Deploying Deep Learning Networks to Embedded GPUs and CPUs

By Pierre Nowodzienski
Deep Learning enablers

- Increased GPU acceleration
- Labeled public datasets

World-class models to be leveraged

- AlexNet: PRETRAINED MODEL
- VGG-16: PRETRAINED MODEL
- ResNet: PRETRAINED MODEL
- Caffe: MODELS
- GoogLeNet: PRETRAINED MODEL
- TensorFlow/Keras: MODELS

human accuracy
Deep Learning Applications:
Image classification, speech recognition, autonomous driving, etc…

Detection of cars and road in autonomous driving systems

Rain Detection and Removal

1. Deep Joint Rain Detection and Removal from a Single Image

Traffic Sign Recognition
GPUs and CUDA programming

GPUs are “hardware on steroids”, but, programming them is hard

- CUDA
- OpenCL
- C/C++
- MATLAB
- Python

Ease of programming (expressivity, algorithmic, …)

faster

- GPU Coder

Performance
Deep learning workflow in MATLAB

Deep Neural Network Design + 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
Deep learning workflow in MATLAB

Deep Neural Network Design + Training

Caffe

Model importer

Train in MATLAB

Trained DNN

Keras

TensorFlow

Model importer

Deep learning

Application design

Localization

Planning

Autonomous systems

Deep learning Perception

Controls

Application logic
Deep learning workflow in MATLAB

Deep Neural Network Design + Training

- Caffe MODELS
  - IDEA
  - Train in MATLAB
  - Keras
  - TensorFlow

Model importer

- Trained DNN

Application design

- Application logic

Standalone Deployment

- GPU Coder

C++/CUDA
GPU Coder for Deployment

Deep Neural Networks
Deep Learning, machine learning

Image Processing and Computer Vision
Image filtering, feature detection/extraction

Signal Processing and Communications
FFT, filtering, cross correlation,

5x faster than TensorFlow
2x faster than MXNet

60x faster than CPUs for stereo disparity

20x faster than CPUs for FFTs

Intel MKL-DNN Library
ARM Compute Library
NVIDIA CUDA C/C++
ARM Compute Library
Intel XEON
ARM
GPU Coder
MATLAB
GPUs and CUDA

CUDA kernels

C/C++

GPU
CUDA Cores

ARM Cortex

GPU Memory Space

CPU Memory Space

SECURITY ENGINES

4K60 VIDEO CODEC

128-bit LPDDR4

IAQI ENGINES

Safety Engine

I/O
Challenges of Programming in CUDA for GPUs

- Learning to program in CUDA
  - Need to rewrite algorithms for parallel processing paradigm

- Creating CUDA kernels
  - Need to analyze algorithms to create CUDA kernels that maximize parallel processing

- Allocating memory
  - Need to deal with memory allocation on both CPU and GPU memory spaces

- Minimizing data transfers
  - Need to minimize while ensuring required data transfers are done at the appropriate parts of your algorithm
GPU Coder Helps You Deploy to GPUs Faster

- Library function mapping
- Loop optimizations
- Dependence analysis

- Data locality analysis
- GPU memory allocation

- Data-dependence analysis
- Dynamic memcpy reduction

GPU Coder

CUDA Kernel creation

Memory allocation

Data transfer minimization
GPU Coder speeds up MATLAB for Image Processing and Computer Vision

- Fog removal: 5x speedup
- Distance transform: 8x speedup
- Ray tracing: 18x speedup
- Frangi filter: 3x speedup
- Stereo disparity: 50x speedup
- SURF feature extraction: 700x speedup
GPU Coder speeds up MATLAB at least 2x (inference)

MATLAB 18a on TitanXP GPU - Linux

Images / Sec

AlexNet 700
ResNet-50 200
VGG-16 100

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA9 - cuDNN 7 - Framework: MATLAB 18a
Benchmark: GPU Coder versus other frameworks (inference)

Single Image Prediction (TitanXP GPU, Linux)

Images / Sec

2x

TensorFlow
MXNet
GPU Coder

AlexNet
ResNet-50
VGG-16

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA9 - cuDNN 7 - Frameworks: TensorFlow 1.6.0, MXNet 1.1.0, MATLAB 18a
Benchmark: **GPU Coder** for embedded GPU on Jetson TX2 (inference)

**Single Image Prediction on AlexNet with Jetson TX2**

- **Images / Sec**
  - Caffe
  - GPU Coder
  - GPU Coder/TensorRT

**Memory Usage [MB]**

- GPU Coder/TensorRT
- GPU Coder
- Caffe
Algorithm Design to Embedded Deployment Workflow

MATLAB algorithm (functional reference)

**GPU Coder**

- **Build type**
  - Call CUDA from MATLAB directly
  - Call CUDA from (C++) hand-coded main()

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

- Desktop GPU
- C++
- Embedded GPU
- Cross-compiled .lib
- .mex
- .lib

**Build types**
- Call CUDA from MATLAB directly
- Call CUDA from (C++) hand-coded main()
Demo: Alexnet Deployment with ‘mex’ Code Generation
Algorithm Design to Embedded Deployment on Tegra GPU

MATLAB algorithm (functional reference) → GPU Coder

Functional test
1. (Test in MATLAB on host)

Deployment unit-test
2. (Test generated code in MATLAB on host + GPU)

Deployment integration-test
3. (Test generated code within C/C++ app on host + GPU)

Real-time test
4. (Test generated code within C/C++ app on Tegra target)

Build type
- Call CUDA from MATLAB directly
- Call CUDA from (C++) hand-coded main()

Cross-compiled .lib
- Call CUDA from (C++) hand-coded main().
  Cross-compiled on host with Linaro toolchain
Alexnet Deployment to Tegra: Cross-Compiled with ‘lib’

Two small changes

1. Change build-type to ‘lib’

2. Select cross-compile toolchain
End-to-End Application: Lane Detection

Output of CNN is lane parabola coefficients according to: $y = ax^2 + bx + c$

Image → **Lane detection CNN** → Left lane coefficients → Post-processing (find left/right lane points) → Right lane coefficients → Image with marked lanes

GPU coder generates code for whole application