MathWorks

MATLAB EXPO 2018

嵌入式GPU和CPU的深度学习网络部署

Bill Chou

MATLAB深度学习框架

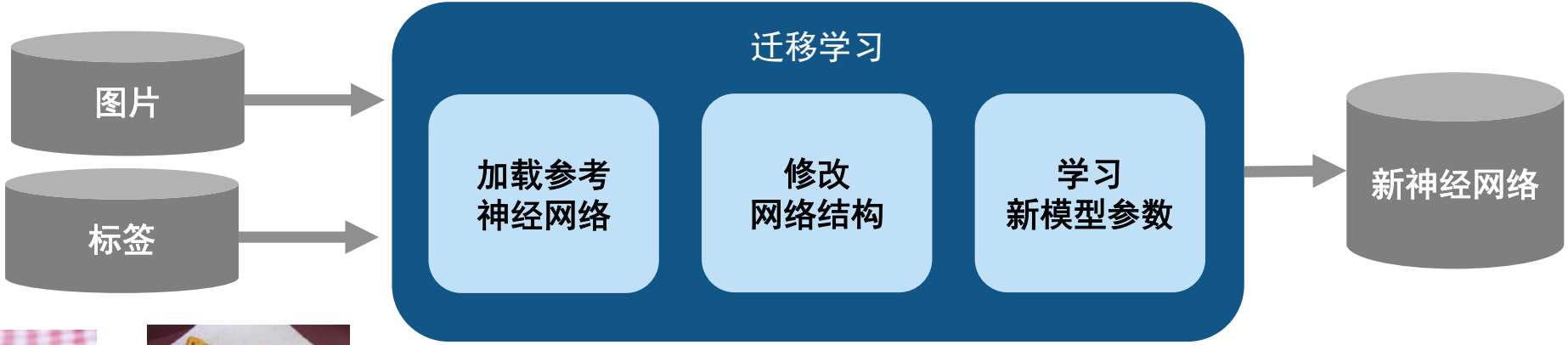
- 管理大型图像集
- 自动化图像标签
- 轻松访问模型

- 利用GPU加速
- 扩展到HPC集群

- 使用GPU Coder自动生成代码到GPU和CPU：
  - 比TensorFlow快5倍
  - 比MXNet快2倍

设计神经网络和视觉算法

迁移学习流程



标签: 热狗，比萨饼，冰淇淋，
巧克力蛋糕，炸薯条

培训数据

示例：用MATLAB作迁移学习

设置
训练数据集

```
%% set up training dataset
cifarFolder = 'cifar10Train';
categories = {'Cars', 'Trucks', 'BigTrucks', 'Suvs', 'Vans'};
imds = imageDatastore(fullfile(cifarFolder, categories), ...
    'LabelSource', 'foldernames');

imds = splitEachLabel(imds, 500, 'randomize'); % we only need 500 images per class
imds.ReadFcn = @readFunctionTrain;
```

加载参考
神经网络

```
%% load reference network
net = alexnet;
layers = net.Layers;
```

修改
网络结构

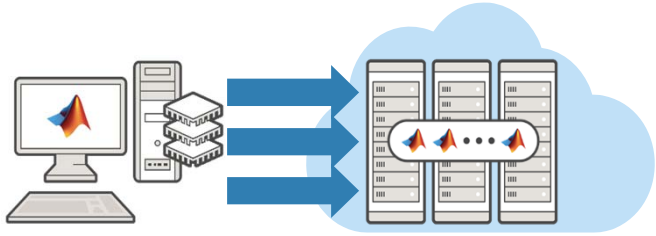
```
%% modify network
layers = layers(1:end-3);

layers(end+1) = fullyConnectedLayer(64, 'Name', 'special_2');
layers(end+1) = reluLayer;
layers(end+1) = fullyConnectedLayer(5, 'Name', 'fc8_2 ');
layers(end+1) = softmaxLayer;
layers(end+1) = classificationLayer();
```

学习
新模型参数

```
%% train!
options = trainingOptions('sgdm', ...
    'LearnRateSchedule', 'none',...
    'InitialLearnRate', .0001,...
    'MaxEpochs', 20, ...
    'MiniBatchSize', 128);

myConvnet = trainNetwork(imds, layers, options);
```

扩大神经网络培训绩效

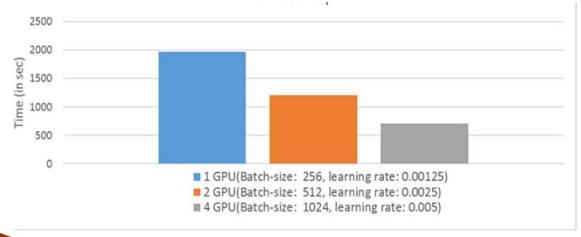
```
'ExecutionEnvironment', 'parallel' );
```



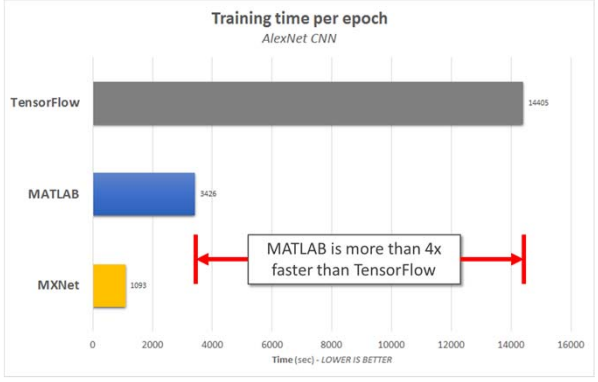
Training on the AWS (EC2)

```
opts = trainingOptions('sgdm', ...
    'MaxEpochs', 100, ...
    'MiniBatchSize', 250, ...
    'InitialLearnRate', 0.00005, ...
    'ExecutionEnvironment', 'auto' );
```

```
'ExecutionEnvironment', 'multi-gpu' );
```



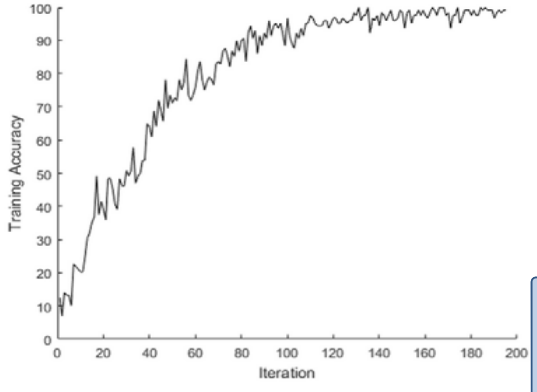
Multiple GPU support



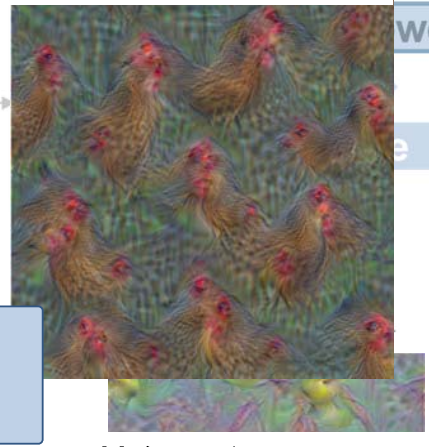
Single GPU performance

可视化和调试网络中间结果

培训准确度可视化

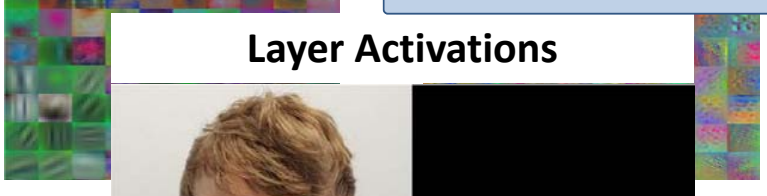


Deep Dream

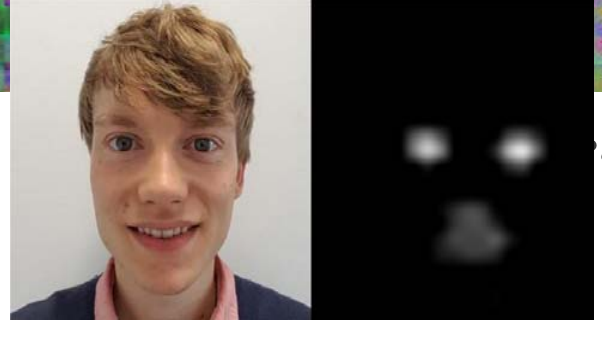


- 许多可视化和调试选项
- 许多示例帮忙您学习

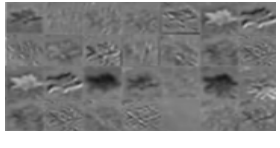
过滤器

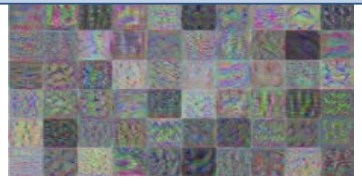


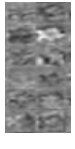
Layer Activations



激活







特征可视化

使用GPU Coder进行部署

MATLAB

↓

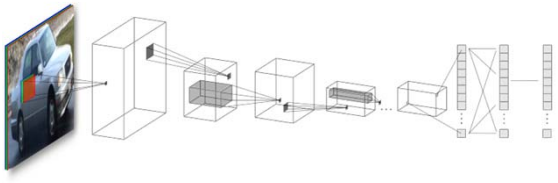
GPU Coder　　在GPU和CPU上加速并行算法

↓

英特尔 MKL-DNN Library　　NVIDIA CUDA C/C++　　ARM Compute Library

深度学习网络

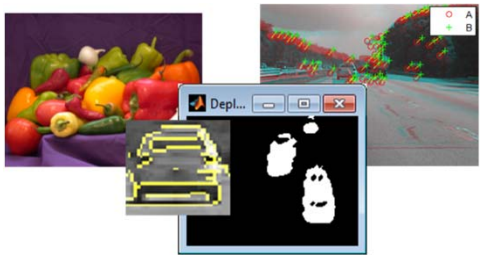
深度学习, 机器学习



比TensorFlow快**5**倍
比MXNet快**2**倍

图像处理和计算机视觉

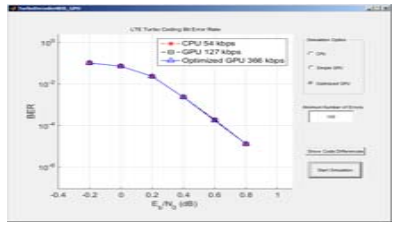
图像过滤，特征检测/提取



立体声视差计算比CPU快**60**倍

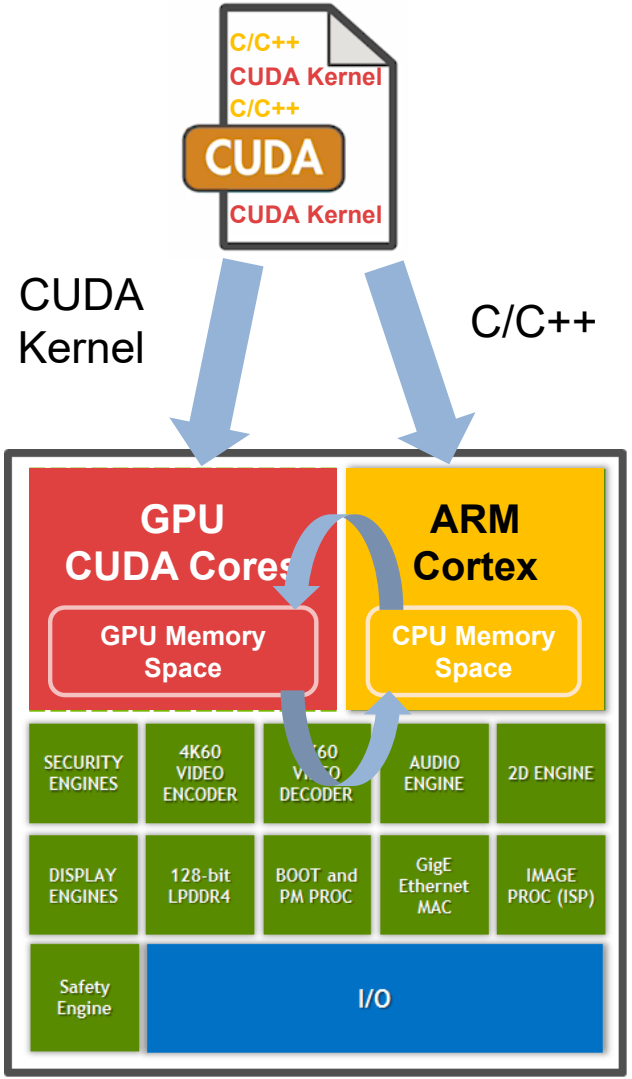
信号处理和通信

FFT，滤波，互相关



FFT计算比CPU快**20**倍

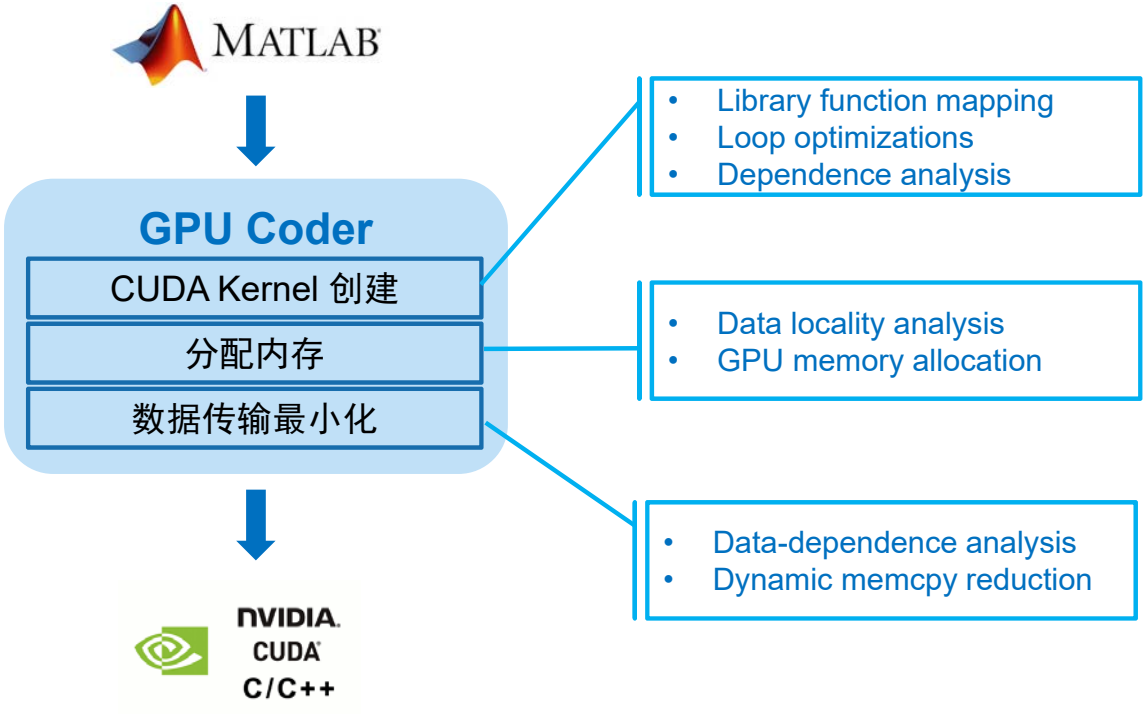
GPU和CUDA


C/C++
CUDA Kernel
C/C++
CUDA
CUDA Kernel
CUDA
Kernel
C/C++
GPU
CUDA Cores
ARM
Cortex
GPU Memory
Space
CPU Memory
Space
SECURITY
ENGINES
4K60
VIDEO
ENCODER
DECODER
AUDIO
ENGINE
2D ENGINE
DISPLAY
ENGINES
128-bit
LPDDR4
BOOT and
PM PROC
GigE
Ethernet
MAC
IMAGE
PROC (ISP)
Safety
Engine
I/O

CUDA编程对GPU的挑战

- 学习CUDA编程
  - 需要重新编写程序于GPU并行编程的运算架构
- 创建CUDA kernels
  - 需要分析算法来创建最大化并行处理的CUDA kernel
- 分配内存
  - 需要处理CPU和GPU memory space的内存分配
- 尽量减少CPU和GPU的数据传输
  - 需要尽量减少，同时确保在算法的适当部分完成所需的数据传输

GPU Coder帮助您更快部署到GPU



GPU Coder从MATLAB生成CUDA代码：saxpy

Scalarized MATLAB

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

GPU Coder

Vectorized MATLAB

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

循环和矩阵计算直接编译到CUDA kernels

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

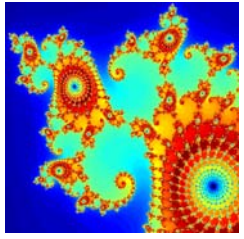
```
static __global__ __launch_bounds__(512, 1) void saxpy_kernel1(const real32_T *y,
  const real32_T *x, real32_T a, real_T *z)
{
  int32_T i;

  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]);
  }
}
```

自动生成CUDA代码针对内存性能进行了优化

GPU Coder会自动优化Kernel数据分配

```
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

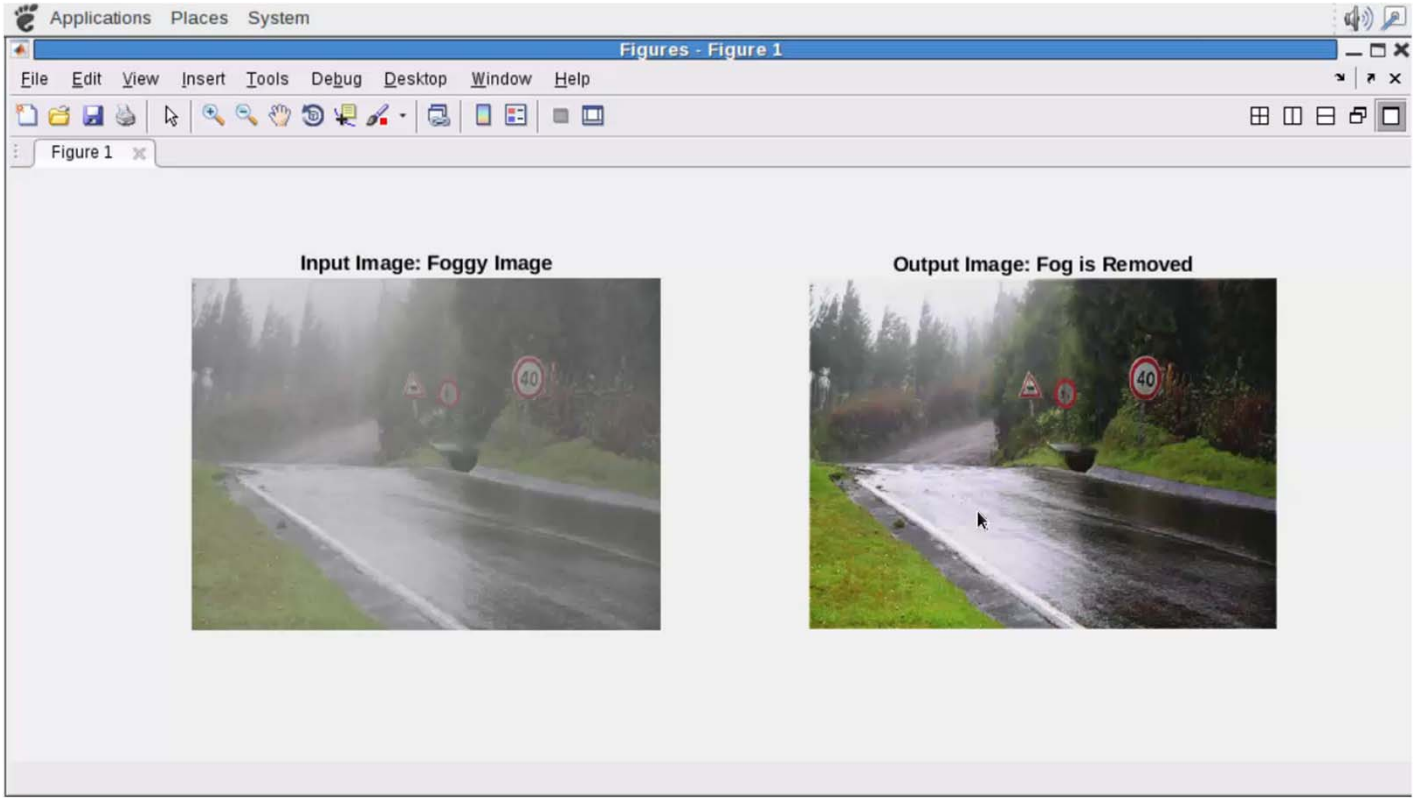
CUDA kernel for GPU parallelization

```
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);
  }
}
```

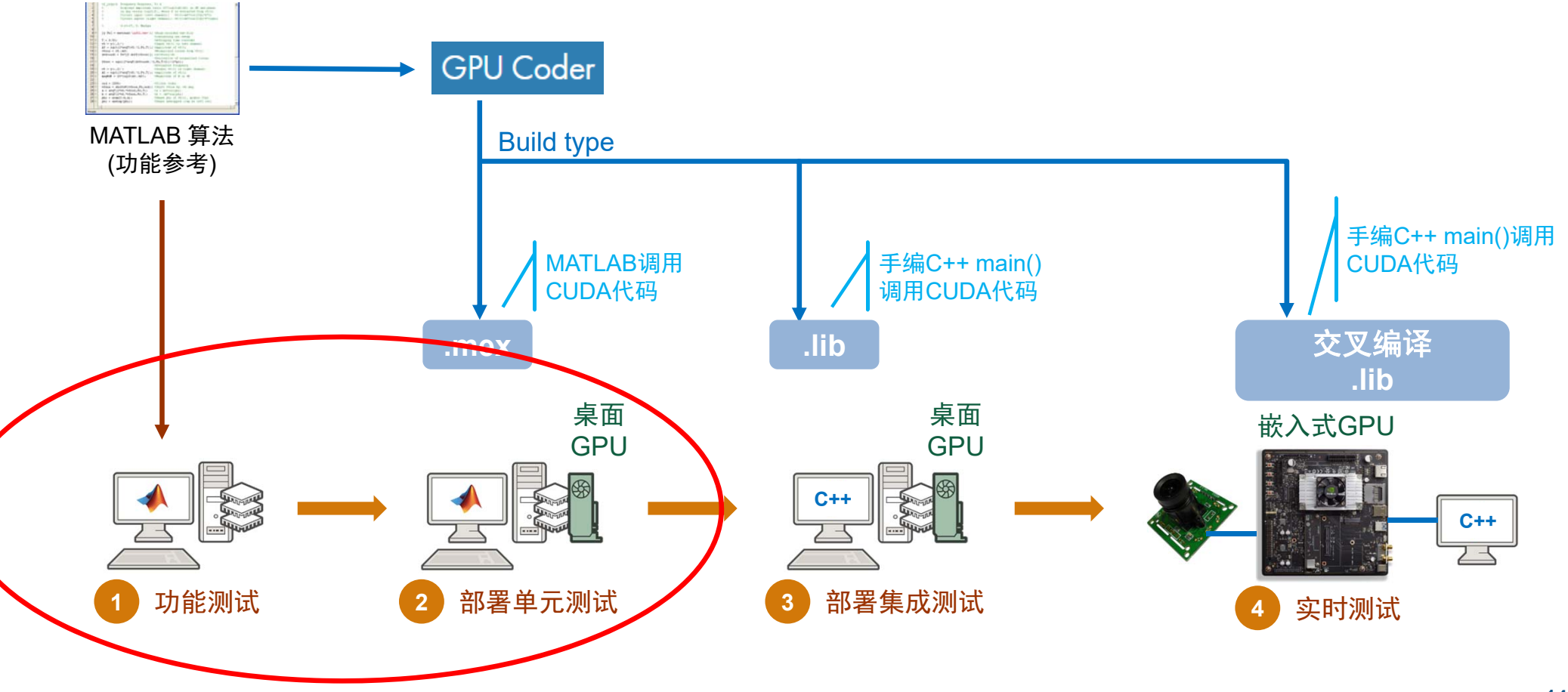
CUDA

```
...
...

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);

...
...
```

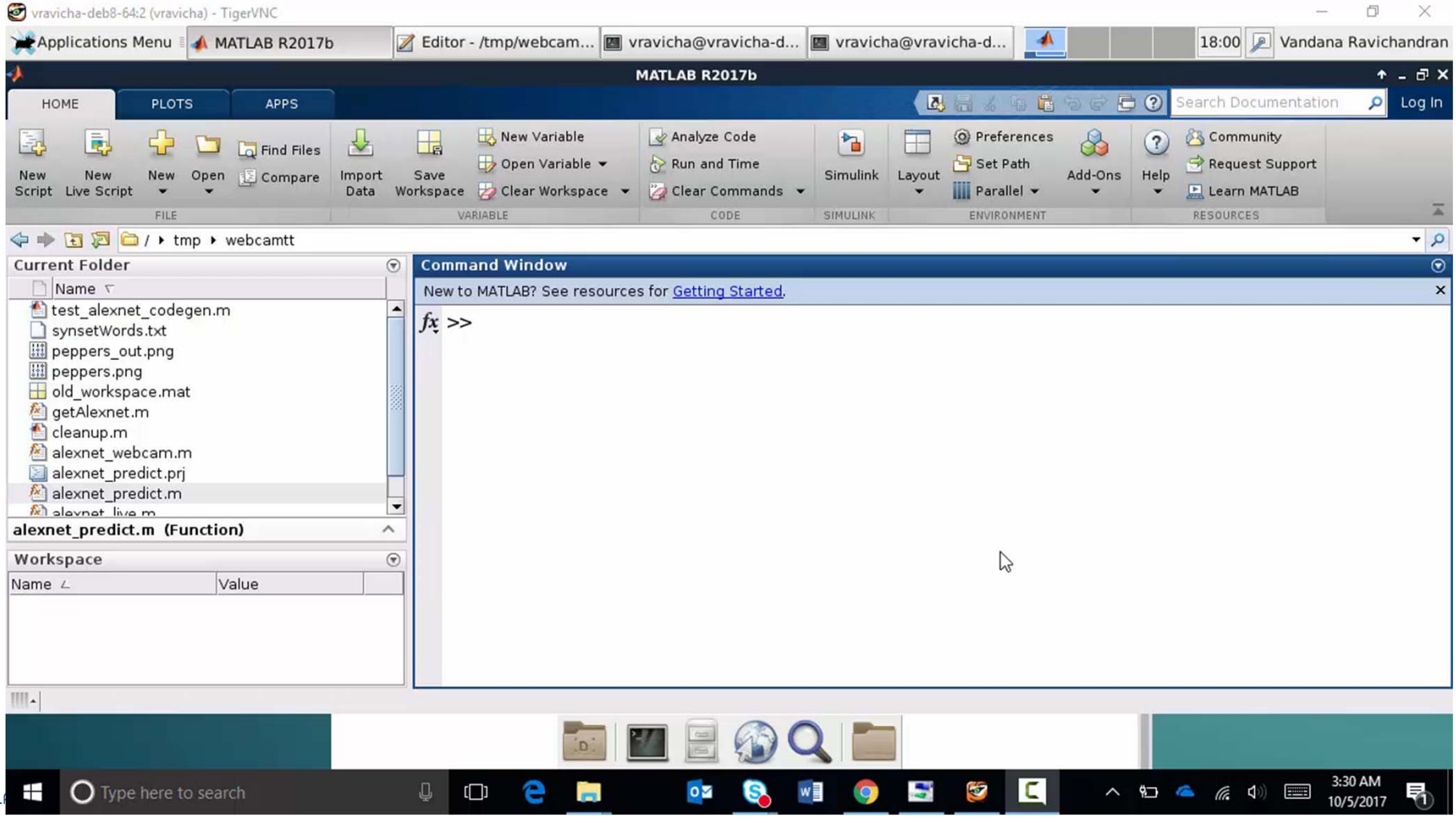
示例：雾校正


Applications Places System
Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Figure 1
Input Image: Foggy Image
Output Image: Fog is Removed

算法设计到嵌入式部署的工作流程



示例：使用 MEX 代码生成部署 Alexnet



算法设计到嵌入式 Tegra GPU 部署的工作流程

MATLAB 算法
(功能参考)

GPU Coder

Build type

MATLAB调用
CUDA代码

.mex

手编C++ main()
调用CUDA代码

.lib

手编C++ main()调用
CUDA代码。使用Linaro
工具链交叉编译主机

交叉编译
.lib

桌面
GPU

Tesla
GPU

Tegra GPU

C++

C++

1 功能测试

2 部署单元测试

3 部署集成测试

4 实时测试

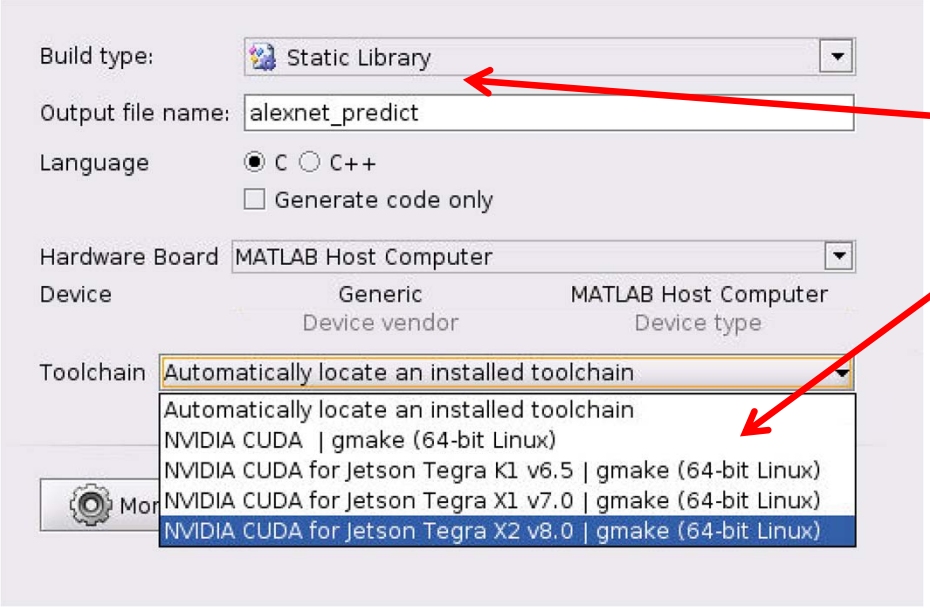
（在主机上的MATLAB测试）

（在主机+ GPU上测试
MATLAB生成的代码）

（在主机+ GPU上测试
C/C++应用程序中的生成代码）

（在Tegra目标上的C/C++应
用程序中测试生成的代码）

Alexnet部署到Tegra：使用 Lib 交叉编译

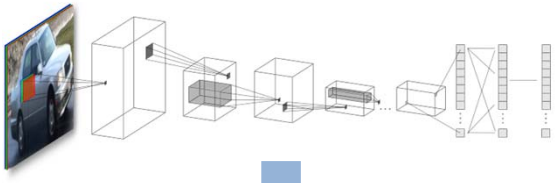


两个小变化

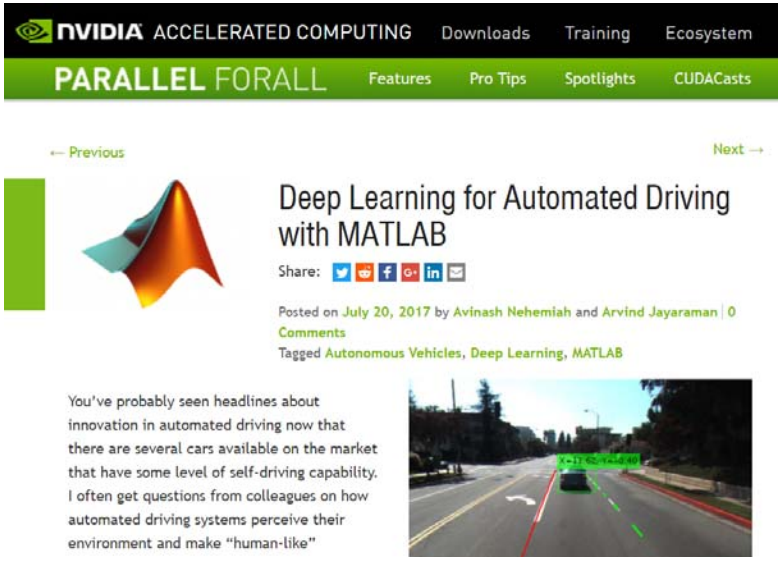
1. 将build-type更改为'lib'
2. 选择交叉编译工具链

End-to-End 示例：车道检测

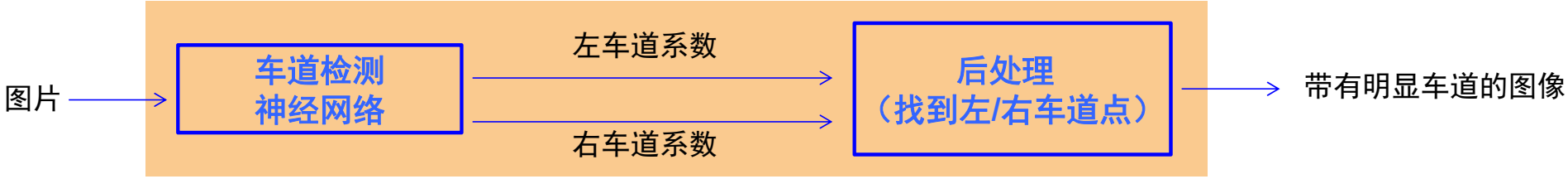
Alexnet



迁移学习



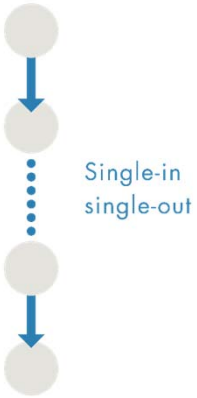
CNN的输出是车道抛物线系数，根据：y = ax^2 + bx + c



GPU Coder为整个应用程序生成代码

神经网络支持（使用Neural Network Toolbox）

SeriesNetwork



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

网络支持:
- MNist
- Alexnet
- YOLO
- VGG
- Lane detection
- Pedestrian detection

DAGNetwork

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

网络支持:
- GoogLeNet
- ResNet

对象检测

- SegNet
- DeconvNet

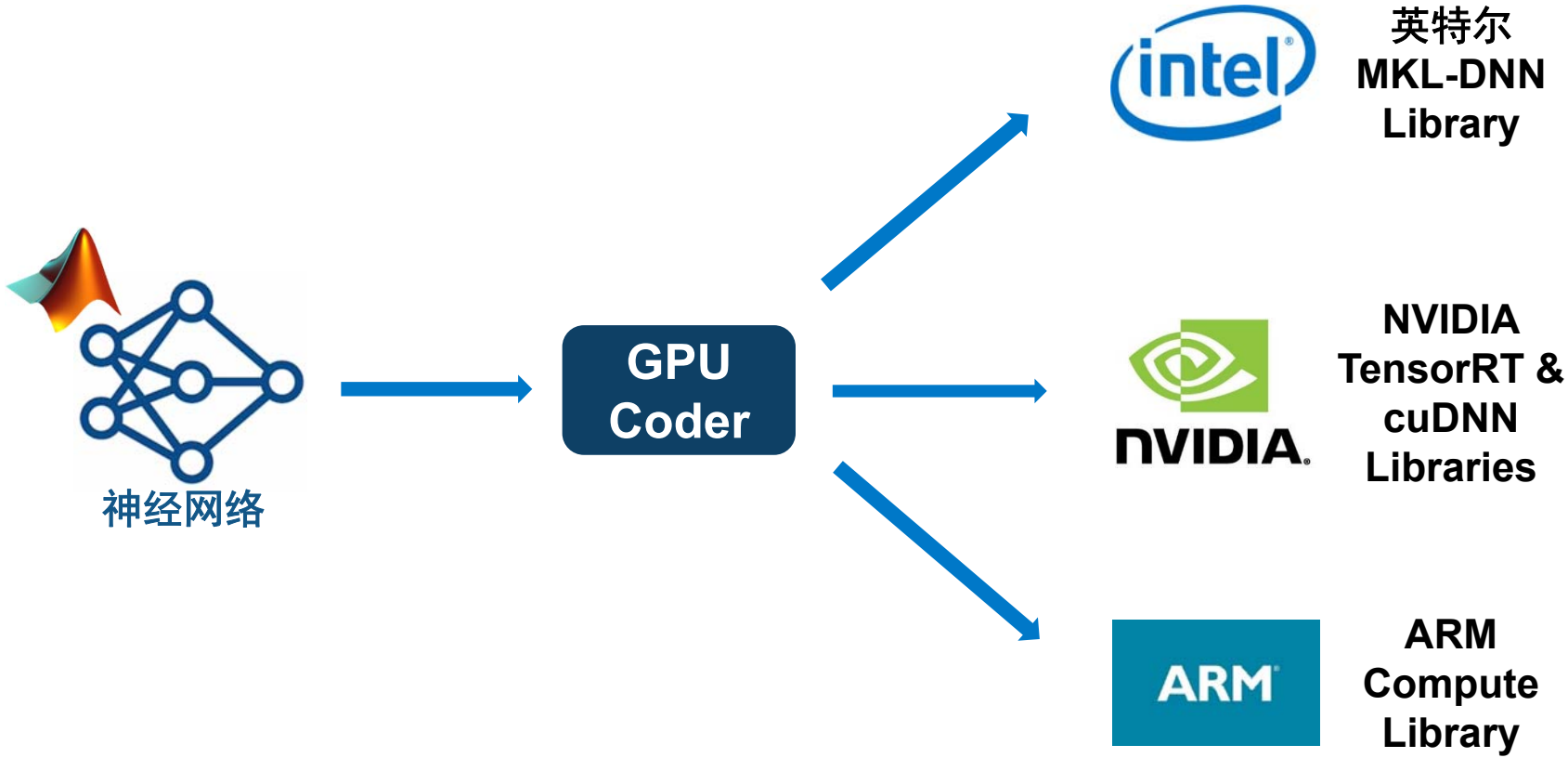
语义分割

语义分割

在MATLAB中运行

来自GPU Coder的生成代码

部署到CPU


神经网络
GPU Coder
英特尔
MKL-DNN
Library
NVIDIA
TensorRT &
cuDNN
Libraries
ARM
Compute
Library

部署到CPU

神经网络

GPU Coder

桌面CPU

NVIDIA

NVIDIA TensorRT & cuDNN Libraries

树莓派

生成代码的性能

- 图像处理和计算机视觉性能
- AlexNet 在Titan XP上的深度学习推理性能
- VGG-16 在Titan XP上的深度学习推理性能
- AlexNet 在Jetson (Tegra) TX2 上的深度学习推理性能

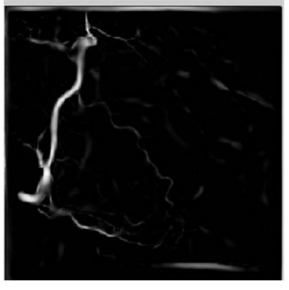
GPU Coder用于图像处理和计算机视觉

除雾

5倍加速

Frangi filter

3倍加速

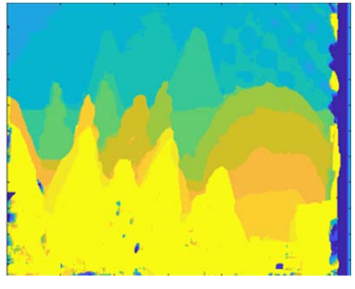


Distance transform

8倍加速

Stereo disparity

50倍加速



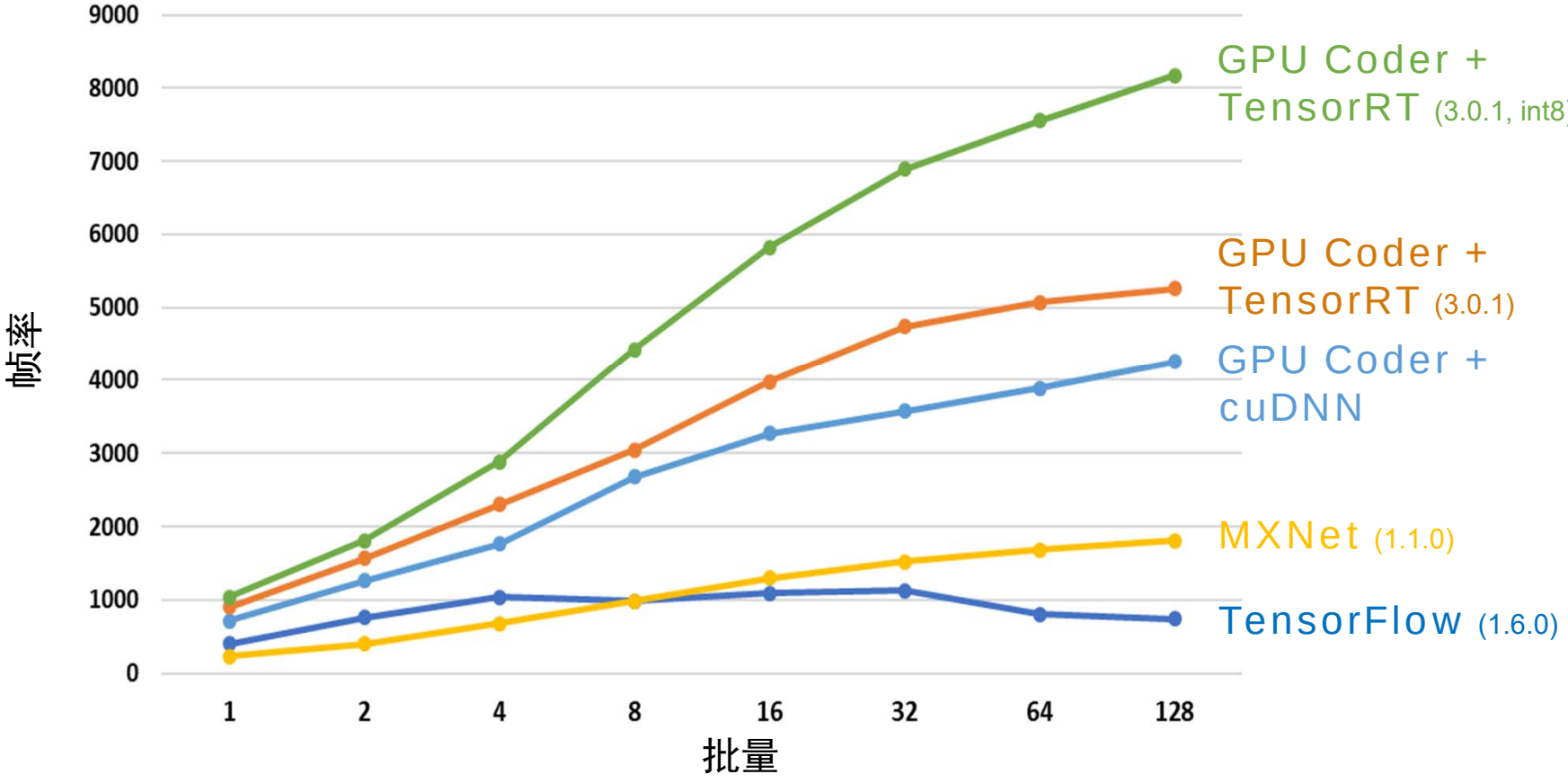
Ray tracing

18倍加速

SURF feature extraction

700倍加速

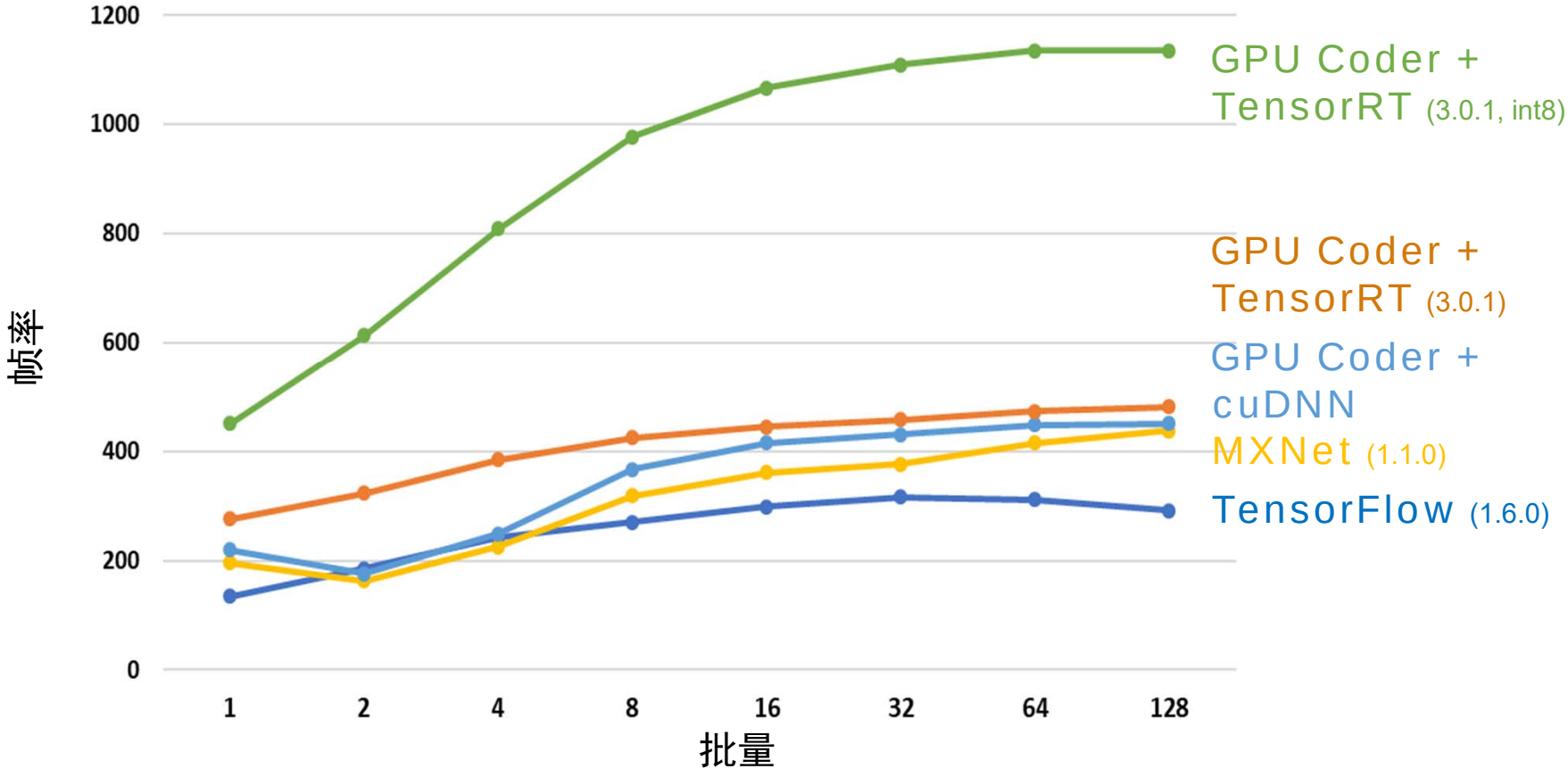
AlexNet 在Titan XP上的深度学习推理



测试平台

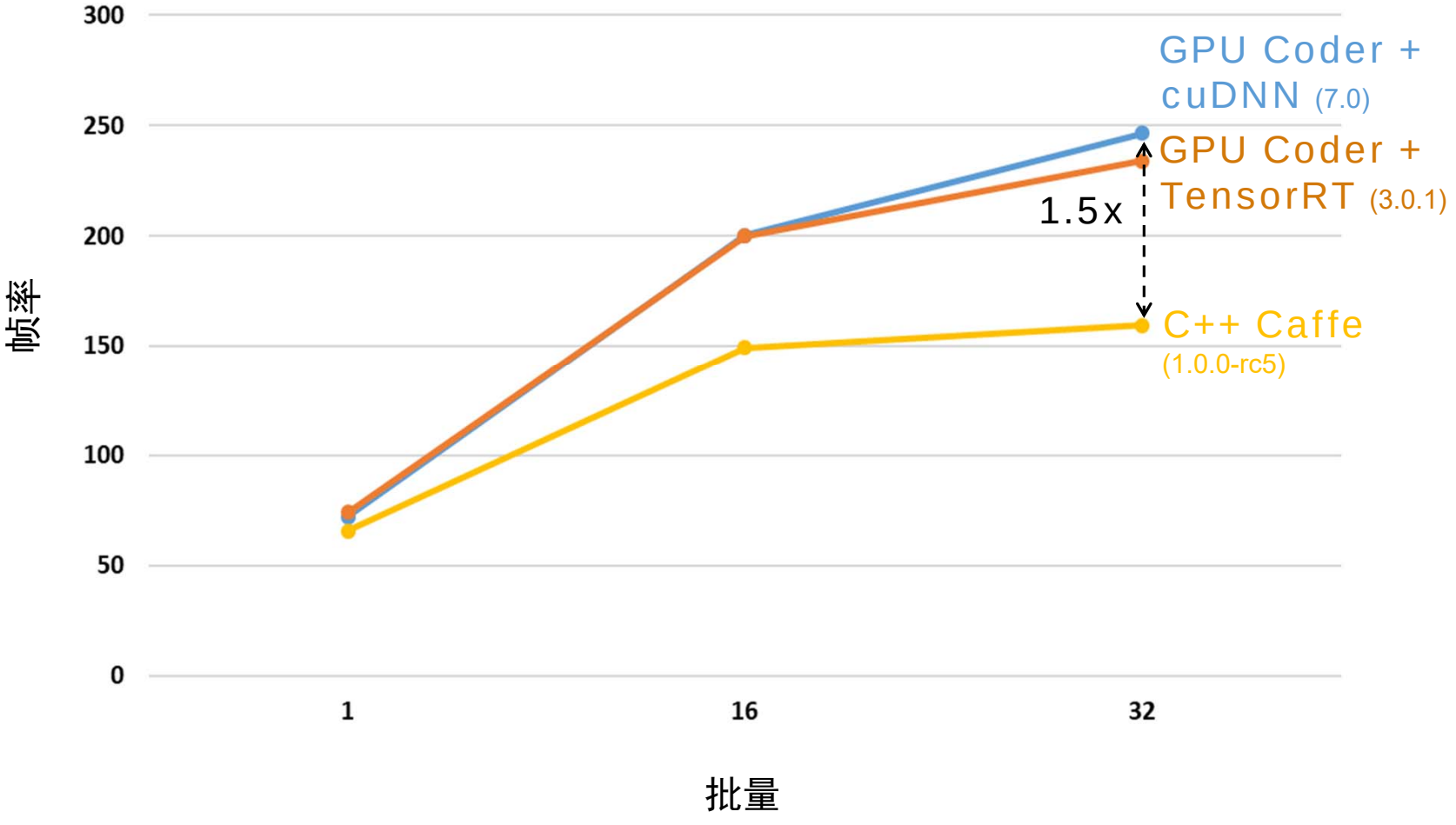
CPU	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz
GPU	Pascal Titan Xp
cuDNN	v7

VGG-16 在Titan XP上的深度学习推理

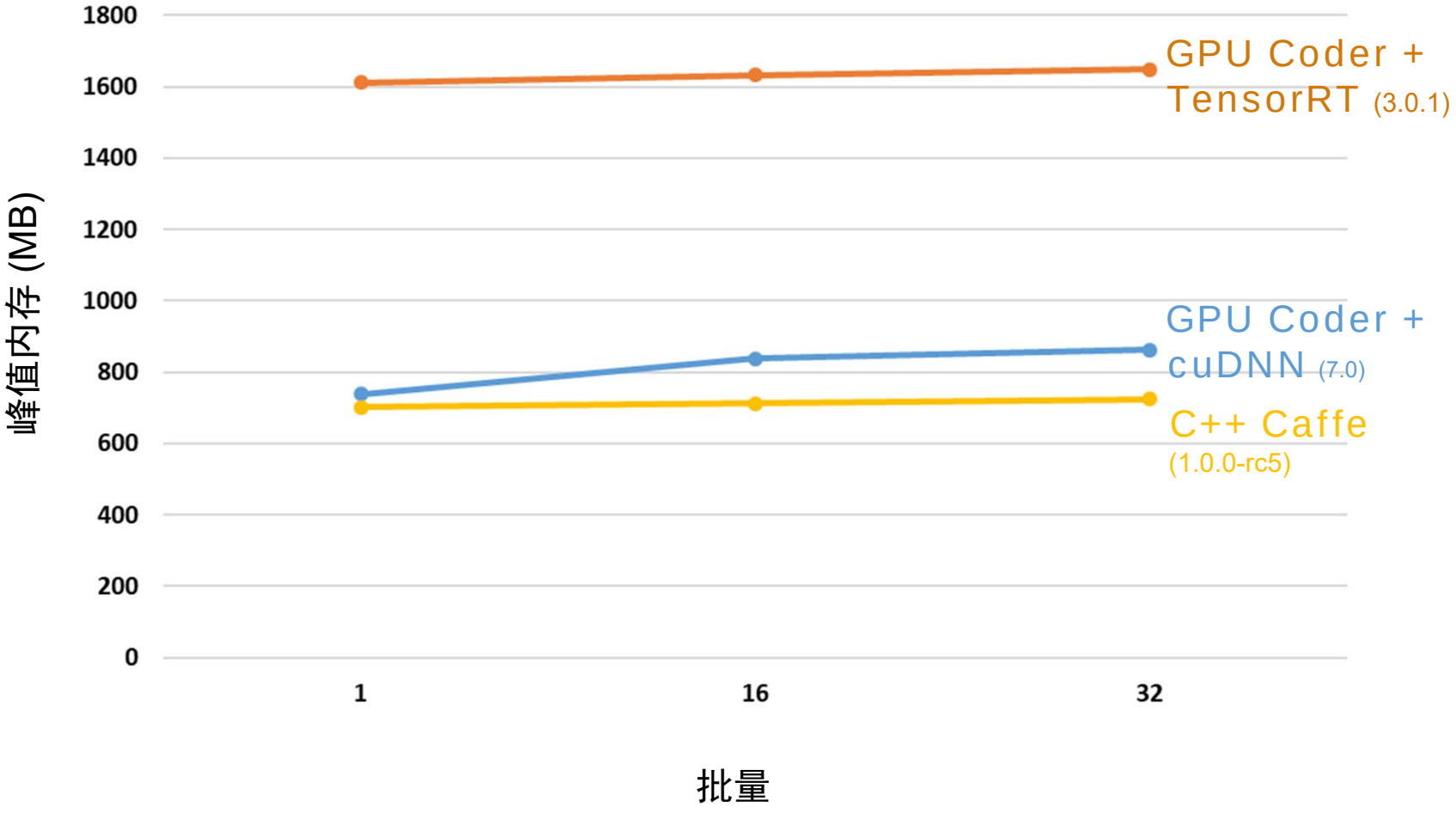


测试平台		
	CPU	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz
	GPU	Pascal Titan Xp
	cuDNN	v7

AlexNet 在Jetson TX2 上的深度学习推理:帧率性能



AlexNet 在Jetson TX2 上的深度学习推理:内存性能



在MATLAB中设计您的神经网络，使用GPU Coder 进行部署

- 管理大型图像集
- 自动化图像标签
- 轻松访问模型

- 利用GPU加速
- 扩展到HPC集群

- 使用GPU Coder自动生成代码到GPU和CPU：
  - 比TensorFlow快5倍
  - 比MXNet快2倍