embedded VISION SUMMIT 2018

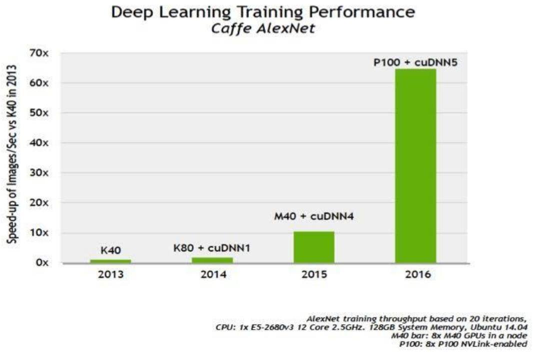
Deep Learning in MATLAB: From Concept to Optimized Embedded Code

Girish Venkataramani, Avinash Nehemiah
MathWorks

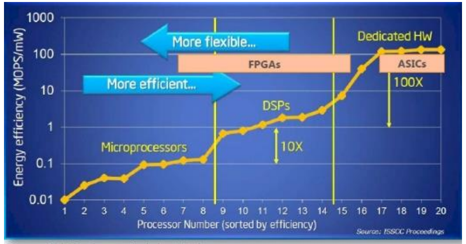
23 May 2018

Embedded Hardware and Deep Learning

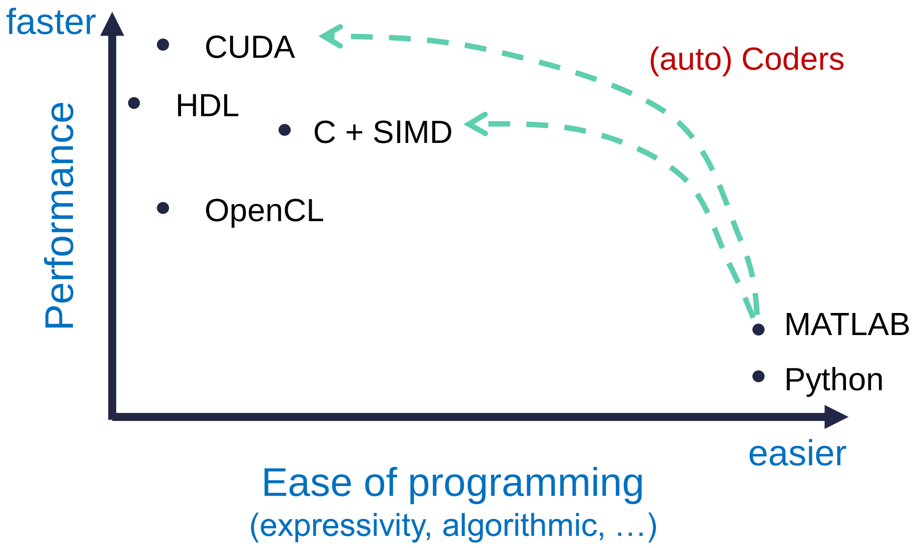
GPUs for raw performance



CPUs, DSPs, FPGAs for energy-efficiency



Source: Bob Broderson, Berkeley Wireless group



GPUs, DSPs, CPUs, FPGAs.
Many choices for deep learning

but, programming them is hard

Example: saxpy

Scalarized MATLAB

```
for i = 1:length(x)
    z(i) = a .* x(i) + y(i);
end
```

C + SIMD

```
void saxpy(float* y, float* x, size_t length, floa
{
  size_t n;
  // execute op on array elements block-wise
  for(n = 0; n < ROUND_DOWN(length, op.BLOCK_SIZE)
    op.block(dst + n, src + n);

  // execute the remaining array elements one by o
  for(; n < length; n++) op(dst + n, src + n);
}
```

Coder

Vectorized MATLAB

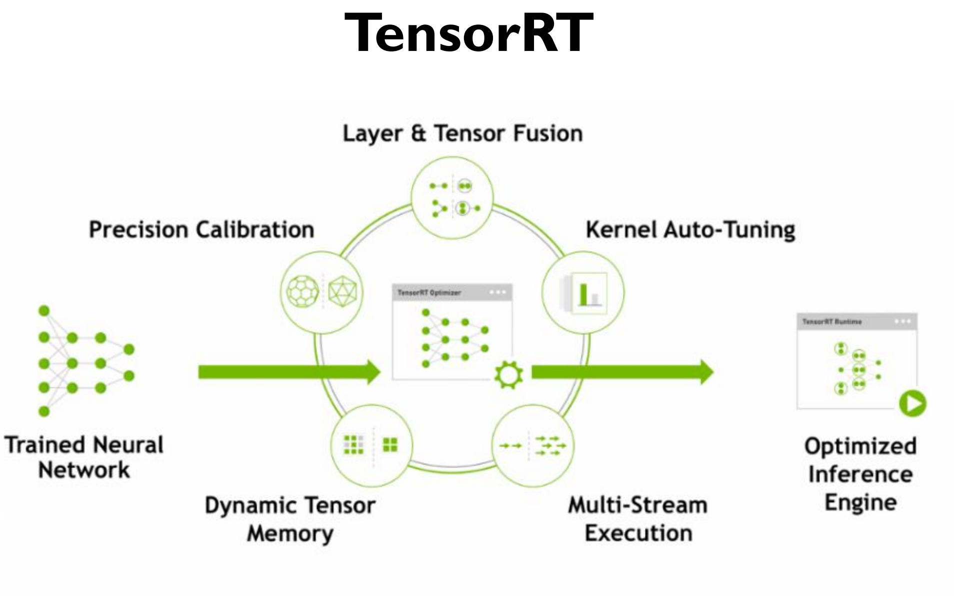
```
z = a .* x + y;
```

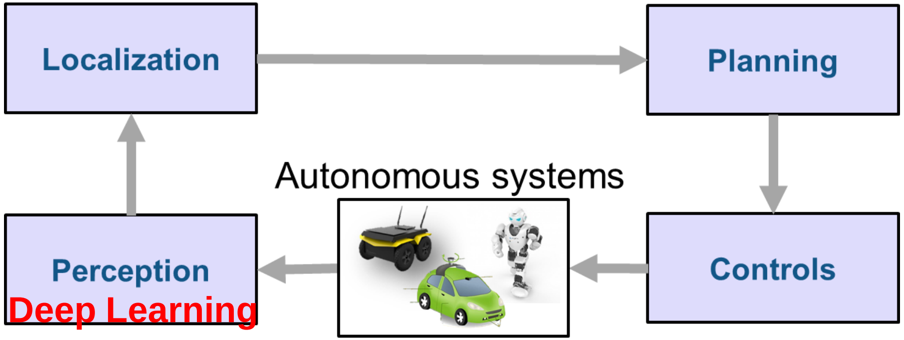
C + CUDA

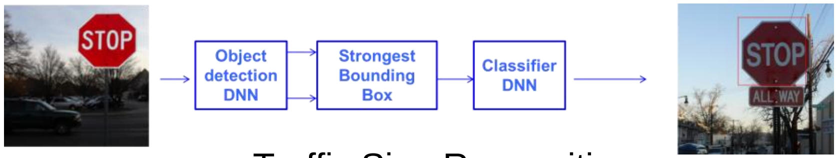
```
void saxpy(real32_T a, const real32_T x[1048576], const real
           real_T z[1048576])
{
  real32_T *gpu_y;
  real32_T *gpu_x;
  real_T *gpu_z;
  cudaMalloc(&gpu_z, 8388608UL);
  cudaMalloc(&gpu_x, 4194304UL);
  cudaMalloc(&gpu_y, 4194304UL);
  cudaMemcpy((void *)gpu_y, (void *)&y[0], 4194304UL, cudaMe
  cudaMemcpy((void *)gpu_x, (void *)&x[0], 4194304UL, cudaMe
  saxpy_kernel1<<<dim3(2048U, 1U, 1U), dim3(512U, 1U, 1U)>>>
    gpu_z);
```

Automatic compilation from an expressive language to a high-performance language

Inference Engines Alone are Insufficient



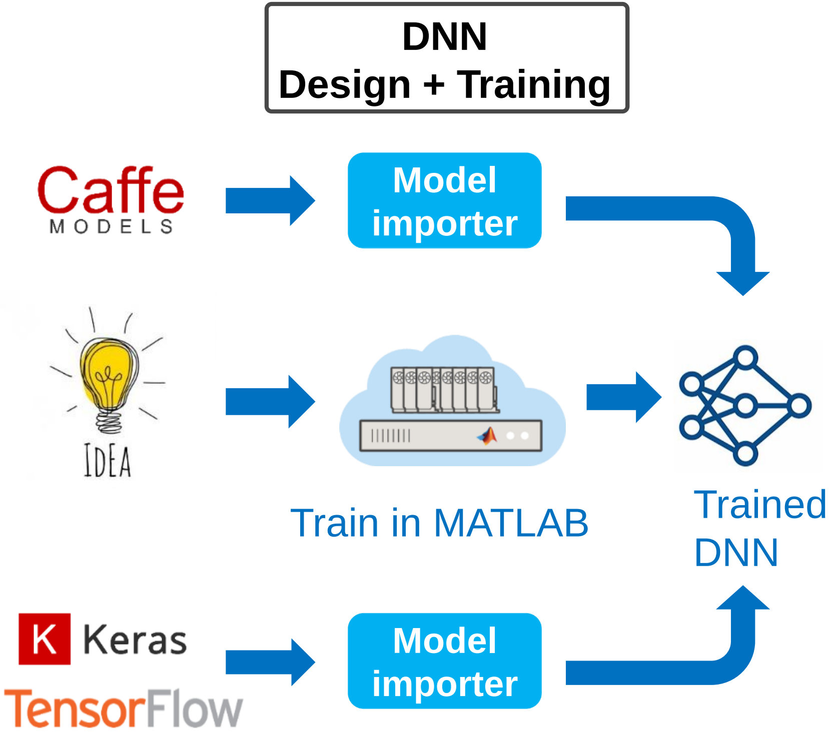




Traffic Sign Recognition

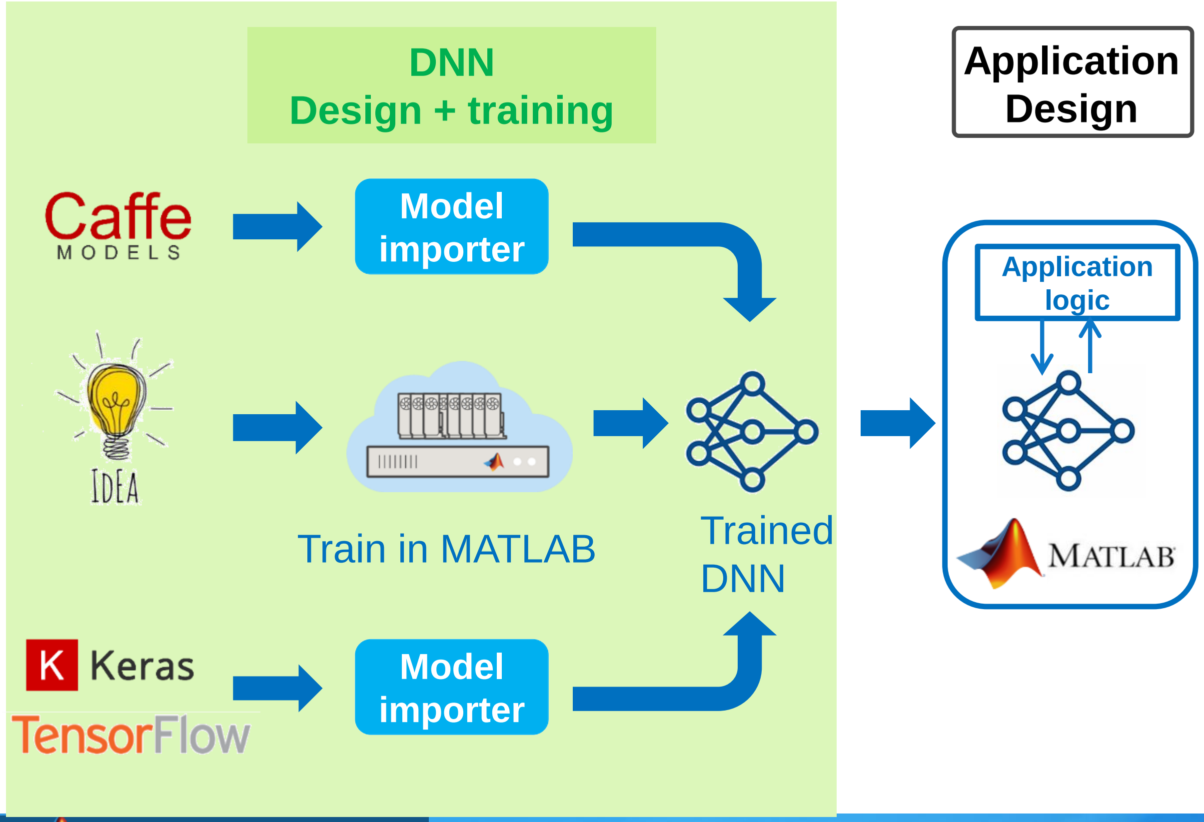
TensorRT is great for inference, … but, applications require more than inference

Deep Learning Workflow in MATLAB



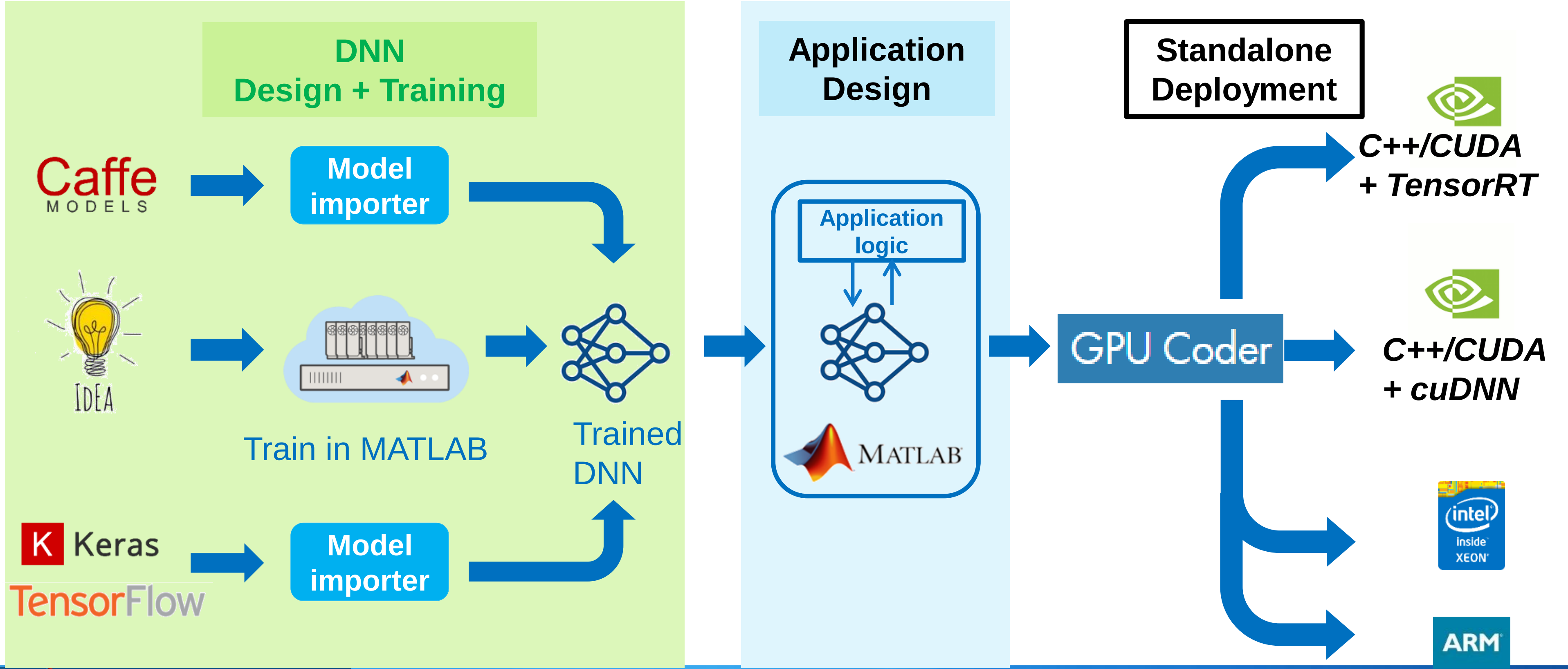
- **Design in MATLAB**
  - **Manage** large image sets
  - **Automate** data labeling
  - **Easy access** to models
- **Training in MATLAB**
  - **Acceleration** with GPU's
  - **Scale** to clusters

Deep Learning Workflow in MATLAB

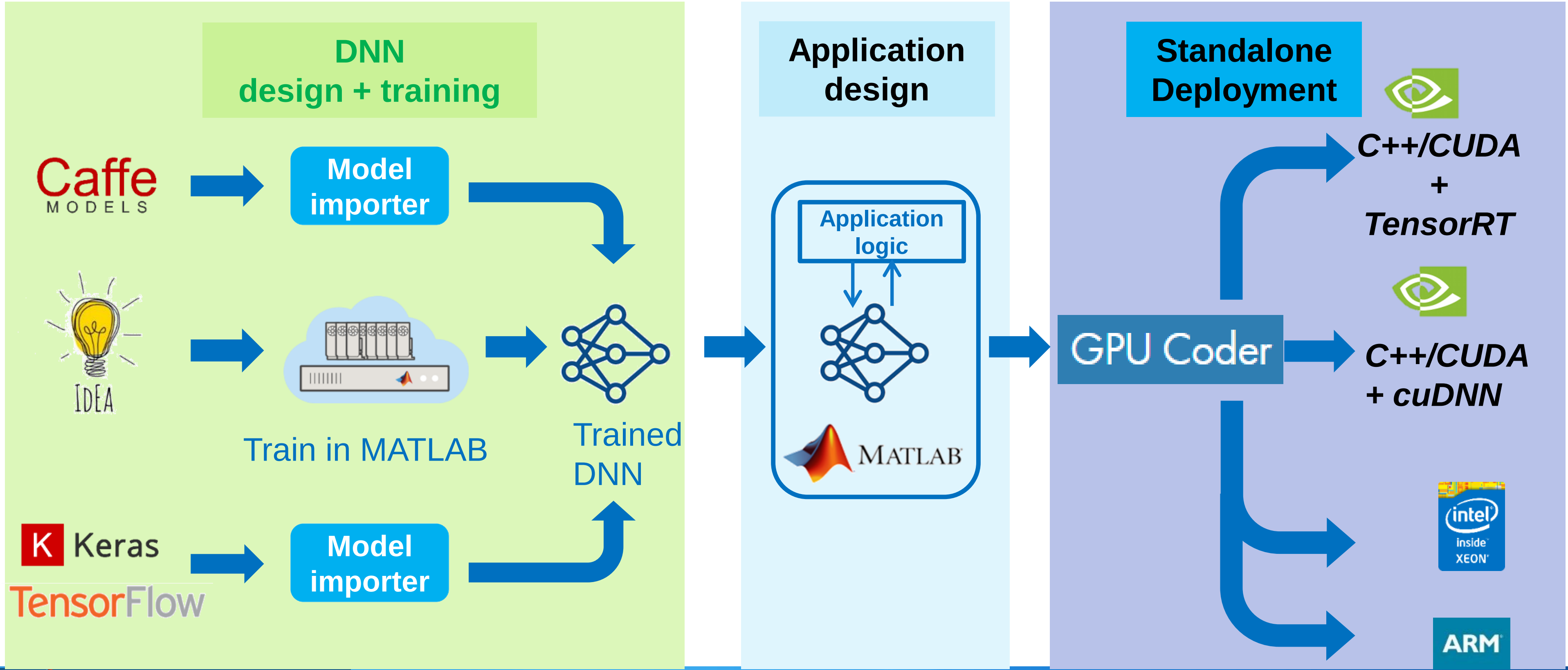


- **MATLAB for application design**
  - **Write arbitrary MATLAB**
  - **Interact with trained net**
  - **Use power of toolboxes (image, vision, signal)**
  - **MATLAB as a test-harness**

Deep Learning Workflow in MATLAB



Deep Learning Workflow in MATLAB



Coders: Easy, Fast and Efficient Deployment

MATLAB

SIMULINK

Unified compiler framework

20+ years

Embedded Coder

Simulink Coder

MATLAB Coder

Deep learning

C, C++

R2006b

HDL Coder

VHDL
Verilog

Deep learning

R2017b

GPU Coder

CUDA

Easy, fast and efficient in so many domains:

- Control systems
- Signal processing and communications
- Image processing and computer vision
- **Deep learning**

GPU Coder Released in September 2017

NEW

R2017b

MATLAB → GPU Coder → NVIDIA CUDA C/C++

Accelerated implementation of parallel algorithms on GPUs

Neural Networks

Deep Learning, machine learning

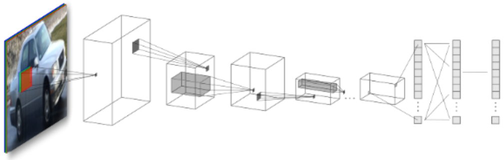
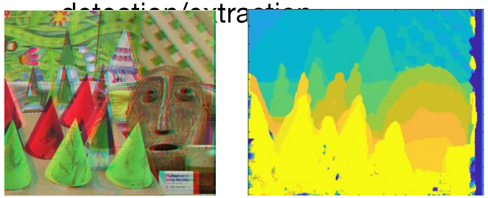


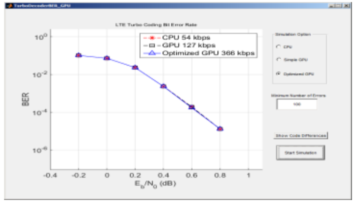
Image Processing and Computer Vision

Image filtering, feature detection/extraction



Signal Processing and Communications

FFT, filtering, cross correlation,



Talk outline

- Introduction
- Code generation
  - Parallel compilation
  - Processor re-targetability
- Deep learning performance benchmarks
- Application demo: LiDAR semantic segmentation

Coders Extract Parallelism in MATLAB

1. Scalarized MATLAB
   ("for-all" loops)

```
%% Pixel processing on the height/width of an image
for i = 1:height
    for j = 1:width
        tmpVal = (width*height);
        for x = 1:width
            disValTmp = temp(x,i);
            dist = ((j-x)^2 + disValTmp^2 );
            if (dist < tmpVal)
                tmpVal = single(dist);
            end
        end
        out(i,j) = tmpVal;
    end
end
```

2. Vectorized MATLAB
   (math operators and library functions)

```
%% Parallel element-wise math to compute
%  Restoration with inverse Koschmieder's law
factor=1.0./(1.0-(diff_im));
restoreOut(:,:,1)= (input(:,:,1)-diff_im).*factor;
restoreOut(:,:,2)= (input(:,:,2)-diff_im).*factor;
restoreOut(:,:,3)= (input(:,:,3)-diff_im).*factor;
```

3. Composite functions in MATLAB
   (maps to cuBlas/mkl-Blas, cuFFT/fftw, lapack/cuSolver, mkl-dnn/cuDNN/TensorRT)

```
C = A * B; % cuBLAS

y = fft(in); % cuFFT

z = A \ B; % cuSolver
```

From composite functions to optimized code

embedded VISION SUMMIT 2018

Core math

- Matrix multiply (cuBLAS, mkl-BLAS)
- Linear algebra (cuSolver, LAPACK)
- FFT functions (cuFFT, fftw)
- Convolution
- …

Image processing
Computer vision

- imfilter
- imresize
- imerode
- imdilate
- bwlabel
- imwarp
- …

Neural Networks

- Deep learning inference (cuDNN, TensorRT, ARM Compute, mkl-DNN)
- …

Over 300+ MATLAB functions are optimized for code generation

Coders are Powerful Parallelizing Compilers

MATLAB

Front – end

Control-flow graph
Intermediate representation
(CFG – IR)

.

Traditional compiler optimizations

:

Library function mapping

Loop optimizations

- Scalarization
- Loop perfectization
- Loop interchange
- Loop fusion
- Scalar replacement

Parallel kernel optimizations

- Parallel loop creation
- Parallel kernel creation
- memcpy minimization
- Shared memory mapping

Code emission → *C, C++, CUDA*

Coder Workflow (for GPUs, CPUs)

MATLAB

↓

GPU Codegen

↓

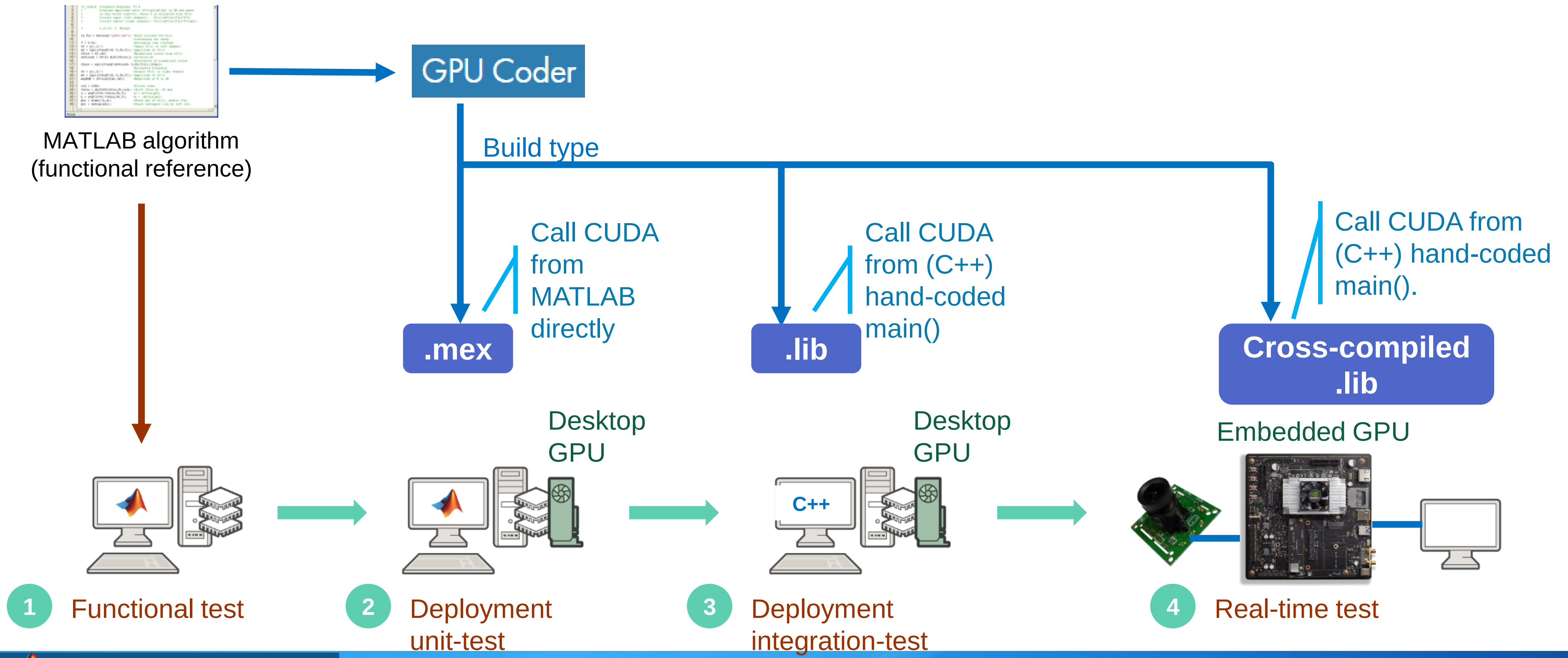
.c, .cu source code, build scripts, Makefile

‘exe’
Stand-alone executable

‘mex’
Call directly from MATLAB

‘lib’ or ‘dll’
Shared library

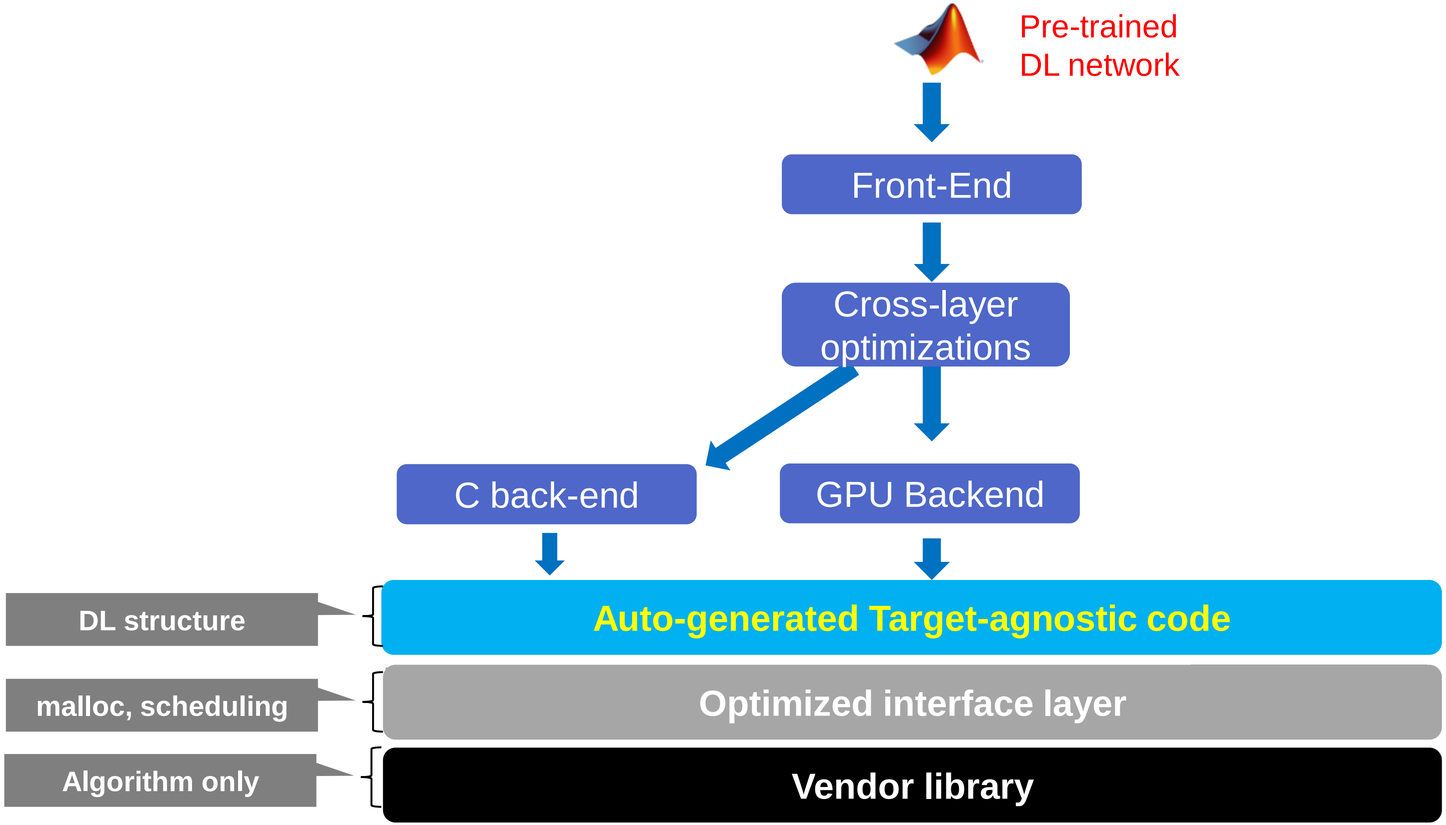
Algorithm Design to Embedded Deployment Workflow



Example: Traffic Sign Detection and Recognition

→ Object detection DNN ⇉ Strongest Bounding Box → Classifier DNN →

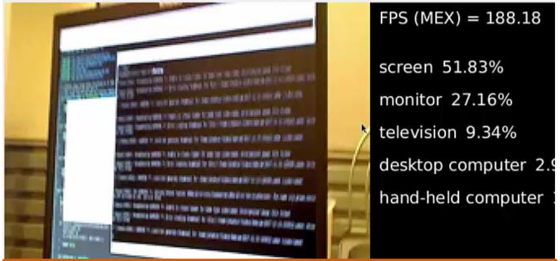
Deep Learning Re-targetability



Talk outline

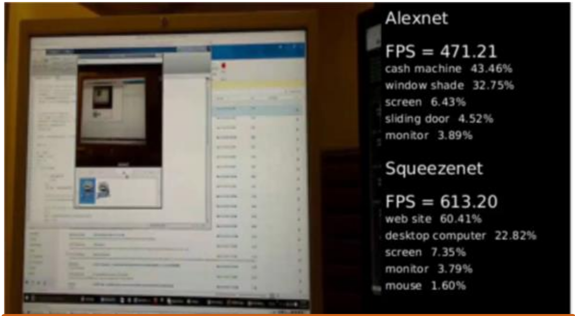
- Introduction
- Code generation
  - Parallel compilation
  - Processor re-targetability
- Deep learning performance benchmarks
- Application demo: LiDAR semantic segmentation

MATLAB and GPU Coder Support State-of-Art Classification Networks



GoogLeNet

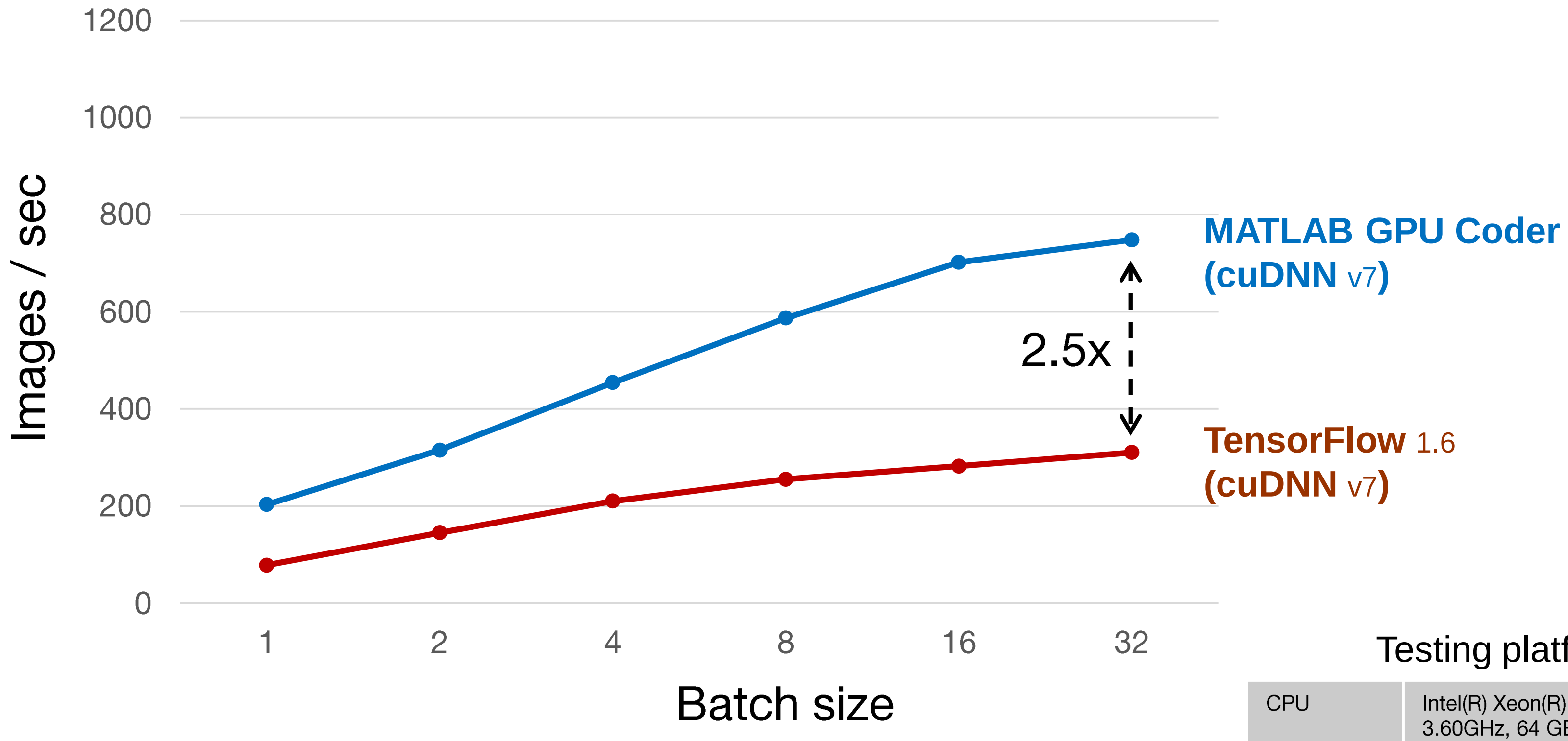
ResNet50



Alexnet vs Squeezenet

Network	# Layers	Size	Frame-rate (GPU Coder)
Alexnet	25	227 MB	500 Fps
ResNet50	177	96 MB	160 Fps
GoogLeNet	144	27 MB	190 Fps
Squeezenet	68	5 MB	615 Fps

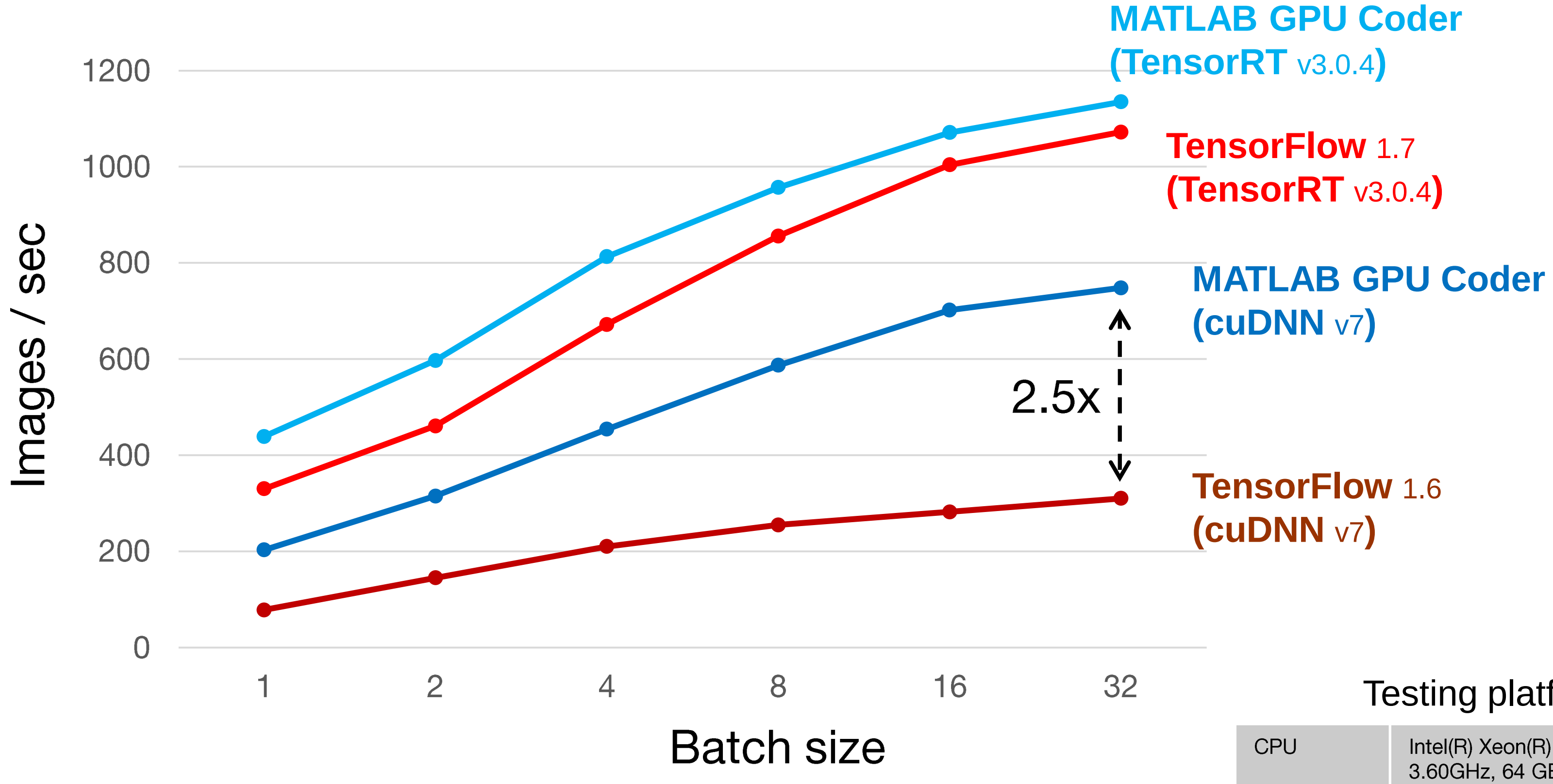
Desktop GPU Benchmarking: Resnet-50



Testing platform

CPU	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz, 64 GB RAM
GPU	Pascal Titan Xp, 12 GB RAM

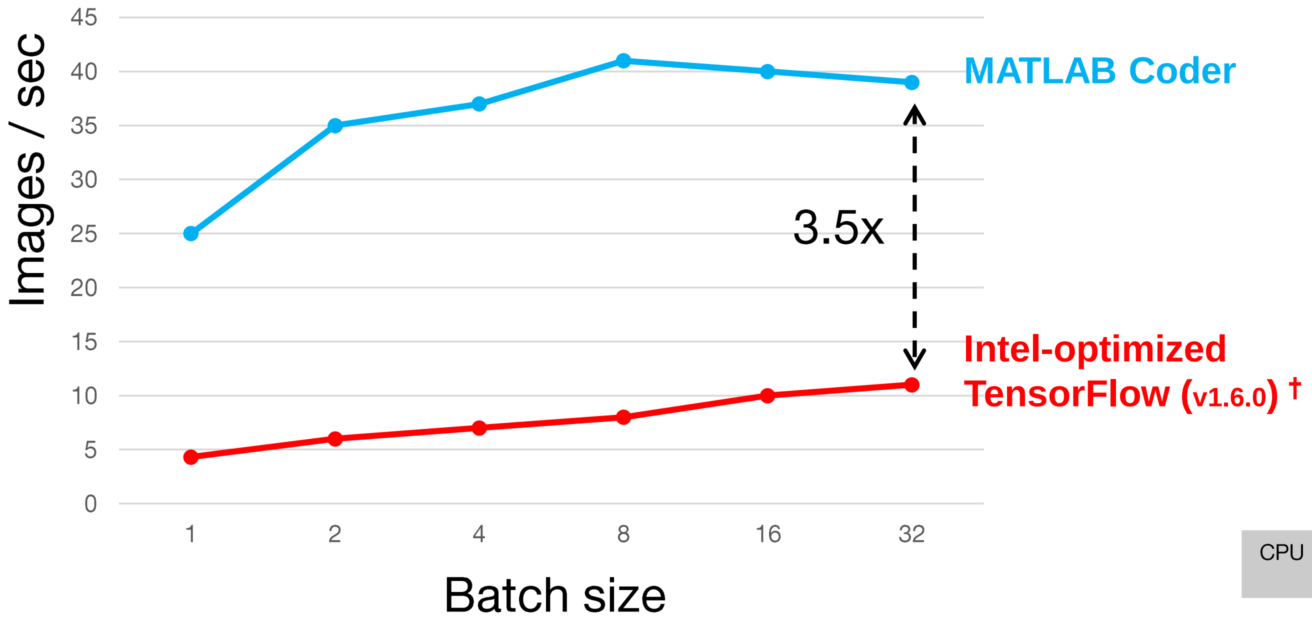
Desktop GPU Benchmarking: Resnet-50



Testing platform

CPU	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz, 64 GB RAM
GPU	Pascal Titan Xp, 12 GB RAM

Desktop CPU Benchmarking: Resnet-50



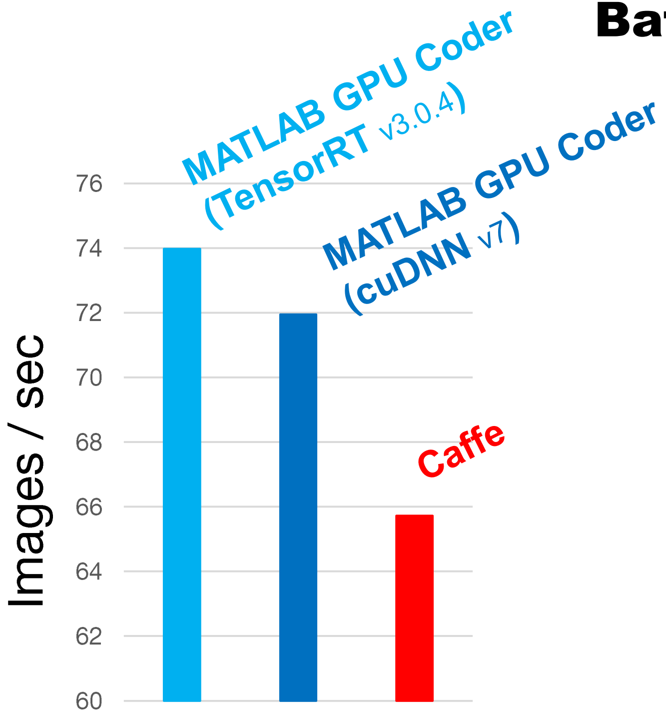
Testing platform

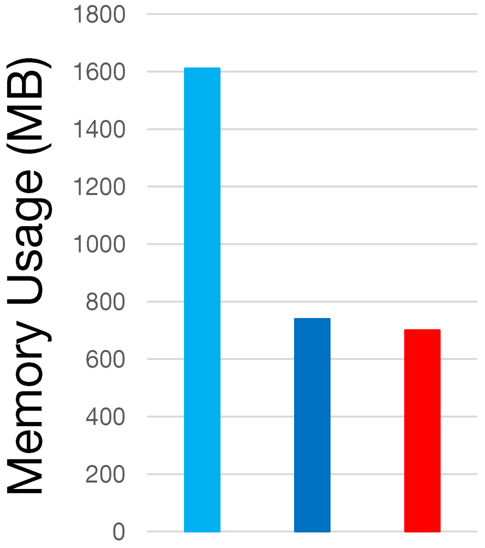
CPU	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz, 64 GB RAM

† https://software.intel.com/en-us/articles/intel-optimized-tensorflow-installation-guide

Embedded GPU Benchmarking: Jetson TX2

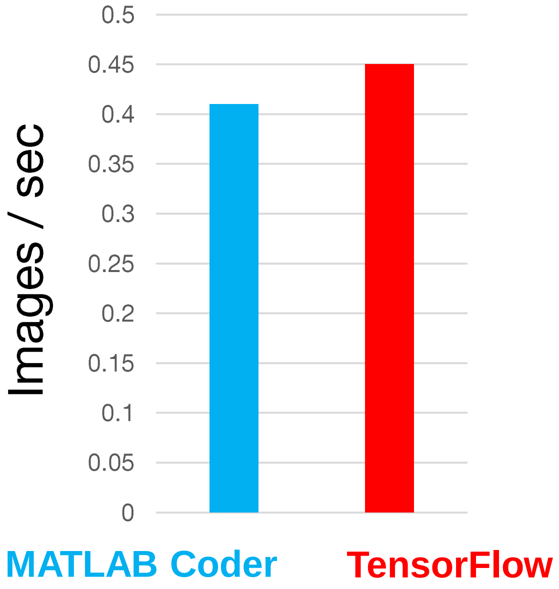
Batch size = 1

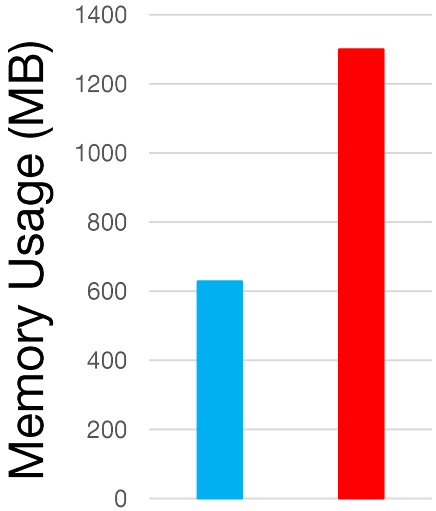




Embedded benchmarking: RPi3 (ARM)

Batch size = 1





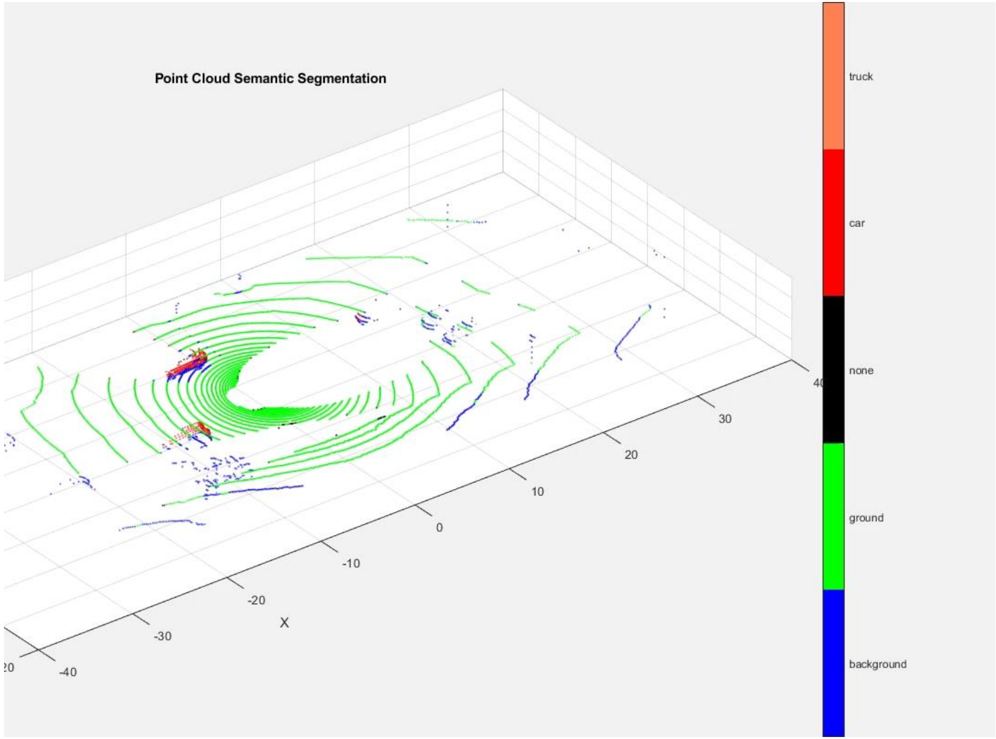
Testing platform

CPU	Raspberry Pi 3 quad core ARM Cortex @1.2GHz, 1GB RAM
ARM CL	V18.03

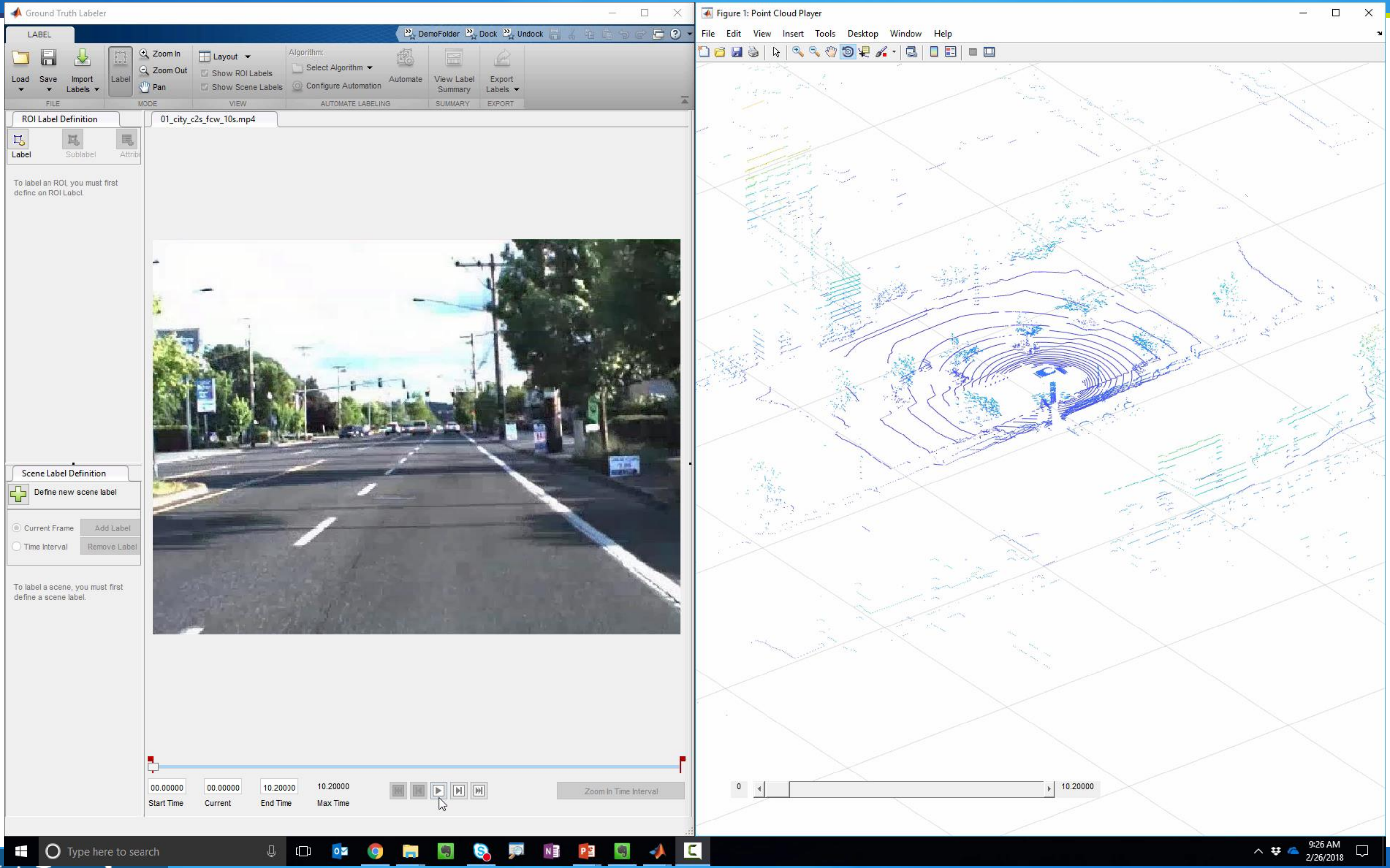
Talk outline

- Introduction
- Code generation
  - Parallel compilation
  - Processor re-targetability
- Deep learning performance benchmarks
- Application demo: LiDAR semantic segmentation

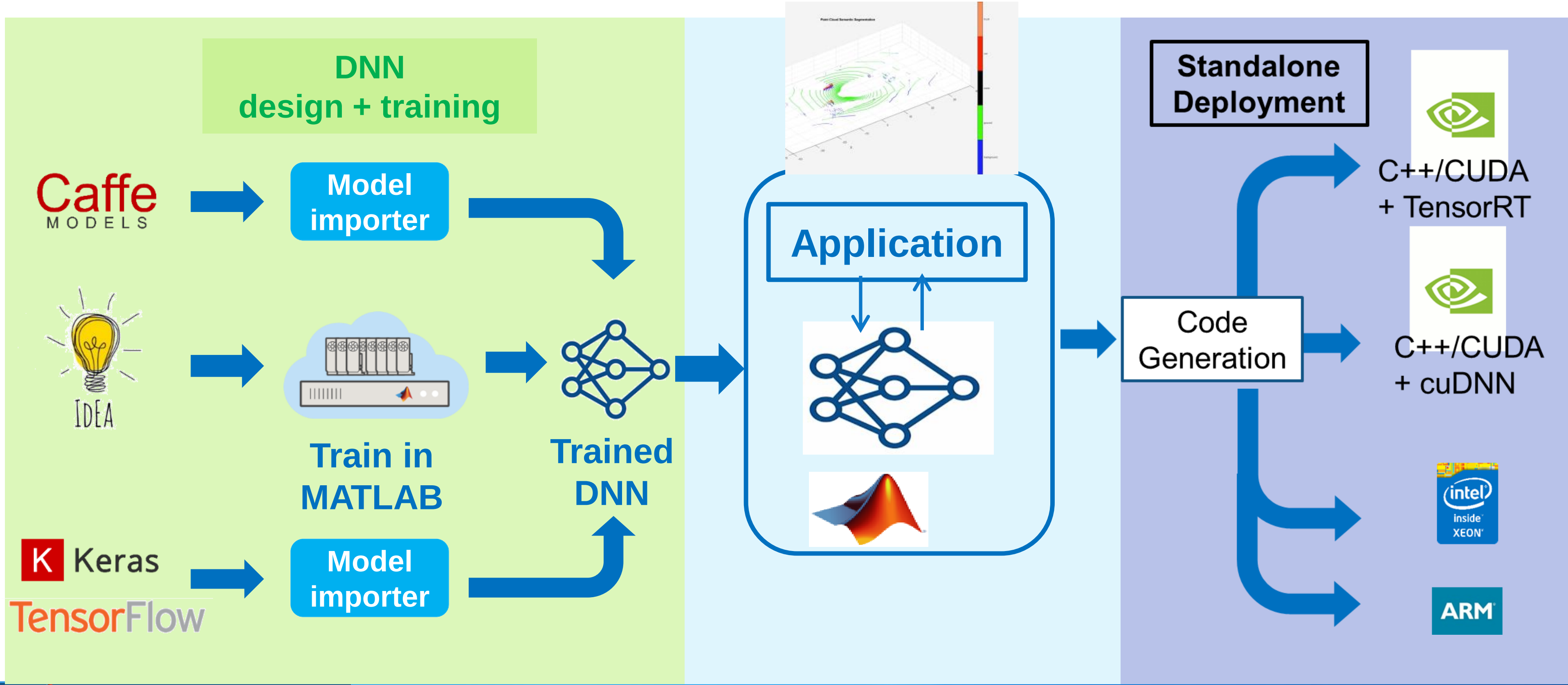
Example: Lidar Semantic Segmentation



What Does Lidar Data Look Like ?

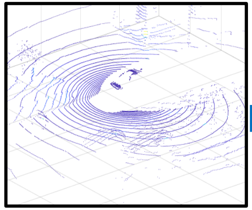


Deep Learning Workflow in MATLAB

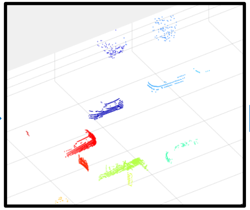


Data Preparation and Labeling

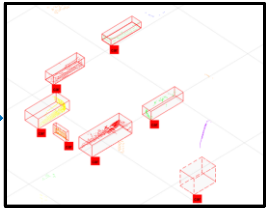
DNN design + training



Access Data



Pre-Process

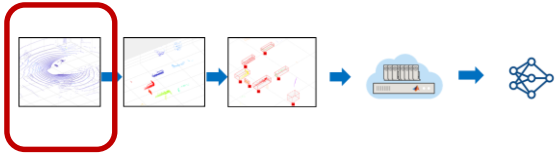


Label Data

Train

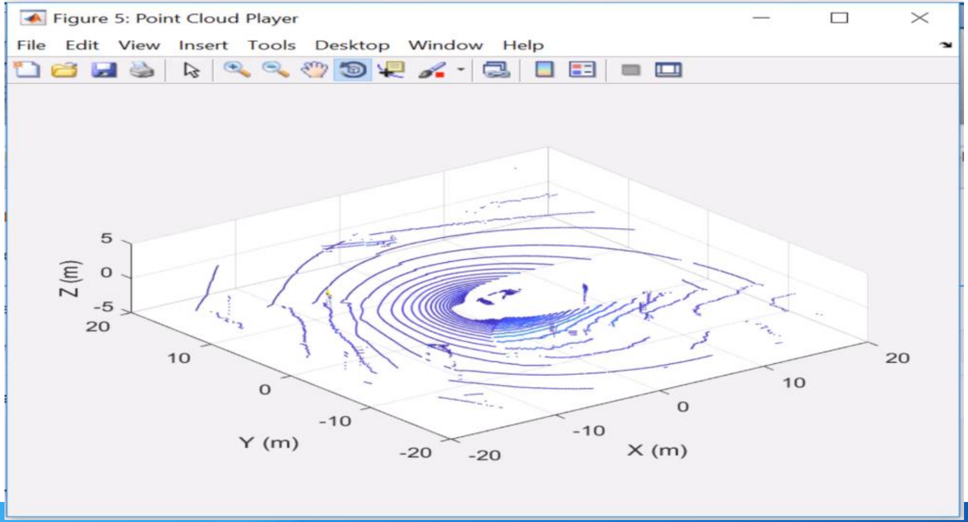
Trained DNN

Access and Visualize Lidar Data

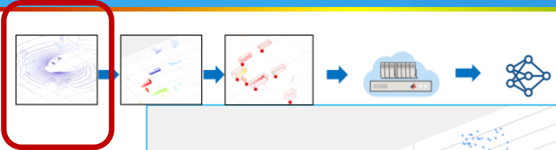


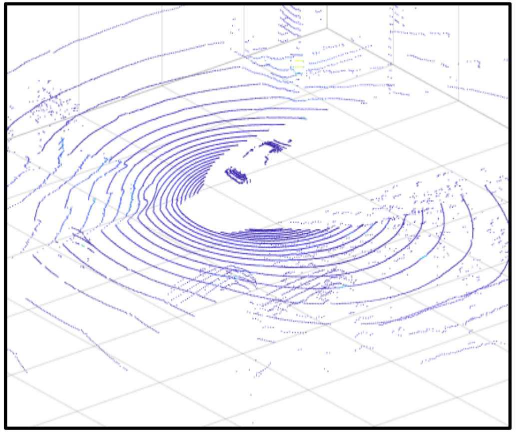
Access Stored Lidar Data
- Velodyne file I/O (pcap)
- Individual point clouds (.pcd,ply)

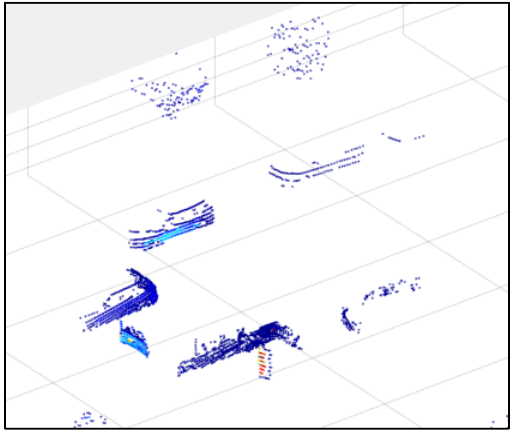
Visualize Lidar Data
- Streaming Lidar player
- Static point cloud display
- Point cloud differences

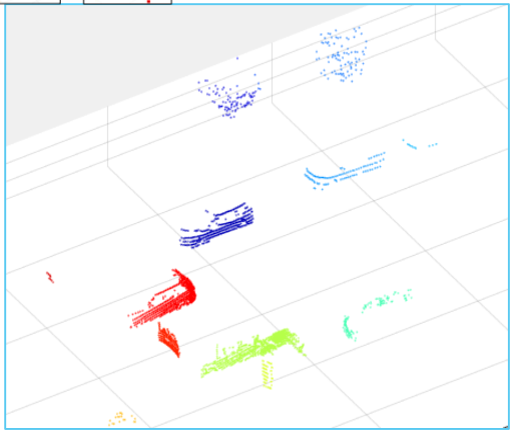


Lidar Preprocessing









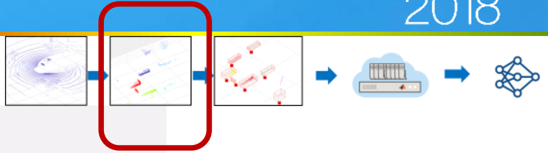
Remove Ground

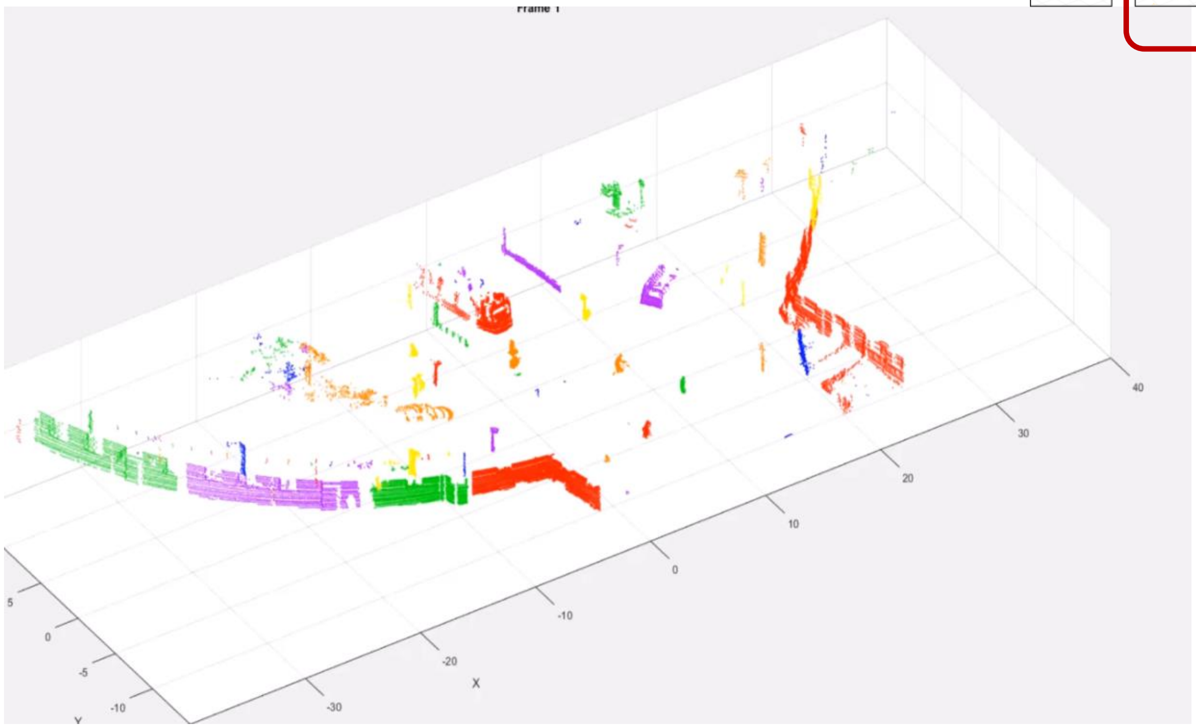
- Fit plane using RANSAC

Cluster

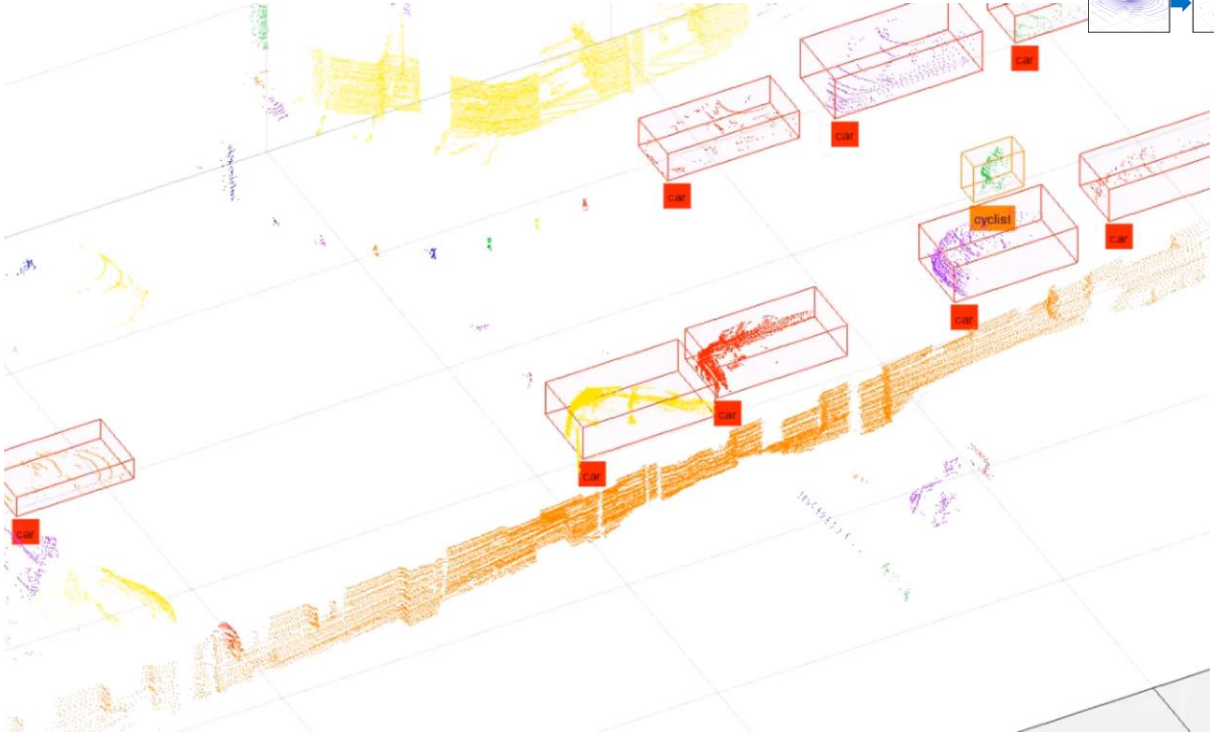
- Segment clusters using Euclidean distance

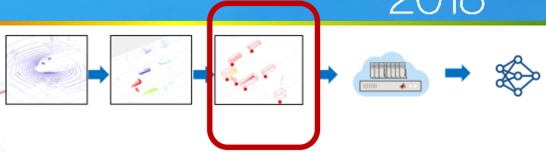
Ground Truth Labeling of Lidar Data



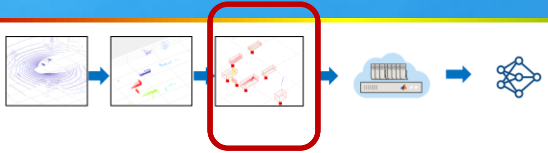


Ground Truth Labeling of Lidar Data





Organize Data for Training



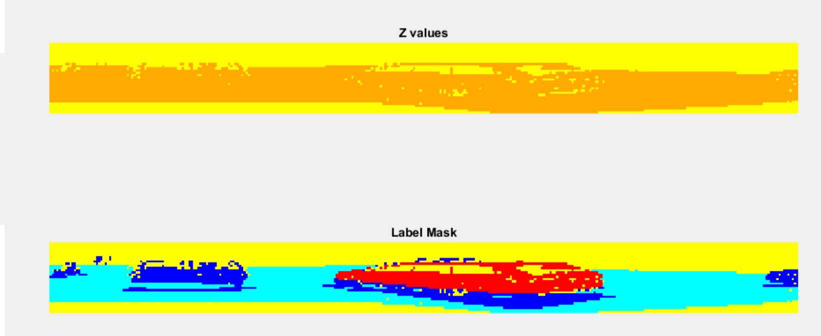
Raw Point Cloud Data

32x1085 double

	1	2	3	4
1	NaN	0.1761	0.2727	0.3
2	0.0770	0.1932	0.3064	0.4
3	NaN	NaN	0.2822	0.4
4	0.0834	0.2183	0.2875	0.3
5	NaN	NaN	NaN	N
6	0.0791	0.2409	0.4033	0.5
7	0.0745	0.2363	0.3987	0.5
8	0.0651	0.2266	0.3885	0.5
9	0.0559	0.2177	0.3799	0.5
10	0.0512	0.2130	0.3749	0.5
11	0.0419	0.2037	0.3657	0.5
12	[illegible]	[illegible]	[illegible]	[illegible]

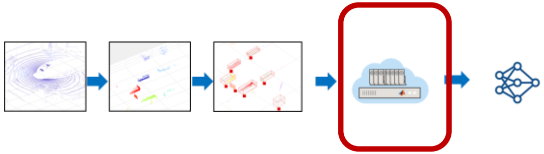
Project to 2D

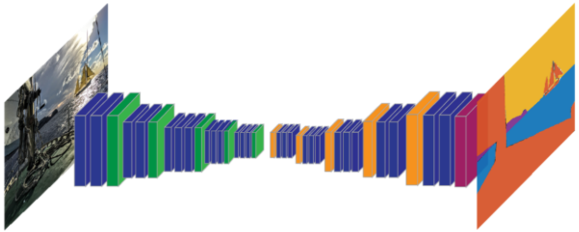
Ground Truth Labels Transformed to Label Mask



MathWorks

Create Network Architecture



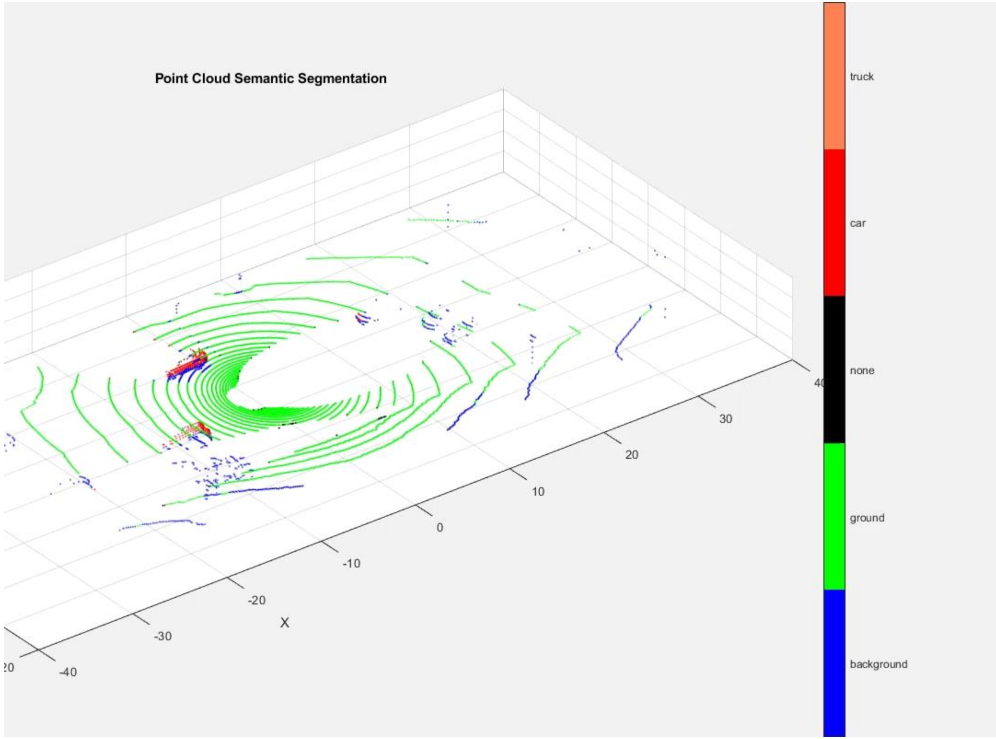


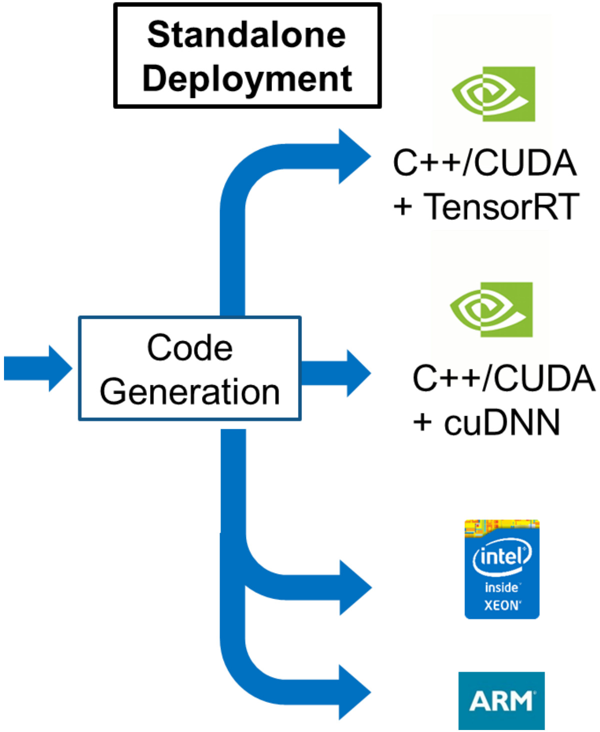
Easy MATLAB API to create network

```
%build encoder
nOutputs = 64;
inputLayerName = 'init_maxpool';
for blockIdx = 1:encoderDepth
    [lGraph, layerOutName] = encoderBlock(lGraph, blockIdx, nOutputs, inputLayerName
    nOutputs = nOutputs * 2;
    inputLayerName = layerOutName;
end

%build decoder
nInputs = nOutputs;
inputLayerName = layerOutName;
for blockIdx = encoderDepth:-1:1
    nOutputs = min(nInputs/2, 64);
    [lGraph, decoderLayerOutName] = decoderBlock(lGraph, blockIdx, nInputs, nOutputs
    if blockIdx ~= 1
        inputLayerName = ['res_add' num2str(blockIdx)];
        lGraph = addLayers(lGraph, additionLayer(2, 'Name', inputLayerName) );
        lGraph = connectLayers(lGraph, ['enc' num2str(blockIdx-1) '_addout'],  [inpu
        lGraph = connectLayers(lGraph, decoderLayerOutName,  [inputLayerName '/in1']
    end
    nInputs = nInputs/2;
end
```

Deploy Model to Multiple Platforms

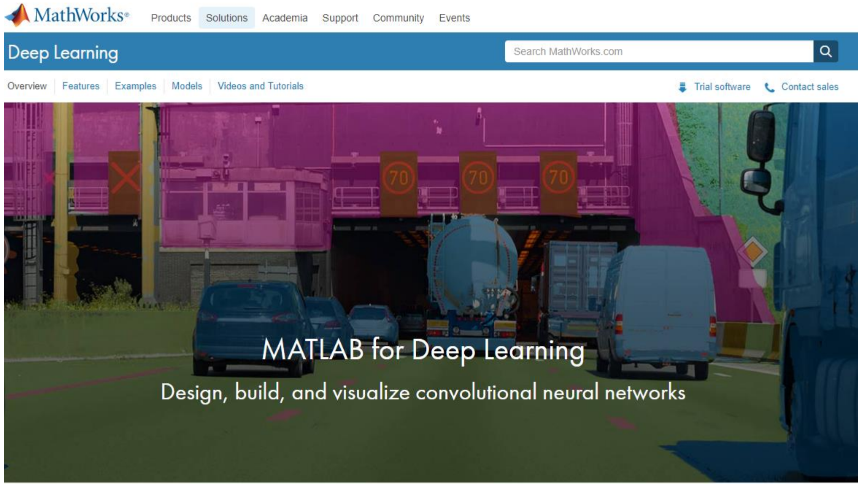


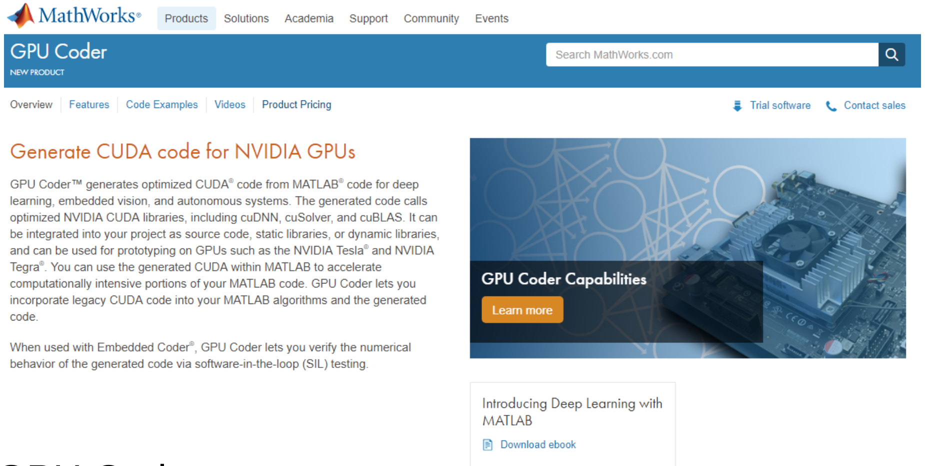


Deep Learning: MATLAB Supports the Full Workflow

1. High performance deep learning inference
2. Prototyping to production workflow
3. Automate ground truth labeling
4. Domain specific pre-processing
5. Post-processing and data analysis

Check Out Deep Learning in MATLAB and GPU Coder





GPU Coder
https://www.mathworks.com/products/gpu-coder.html

Deep learning in MATLAB
https://www.mathworks.com/solutions/deep-learning.html

Deep learning On-Ramp: Free self-paced, online training
https://matlabacademy.mathworks.com/R2017b/portal.html?course=deeplearning