MATLAB EXPO 2017
Deep Learning: Transforming Engineering and Science
DEEP LEARNING: TRANSFORMING ENGINEERING AND SCIENCE
THE RISE OF GPU COMPUTING

40 Years of Microprocessor Trend Data

GPU-Computing perf
1.5X per year
1.1X per year

1000X by 2025

The Big Bang of Deep Learning

NVIDIA IS THE WORLD’S LEADING AI PLATFORM

ONE ARCHITECTURE — CUDA
AMAZING ACHIEVEMENTS IN AI

- NVIDIA Interactive Ray Tracing
- NVIDIA / Remedy Audio-driven Facial Animation
- WRNCH Pose Estimation
- University of Edinburgh Character Animation
- UC Berkeley / OpenAI One-shot Imitation Learning
A WORLD OF INTELLIGENT MACHINES

10% of Manufacturing Tasks Are Automated
1M Pizzas Delivered Per Day by Domino’s
100M People 80+ Years Old
Ag Tech: 70% Increase in Farm Yields by 2050
600K Bridges to Inspect in the U.S.
300M Operations per Year WW
JETSON TX2
SUPERCOMPUTER FOR AI AT THE EDGE

2 Core i7 PCs in <10W
256 CUDA cores
>1 TFLOPS

cuDNN, TensorRT
CUDA
Linux, ROS
<table>
<thead>
<tr>
<th></th>
<th>JETSON TX1</th>
<th>JETSON TX2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPU</strong></td>
<td>Maxwell</td>
<td>Pascal</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>64-bit A57 CPUs</td>
<td>64-bit Denver 2 and A57 CPUs</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>4 GB 64 bit LPDDR4 25.6 GB/s</td>
<td>8 GB 128 bit LPDDR4 58.4 GB/s</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>16 GB eMMC</td>
<td>32 GB eMMC</td>
</tr>
<tr>
<td><strong>Wi-Fi/BT</strong></td>
<td>802.11 2x2 ac/BT Ready</td>
<td>802.11 2x2 ac/BT Ready</td>
</tr>
<tr>
<td><strong>Video Encode</strong></td>
<td>2160p @ 30</td>
<td>2160p @ 60</td>
</tr>
<tr>
<td><strong>Video Decode</strong></td>
<td>2160p @ 60</td>
<td>2160p @ 60</td>
</tr>
<tr>
<td></td>
<td>12 bit support for H.265, VP9</td>
<td></td>
</tr>
<tr>
<td><strong>Camera</strong></td>
<td>1.4Gpix/s Up to 1.5Gbps per lane</td>
<td>1.4Gpix/s Up to 2.5Gbps per lane</td>
</tr>
<tr>
<td><strong>Mechanical</strong></td>
<td>50mm x 87mm</td>
<td>400-pin Compatible Board to Board Connector</td>
</tr>
</tbody>
</table>
INDUSTRY ADOPTION

Manufacturing

Agriculture

Construction

Inventory Management

Logistics/Retail

Security

Delivery

Inspection

Autonomous UAV

Social
RESEARCH & EDUCATION ADOPTION
<table>
<thead>
<tr>
<th>Sample Code</th>
<th>Nsight Developer Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multimedia API</strong></td>
<td></td>
</tr>
<tr>
<td>TensorRT</td>
<td>VisionWorks</td>
</tr>
<tr>
<td>cuDNN</td>
<td>OpenCV</td>
</tr>
<tr>
<td></td>
<td>Vulkan</td>
</tr>
<tr>
<td></td>
<td>OpenGL</td>
</tr>
<tr>
<td></td>
<td>libargus</td>
</tr>
<tr>
<td></td>
<td>Video API</td>
</tr>
<tr>
<td><strong>Deep Learning</strong></td>
<td><strong>Computer Vision</strong></td>
</tr>
<tr>
<td>CUDA, Linux4Tegra, ROS</td>
<td></td>
</tr>
<tr>
<td>Jetson Embedded Supercomputer: Advanced GPU, 64-bit CPU, Video CODEC, VIC, ISP</td>
<td></td>
</tr>
</tbody>
</table>
How do we target the Jetson TX2 from MATLAB?
Introducing **GPU Coder**

- Generates **CUDA** code, which can be used only on NVIDIA GPUs*

- CUDA extends C/C++ code with constructs for parallel computing

*Any modern CUDA-enabled GPU with **compute capability 3.2** or higher
Why Use GPU Coder?

**Neural Networks**
Deep Learning, machine learning

*Up to 7x faster than state-of-art*

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

*Up to 700x faster than CPUs for feature extraction*

**Signal Processing and Communications**
FFT, filtering, cross correlation,

*Up to 20x faster than CPUs for FFTs*

Performance
How fast is GPU Coder?

Orders magnitude speedup over optimized C code.

- Fog removal: 5x speedup
- Distance transform: 8x speedup
- Ray tracing: 18x speedup
- Stereo disparity: 50x speedup
- SURF feature extraction: 700x speedup
How to Use GPU Coder? Workflow to Embedded Jetson GPU

MATLAB algorithm (functional reference)

1. Functional test
   - (Test in MATLAB on host)

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

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

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

Build type

- Call CUDA from MATLAB directly
- Call CUDA from (C++) hand-coded main() or via SIL

Cross-compiled .lib

- Call CUDA from (C++) hand-coded main() or Cross-compiled on host with Linaro toolchain

GPU Coder

.mex

.Tesla GPU

.lib

.C++

.Tegra GPU

.mex, .lib

Cross-compiled .lib

.C++
Demo: Generate CUDA Code for AlexNet Prediction

“Hello World” for Deep Learning

```matlab
while true
    % Grab a frame from the camera
    ipicture = camera.snapshot;

    % Resize to Alexnet size
    picture = imresize(ipicture,[227,227]);

    % Call MEX function for alexnet prediction
    tic;
    pout = alexnet_predict(single(picture));
    newt = toc;

    % Compute Frames per second (fps)
    fps = .9*fps + .1*(1/newt);

    % top 5 scores
```
Deployment to NVIDIA Jetson: Cross-Compiled ‘lib’

Two small changes
1. Change build-type to ‘lib’
2. Select cross-compile toolchain
Alexnet Inference on NVIDIA Titan XP

Frames per second vs Batch Size

MATLAB GPU Coder (R2017b)

mxNet (0.10)

MATLAB (R2017b)

Caffe2 (0.8.1)

TensorFlow (1.2.0)

Testing platform

<table>
<thead>
<tr>
<th>CPU</th>
<th>Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>Pascal Titan Xp</td>
</tr>
<tr>
<td>cuDNN</td>
<td>v5</td>
</tr>
</tbody>
</table>
Alexnet Inference on **Jetson TX2**: Frame-Rate Performance

![Graph showing frame-rate performance对比 MATLAB GPU Coder and C++ Caffe with Jetson TX2. MATLAB GPU Coder shows a significant increase in performance compared to C++ Caffe.](image)
**Why is GPU Coder Faster than OSS Deep Learning Frameworks?**

- **OSS frameworks** are designed to do many things, including:
  - Training
  - Inference
  - Support various data types (singles, FP16, int8, etc)

- **Tensorflow** has the Python overhead

- **GPU Coder** generates code for the *specific* DNN with *specific* data types
  - Much less overhead
Additional Features: Optimizations for CUDA Code

- NVIDIA accelerated library support:
  - **cuSolver**: Dense and sparse direct solvers to accelerate computer vision and linear optimization applications
  - **cuFFT**: High-performance computation of FFTs
  - **cuBLAS**: GPU-accelerated implementation of the standard BLAS
  - **cuDNN**: GPU-accelerated library of primitives for deep neural networks
Lots of Examples to Get Started

GPU Coder Examples

Getting Started

- GPU Code Generation: The Mandelbrot Set
  - Generate CUDA® code from a simple MATLAB® function by using GPU Coder™. A Mandelbrot set implementation by using standard
  - Open Script

- Simulate Diffraction Patterns using CUDA FFT Libraries
  - Demonstrates how to use GPU Coder™ to leverage the CUDA® Fast Fourier Transform library (cuFFT) and compute two-
  - Open Script

- Benchmarking Alb with GPU Coder
  - Looks at how we can benchmark the solving of a linear system by generating GPU code. The MATLAB® code to solve for x in A\*x
  - Open Script

- Integrating GPU Coder™ into Simulink®
  - Integrate GPU Code™ into Simulink®. While GPU Coder is not supported for Simulink blocks, you can still leverage GPUs in Simulink
  - Open Script

Image Processing and Computer Vision

- Stays alive
  - Image input Image
  - Background Estimate Image

- Image output Image
▪ Easily target Jetson TX 2 from MATLAB

▪ Best in class performance for deep learning

Come See the Demo Live!
Sign Up for 50% Discount on Jetson TX2