MATLAB EXPO 2019

Deploying Deep Neural Networks to Embedded GPUs and CPUs

Dr Rishu Gupta
Senior Application Engineer
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

Multi-Platform Deep Learning Deployment
Algorithm Design to Embedded Deployment Workflow

1. DNN Design & Train
   - High-level language
   - Deep learning framework
   - Large, complex software stack

2. Application Design
   - 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
   - C/C++

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
Solution: Use MATLAB Coder & GPU Coder for Deep Learning Deployment
Musashi Seimitsu Industry Co., Ltd.
Detect Abnormalities in Automotive Parts

MATLAB use in project:
- Preprocessing of captured images
- Image annotation for training
- Deep learning based analysis
  - Various transfer learning methods (Combinations of CNN models, Classifiers)
  - Estimation of defect area using Class Activation Map (CAM)
  - Abnormality/defect classification
- Deployment to NVIDIA Jetson using GPU Coder

Automated visual inspection of 1.3 million bevel gear per month
Deep Learning Deployment Workflows

**INFERENACE ENGINE DEPLOYMENT**

- Trained DNN
  - `cnncodegen`
  - Portable target code

**INTEGRATED APPLICATION DEPLOYMENT**

- Pre-processing
  - Trained DNN
  - `codegen`
  - Portable target code
- Post-processing
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. Build the code for the inference engine
   `>> make -C ./codegen -f ...mk`

4. Use hand written main function to call inference engine

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

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

MATLAB Coder

Application logic

GPU Coder

MATLAB

ARM NEON™
Deep Learning Inference Deployment

Pedestrian Detection

MATLAB Coder

Target Libraries

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library
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

- TensorFlow 1.13.0
- MXNet 1.4.0
- GPU Coder (R2019a)
- PyTorch 1.0.0
Single Image Inference on Jetson TX2

![Bar chart showing performance comparison between TensorFlow + TensorRT and GPU Coder + TensorRT for ResNet-50.][1]

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

Frameworks: TensorFlow 1.6.0, MXNet 1.2.1, PyTorch 0.3.1

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

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

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

- **Lane Detection**
  - AlexNet-based
  - cuDNN/TensorRT optimized code

- **Post-processing**
  - CUDA optimized code

- **Object Detection**
  - YOLO v2

- **Strongest Bounding Box**

MATLAB EXPO 2019
(3) Generate Code and Test on Jetson AGX Xavier GPU

**CUDA optimized code**

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

**cuDNN/TensorRT optimized code**
Lane and Object Detection using YOLO v2

Workflow:
1) Test in MATLAB
2) Generate code and test on desktop
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('gpucoder-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';
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([buildDir '/alexnet_test.elf', 'input2.txt'])

%% copy the output file back to host machine
hwobj.getFile([buildDir '/out.txt'])
TensorFlow (1.13.0)
MXNet (1.4.0)
GPU Coder (R2019a)
PyTorch (1.0.0)
How does MATLAB Coder and GPU Coder achieve these results?
Coders Apply Various Optimizations

MATLAB

Traditional compiler optimizations

Loop optimizations

CUDA kernel lowering

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
Deep Learning Workflow in MATLAB

Deep Neural Network Design + Training

Reference model

Train in MATLAB

Transfer learning

Trained DNN

Model importer

Application Design

Application logic

Standalone Deployment

Intel MKL-DNN Library

NVIDIA TensorRT and cuDNN Libraries

ARM Compute Library

Coders

ARM NEON™
Deep Learning with MATLAB
This two-day course provides a comprehensive introduction to practical deep learning using MATLAB®.

Topics include:
- Importing image and sequence data
- Using convolutional neural networks for image classification, regression, and object detection
- Using long short-term memory networks for sequence classification and forecasting
- Modifying common network architectures to solve custom problems
- Improving the performance of a network by modifying training options
Please provide feedback for this block of sessions

- Scan this QR Code or log onto link below (link also sent to your phone and email)
- Enter the registration id number displayed on your badge
- Provide feedback for this session

Email: rishu.g@mathworks.com,
LinkedIn: https://www.linkedin.com/in/rishu-gupta-72148914/