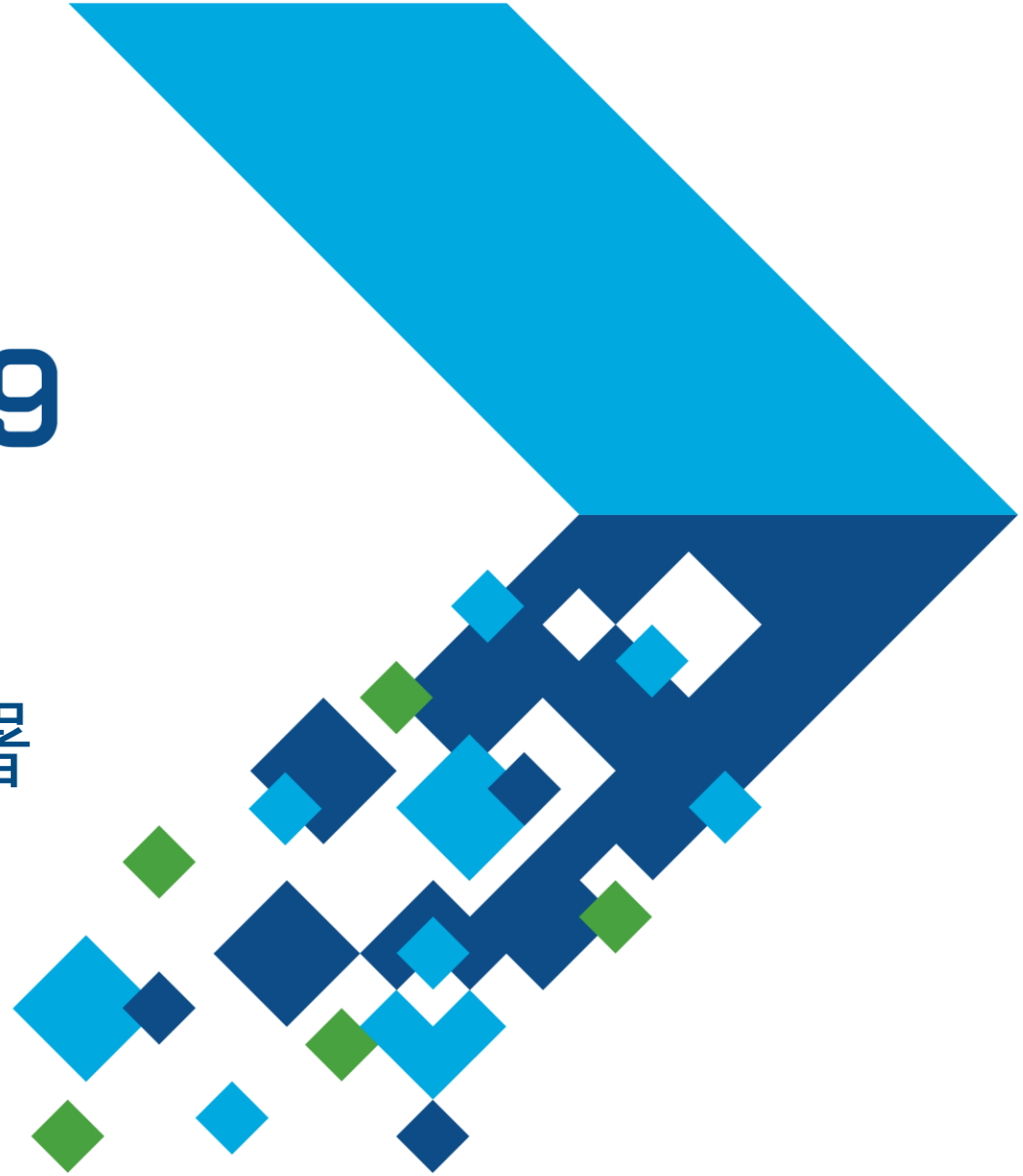
MATLAB EXPO 2019

在嵌入式 GPU 和 CPU 上部署深度神经网络

刘海伟



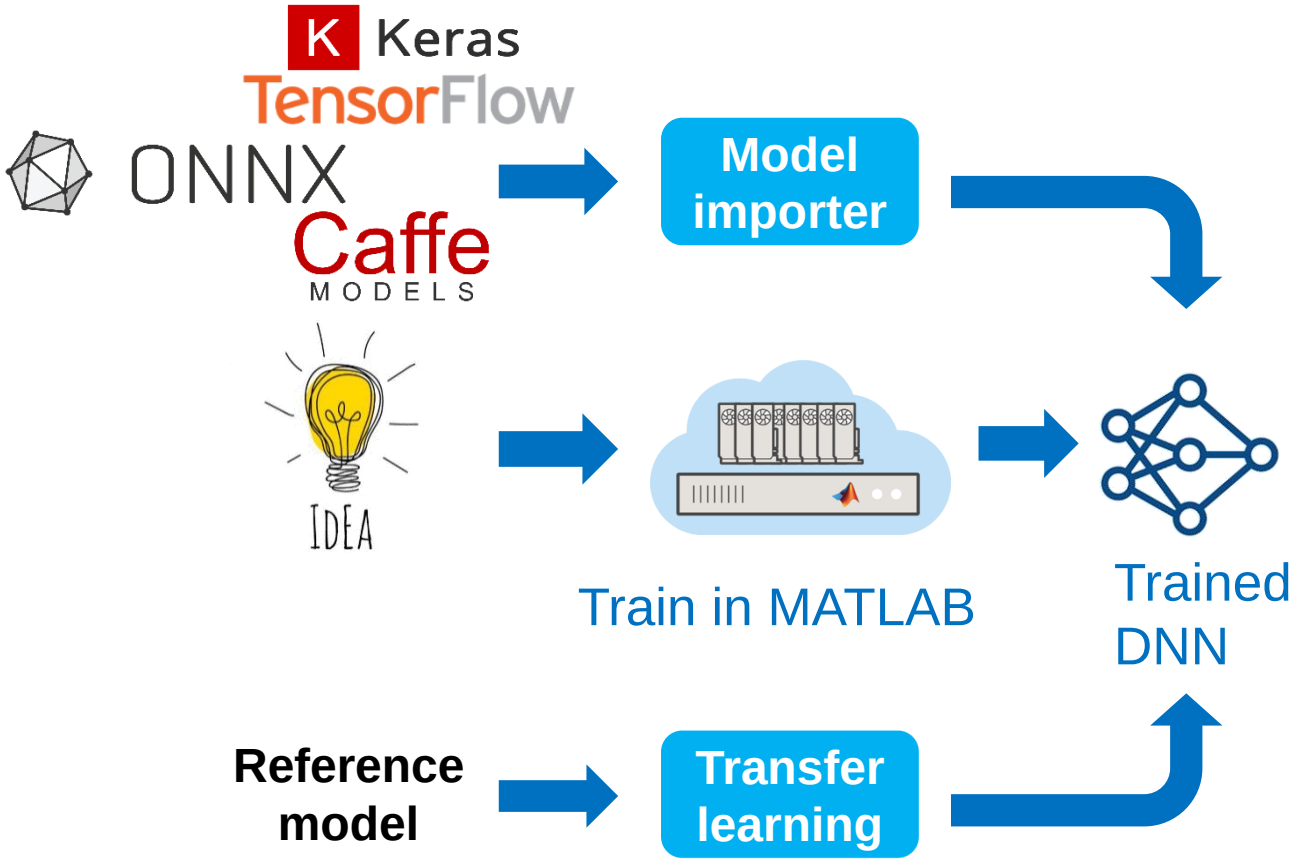
MATLAB 深度学习流程

深度神经网络
设计 + 训练

应用程序设计

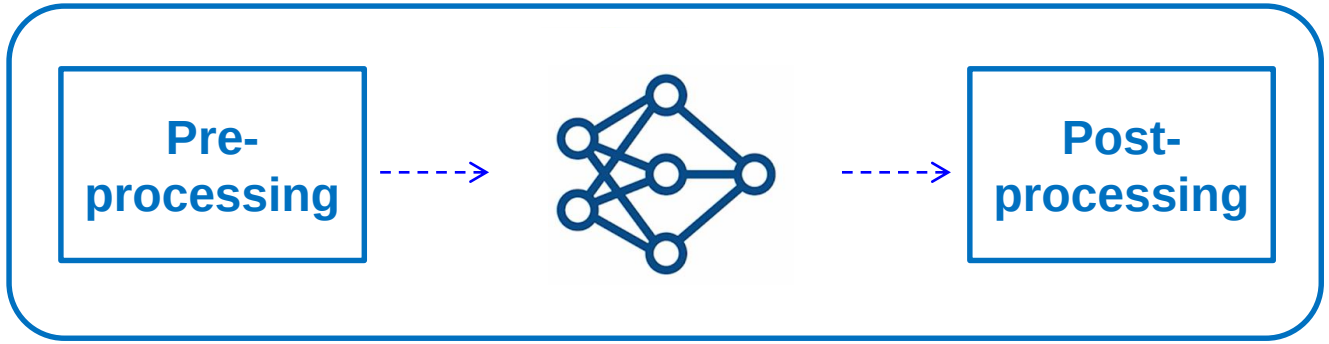
独立部署

深度神经网络设计与训练

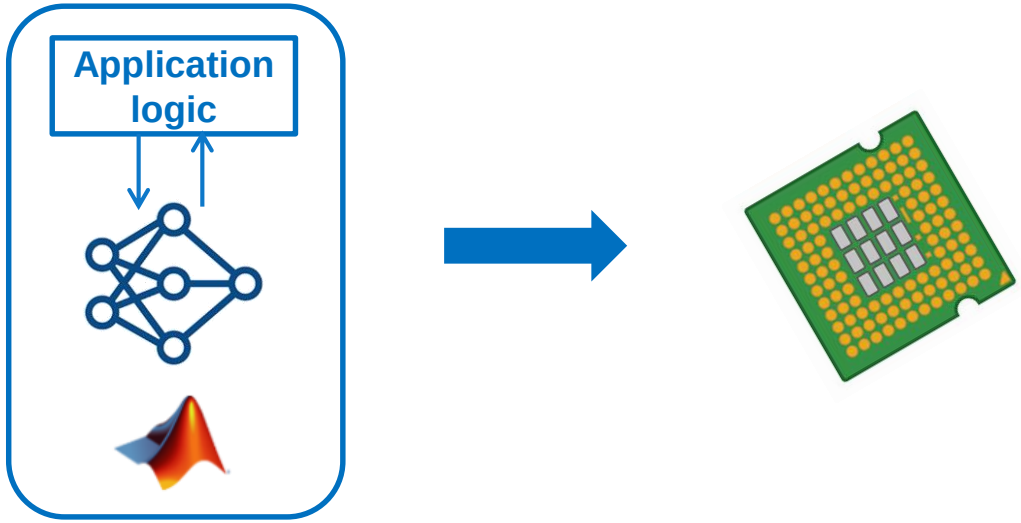


- 在 MATLAB 中设计
  - 管理大数据集
  - 自动数据标签
  - 模型易于访问
- 在 MATLAB 中训练
  - 使用 GPU 加速
  - 扩展到集群

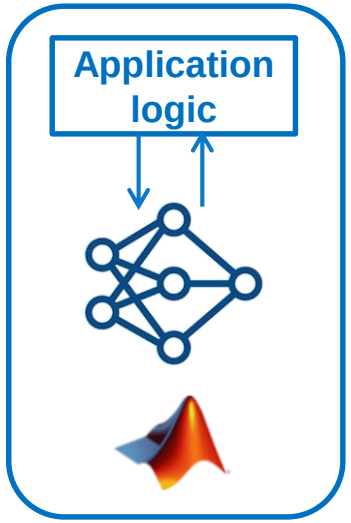
应用设计

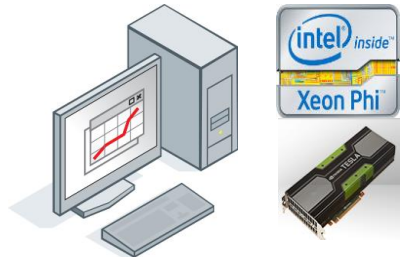


深度学习多平台部署



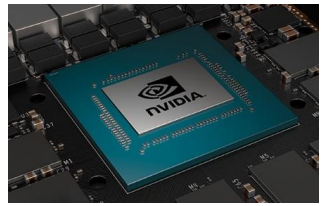
深度学习多平台部署



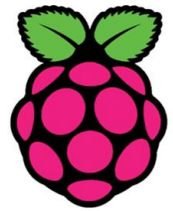


Desktop

Data Center

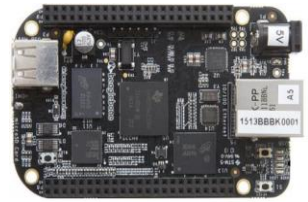


NVIDIA Jetson



Raspberry pi

Mobile

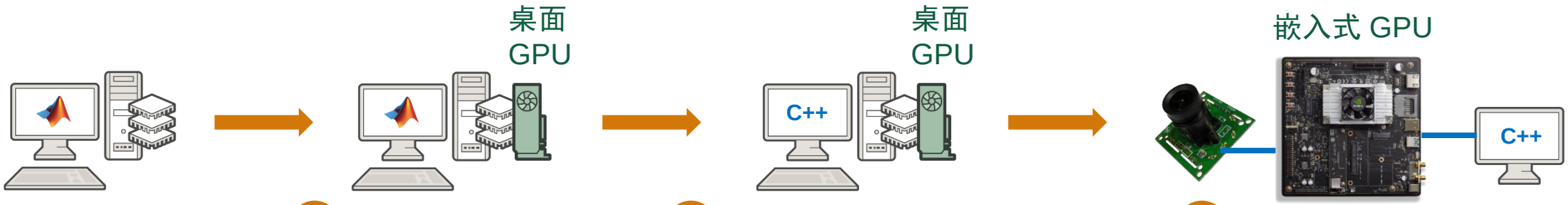


Beaglebone

. . .

嵌入式

从算法设计到嵌入式部署流程
传统方法



1. 功能性测试
2. 单元测试部署

高级语言
深度学习框架
大型、复杂的软件堆栈

3. 集成测试部署

C/C++
底层 APIs
特定应用库

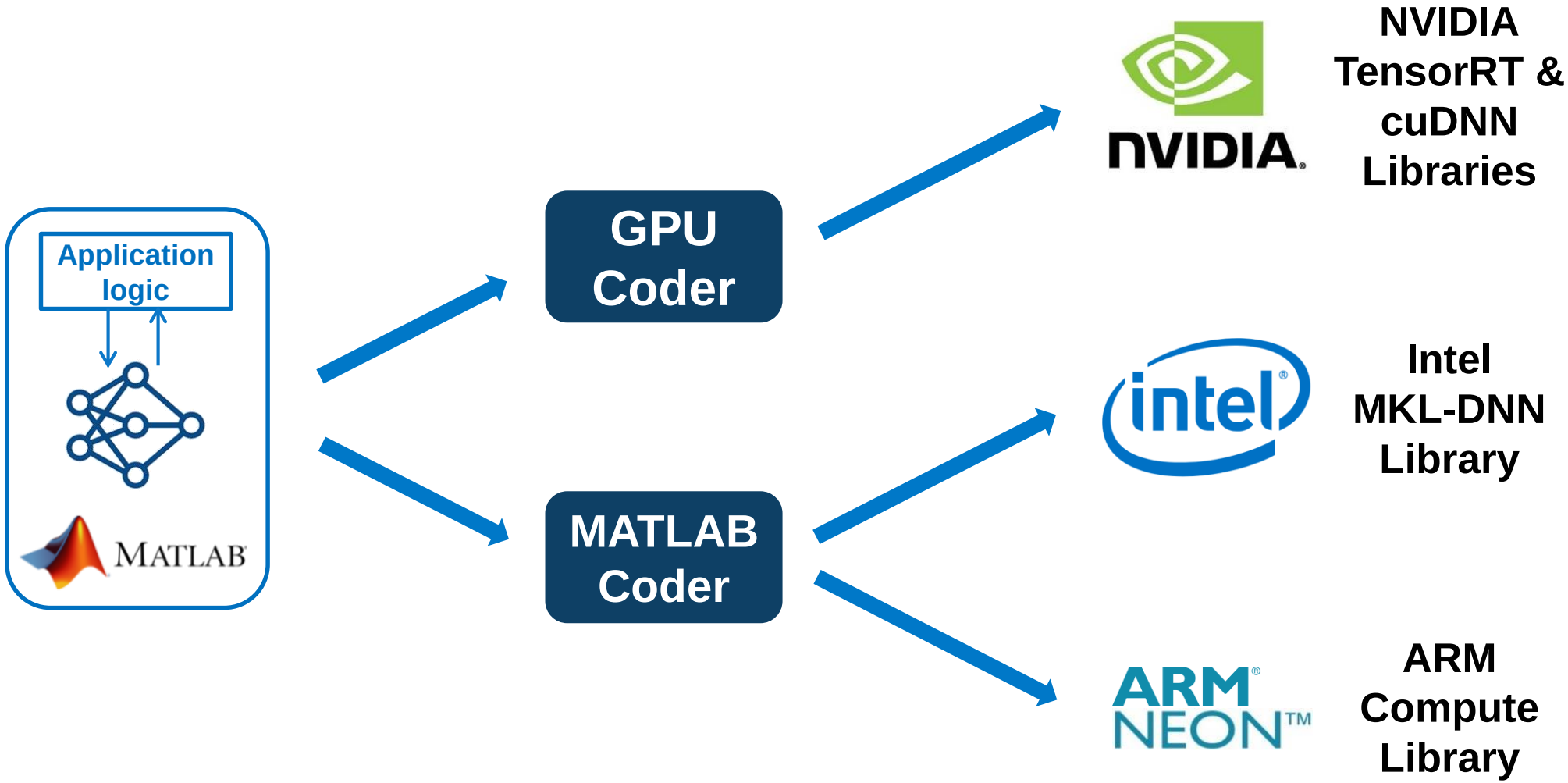
4. 实时测试

C/C++
目标优化库
内存与速度优化

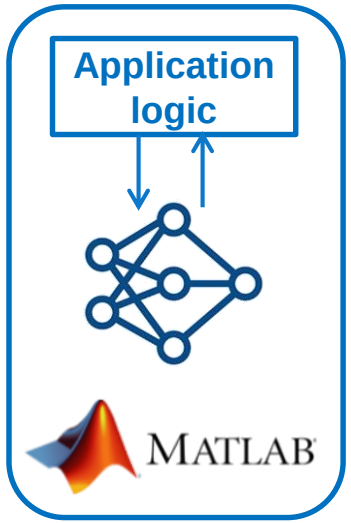
挑战

- 集成多个库和包
- 验证和维护多个实现
- 算法和供应商锁定

解决方案：使用 MATLAB Coder 和 GPU Coder 进行深度学习部署



解决方案：使用 MATLAB Coder 和 GPU Coder 进行深度学习部署



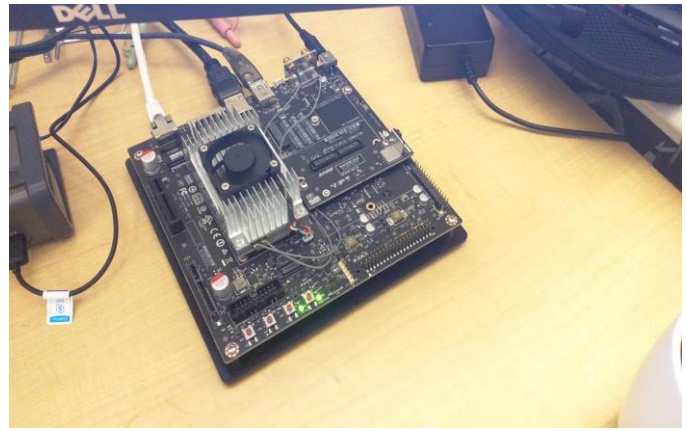
GPU Coder

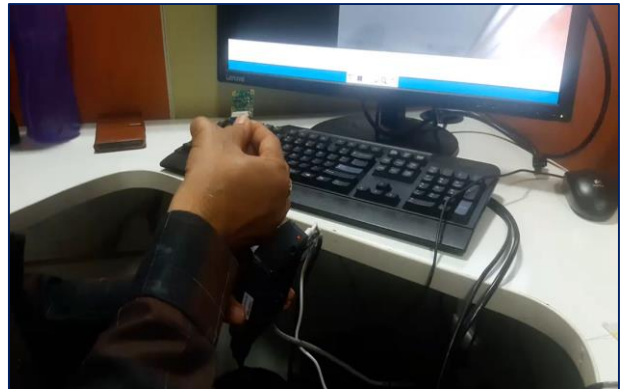
MATLAB Coder

NVIDIA

intel

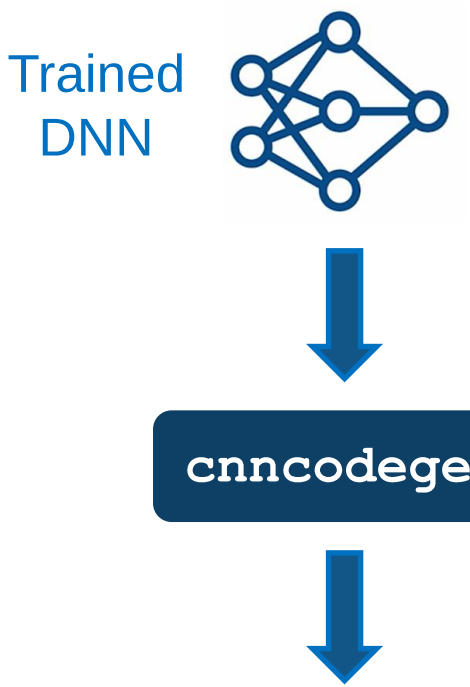
ARM NEON™



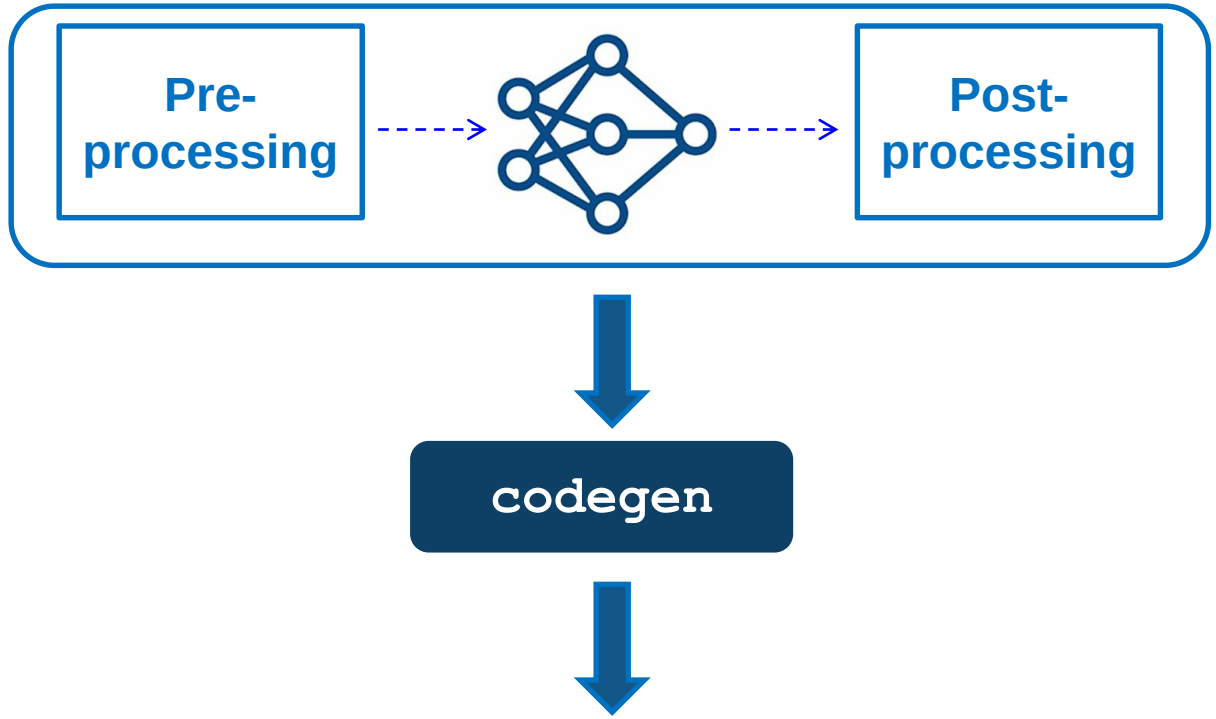


深入学习部署流程

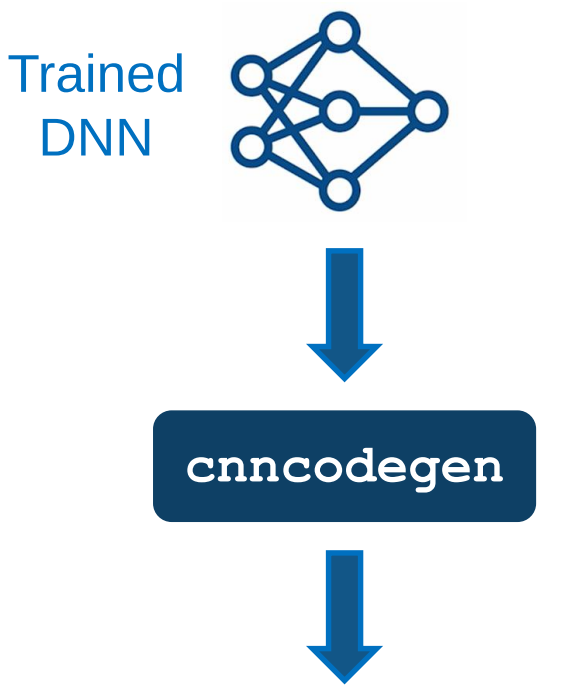
推断引擎部署



应用程序集成部署



推断引擎部署流程



推断引擎部署的步骤

1. 为训练好的模型生成代码

```
>> cnncodegen(net, 'targetlib', 'arm-compute')
```

2. 将生成的代码复制到目标器件上

3. 生成推断引擎

```
>> make -C ./codegen -f …mk
```

4. 手写主函数，用以调用推断引擎

5. 生成并测试可执行 exe

```
>> make -C ./ ……
```

深度学习推断部署

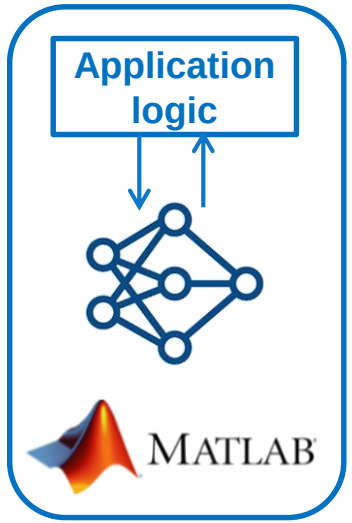
行人检测

目标库

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

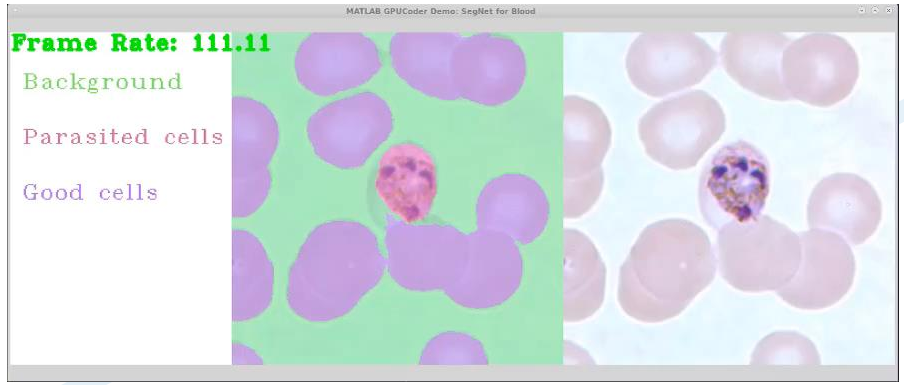
ARM Compute Library

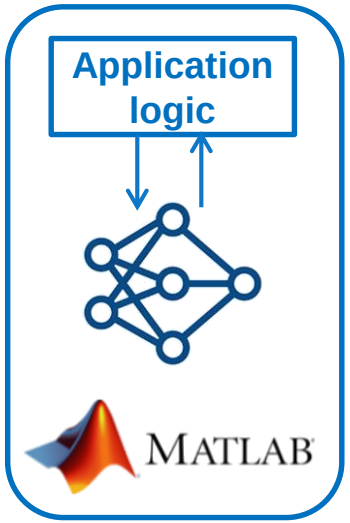


MATLAB Coder

深度学习推断部署

血液涂片分割





MATLAB Coder

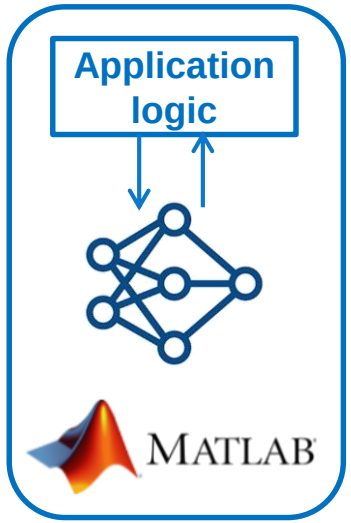
目标库

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library

深度学习推断部署



GPU Coder

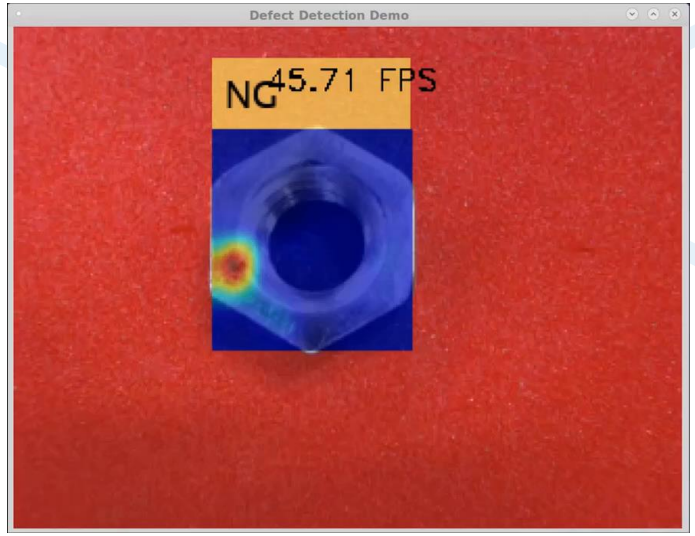
缺陷分类与检测

目标库

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library



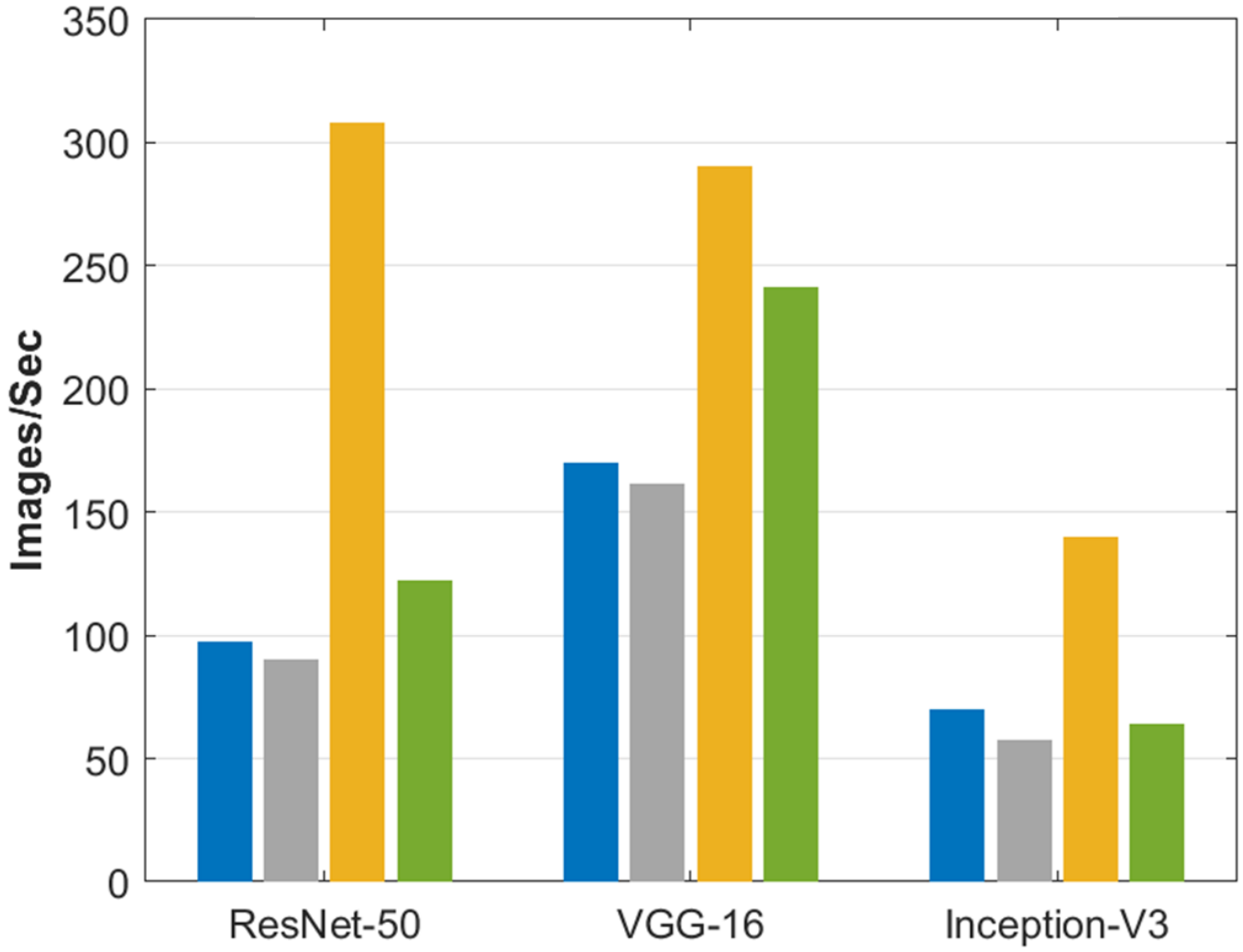

目标库
NVIDIA
TensorRT &
cuDNN
Libraries
Application
logic
MATLAB
GPU
Coder
MATLAB
Coder
Intel
MKL-DNN
Library
ARM
Compute
Library

代码性能怎么样？

生成代码的性能

- Titan V GPU 上进行 CNN 推断(ResNet-50, VGG-16, Inception V3)
- Jetson TX2 上进行 CNN 推断(ResNet-50)
- Intel Xeon CPU 上进行 CNN推断(ResNet-50 , VGG-16, Inception V3)

使用 Titan V cuDNN 进行单图像推断



TensorFlow (1.13.0)

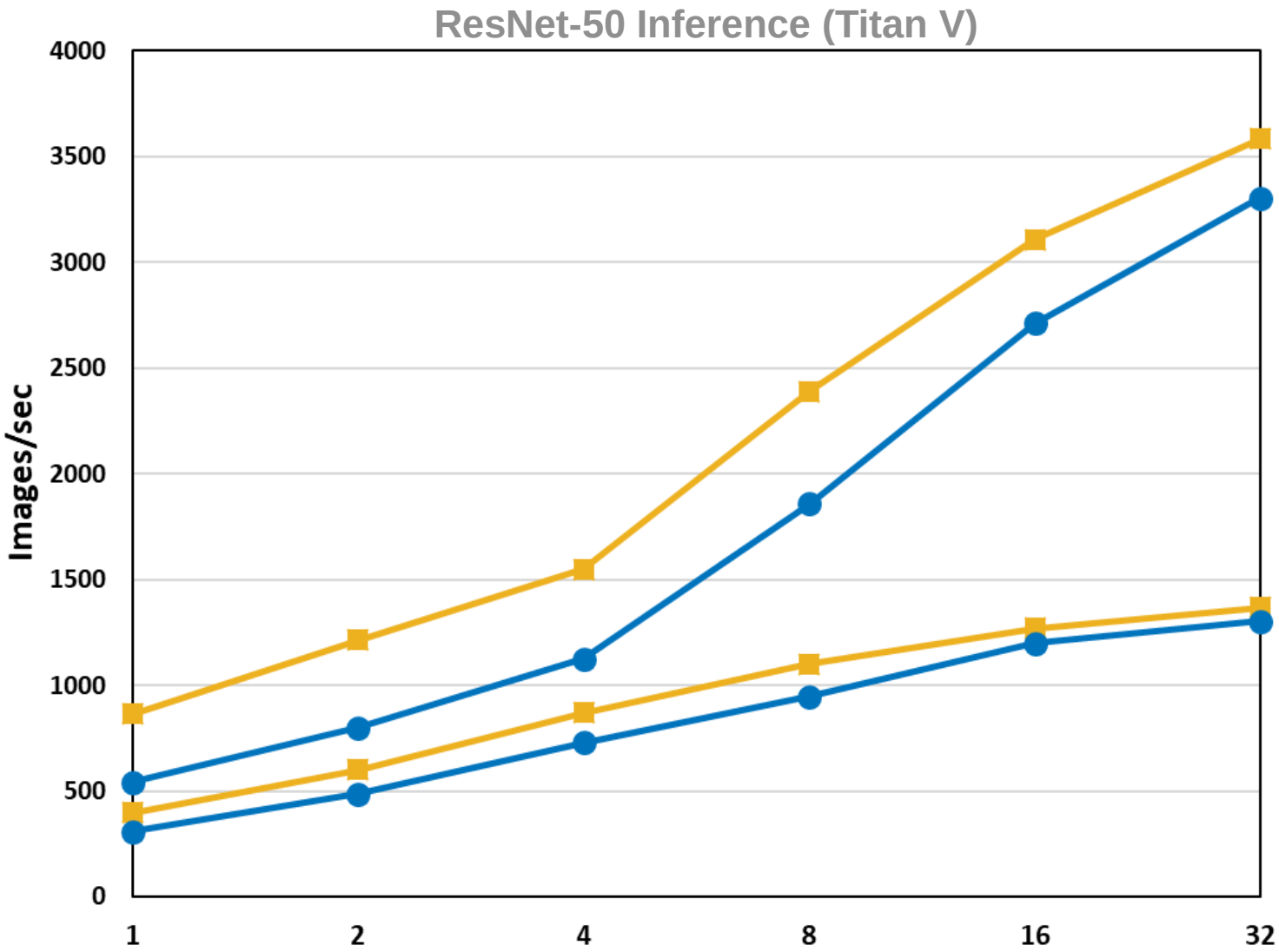
MXNet (1.4.0)

GPU Coder (R2019a)

PyTorch (1.0.0)

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7 - Frameworks: TensorFlow 1.13.0, MXNet 1.4.0 PyTorch 1.0.0

使用 TensorRT INT8 更强大的性能



GPU Coder + TensorRT (INT8)

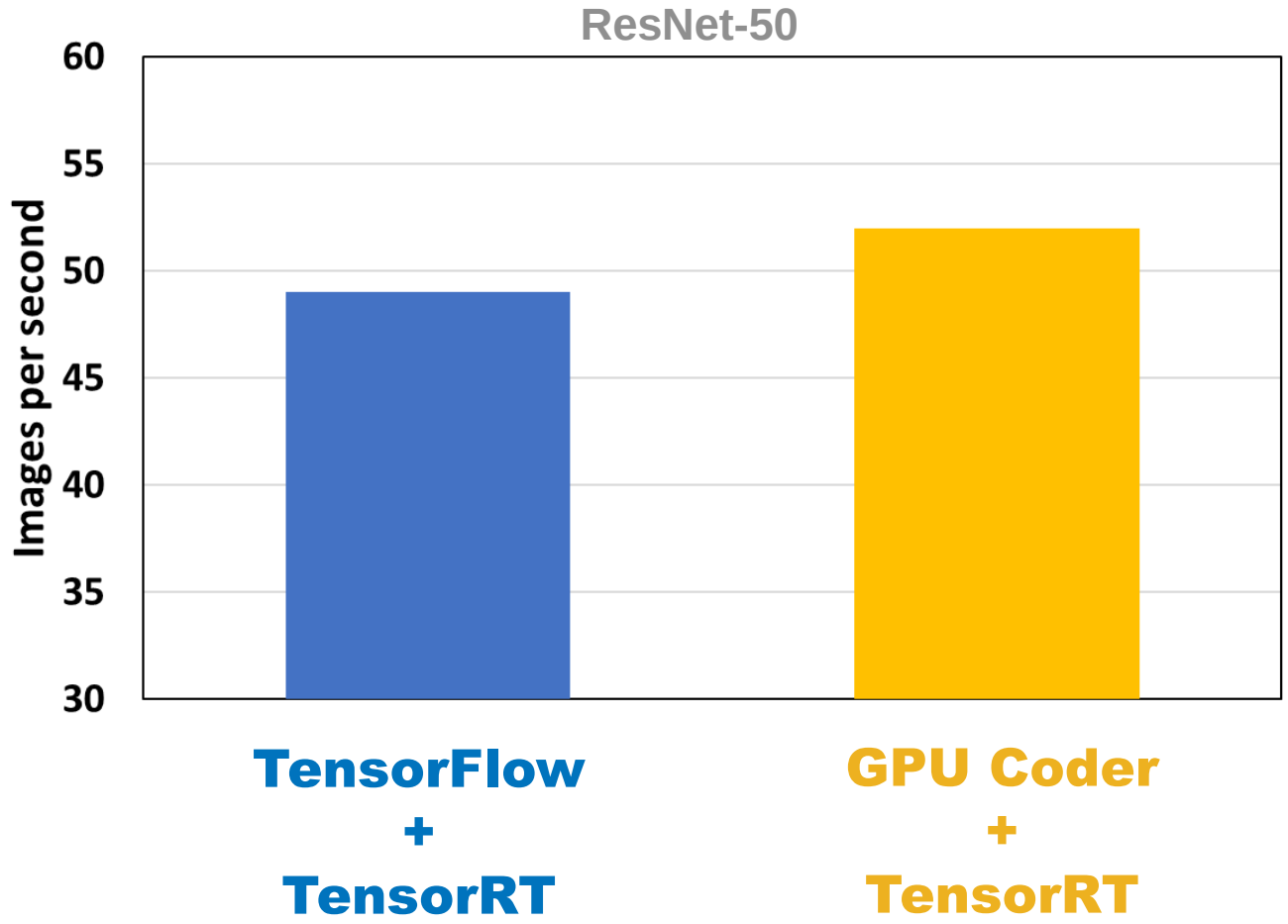
TensorFlow + TensorRT (INT8)

GPU Coder + TensorRT (FP32)

TensorFlow + TensorRT (FP32)

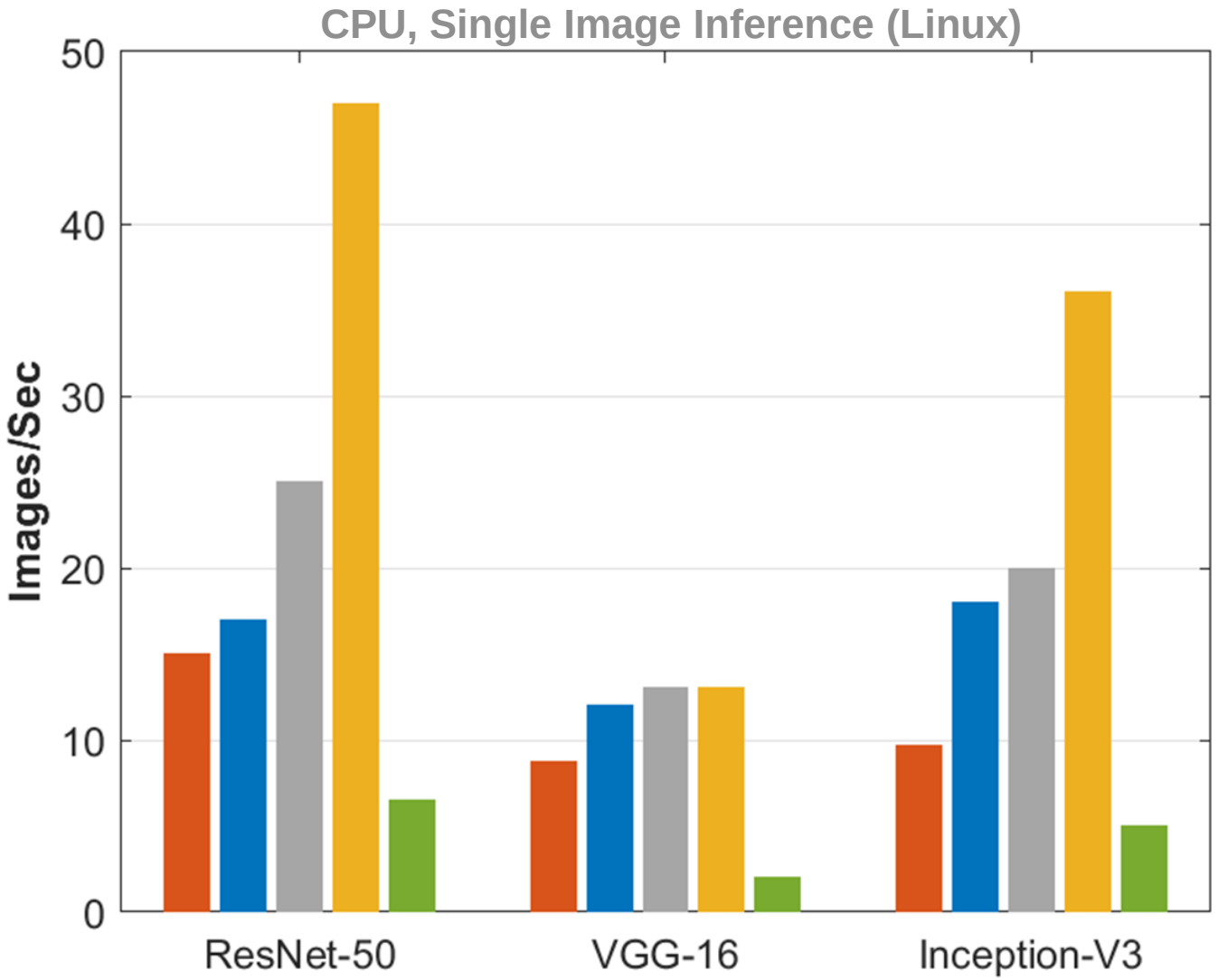
Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7.4.1 – TensorRT 5.0.2.6 - Frameworks: TensorFlow 1.13.0_rc0

Jetson TX2上的单图像推理



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

CPU 性能



MATLAB

TensorFlow

MXNet

MATLAB Coder

PyTorch

MATLAB

Intel® Xeon® CPU 3.6 GHz - Frameworks: TensorFlow 1.6.0, MXNet 1.2.1, PyTorch 0.3.1

简要总结

DNN 库非常适合推断**……**

MATLAB Coder 和 GPU Coder 生成的代码可以利用：

- NVIDIA® CUDA 库，包括 TensorRT & cuDNN
- 用于深度神经网络的 Intel® 数学内核库 (MKL-DNN)
- ARM® 移动平台计算库

简要总结

DNN 库非常适合推断.....

MATLAB Coder 和 GPU Coder 生成的代码可以利用：

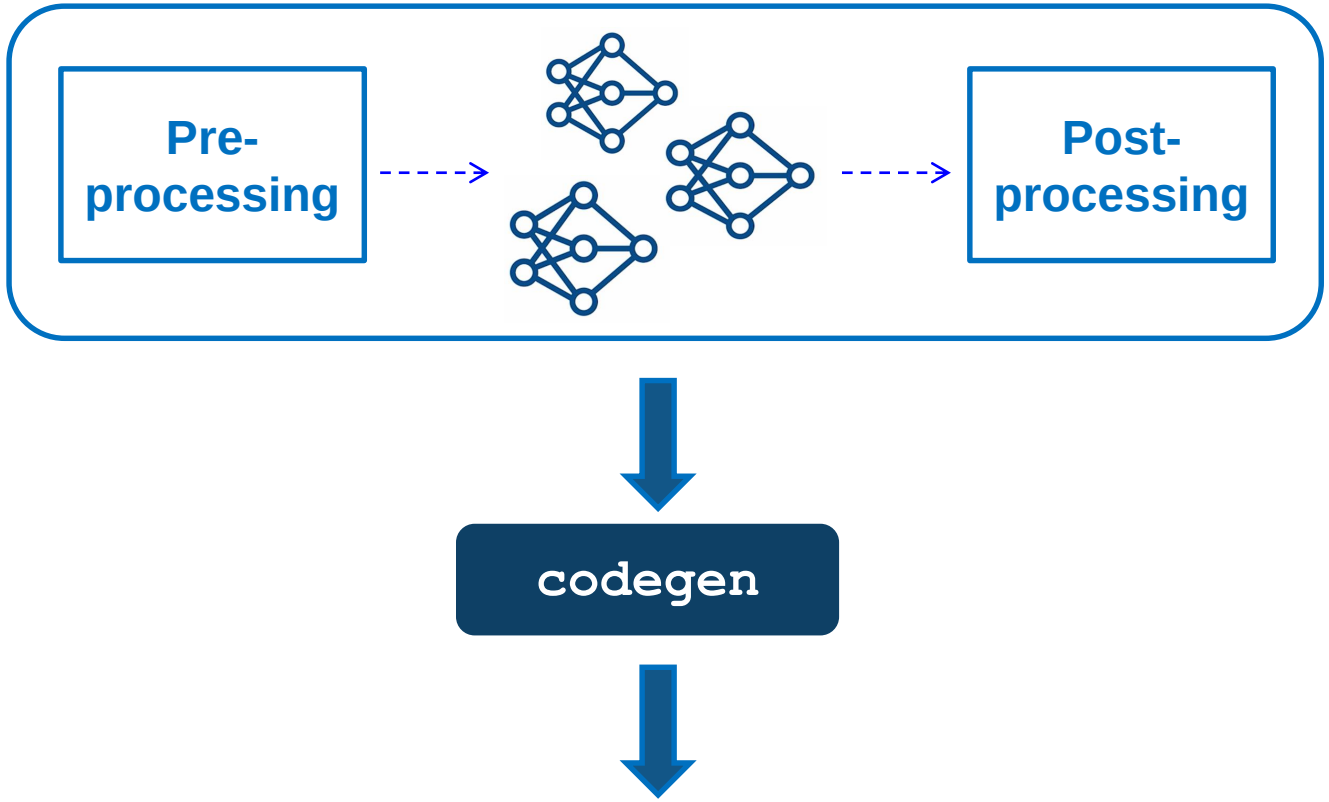
NVIDIA® CUDA 库，包括 TensorRT & cuDNN

用于深度神经网络的 Intel® 数学内核库 (MKL-DNN)

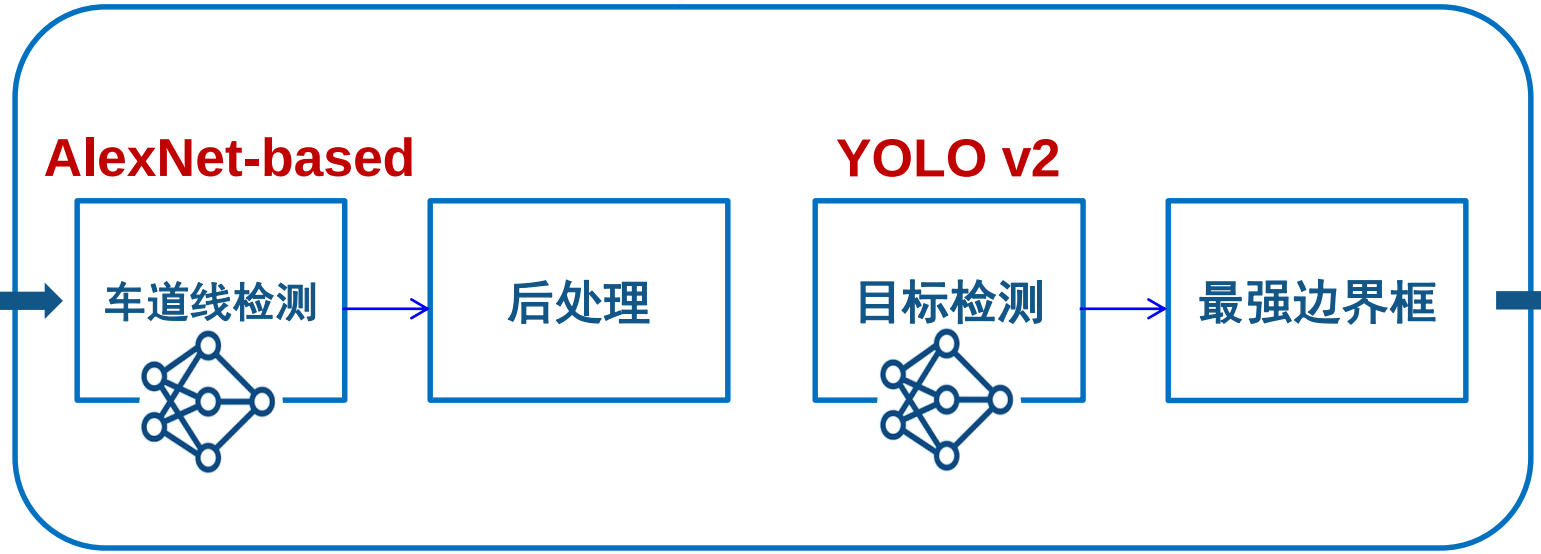
ARM® 移动平台计算库

但是，应用程序需要的
不仅仅是推断

深入学习流程：应用程序集成部署



使用 YOLO v2 进行车道线和目标检测



流程：

1) 在 CPU 上的 MATLAB 进行测试
2) 在桌面 GPU 上生成代码和测试
3) 在Jetson AGX Xavier GPU 上生成代码和测试

(1) 在 CPU 上用 MATLAB 进行测试

AlexNet-based

车道线检测

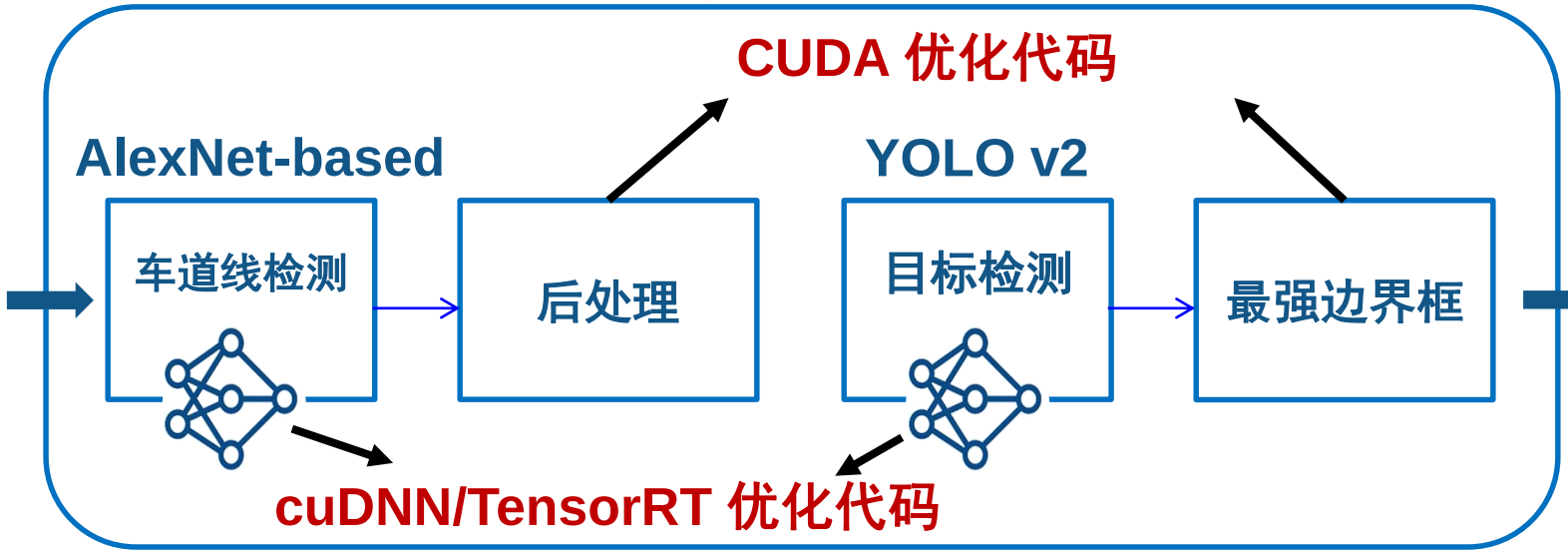
后处理

YOLO v2

目标检测

最强边界框

(2) 在桌面 GPU 上生成代码并测试



(3) 在Jetson AGX Xavier GPU 上生成代码并测试

MATLAB

CUDA 优化代码

AlexNet-based

YOLO v2

车道线检测

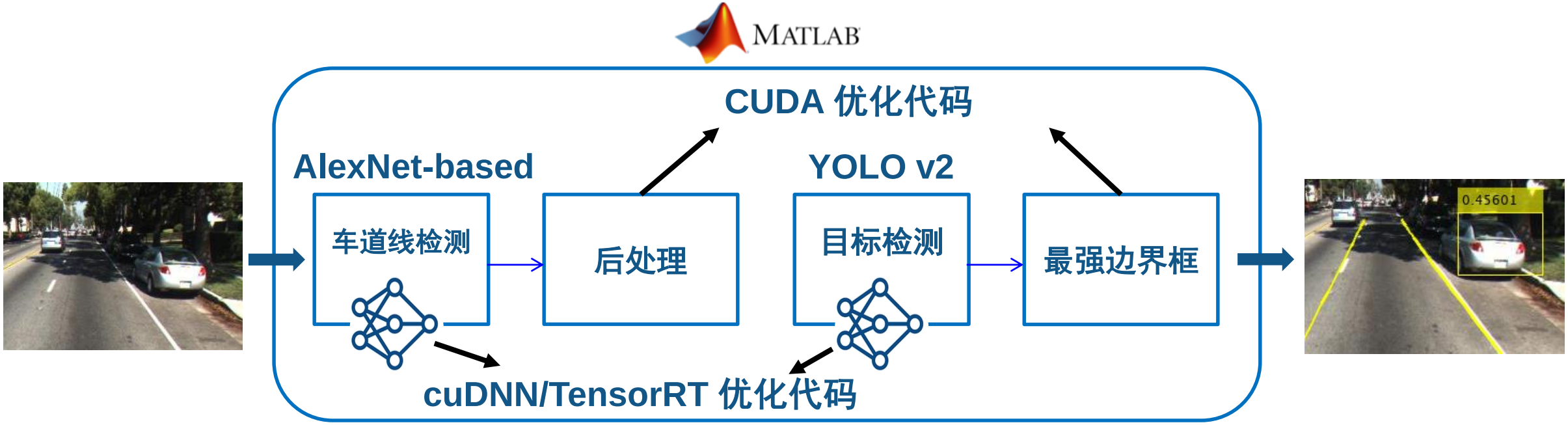
后处理

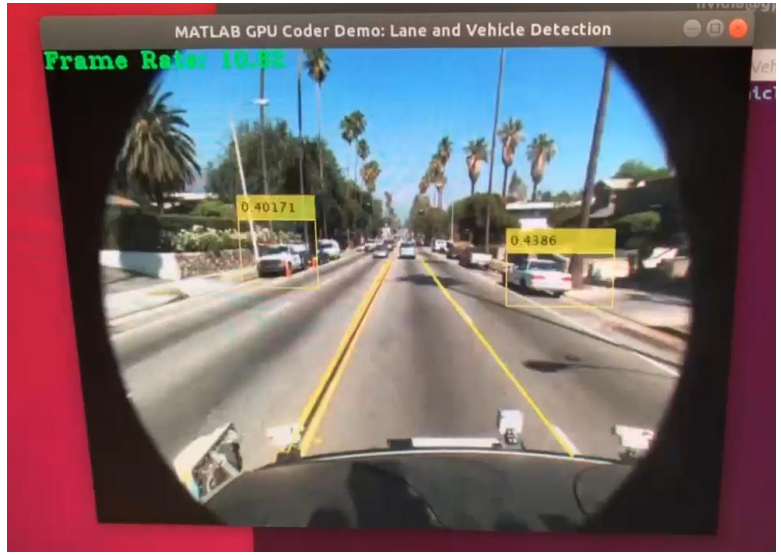
目标检测

最强边界框

cuDNN/TensorRT 优化代码

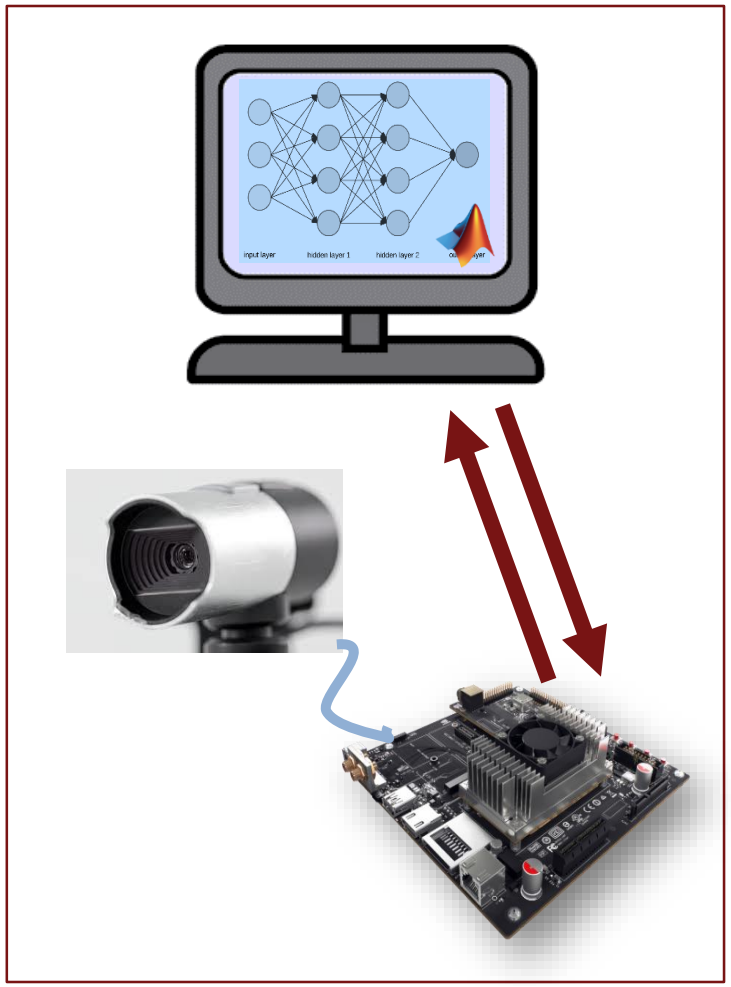
使用YOLO v2进行车道线和目标检测



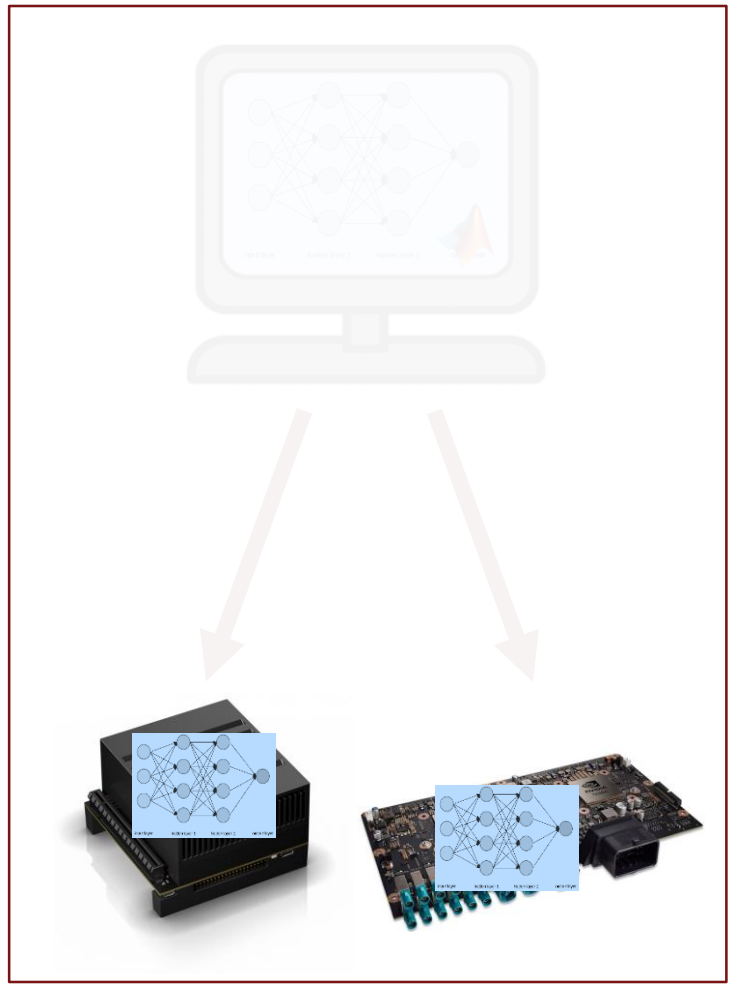


1) CPU 上运行
2) 在桌面 GPU 上生成代码，运行速度提高7倍
3) 在Jetson AGX Xavier GPU 上生成代码和测试

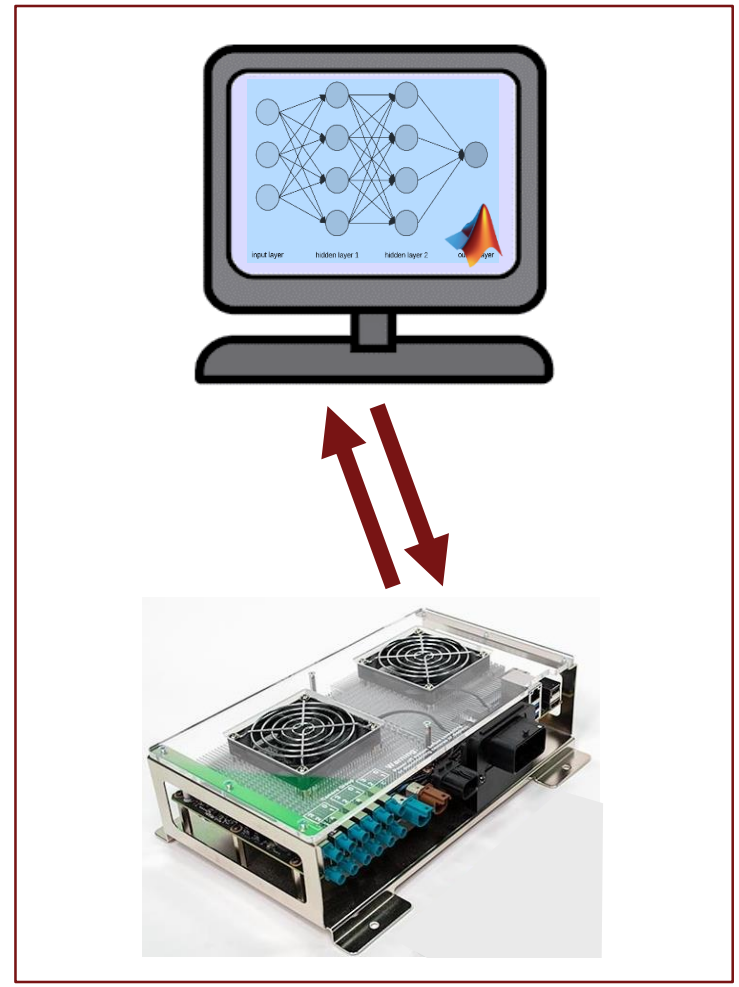
访问硬件



从 MATLAB 访问外设

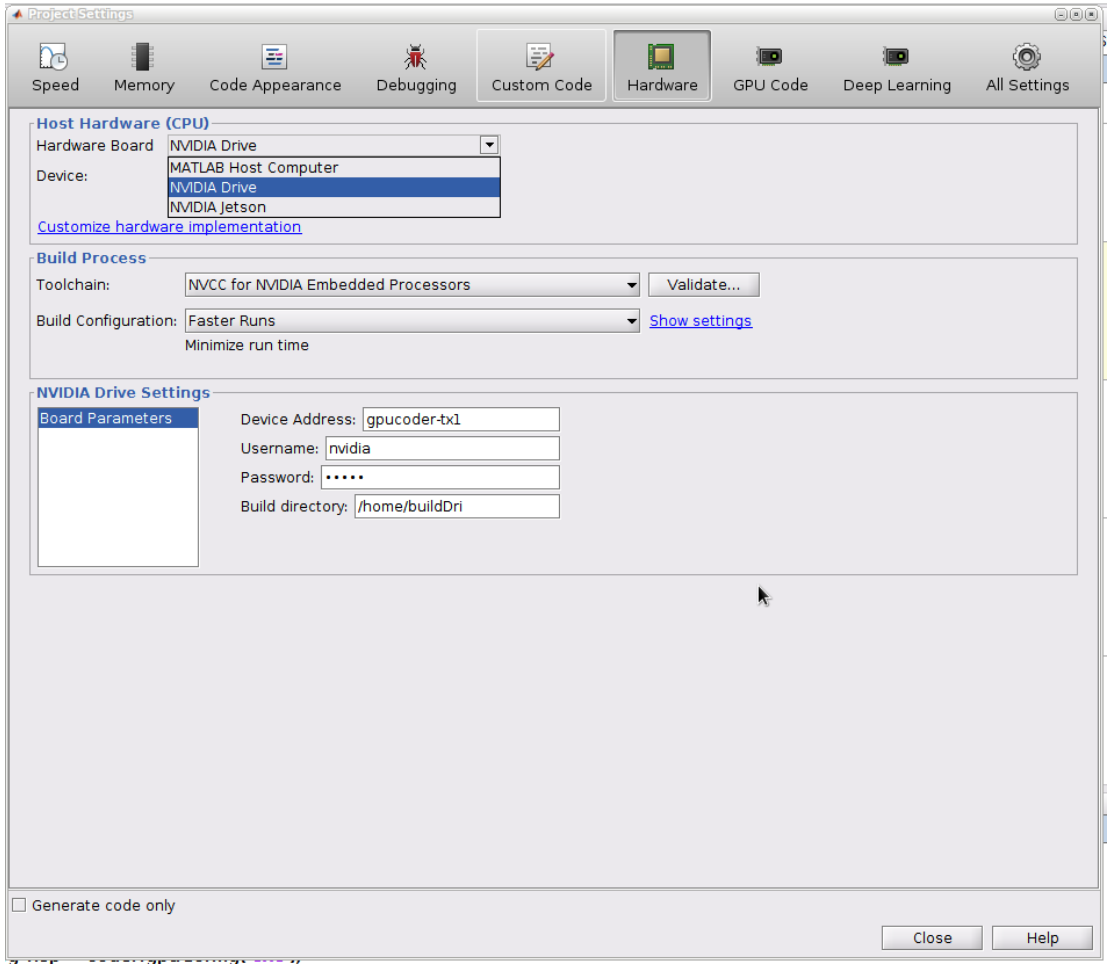


部署独立应用程序

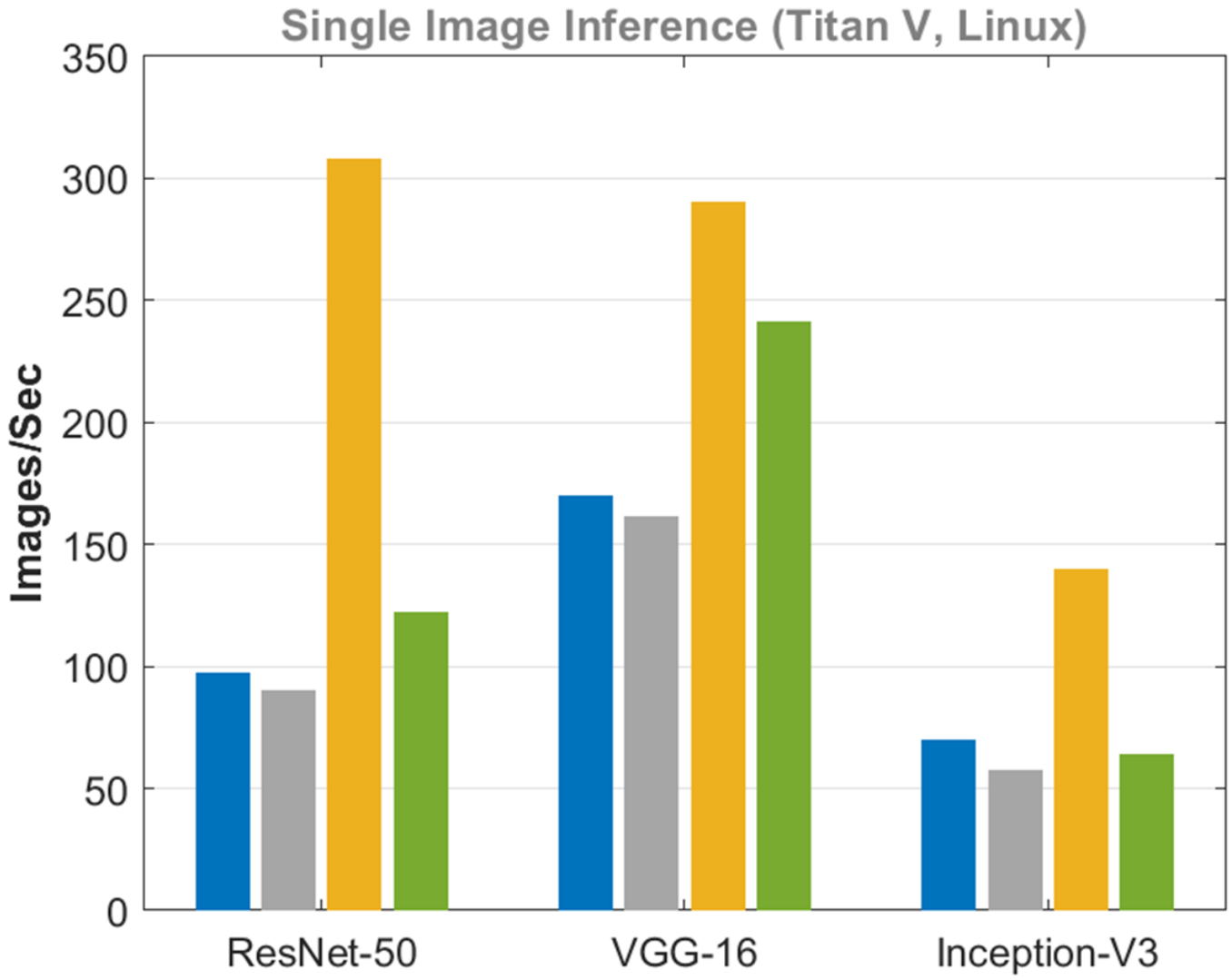


处理器在环验证

通过 Apps 和命令行实现目标硬件部署




```
%% Deploy and launch through  NVIDIA HSP

%% setup hardware object
% create jetson/drive hardware object with IP or hostname of jeston/drive
%also pass credentials for login
hwObj = jetson('gpucoder-tx2-2','ubuntu','ubuntu');
hwObj.setupCodegenContext;

%% setup codegen config object
% create congen config and connect to hardware object.
cfg_hsp = coder.gpuConfig('exe');
cfg_hsp.Hardware = coder.hardware(hwObj.BoardPref);
buildDir ='~/buildDir';
cfg_hsp.Hardware.BuildDir = buildDir;

%% 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', buildDir);
hwObj.putFile('synsetWords.txt', buildDir);

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

%% copy the output file back to host machine
hwObj.getFile([buildDir '/tOut.txt']);
```

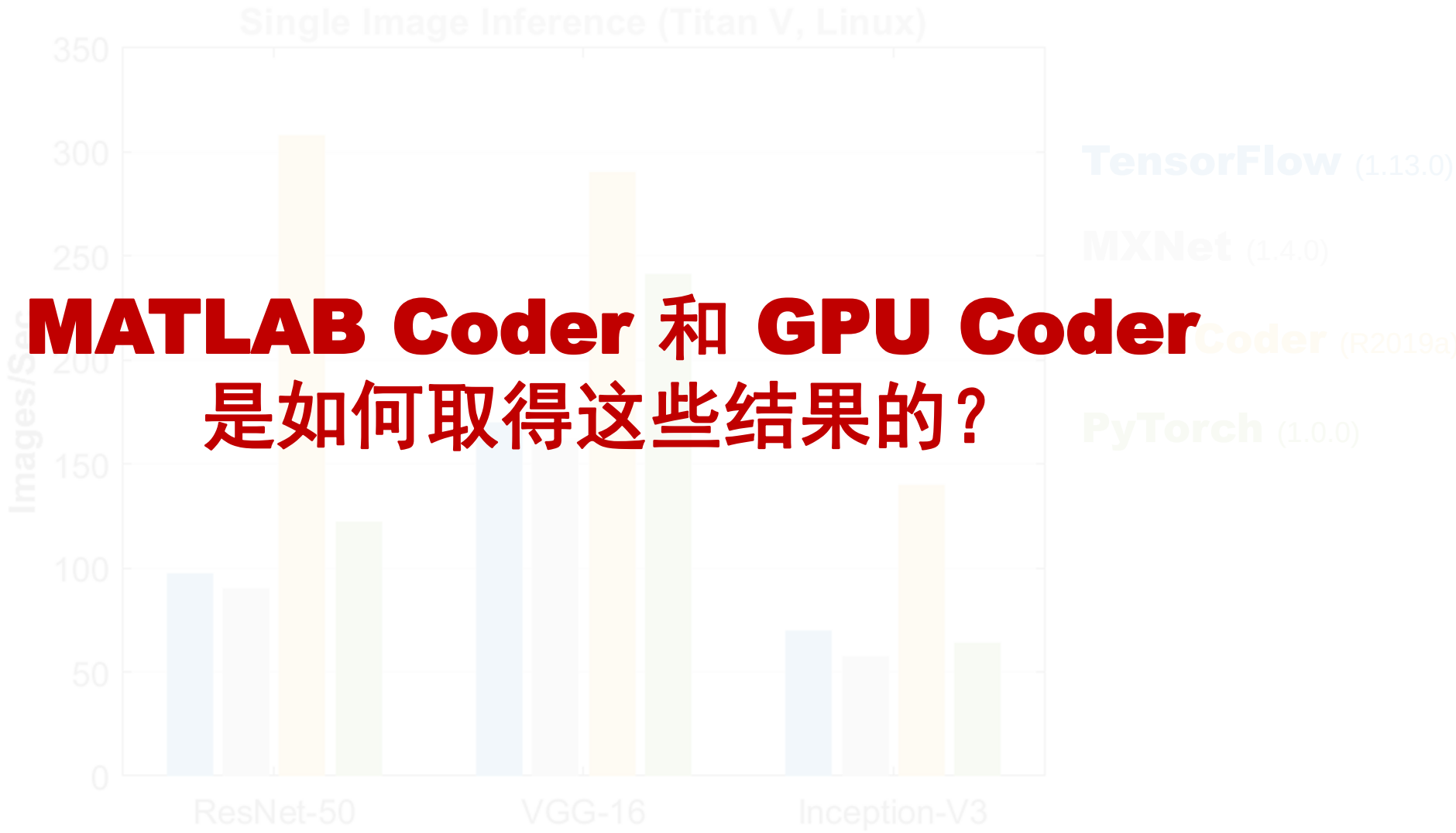


TensorFlow (1.13.0)

MXNet (1.4.0)

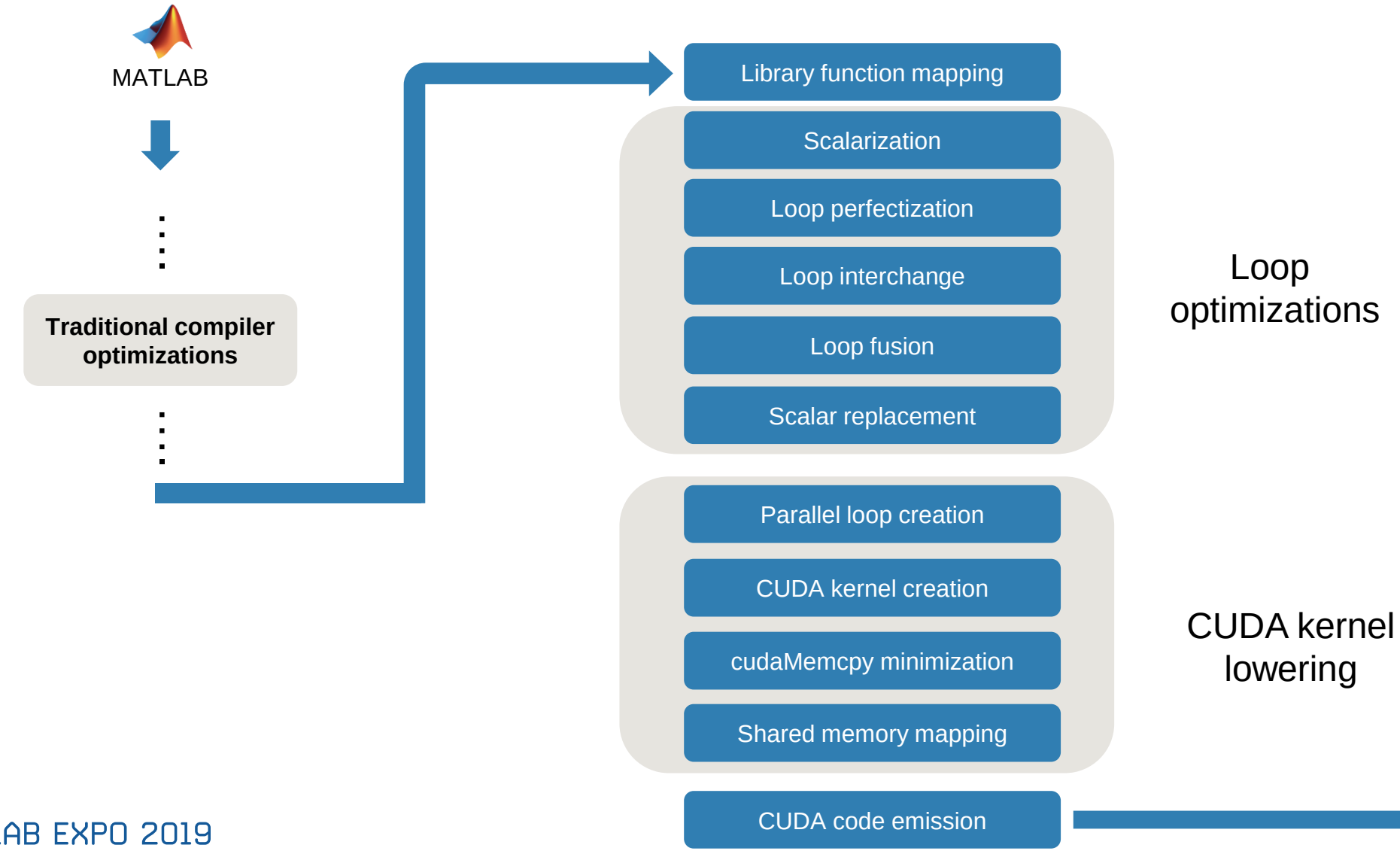
GPU Coder (R2019a)

PyTorch (1.0.0)

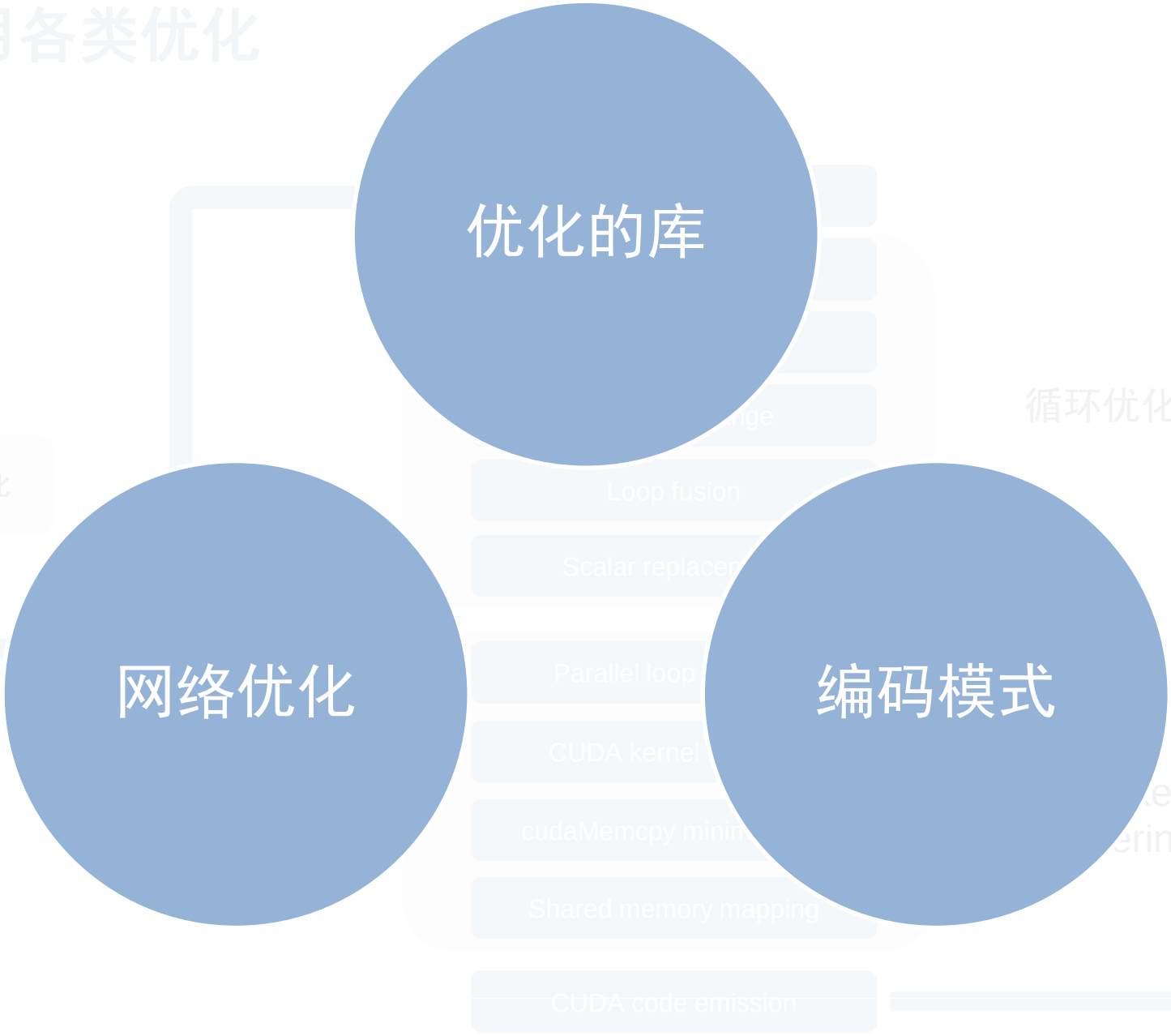


MATLAB Coder 和 GPU Coder
是如何取得这些结果的？

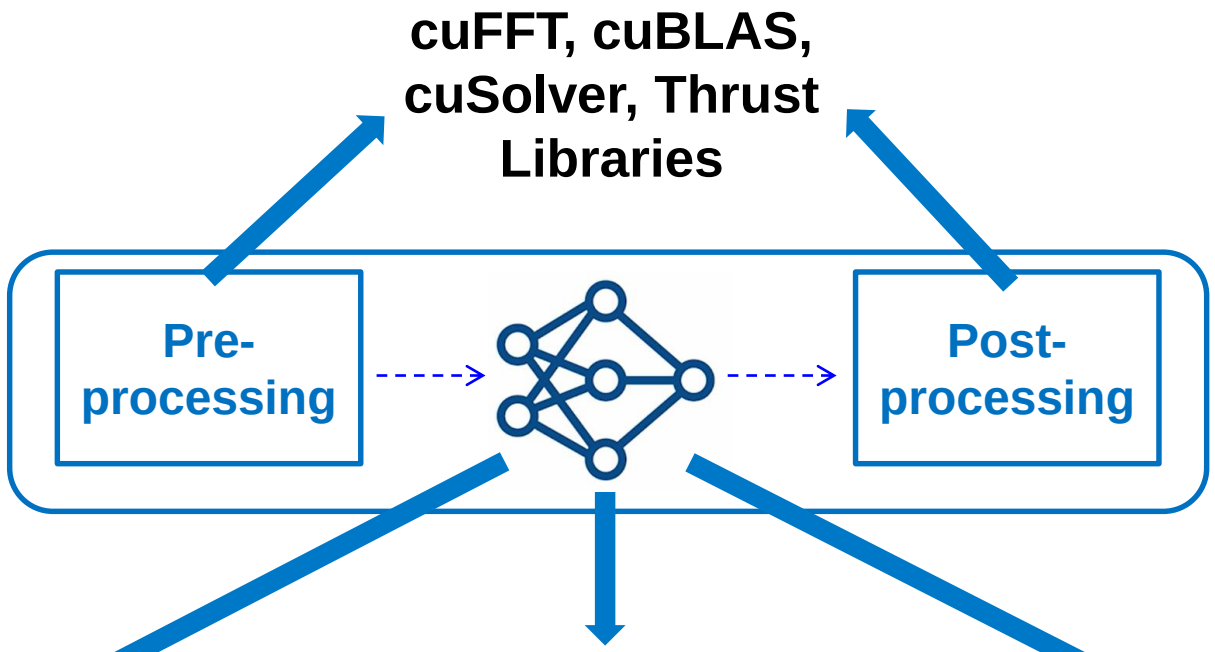
Coders 应用各类优化



Coders 应用各类优化



生成的代码调用优化的库



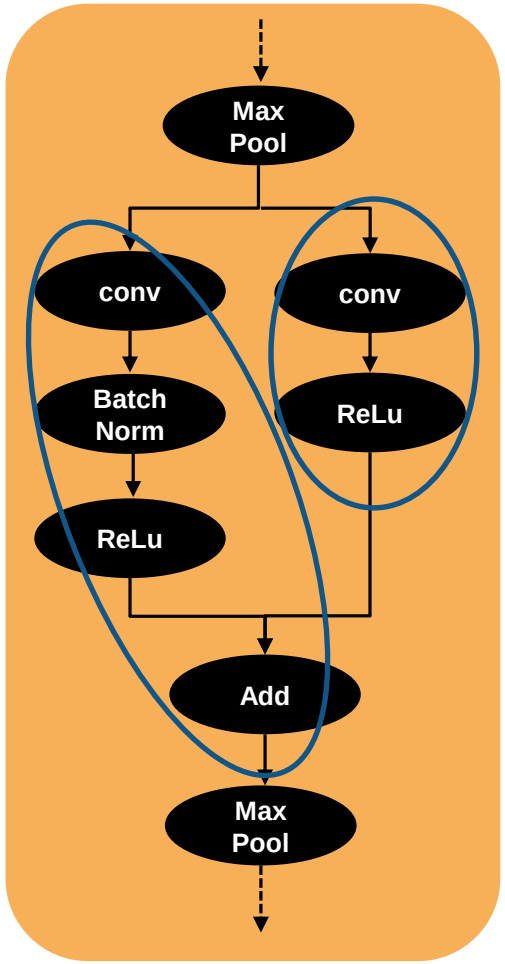
Intel MKL-DNN Library

NVIDIA TensorRT & cuDNN Libraries

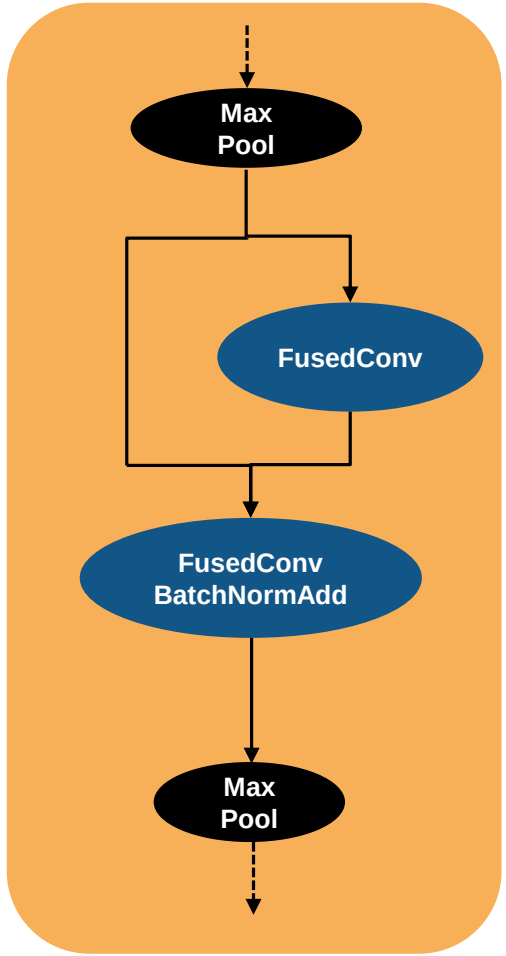
ARM Compute Library

深度学习网络优化

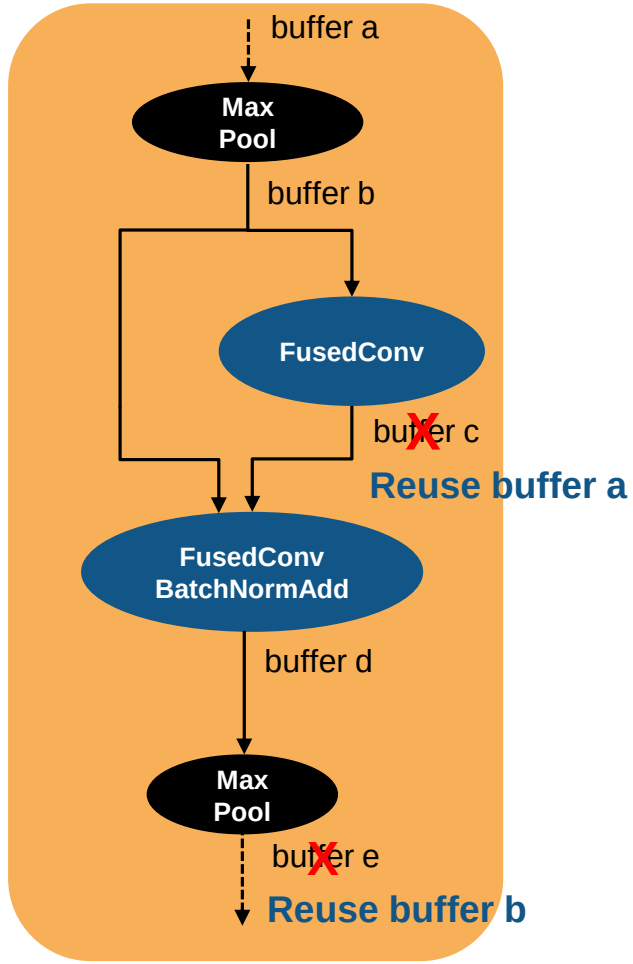
性能
1. 优化的库
2. **网络优化**
3. 编码模式



网络



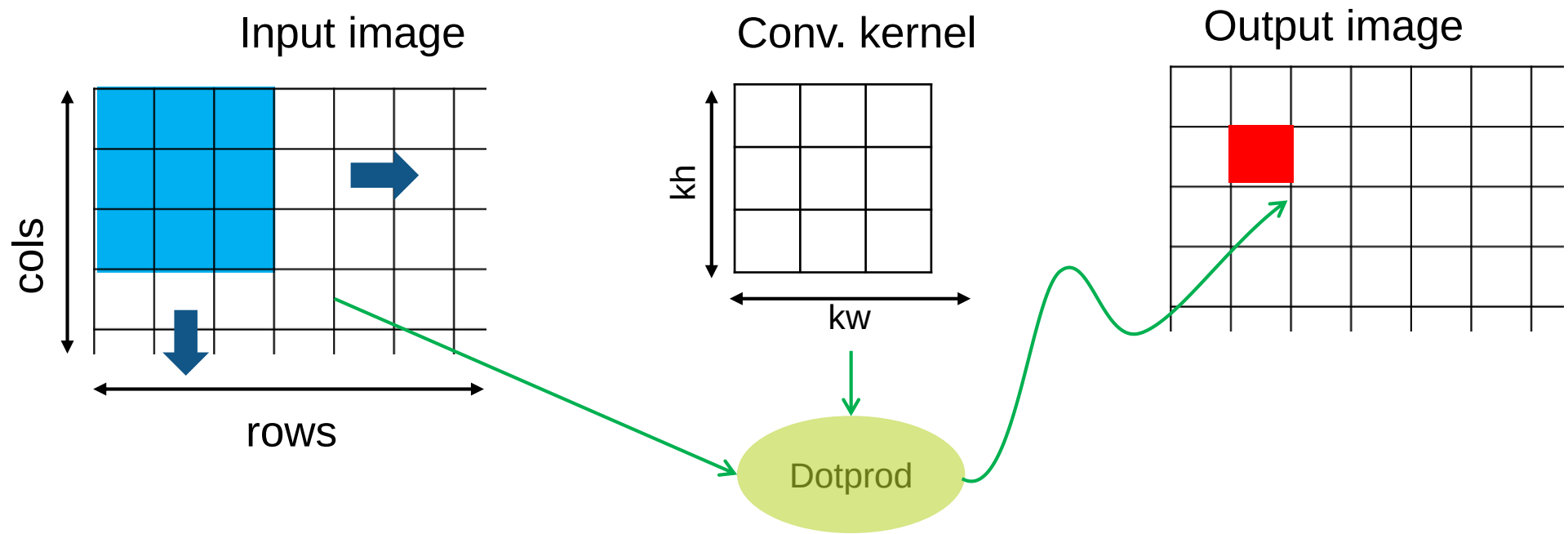
层融合
优化计算



缓冲区最小化
优化内存

编码模式：Stencil 内核

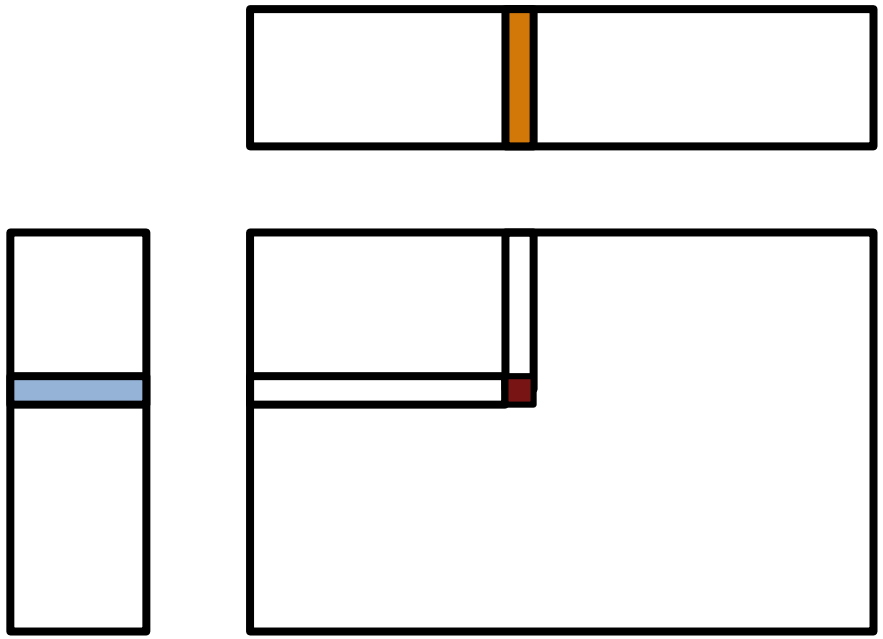
- 自动应用于图像处理函数 (e.g. imfilter, imerode, imdilate, conv2, …)
- 使用 *gpucoder.stencilKernel()* 手动启用

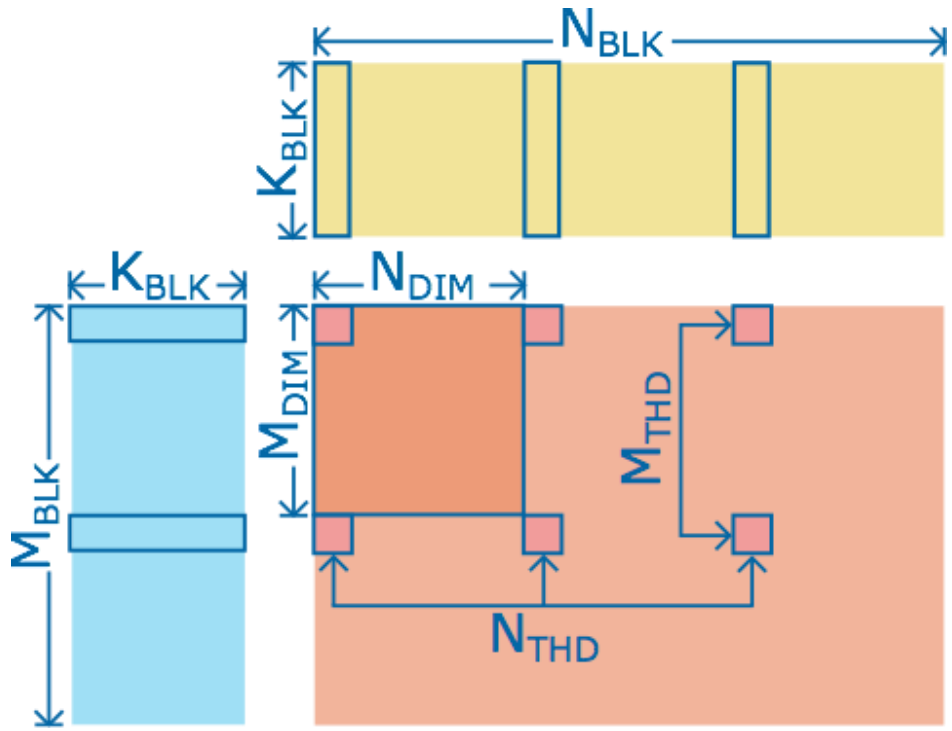


编码模式：Matrix-Matrix 内核

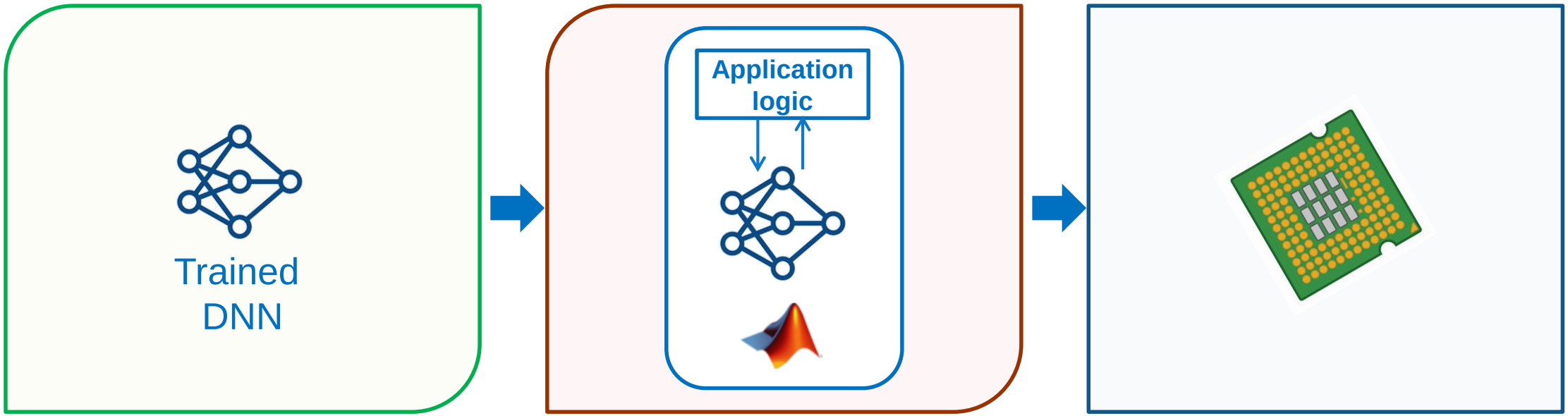
性能
1. 优化的库
2. 网络优化
3. **编码模式**

- 自动应用于很多 MATLAB 函数 (e.g. matchFeatures SAD, SSD, pdist, …)
- 使用 *gpucoder.matrixMatrixKernel()* 手动启动





MATLAB 深度学习流程



深度神经网络
设计 + 训练

应用程序设计

独立部署

MATLAB 中的深度学习流程