Applying Artificial Intelligence to Product Development

Sebastian Bomberg, Application Engineering
Diverse Set of Automotive Customers use MATLAB for AI

**Caterpillar**
Cloud Based Data Labeling

**Veoneer**
Radar Sensor Verification

**Alpine**
Ground Detection

**Musashi Seimitsu**
Automotive Part Defect Detection

**Results - Accuracy**
- Achieved high recognition rates in our test cases.
- Median recognition rate on ten test cases: 99%
- Outliers at the low end due to “no ground” in view
- Slightly high false positive with a 3x5 median.
- Non-ground being recognized as ground
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

AI Application

- Lane Detection Network
- Co-ordinate Transform
- YOLOv2 Network
- Bounding Box Processing
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 → Labels for Training
Interactive Tools for Ground Truth Labeling

**ROI Labels**
- Bound boxes
- Pixel labels
- Poly-lines

**Scene Labels**
Automate Ground Truth Labeling

Pre-built Automation

User authored automation
Automating Labeling of Lane Markers

Run automation algorithm
Automate Labeling of Bounding Boxes for Vehicles
Export Labeled Data for Training

```
>> gTruth

gTruth =

groundTruth with properties:

    DataSource: [1x1 groundTruthDataSource]
    LabelDefinitions: [4x3 table]
    LabelData: [250x4 timetable]

>> gTruth.LabelData

ans =

250x4 timetable

<table>
<thead>
<tr>
<th>Time</th>
<th>Car</th>
<th>LaneMarker</th>
<th>Sunny</th>
<th>Shadow</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 sec</td>
<td>[2x4 double]</td>
<td>[2x1 cell]</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>0.033333 sec</td>
<td>[2x4 double]</td>
<td>[2x1 cell]</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>0.066667 sec</td>
<td>[]</td>
<td>[]</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
```

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 Network

Co-ordinate Transform

YOLOv2 Network

Bounding Box Processing

AI Application
Lane Detection Algorithm

Pretrained Network (E.g. AlexNet) → Modify Network for Lane Detection → Coefficients of parabola → Transform to Image Coordinates

```
regressionOutputs =
12x9 table

<table>
<thead>
<tr>
<th></th>
<th>leftlens_a</th>
<th>leftlens_b</th>
<th>leftlens_c</th>
<th>rightlens_a</th>
<th>rightlens_b</th>
<th>rightlens_c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.8402e-05</td>
<td>0.0360327</td>
<td>1.7899</td>
<td>-0.00115832</td>
<td>0.032235</td>
<td>-0.5919</td>
</tr>
<tr>
<td>-4.5134e-05</td>
<td>3.14151</td>
<td>2.001</td>
<td>-0.002075245</td>
<td>0.02595</td>
<td>-2.7840</td>
<td></td>
</tr>
<tr>
<td>-4.7533e-07</td>
<td>-0.006213</td>
<td>1.726</td>
<td>-0.00439502</td>
<td>3.002121</td>
<td>-1.8200</td>
<td></td>
</tr>
<tr>
<td>-0.00234646</td>
<td>0.0388204</td>
<td>1.8108</td>
<td>-0.00003986</td>
<td>-3.005166</td>
<td>-1.973</td>
<td></td>
</tr>
<tr>
<td>-0.00035887</td>
<td>0.0322994</td>
<td>2.0174</td>
<td>-0.00031608</td>
<td>3.0007862</td>
<td>-1.8200</td>
<td></td>
</tr>
</tbody>
</table>
```

Lane Detection Network Output
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
Transparency Scale Compute for Training

Specify Training on:

- 'CPU'
- 'gpu'
- 'multi-gpu'

Quickly change training hardware

Works on Windows (no additional setup)

```python
opts = trainingOptions('sgdm', ...
  '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

Sample Ground Truth Data for Left Lane Boundary

Bird's-Eye Plot of Comparison Results

Bird's-Eye View of Comparison Results
Example Used in Today’s Talk

AI Application

Lane Detection Network
Co-ordinate Transform
YOLOv2 Network
Bounding Box Processing
YOLO v2 Object Detection

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

Detection Subnetwork
YOLO CNN Network

YOLO CNN Network

Decide Predictions

Two anchor boxes
- Class: airplane

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

Class: sailboat
Model Exchange with MATLAB

Open Neural Network Exchange
Import Pretrained Network in ONNX Format

```matlab
load resnetClassNames.mat
net = importONNXNetwork('resnet50.onnx', ...
    'OutputLayerType', 'classification', ...
    'ClassNames', classnames);
analyzeNetwork(net)
```
Import Pretrained Network in ONNX Format
Modify Network

```matlab
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

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

```matlab
>> detector = trainYOLOv2ObjectDetector(trainingData, lgraph, options)
```
Evaluate Performance of Trained Network

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

```matlab
>> [ap, recall, precision] = evalDetectPrecision(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

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

MATLAB

Front-end

Control-flow graph
Intermediate representation
(CFG – IR)

Loop optimizations

Traditional compiler optimizations

Library function mapping

Scalarization

CUDA kernel optimizations

Loop perfectization

CUDA kernel creation

Loop interchange

cudaMemcpy minimization

Loop fusion

Shared memory mapping

CUDA code emission

Scalar replacement
Example Used in Today’s Talk

AI Application

Lane Detection Network  Co-ordinate Transform  YOLOv2 Network  Bounding Box Processing

Optimized TensorRT Code for Models
```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
9
frame = imresize(In, [227,227]);
11 persistent lannet;
12 if isempty(lannet)
13     lannet = coder.loadDeepLearningNetwork('LaneDetectionNet.net', 'Lannet');
14 end
15 lanecoeffsNetworkOutput = lannet.predict(frame);
17 % Recover original coefs by reversing the normalization steps
18 laneCoeffsMeans = [-0.0002, 0.0002, 1.4740, 0.0002, 0.0045, -1.8787];
19 laneCoeffsScales = [0.0038, 0.0765, 0.6313, 0.6262, 0.0735, 0.3946];
20 params = lanecoeffsNetworkOutput * laneCoeffsScales + laneCoeffsMeans;
22 % should be more than 0.5 for it to be a lane
24 isRightLaneFound = abs(params(0)) > 0.5;
25 isLeftLaneFound = abs(params(3)) > 0.5;
27 vehiclePoints = zeros(2,6); % vectors, ahead of the sensor
28 leftPoints = coder.multipoly2(xyz(26,2), 'single');
29 rightPoints = coder.multipoly2(xyz(26,2), 'single');
31 % map vehicle to image coordinates
32 if isRightLaneFound && isLeftLaneFound
```

New to MATLAB? See resources for Getting Started.
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]
With GPU Coder, MATLAB is fast

Faster than TensorFlow, MXNet, and PyTorch
TensorRT speeds up inference for TensorFlow and GPU Coder

Single Image Inference with ResNet-50 (Titan V)

<table>
<thead>
<tr>
<th></th>
<th>Images/Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>cuDNN</td>
<td></td>
</tr>
<tr>
<td>TensorFlow</td>
<td>300</td>
</tr>
<tr>
<td>GPU Coder</td>
<td>330</td>
</tr>
<tr>
<td>TensorRT</td>
<td>370</td>
</tr>
</tbody>
</table>

R2019a
GPU Coder with TensorRT faster across various Batch Sizes

ResNet-50 Inference (Titan V)

Images/Sec

Batch Size

1 2 4 8 16 32

GPU Coder + TensorRT
TensorFlow + TensorRT

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)

ResNet-50 Inference (Titan V)

- GPU Coder + TensorRT (int8)
- TensorFlow (int8)
- GPU Coder + TensorRT (fp32)
- TensorFlow + TensorRT

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

MATLAB algorithm (functional reference)

GPU Coder

Build type

Call compiled application from MATLAB directly

Call compiled application from hand-coded main()

.mex

.lib

Desktop GPU

Desktop GPU

Embedded GPU

1 Functional test

2 Deployment unit-test

3 Deployment integration-test

4 Real-time test

Deploy to target and run with hardware-in-loop

Deploy to target
Hardware in the loop workflow with Jetson/DRIVE device

Stream Webcam Images from HW

MATLAB

Run model in MATLAB

Update parameters

Model + Code

Jetson/DRIVE

Deploy and launch on Target hardware

- Generate CUDA and TensorRT code
- Deploy and build on target
- Launch executable on the target.

Results for Verification
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);
    mlttime = toc;

    % Calculate fps

end
Processor in the loop verification with Jetson/Drive devices

```matlab
% Set up connection to Jetson device
hwobj = jetson('gpucoder-xavier-l','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
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