MATLAB EXPO 2019

Deploying Deep Neural Networks to Embedded GPUs and CPUs

Steven Thomsett
Deep Learning Workflow in MATLAB

Deep Neural Network Design + Training

Trained DNN

Application Design

Application logic

Standalone Deployment
Deep Neural Network Design and Training

- Design in MATLAB
  - Manage large data sets
  - Automate data labeling
  - Easy access to models

- Training in MATLAB
  - Acceleration with GPU’s
  - Scale to clusters

MATLAB EXPO 2019
Application Design

Pre-processing → [Diagram] → Post-processing
Multi-Platform Deep Learning Deployment
Multi-Platform Deep Learning Deployment

Application logic

Desktop
NVIDIA Jetson
Raspberry pi

Data Center
 Mobile
Beaglebone

Embedded
Algorithm Design to Embedded Deployment Workflow
Conventional Approach

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

Challenges
- Integrating multiple libraries and packages
- Verifying and maintaining multiple implementations
- Algorithm & vendor lock-in

High-level language
Deep learning framework
Large, complex software stack

C/C++
Low-level APIs
Application-specific libraries

C/C++
Target-optimized libraries
Optimize for memory & speed
Solution: Use MATLAB Coder & GPU Coder for Deep Learning Deployment

MATLAB Coder

GPU Coder

Target Libraries

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM NEON Compute Library
Solution: Use MATLAB Coder & GPU Coder for Deep Learning Deployment
Deep Learning Deployment Workflows

INFERENCE ENGINE DEPLOYMENT

Pre-processing

Trained DNN

cnncodegen

Portable target code

INTEGRATED APPLICATION DEPLOYMENT

Post-processing

Trained DNN

codegen

Portable target code
Workflow for Inference Engine Deployment

Steps for inference engine deployment

1. Generate the code for trained model
   \[ \text{cnncodegen(net, 'targetlib', 'arm-compute')} \]

2. Copy the generated code onto target board

3. Build the code for the inference engine
   \[ \text{make -C ./codegen -f ...mk} \]

4. Use hand written main function to call inference engine

5. Generate the exe and test the executable
   \[ \text{make -C ./ ....} \]
Deep Learning Inference Deployment

Pedestrian Detection

MATLAB Coder

Target Libraries
- NVIDIA TensorRT & cuDNN Libraries
- Intel MKL-DNN Library
- ARM Compute Library
Deep Learning Inference Deployment

Blood Smear Segmentation

Frame Rate: 111.11
Background
Parasited cells
Good cells

MATLAB Coder

Target Libraries

NVIDIA TensorRT & cuDNN Libraries
Intel MKL-DNN Library
ARM Compute Library

MATLAB EXPO 2019
Deep Learning Inference Deployment

Target Libraries

- NVIDIA TensorRT & cuDNN Libraries
- Intel MKL-DNN Library
- ARM NEON Compute Library

GPU Coder

Defect Classification & Detection

MATLAB EXPO 2019
How is the Performance?
Performance of Generated Code

- CNN inference (ResNet-50, VGG-16, Inception V3) on Titan V GPU
- CNN inference (ResNet-50) on Jetson TX2
- CNN inference (ResNet-50, VGG-16, Inception V3) on Intel Xeon CPU
Single Image Inference on Titan V using cuDNN

**Frameworks:**
- **TensorFlow** (1.13.0)
- **MXNet** (1.4.0)
- **GPU Coder** (R2019a)
- **PyTorch** (1.0.0)
Even Stronger Performance with INT8 using TensorRT

ResNet-50 Inference (Titan V)

- GPU Coder + TensorRT (INT8)
- TensorFlow + TensorRT (INT8)

- GPU Coder + TensorRT (FP32)
- TensorFlow + TensorRT (FP32)

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7.4.1 – TensorRT 5.0.2.6 - Frameworks: TensorFlow 1.13.0_rc0
Single Image Inference on Jetson TX2

ResNet-50

<table>
<thead>
<tr>
<th>Images per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>35</td>
</tr>
<tr>
<td>30</td>
</tr>
</tbody>
</table>

- TensorFlow + TensorRT
- GPU Coder + TensorRT

NVIDIA libraries: CUDA9 - cuDNN 7 – TensorRT 3.0.4 - Frameworks: TensorFlow 1.12.0
CPU Performance

CPU, Single Image Inference (Linux)

- **MATLAB**
- **TensorFlow**
- **MXNet**
- **MATLAB Coder**
- **PyTorch**

Intel® Xeon® CPU 3.6 GHz - Frameworks: TensorFlow 1.6.0, MXNet 1.2.1, PyTorch 0.3.1
Brief Summary

DNN libraries are great for inference, ...

MATLAB Coder and GPU Coder generates code that takes advantage of:

- NVIDIA® CUDA libraries, including TensorRT & cuDNN
- Intel® Math Kernel Library for Deep Neural Networks (MKL-DNN)
- ARM® Compute libraries for mobile platforms
Brief Summary

DNN libraries are great for inference, ...

MATLAB Coder and GPU Coder generates code that takes advantage of:

- NVIDIA® CUDA libraries, including TensorRT & cuDNN
- Intel® Math Kernel Library for Deep Neural Networks (MKL-DNN)
- ARM® Compute libraries for mobile platforms

But, Applications Require More than just Inference
Deep Learning Workflows: Integrated Application Deployment

Pre-processing → Codegen → Post-processing

Portable target code
Lane and Object Detection using YOLO v2

Workflow:
1) Test in MATLAB on CPU
2) Generate code and test on desktop GPU
3) Generate code and test on Jetson AGX Xavier GPU
(1) Test in MATLAB on CPU

AlexNet-based Lane Detection → Post-processing

YOLO v2 Object Detection → Strongest Bounding Box
(2) Generate Code and Test on Desktop GPU

- **AlexNet-based Lane Detection**
- **YOLO v2 Object Detection**
- **Post-processing**
- **Strongest Bounding Box**

**CUDA optimized code**

**cuDNN/TensorRT optimized code**
(3) Generate Code and Test on Jetson AGX Xavier GPU

MATLAB

MATLAB EXPO 2019
Lane and Object Detection using YOLO v2

1) Running on CPU
2) 7X faster running generate code on desktop GPU
3) Generate code and test on Jetson AGX Xavier GPU
Accessing Hardware

Access Peripheral from MATLAB

Deploy Standalone Application

Processor-in-Loop Verification
Deploy to Target Hardware via Apps and Command Line

%% Deploy and launch through NVIDIA HSP
%
%% setup hardware object
%% create jetson/drive hardware object with IP or hostname of jetson/drive
%% also pass credentials for login
hwobj = jetson('gpcoder-tx2-2', 'ubuntu', 'ubuntu');
hwobj.setupCodegenContext;
%
%% setup codegen config object
%% create conegen config and connect to hardware object.
cfg_hsp = coder.gpuConfig('exe');
cfg_hsp.Hardware = coder.hardware(hwobj.BoardPref);
buildDir = '~/buildDir';
cfg_hsp.Hardware.BuildDir = buildDir;
%
%% add user written main files for building executable
%% and generate/build the code.
cfg_hsp.CustomSource = 'driver_files alexnet/main.cu';
cfg_hsp.CustomInclude = 'driver_files alexnet/';
codegen -config cfg_hsp -args {im, coder.Constant(cnnMatFile)} alexnet_test
%
%% copy input and run the executable
hwobj.putFile(input2.txt, buildDir);

hwobj.putFile(synsetWords.txt, buildDir);
%
%% execute on Jetson
hwobj.runExecutable(['alexnet_test.elf', 'input2.txt'])
%
%% copy the output file back to host machine
hwobj.getFile([buildDir '/Out.txt']);
How does MATLAB Coder and GPU Coder achieve these results?
Coders Apply Various Optimizations

- Library 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

Traditional compiler optimizations

MATLAB

Loop optimizations

CUDA kernel lowering
Coders Apply Various Optimizations

- MATLAB
- Traditional compiler optimizations
  - Loop optimizations
    - Loop fusion
    - Scalar replacements
    - Parallel loop
    - CUDA kernel
    - cudaMemcpy
    - Shared memory mapping
    - CUDA code emission

- Optimized Libraries
- Network Optimization
- Coding Patterns

MATLAB EXPO 2019
Generated Code Calls Optimized Libraries

- **Pre-processing**
- **Post-processing**

- cuFFT, cuBLAS, cuSolver, Thrust Libraries

- Intel MKL-DNN Library
- NVIDIA TensorRT & cuDNN Libraries
- ARM Compute Library

Performance
1. Optimized Libraries
2. Network Optimizations
3. Coding Patterns
Deep Learning Network Optimization

1. Optimized Libraries
2. Network Optimizations
3. Coding Patterns

Network

Layer fusion
Optimized computation

Buffer minimization
Optimized memory

Optimized computation

FusedConv
BatchNormAdd

Max Pool

Add

Batch Norm

ReLu

conv

conv

Max Pool

FusedConv

Max Pool

Buffer minimization

Reuse buffer a

Reuse buffer b

buffer a

buffer b

buffer c

buffer d

buffer e

Performance
Coding Patterns: Stencil Kernels

- Automatically applied for image processing functions (e.g. imfilter,imerode, imdilate, conv2, …)
- Manually apply using gpucoder.stencilKernel()
Coding Patterns: Matrix-Matrix Kernels

- Automatically applied for many MATLAB functions (e.g. matchFeatures SAD, SSD, pdist, …)
- Manually apply using `gpucoder.matrixMatrixKernel()`
Deep Learning Workflow in MATLAB

Deep Neural Network Design + Training

Application Design

Standalone Deployment
Deep Learning Workflow in MATLAB

Deep Neural Network Design + Training

- Keras
- TensorFlow
- ONNX
- Caffe

Train in MATLAB

Model importer

Trained DNN

Transfer learning

Reference model

Application Design

Application logic

Standalone Deployment

- Intel MKL-DNN Library
- NVIDIA TensorRT and cuDNN Libraries
- ARM Compute Library

Coders