MATLAB EXPO 2019

Deploying Deep Neural Networks to Embedded GPUs and CPUs

Abhijit Bhattacharjee
Deep Learning Workflow in MATLAB

Deep Neural Network Design + Training

Trained DNN

Application Logic

Application Design

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

Application logic

Desktop

Data Center

NVIDIA Jetson

Raspberry pi

Mobile

Beaglebone

Embedded

MATLAB EXPO 2019
Algorithm Design to Embedded Deployment Workflow

Conventional Approach

1. Functional test
   - High-level language
   - Deep learning framework
   - Large, complex software stack

2. Deployment unit-test
   - C/C++
   - Low-level APIs
   - Application-specific libraries

3. Deployment integration-test
   - C/C++
   - Target-optimized libraries
   - Optimize for memory & speed

4. Real-time test
   - Embedded GPU

Challenges
- Integrating multiple libraries and packages
- Verifying and maintaining multiple implementations
- Algorithm & vendor lock-in
Solution: Use MATLAB Coder & GPU Coder for Deep Learning Deployment

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

**INFERENCE ENGINE DEPLOYMENT**
- Trained DNN
- 
  \[ \text{cnncodegen} \]
- Portable target code

**INTEGRATED APPLICATION DEPLOYMENT**
- Pre-processing
- Trained DNN
- 
  \[ \text{codegen} \]
- Post-processing
- Portable target code

MATLAB EXPO 2019
Workflow for Inference Engine Deployment

Steps for inference engine deployment

1. Generate the code for trained model
   >> cnncodegen(net, 'targetlib', 'arm-compute')

2. Copy the generated code onto target board

3. Use hand written main function to call inference engine

4. Generate the exe and test the executable
   >> make -C ./ ......
Deep Learning Inference Deployment

Pedestrian Detection

Target Libraries
- NVIDIA TensorRT & cuDNN Libraries
- Intel MKL-DNN Library
- ARM NEON Compute Library
  Includes ARM Cortex-A support

MATLAB Coder

Application logic

MATLAB

MATLAB EXPO 2019
Deep Learning Inference Deployment

Blood Smear Segmentation

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

MATLAB Coder

MATLAB EXPO 2019
Deep Learning Inference Deployment

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

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

Inference Speed - ResNet-50 (Img/Sec)

- TensorFlow XLA: 120
- PyTorch JIT: 123
- GPU Coder: 289
TensorRT Accelerates Inference Performance on Titan V

Single Image Inference with ResNet-50 (Titan V)

Images per second

- cuDNN
- TensorFlow
- TensorRT (FP32)
- TensorRT (INT8)

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10.0/1 - cuDNN 7.5.0 - TensorRT 5.1.2 - Frameworks: TensorFlow 1.13.1
Single Image Inference on Jetson TX2

![Graph showing performance comparison between TensorFlow and GPU Coder with TensorRT]

- **ResNet-50**
- **Images per second**

**TensorFlow + TensorRT**
- Performance: ~45 images per second

**GPU Coder + TensorRT**
- Performance: ~47 images per second

**NVIDIA libraries:**
- CUDA 10.0/1
- cuDNN 7.5.0
- TensorRT 5.1.2

**Frameworks:**
- TensorFlow 1.13.1

---

MATLAB EXPO 2019

NVIDIA libraries: CUDA10.0/1 - cuDNN 7.5.0 - TensorRT 5.1.2 - Frameworks: TensorFlow 1.13.1
CPU Performance

CPU, Single Image Inference (Linux)

Images/Sec

ResNet-50  VGG-16  Inception-V3  SqueezeNet

MATLAB  TensorFlow  MXNet  MATLAB Coder  PyTorch
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 $\rightarrow$ codegen $\rightarrow$ 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
- Post-processing
- YOLO v2 Object Detection
- Strongest Bounding Box

CUDA optimized code

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

- **Lane Detection** (AlexNet-based)
  - Post-processing

- **YOLO v2**
  - Object Detection
  - Strongest Bounding Box

- **cuDNN/TensorRT optimized code**
- **CUDA optimized code**
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

```matlab
%% 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
hwoObj = jetson('gpcoder-tx2-2', 'ubuntu', 'ubuntu');
hwoObj.setupCodegenContext;

%% setup codegen config object
% create conegen config and connect to hardware object.
cfg_hsp = coder.gpuConfig('exe');
cfg_hsp.Hardware = coder.hardware(hwoObj.BoardPref);
blddir = 'buildDir';
cfg_hsp.Hardware.BuildDir = blddir;

%% add user written main files for building executable
% and generate/build the code.
cfg_hsp.CustomSource = 'driver_files alexnet/main.m';
cfg_hsp.PrivateInclude = 'driver_files alexnet/';
codegen -config cfg_hsp -args {im, coder.Constant(cnnMatFile)) alexnet_test

%% copy input and run the executable
hwoObj.putFile(input2.txt, blddir);
hwoObj.runExecutable(['alexnet_test.elf', 'input2.txt'])

%% copy the output file back to host machine
hwoObj.getFile(['/Out.txt'])
```
Inference Speed - ResNet-50 (Img/Sec)

- TensorFlow XLA: 120
- PyTorch JIT: 123
- GPU Coder: 289
How does MATLAB Coder and GPU Coder achieve these results?
Coders Apply Various Optimizations

MATLAB

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

Loop optimizations
CUDA kernel lowering

Traditional compiler optimizations
Coders Apply Various Optimizations

- CUDA kernel lowering
- Traditional compiler optimizations
- MATLAB Library function mapping
- Parallel loop creation
- CUDA kernel creation
- cudaMemcpy minimization
- Shared memory mapping
- CUDA code emission
- Loop optimizations
  - Loop fusion
  - Scalar replacement
  - Parallel loop
  - CUDA kernel
  - cudaMemcpy
  - Shared memory mapping
  - CUDA code emission

- Optimized Libraries
- Network Optimization
  - Coding Patterns
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
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()`

Performance
1. Optimized Libraries
2. Network Optimizations
3. Coding Patterns
Deep Learning Workflow in MATLAB

Deep Neural Network Design + Training

Application Design

Standalone Deployment
Deep Learning Workflow in MATLAB

Deep Neural Network Design + Training

- Model importer
- Train in MATLAB
- 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
Call to action

- Visit the Deep Learning Booth!
- Related upcoming talks:
  - AI Techniques for Signals, Time-series, and Text Data
  - Sensor Fusion and Tracking for Autonomous Systems
  - Deploying Deep Neural Networks to Embedded GPUs and CPUs