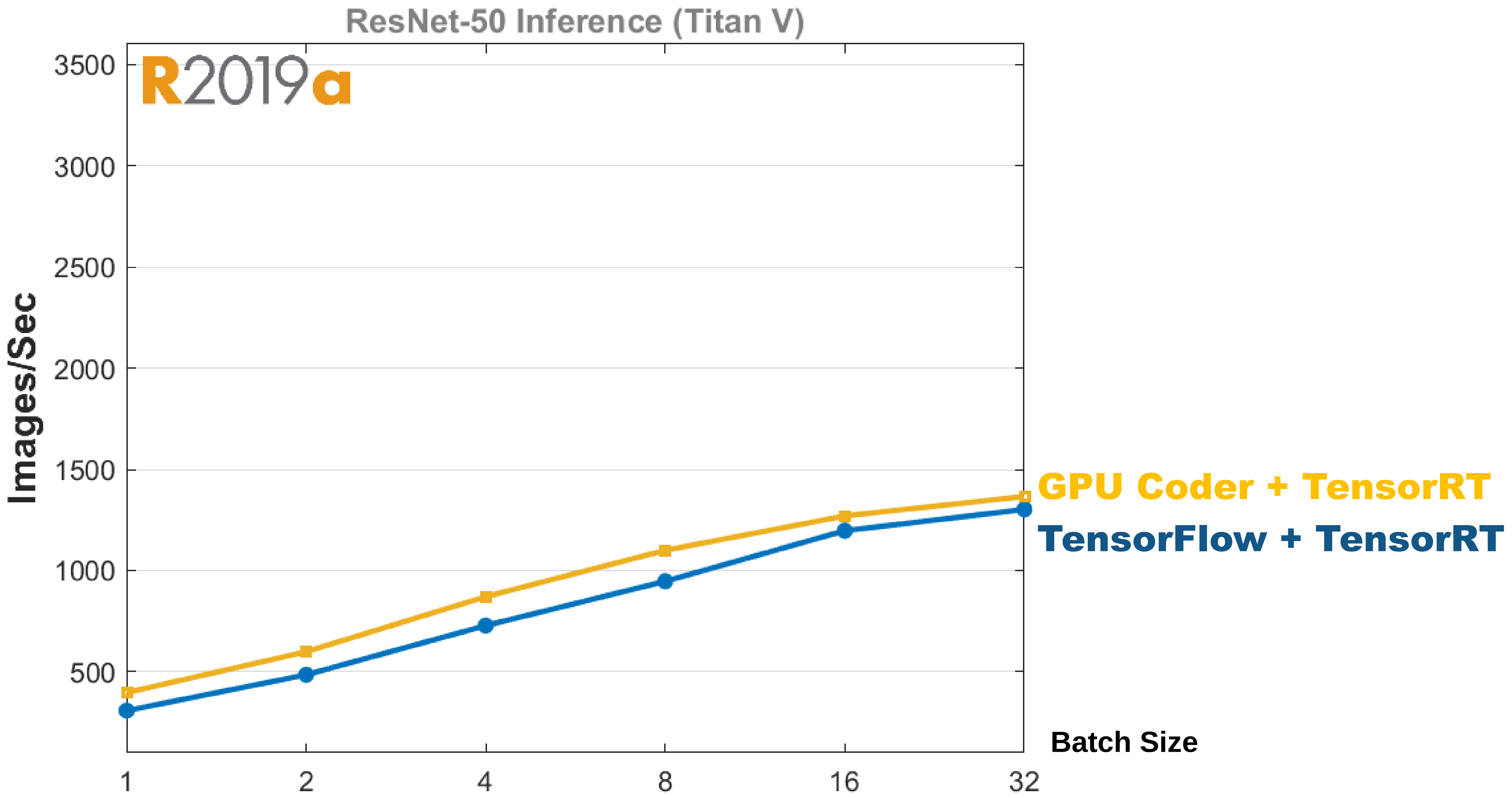
Faster than TensorFlow, MXNet, and PyTorch

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7 - Frameworks: TensorFlow 1.13.0, MXNet 1.4.0 PyTorch 1.0.0

TensorRT speeds up inference for TensorFlow and GPU Coder

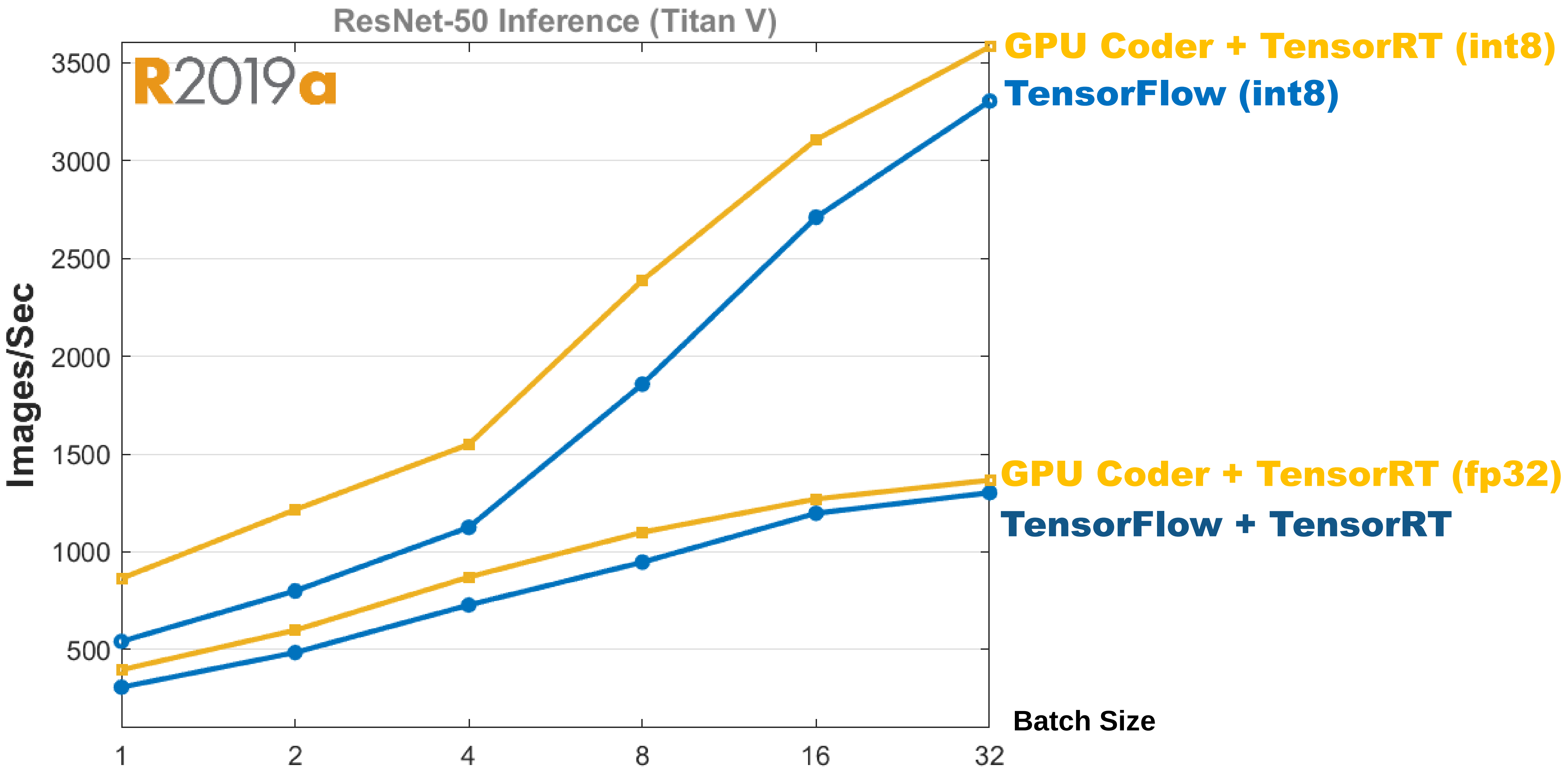


GPU Coder with TensorRT faster across various Batch Sizes



Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7 – Tensor RT 5.0.2.6. Frameworks: TensorFlow 1.13.0, MXNet 1.4.0 PyTorch 1.0.0

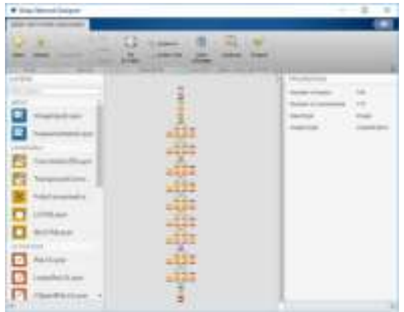
Even higher Speeds with Integer Arithmetic (int8)



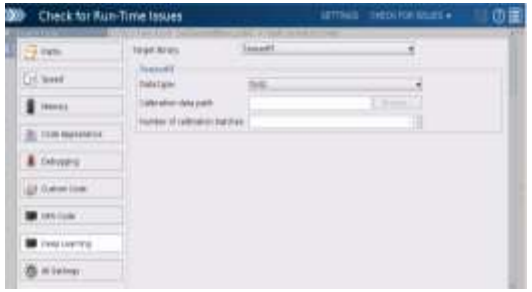
Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7 – Tensor RT 5.0.2.6. Frameworks: TensorFlow 1.13.0, MXNet 1.4.0 PyTorch 1.0.0

Outline

Ground Truth Labeling



Network Design and Training



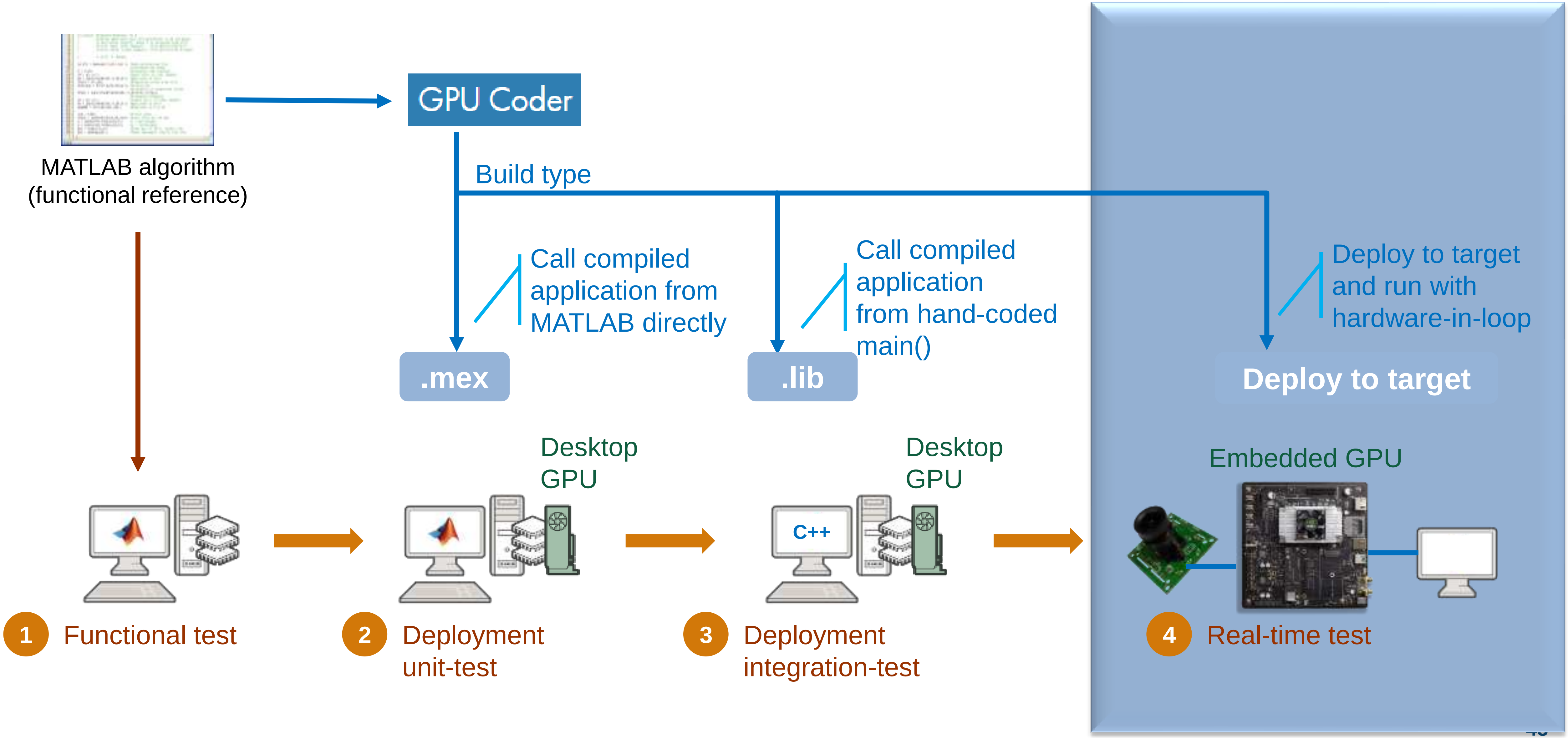
CUDA and TensorRT Code Generation

Jetson Xavier and DRIVE Xavier Targeting

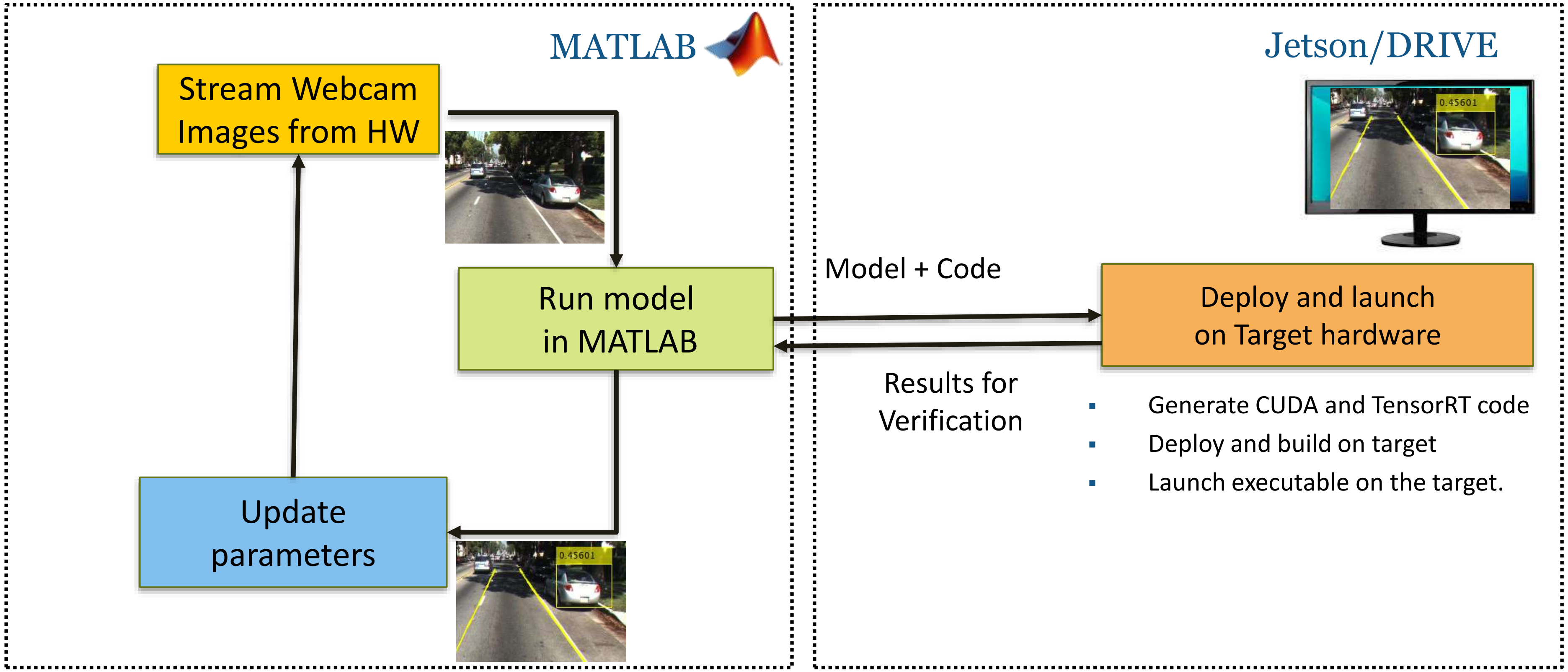
Key Takeaways

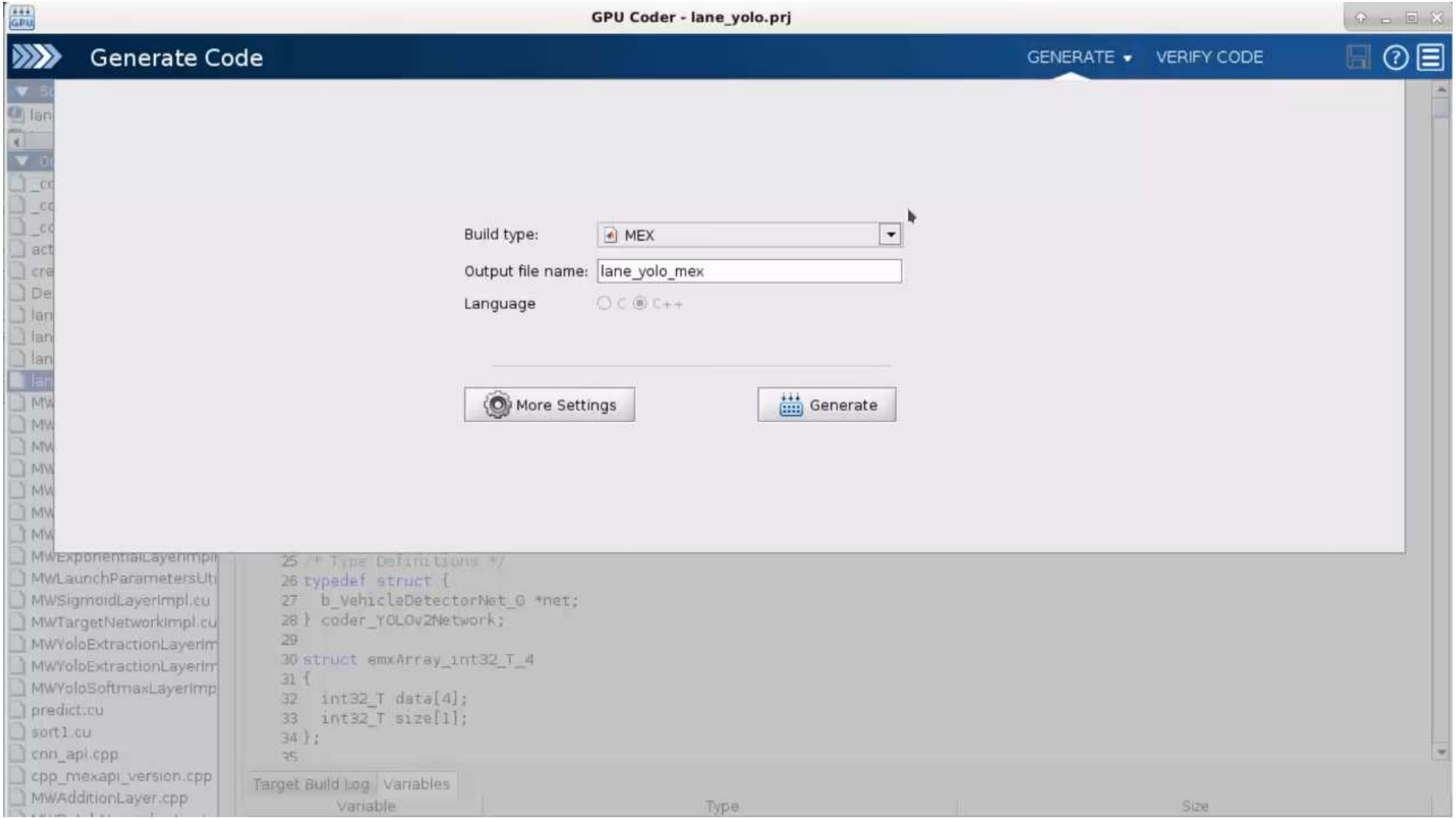
Optimized CUDA and TensorRT code generation

Deploy to Jetson and Drive



Hardware in the loop workflow with Jetson/DRIVE device




GPU Coder - lane_yolo.prj
Generate Code
GENERATE
VERIFY CODE
Build type:
MEX
Output file name:
lane_yolo_mex
Language
C
C++
More Settings
Generate
MWLaunchParametersUti
MWSigmoidLayerImpl.cu
MWTargetNetworkImpl.cu
MWYoloSoftmaxLayerImp
predict.cu
sort1.cu
cnn_api.cpp
cpp_mexapi_version.cpp
MWAdditionLayer.cpp
26 typedef struct {
27 b_VehicleDetectorNet_0 *net;
28 } coder_YOLOv2Network;
30 struct emxArray_int32_T_4
31 {
32 int32_T data[4];
33 int32_T size[1];
34 };
Target Build Log
Variables
Variable
Type
Size

Editor - /mathworks/devel/sandbox/jshankar/GTC2019/demofolder/demo_files/lane_and_vehicleDetection.m

lane_yolo.m | lane_and_vehicleDetection.m | +

```
function lane_and_vehicleDetection

videoFileReader = VideoReader('caltech_washington1.avi');
depVideoPlayer = vision.DeployableVideoPlayer('Name', 'simulation');
fps = 0;
while hasFrame(videoFileReader)
    % grab frame from video
    I = readFrame(videoFileReader);

    % Run the detector on the input test image
    tic;
    sim_frame = lane_yolo_mex(I);
    mltime = toc;

    % Calculate fps
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> h
```

Processor in the loop verification with Jetson/Drive devices

```
% Set up connection to Jetson device
hwobj = jetson('gpucoder-xavier-1','ubuntu','ubuntu');
```

```
% Set up code generation to Processor-in-loop mode
cfg = coder.gpuConfig('lib');
cfg.VerificationMode = 'PIL';
cfg.Hardware = coder.hardware('NVIDIA Jetson');
```

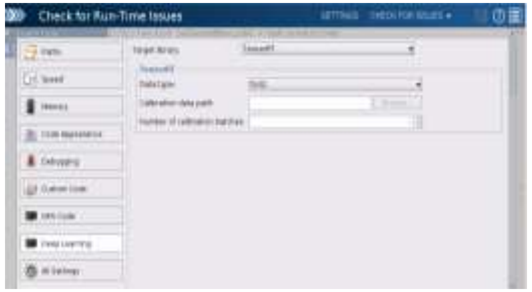
```
% Generate code for application using CUDA and TensorRT
cfg.DeepLearningConfig = coder.DeepLearningConfig('tensorrt');
codegen -config cfg detect_lane_yolo_full -args {ones(480,640,3,'uint8')}
```

Generates a wrapper
detect_lane_yolo_full_pil

Outline

Ground Truth Labeling

Network Design and Training



CUDA and TensorRT Code Generation

Jetson Xavier and DRIVE Xavier Targeting

Key Takeaways

Platform Productivity: Workflow automation, ease of use

Framework Interoperability: ONNX, Keras-TensorFlow, Caffe

Key Takeaways

Optimized CUDA and TensorRT code generation

Jetson Xavier and DRIVE Xavier targeting

Processor-in-loop(PIL) testing and system integration

Thank You