MATLAB EXPO 2018

Deploying Deep Learning Networks to Embedded GPUs and CPUs

Daniel Aronsson

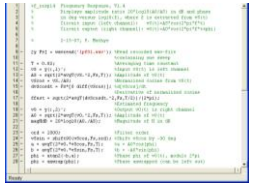
Agenda

- What is CUDA code?
- What is GPU Coder?
- Why use GPU Coder?
- How to use GPU Coder?
- How fast is GPU Coder?
- Key takeaways

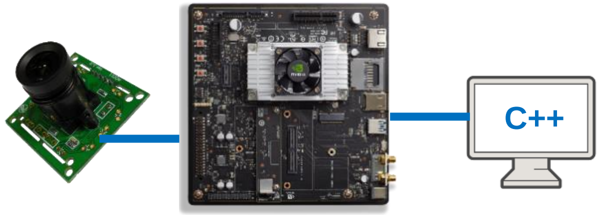
Algorithm Design to Embedded Deployment Workflow

?

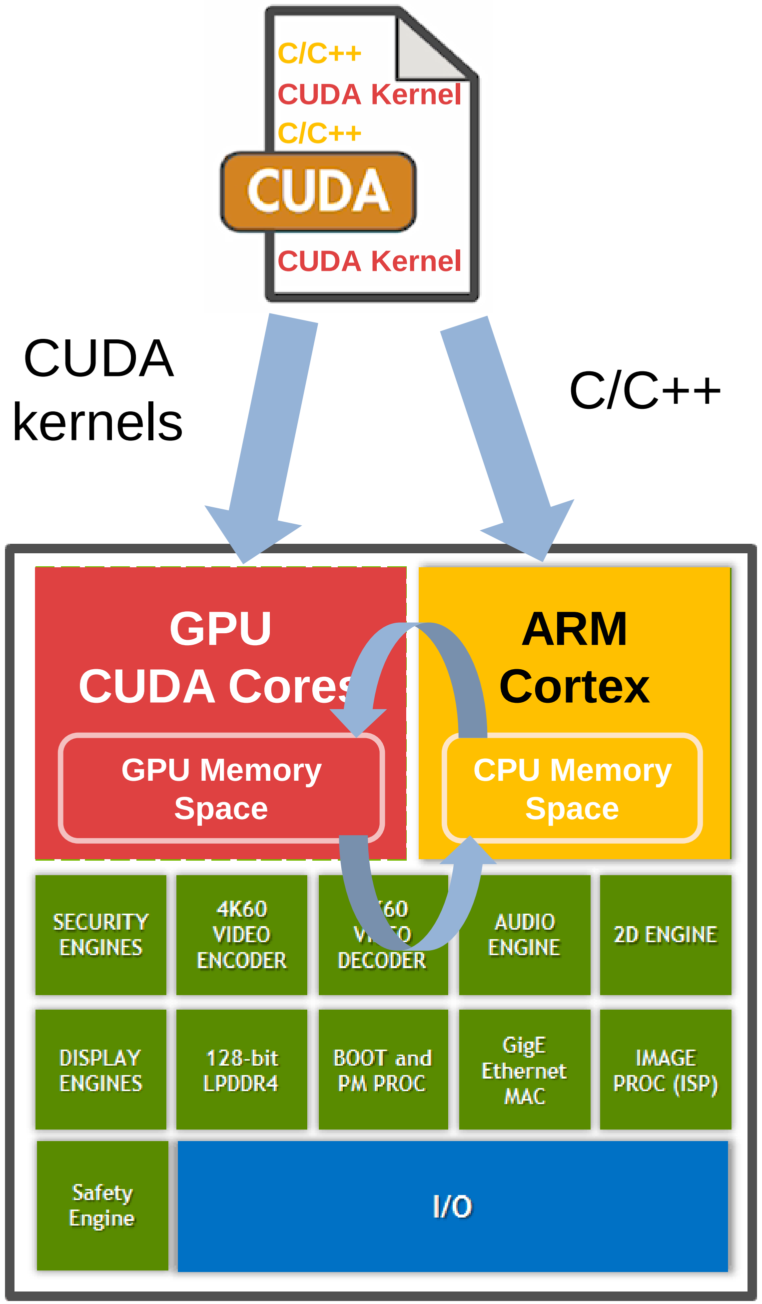
Embedded GPU



MATLAB algorithm
(functional reference)

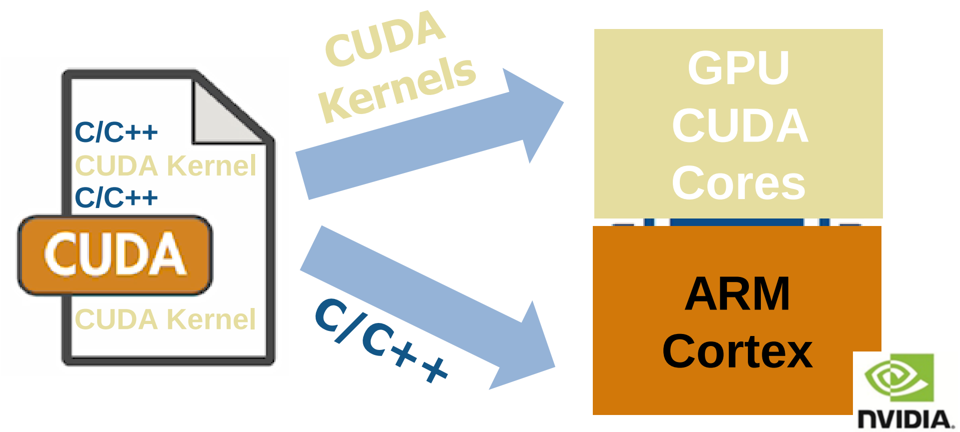


GPUs and CUDA



What is CUDA code?

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



What does CUDA code look like?

```
void foo(const real_T A[100000000], const real_T B[100000000],
real_T C[100000000])
{
  real_T *gpu_B;
  real_T *gpu_A;
  real_T *gpu_C;
  cudaMalloc(&gpu_C, 800000000ULL);
  cudaMalloc(&gpu_A, 800000000ULL);
  cudaMalloc(&gpu_B, 800000000ULL);
  cudaMemcpy((void *)gpu_B, (void *)&B[0], 800000000ULL,
cudaMemcpyHostToDevice);
  cudaMemcpy((void *)gpu_A, (void *)&A[0], 800000000ULL,
cudaMemcpyHostToDevice);
  foo_kernel1<<<dim3(313U, 313U, 1U), dim3(32U, 32U,
1U)>>>(gpu_B, gpu_A, gpu_C);
  cudaMemcpy((void *)&C[0], (void *)gpu_C, 800000000ULL,
cudaMemcpyDeviceToHost);
  cudaFree(gpu_B);
  cudaFree(gpu_A);
  cudaFree(gpu_C);
}
```

```
static __global__ __launch_bounds__(1024, 1)
  void foo_kernel1(const real_T *B,
    const real_T *A, real_T *C)
{
  uint32_T threadId;
  int32_T i0;
  threadId = (uint32_T)mwGetGlobalThreadIndex();
  i0 = (int32_T)threadId;
  if (!(i0 >= 100000000)) {
    C[i0] = A[i0] * B[i0];
  }
}
```

```
function C = foo(A,B)
C = A*B;
```

Challenges for the CUDA programmer

- 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

What is GPU Coder?

- Generates **CUDA** code for NVIDIA GPUs
- Also generates code for Deep Neural Networks for Intel CPUs and ARM Cortex-A platforms.

GPU Coder Helps You Deploy to GPUs Faster

MATLAB

GPU Coder

- CUDA Kernel creation
  - Library function mapping
  - Loop optimizations
  - Dependence analysis
- Memory allocation
  - Data locality analysis
  - GPU memory allocation
- Data transfer minimization
  - Data-dependence analysis
  - Dynamic memcpy reduction

NVIDIA
CUDA
C/C++

GPU Coder Generates CUDA from MATLAB: saxpy

Scalarized MATLAB

```
for i = 1:length(x)
    z(i) = a .* x(i) + y(i);
end
```

GPU Coder

Vectorized MATLAB

```
z = a .* x + y;
```

CUDA

```
cudaMalloc(&gpu_z, 8388608UL);
cudaMalloc(&gpu_x, 4194304UL);
cudaMalloc(&gpu_y, 4194304UL);
cudaMemcpy((void *)gpu_y, (void *)y, 4194304UL, cudaMemcpyHostToDevice);
cudaMemcpy((void *)gpu_x, (void *)x, 4194304UL, cudaMemcpyHostToDevice);
saxpy_kernel1<<<dim3(2048U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_y, gpu_x, a,
  gpu_z);
cudaMemcpy((void *)z, (void *)gpu_z, 8388608UL, cudaMemcpyDeviceToHost);
cudaFree(gpu_y);
cudaFree(gpu_x);
cudaFree(gpu_z);
```

CUDA kernel for GPU parallelization

```
static __global__ __launch_bounds__(512, 1) void saxpy_kernel1(const real32_T *y,
  const real32_T *x, real32_T a, real_T *z)
{
  int32_T i;

  i = (int32_T)(((((gridDim.x * gridDim.y * blockIdx.z + gridDim.x * blockIdx.y)
                  + blockIdx.x) * (blockDim.x * blockDim.y * blockDim.z) +
                 threadIdx.z * blockDim.x * blockDim.y) + threadIdx.y *
                blockDim.x) + threadIdx.x);
  if (!(i >= 1048576)) {
    z[i] = (real_T)(a * x[i] + y[i]);
  }
}
```

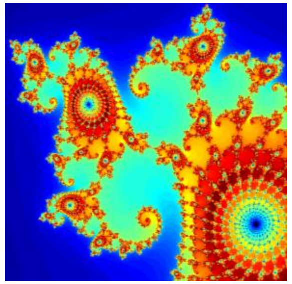
Loops and matrix operations are directly compiled into kernels

Generated CUDA Optimized for Memory Performance

Kernel data allocation is automatically optimized

```
z = z0;
for n = 0:maxIterations
    z = z.*z + z0;
    inside = abs( z )<=2;
    count = count + inside;
end
count = log( count );
```

GPU Coder



Mandelbrot space

CUDA kernel for GPU parallelization

```
static __global__ __launch_bounds__(512, 1) void kernel3(creal_T *z0, real_T
  *count, creal_T *z)
{
  real_T z_im;
  real_T y[1000000];
  int32_T threadIdX;
  threadIdX = (int32_T)(blockDim.x * blockIdx.x + threadIdx.x);
  if (!(threadIdX >= 1000000)) {
    z_im = z[threadIdX].re * z[threadIdX].im + z[threadIdX].im * z[threadIdX].re;
    z[threadIdX].re = (z[threadIdX].re * z[threadIdX].re - z[threadIdX].im *
                       z[threadIdX].im) + z0[threadIdX].re;
    z[threadIdX].im = z_im + z0[threadIdX].im;
    y[threadIdX] = hypot(z[threadIdX].re, z[threadIdX].im);
    count[threadIdX] += (real_T)(y[threadIdX] <= 2.0);
  }
}
```

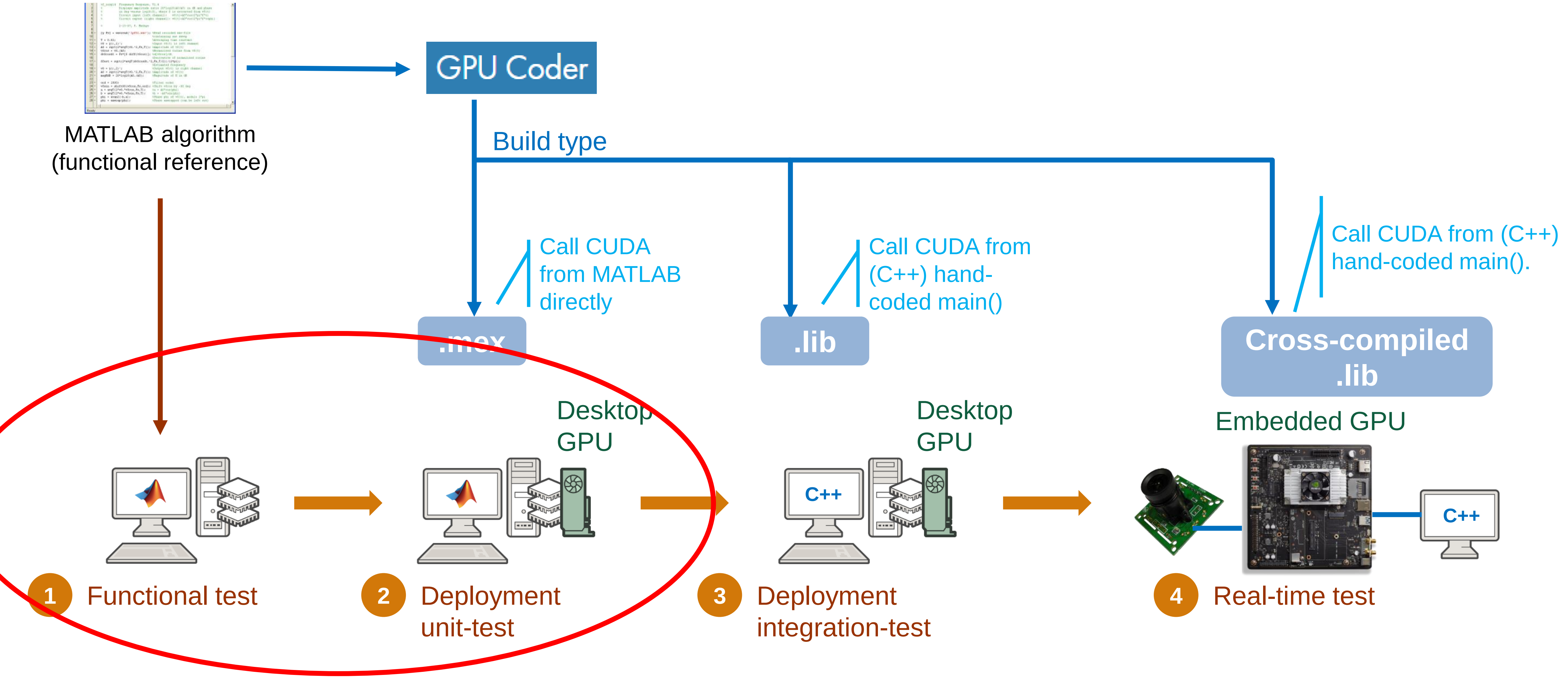
CUDA

...
...

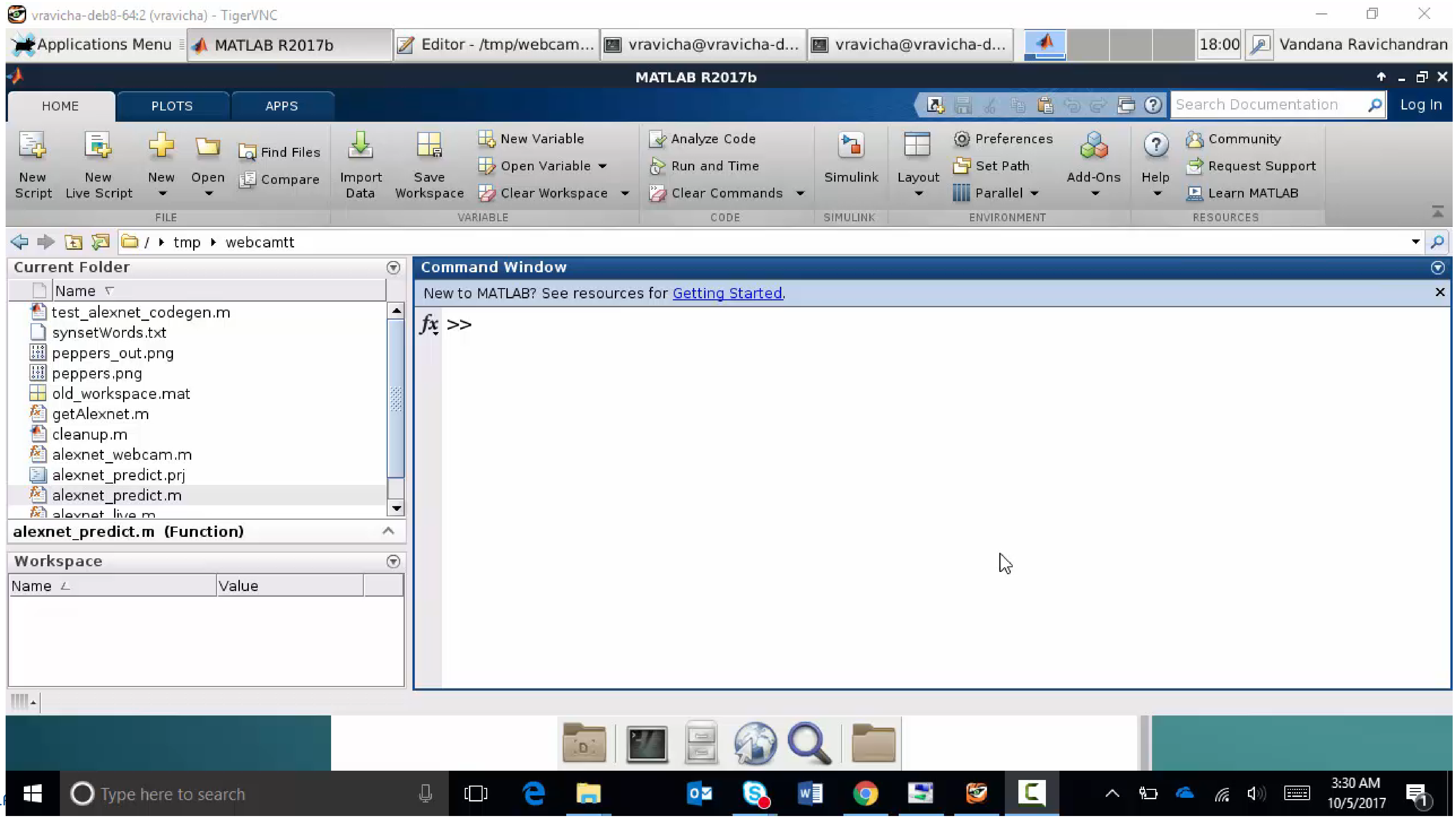
```
cudaMalloc(&gpu_xGrid, 8000000U);
cudaMalloc(&gpu_yGrid, 8000000U);

/*  mandelbrot computation */
cudaMemcpy(gpu_yGrid, yGrid, 8000000U, cudaMemcpyHostToDevice);
cudaMemcpy(gpu_xGrid, xGrid, 8000000U, cudaMemcpyHostToDevice);
kernel1<<<dim3(1954U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_yGrid, gpu_xGrid,
  gpu_z, gpu_count, gpu_z0);
for (n = 0; n < (int32_T)(maxIterations + 1.0); n++) {
  kernel3<<<dim3(1954U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_z0, gpu_count,
    gpu_z);
}

kernel2<<<dim3(1954U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_count);
cudaMemcpy(count, gpu_count, 8000000U, cudaMemcpyDeviceToHost);
cudaFree(gpu_yGrid);
```

...
...

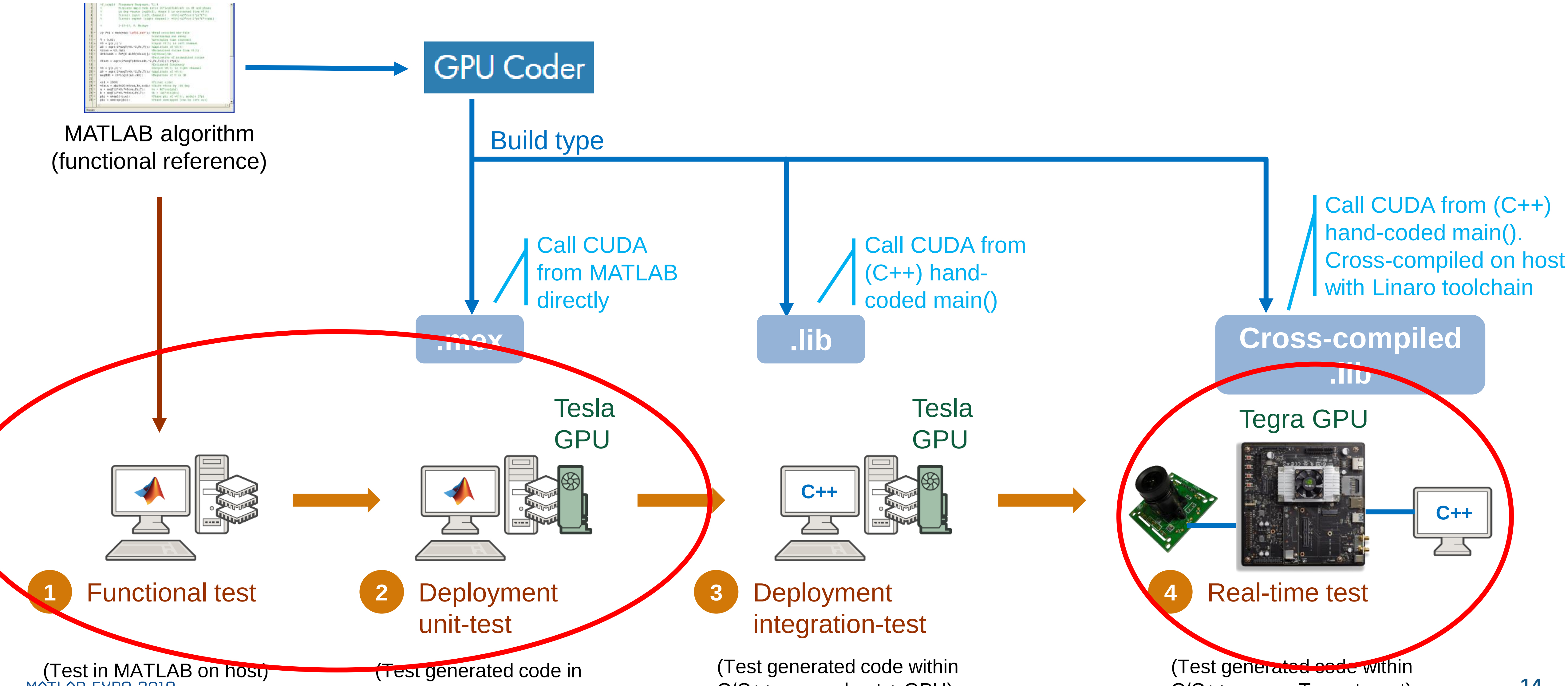
Algorithm Design to Embedded Deployment Workflow



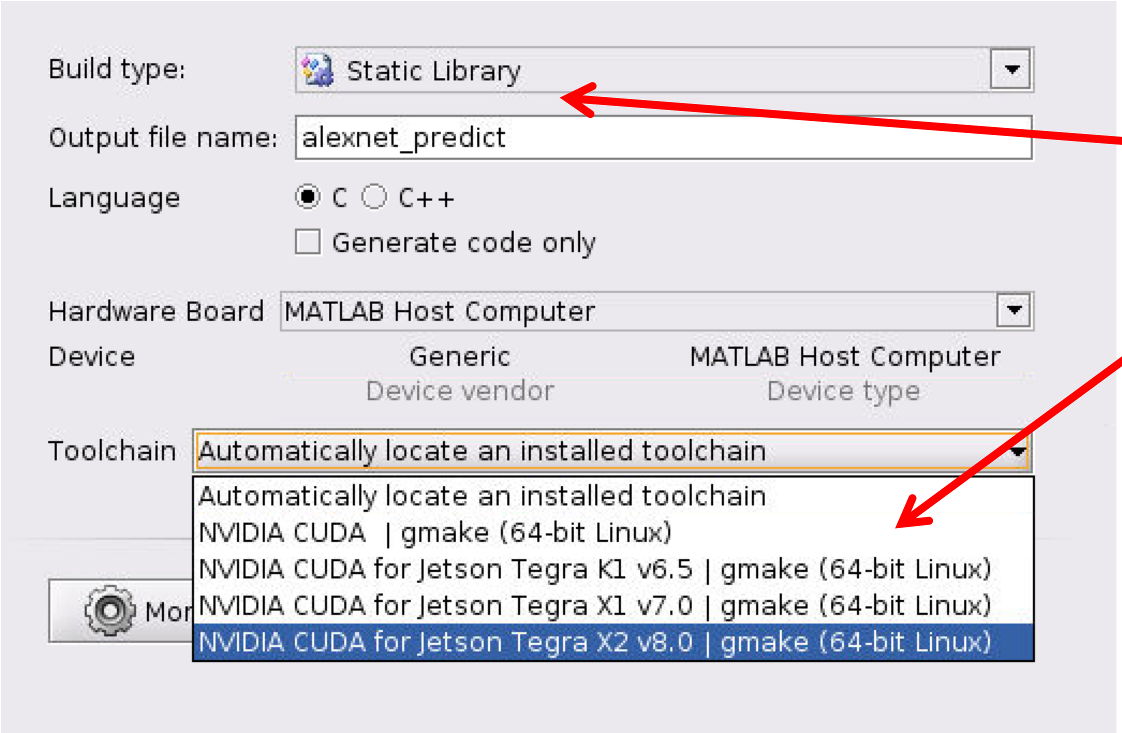
Demo: Alexnet Deployment with ‘mex’ Code Generation



Algorithm Design to Embedded Deployment on Tegra GPU



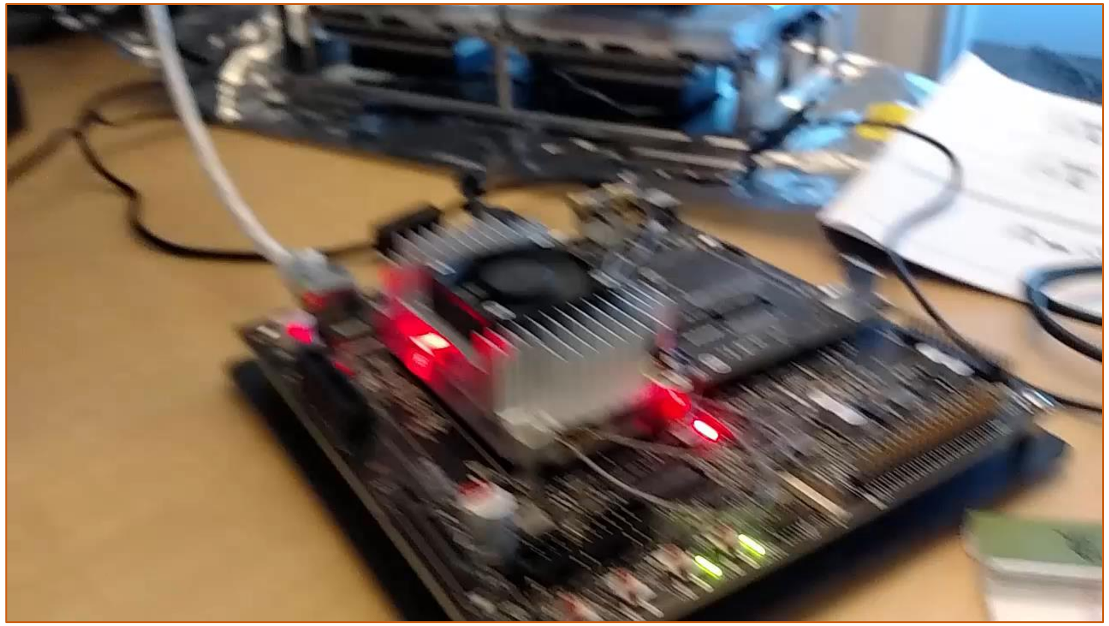
Alexnet Deployment to Tegra: Cross-Compiled with 'lib'



<u>Two small changes</u>

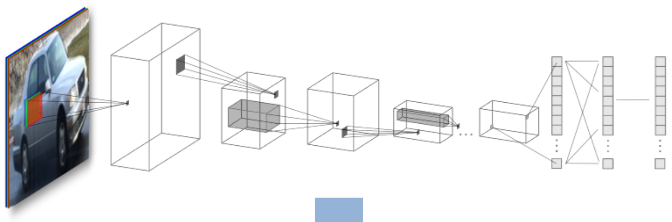
1. Change build-type to 'lib'

2. Select cross-compile toolchain

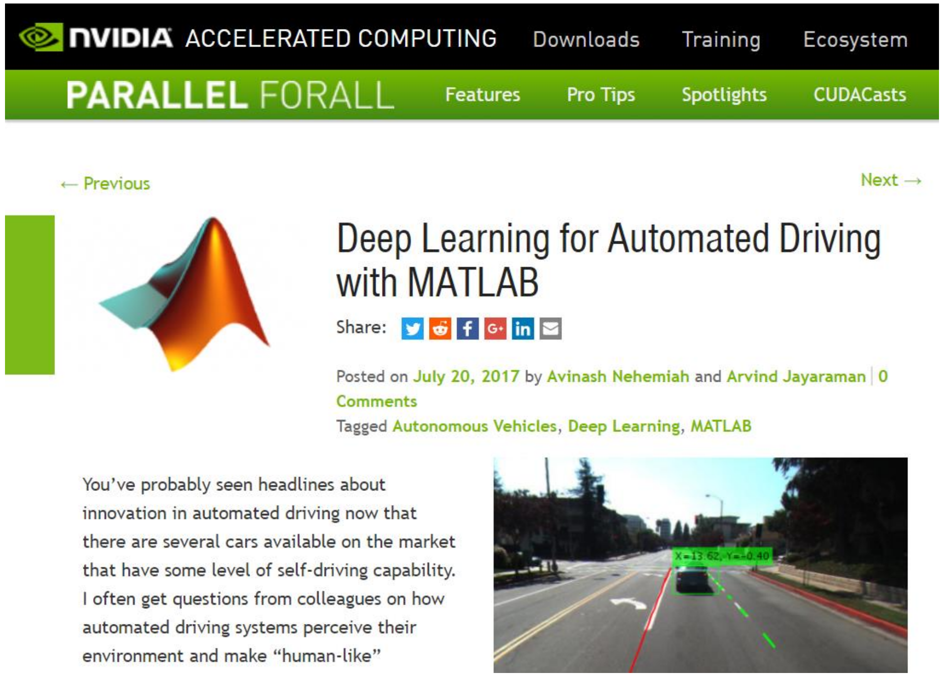


End-to-End Application: Lane Detection

Alexnet



Transfer Learning


NVIDIA ACCELERATED COMPUTING Downloads Training Ecosystem

PARALLEL FORALL Features Pro Tips Spotlights CUDACasts

← Previous Next →

Deep Learning for Automated Driving with MATLAB

Share:

Posted on July 20, 2017 by Avinash Nehemiah and Arvind Jayaraman | 0 Comments

Tagged Autonomous Vehicles, Deep Learning, MATLAB

You've probably seen headlines about innovation in automated driving now that there are several cars available on the market that have some level of self-driving capability. I often get questions from colleagues on how automated driving systems perceive their environment and make "human-like"

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

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

GPU coder generates code for whole application

GPU Coder for Deployment

MATLAB

GPU Coder

Accelerated implementation of parallel algorithms on GPUs & CPUs

Intel MKL-DNN Library

intel inside XEON

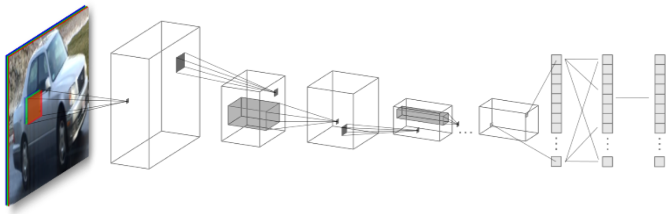
NVIDIA CUDA C/C++

ARM

ARM Compute Library

Deep Neural Networks

Deep Learning, machine learning



5x faster than TensorFlow

2x faster than MXNet

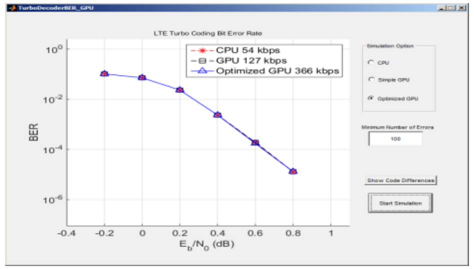
Image Processing and Computer Vision

Image filtering, feature detection/extraction

60x faster than CPUs for stereo disparity

Signal Processing and Communications

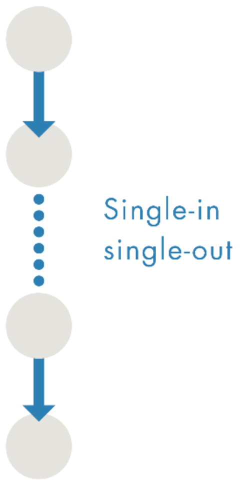
FFT, filtering, cross correlation,



20x faster than CPUs for FFTs

Deep Learning Network Support (with Neural Network Toolbox)

SeriesNetwork



GPU Coder: **R**2017**b**

Networks:
- MNist
- Alexnet
- YOLO
- VGG
- Lane detection
- Pedestrian detection

DAGNetwork

GPU Coder: **R**2018**a**

Networks:
- GoogLeNet
- ResNet
- SegNet
- DeconvNet

GoogLeNet, ResNet: Object detection

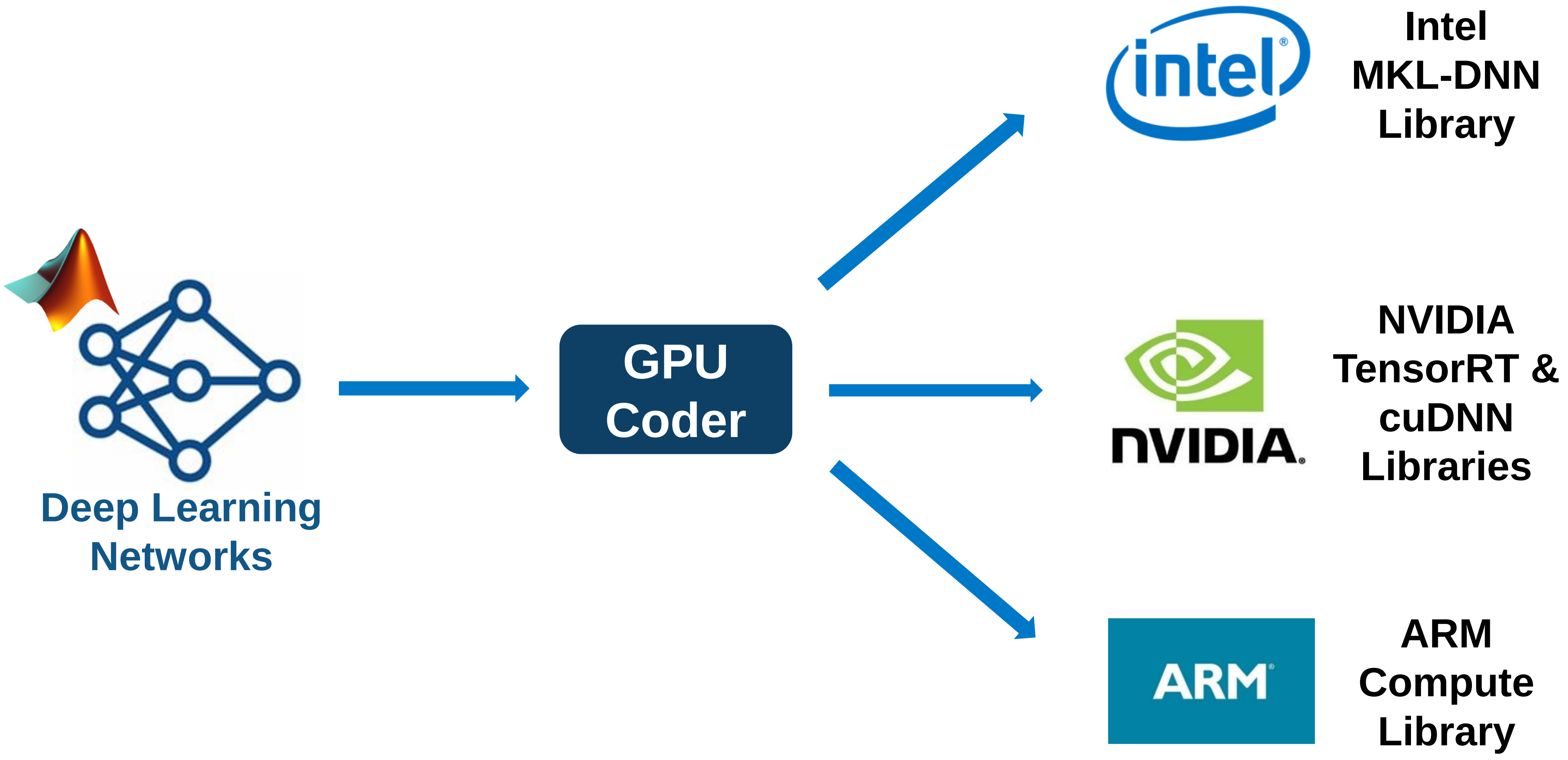
SegNet, DeconvNet: Semantic segmentation

Semantic Segmentation

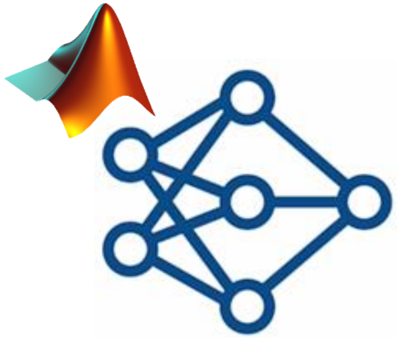
Running in MATLAB

Generated Code from GPU Coder

Deploying to CPUs



Deploying to CPUs

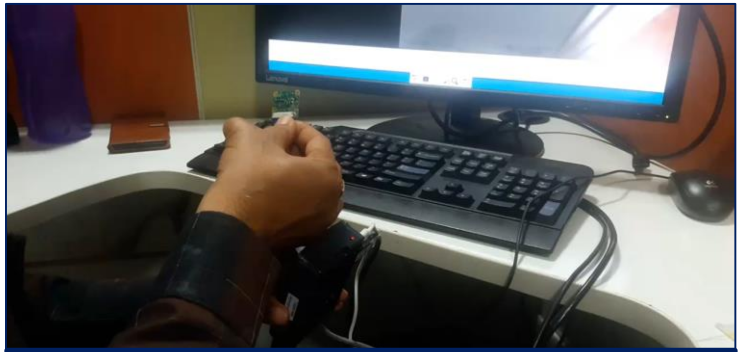


Deep Learning Networks

GPU Coder

Desktop CPU

NVIDIA TensorRT & cuDNN Libraries



Raspberry Pi board

How Good is Generated Code Performance

- Performance of image processing and computer vision
- Performance of CNN inference (Alexnet) on Titan XP GPU
- Performance of CNN inference (Alexnet) on Jetson (Tegra) TX2

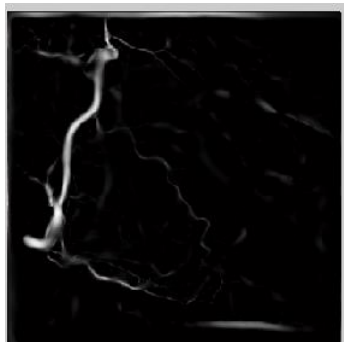
GPU Coder for Image Processing and Computer Vision

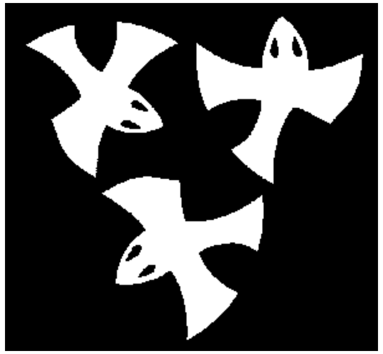
Fog removal

5x speedup

Frangi filter

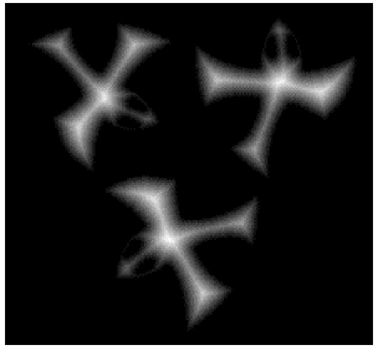
3x speedup





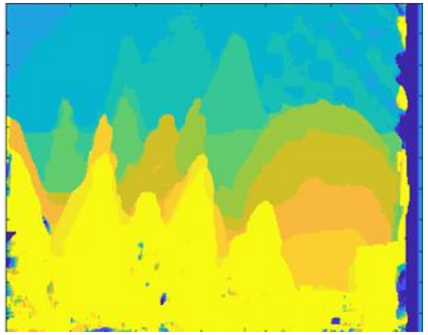
Distance transform

8x speedup



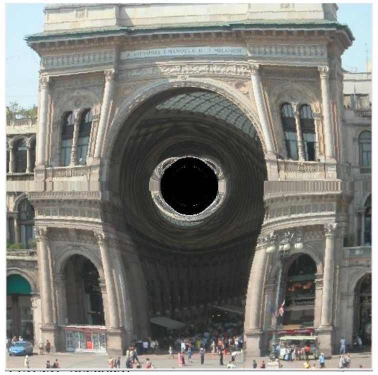
Stereo disparity

50x speedup



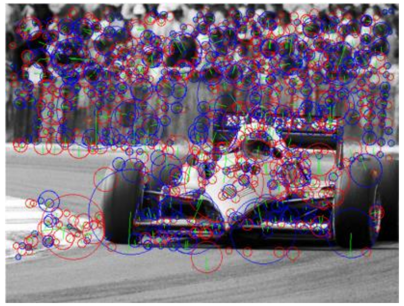
Ray tracing

18x speedup

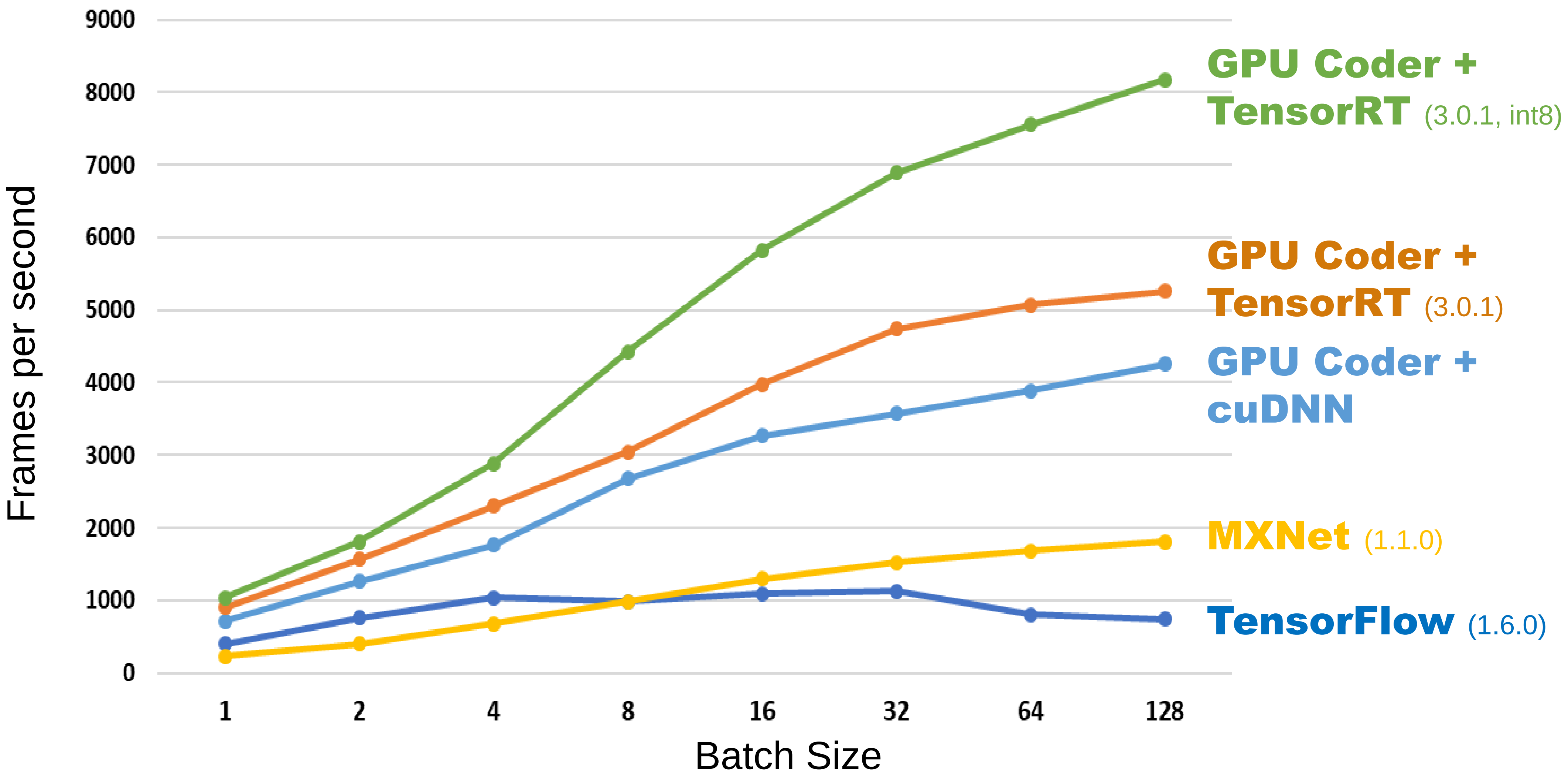


SURF feature extraction

700x speedup



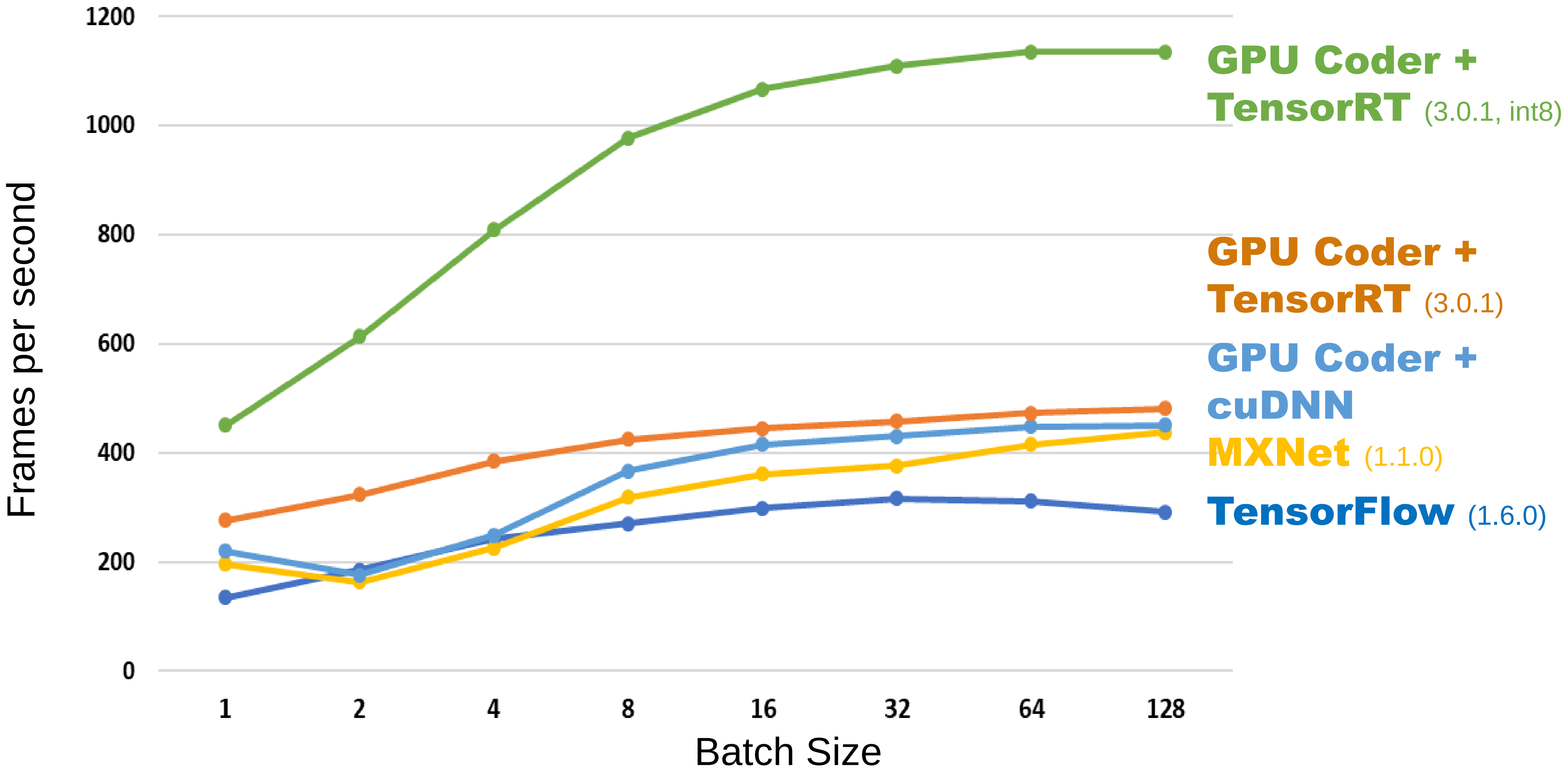
Alexnet Inference on NVIDIA Titan Xp



Testing platform

CPU	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz
GPU	Pascal Titan Xp
cuDNN	v7

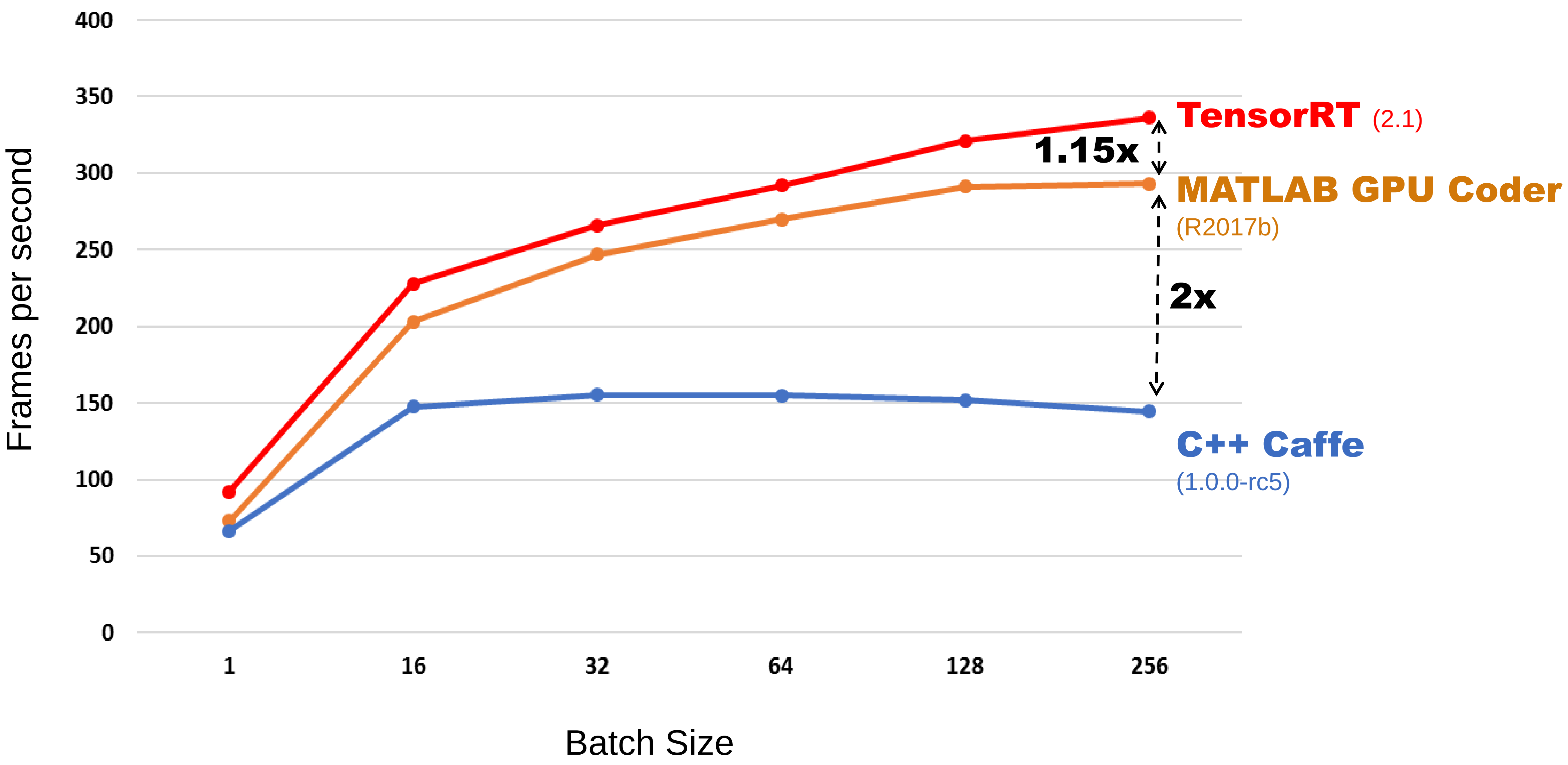
VGG-16 Inference on NVIDIA Titan Xp



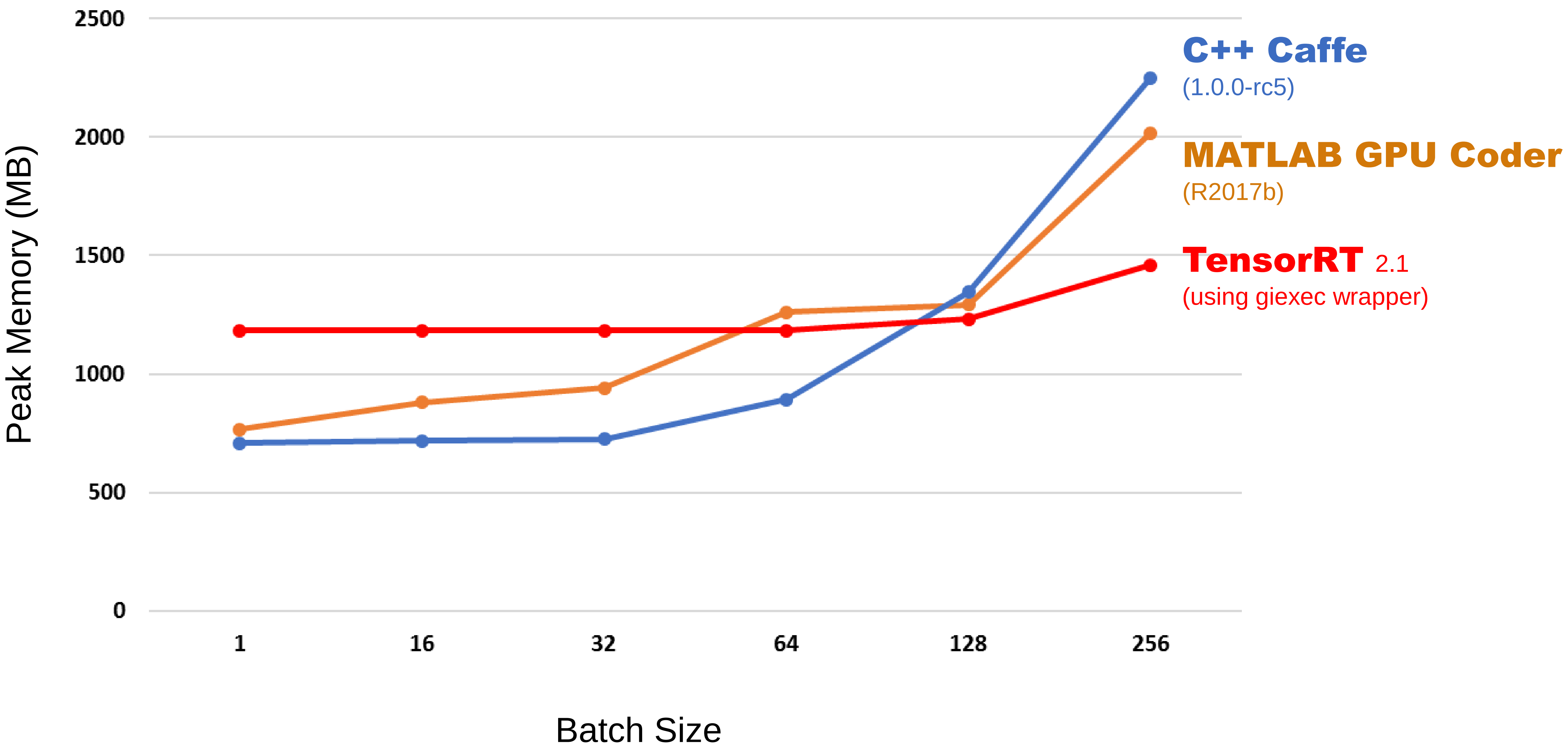
Testing platform

CPU	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz
GPU	Pascal Titan Xp
cuDNN	v7

Alexnet Inference on Jetson TX2: Frame-Rate Performance



Alexnet Inference on Jetson TX2: Memory Performance



Key Takeaways

- GPU Coder automates the process of writing CUDA code for general algorithms – not only Deep Learning
- GPU Coder generates code for DNN for multiple platforms
- GPU Coder performs in most times better than other common Deep Learning platforms