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 35% 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

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

AI Application
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

- Bounding Boxes Labels
- Polylne 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

AI Application

Lane Detection Network

Co-ordinate Transform

YOLOv2 Network

Bounding Box Processing
Lane Detection Algorithm

Pretrained Network (E.g. AlexNet) → Modify Network for Lane Detection → Coefficients of parabola → 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
Specify Training on:

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

Quickly change training hardware

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

Works on Windows (no additional setup)
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

`evaluateLaneBoundaries`

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)

YOLO CNN Network

Detection Subnetwork

Decode Predictions
Model Exchange with MATLAB

ONNX

PyTorch

Caffe2

MXNet

Core ML

CNTK

MATLAB

Keras-Tensorflow

Caffe

Open Neural Network Exchange
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
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

\[
\text{>> } lgraph = \text{yolov2Layers}(\text{imageSize}, \text{numClasses}, \text{anchorBoxes}, \text{network}, \text{featureLayer})
\]

\[
\text{>> detector = trainYOLOv2ObjectDetector(trainingData, lgraph, options)}
\]
Evaluate Performance of Trained Network

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

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

MATLAB

Front-end

Control-flow graph
Intermediate representation
(CFG – IR)

Traditional compiler optimizations

Libray function mapping
Scalarization
Loop perfectization
Loop interchange
Loop fusion
Scalar replacement

Parallel loop creation
CUDA kernel creation
cudaMemcpy minimization
Shared memory mapping
CUDA code emission

Loop optimizations

CUDA kernel optimizations
Example Used in Today’s Talk

AI Application

Lane Detection Network

Co-ordinate Transform

YOLOv2 Network

Bounding Box Processing

Optimized TensorRT Code for Models
Code generation workflow (demo)

Deployment unit

Desktop

GPU .mex

Build type

Call compiled application from MATLAB directly
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

Single Image Inference (Titan V, Linux)

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

Single Image Inference with ResNet-50 (Titan V)

R2019a

![Comparison chart showing the performance of cuDNN and TensorRT in images/sec](chart.png)

- **TensorFlow**
- **GPU Coder**
GPU Coder with TensorRT faster across various Batch Sizes

ResNet-50 Inference (Titan V)

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA 10 - 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

Images/Sec

Batch Size

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

1. Functional test
2. Deployment unit-test
3. Deployment integration-test
4. Real-time test

Deploy to target and run with hardware-in-loop
Hardware in the loop workflow with Jetson/DRIVE device

Stream Webcam Images from HW → MATLAB
Run model in MATLAB → Model + Code
Update parameters

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