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

- **Veoneer**
  - Radar Sensor Verification

- **Alpine**
  - Ground Detection

- **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

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 =
```

```
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
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
Transparently Scale Compute for Training

Specify Training on:

'CPU'
'gpu'
'multi-gpu'

Quickly change training hardware

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](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

23
YOLO v2 Object Detection

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

YOLO CNN Network

Detection Subnetwork

Decode Predictions

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

Two anchor boxes

Class: airplane
Class: sailboat
Model Exchange with MATLAB

Open Neural Network Exchange

- PyTorch
- ONNX
- Caffe2
- MXNet
- Core ML
- CNTK
- Keras-Tensorflow
- MATLAB
- Caffe

MathWorks
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

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

\[
> \text{detector} = \text{trainYOLOv2ObjectDetector}(	ext{trainingData}, lgraph, \text{options})
\]
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
- Loop optimizations
- CUDA kernel optimizations

### Library function mapping
- Scalarization
- Loop perfectization
- Loop interchange
- Loop fusion
- Scalar replacement

### CUDA kernel optimizations
- Parallel loop creation
- CUDA kernel creation
- cudaMemcpy minimization
- Shared memory mapping
- CUDA code emission
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
TensorRT speeds up inference for TensorFlow and GPU Coder

Single Image Inference with ResNet-50 (Titan V)

R2019a

Images/Sec

TensorFlow

GPU Coder

cuDNN

TensorRT
GPU Coder with TensorRT faster across various Batch Sizes
Even higher Speeds with Integer Arithmetic (int8)

ResNet-50 Inference (Titan V)

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

Graph showing images per second vs. batch size.
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

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

MATLAB algorithm (functional reference)

GPU Coder

Build type

Call compiled application from MATLAB directly

Call compiled application from hand-coded main()

Deploy to target

Deploy to target and run with hardware-in-loop

Desktop GPU

Deploy to target

Embedded GPU

.C++
Hardware in the loop workflow with Jetson/DRIVE device

- Stream Webcam Images from HW
- Run model in MATLAB
- Update parameters

MATLAB

- Model + Code
- Results for Verification

Jetson/DRIVE

- Deploy and launch on Target hardware
- Generate CUDA and TensorRT code
- Deploy and build on target
- Launch executable on the target.
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
    end
end
Processor in the loop verification with Jetson/Drive devices

```matlab
% Set up connection to Jetson device
hwobj = jetson('gupcoder-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
cfg.DeepLearningConfig = coder.DeepLearningConfig('tensortt');
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