임베디드 하드웨어로의 딥러닝 응용프로그램 배포

송완빈
Deep Learning Workflow in MATLAB
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 GPUs
  - Scale to clusters

Reference model → Transfer learning → Trained DNN → Train in MATLAB → Model importer → Keras TensorFlow ONNX Caffe IDEAS
Application Design

Pre-processing → [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 System
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**
   - Desktop GPU
   - C++
   - Low-level APIs
   - Application-specific libraries

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

4. **Real-time test**
   - Embedded GPU
   - 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
Deep Learning Deployment Workflows

**INFERENCE ENGINE DEPLOYMENT**

- Trained DNN
  - `cnncodegen`
  - Portable target code

**INTEGRATED APPLICATION DEPLOYMENT**

- Pre-processing
  - `cnncodegen`
  - Portable target code
- Post-processing
Workflow for Inference Engine Deployment

Steps for inference engine deployment

1. Generate the code for trained model
   `>> cnnocodegen(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

Pedestrian Detection

MATLAB Coder

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

Blood Smear Segmentation

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

MATLAB Coder

Frame Rate: 111.11
Background
Parasited cells
Good cells
Deep Learning Inference Deployment

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

Application logic

GPU Coder

Defect Classification & Detection

MATLAB Coder

Deep Learning Inference Deployment
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)
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)
Single Image Inference on Jetson TX2

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

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
- Post-processing
- codegen

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

MATLAB EXPO 2019
(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

**Matlab**

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

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

AlexNet-based
Lane Detection

Post-processing

YOLO v2
Object Detection

Strongest Bounding Box

cudaDNN/TensorRT optimized code

CUDA optimized code

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('gpucoder-tx2-2','ubuntu','ubuntu');
% setup Codegen context

% setup codegen config object
% create conegen config and connect to hardware object.
cfg_hsp = coder.gpuConfig('exe');
cfg_hsp.Hardware = coder.hardware(hwObj, 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.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', bldDir);
hwObj.putFile('synsetWords.txt', bldDir);

% execute on Jetson
hwObj.runExecutable([bldDir '/alexnet_test.elf'], 'input2.txt')

% copy the output file back to host machine
hwObj.getFile([bldDir '/Out.txt']);
Single Image Inference (Titan V, Linux)

- **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

- 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

CUDA kernel lowering
Loop optimizations
Coders Apply Various Optimizations

MATLAB

Traditional compiler optimizations

Optimized Libraries

CUDA kernel lowering

Parallel loop optimization

CUDA kernel creation

cudaMemcpy minimization

Shared memory mapping

CUDA code emission

Network Optimization

Loop fusion

Scalar replacement

Parallel loop fusion

CUDA kernel optimization

cudaMemcpy minimization

Shared memory mapping

CUDA code emission

Coding Patterns
Generated Code Calls Optimized Libraries

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

Pre-processing

Post-processing

cuFFT, cuBLAS, cuSolver, Thrust Libraries

Intel MKL-DNN Library

NVIDIA TensorRT & cuDNN Libraries

ARM Compute Library
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()`

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

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
ARM Compute Library

Coders
MATLAB EXPO 2019

데모 부스와 상담부스로 질문 하시기 바랍니다.

감사합니다