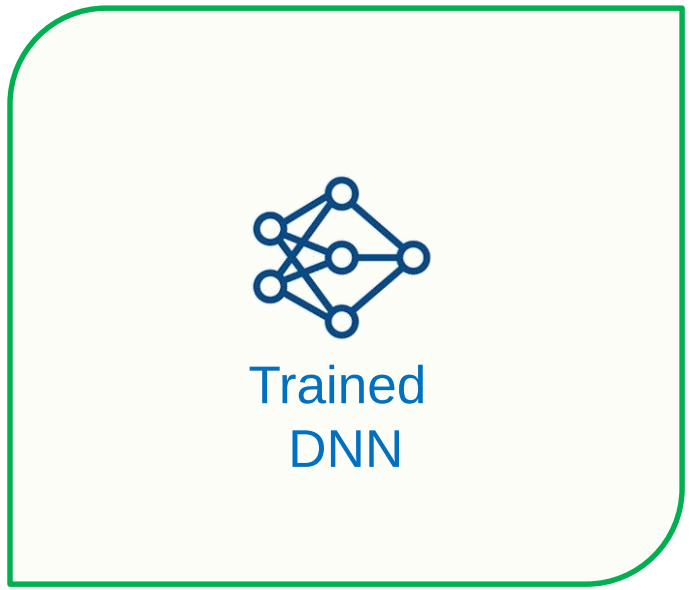
MATLAB EXPO 2019

Deploying Deep Neural Networks to Embedded GPUs and CPUs

Steven Thomsett

Deep Learning Workflow in MATLAB

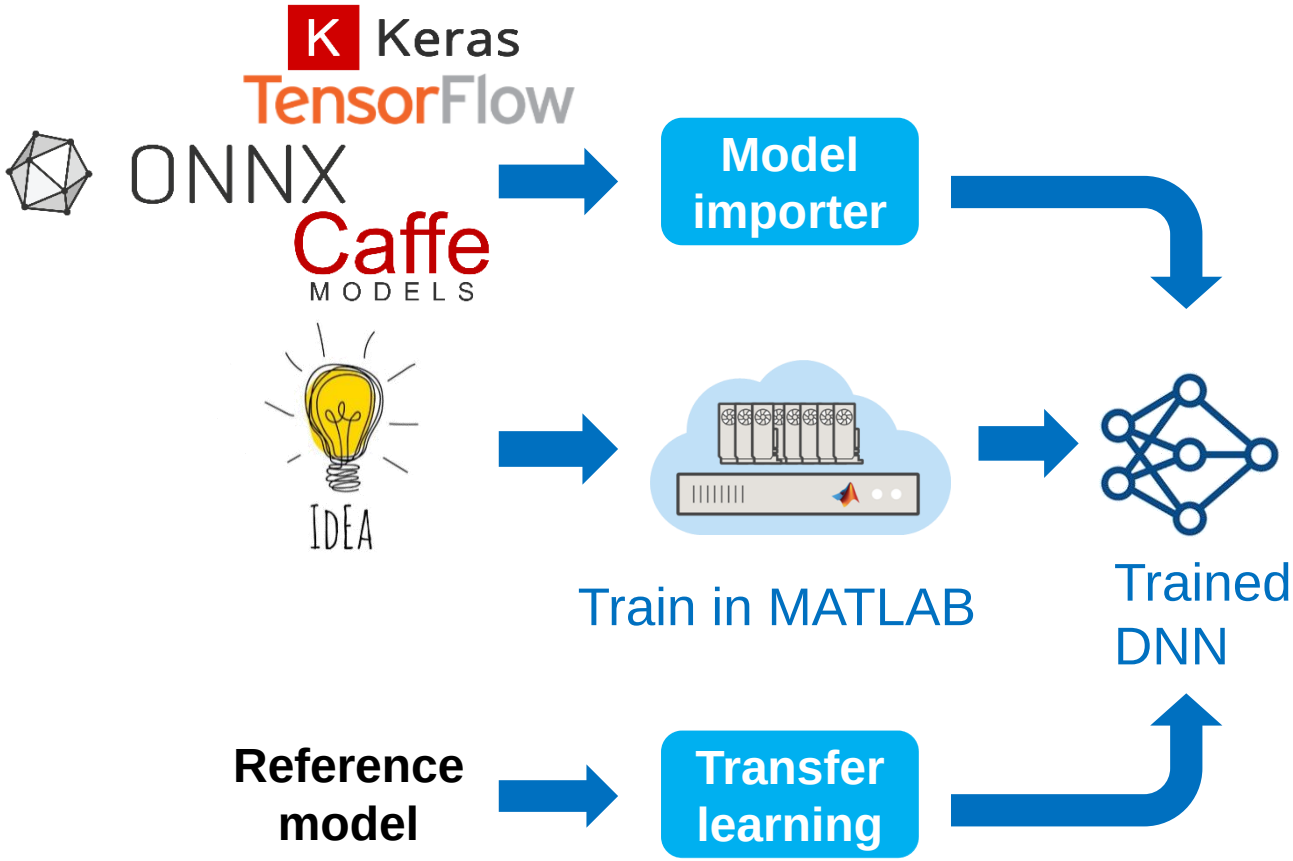


Deep Neural Network Design + Training

Application Design

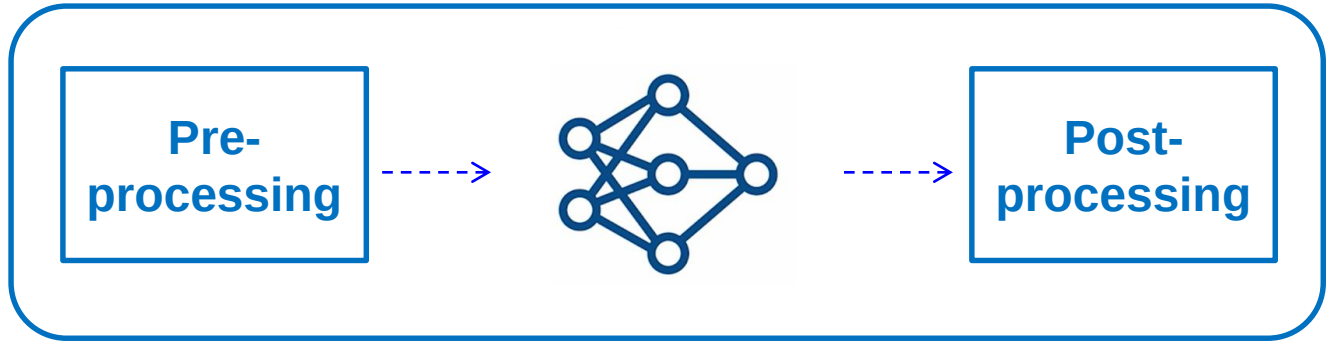
Standalone Deployment

Deep Neural Network Design and Training

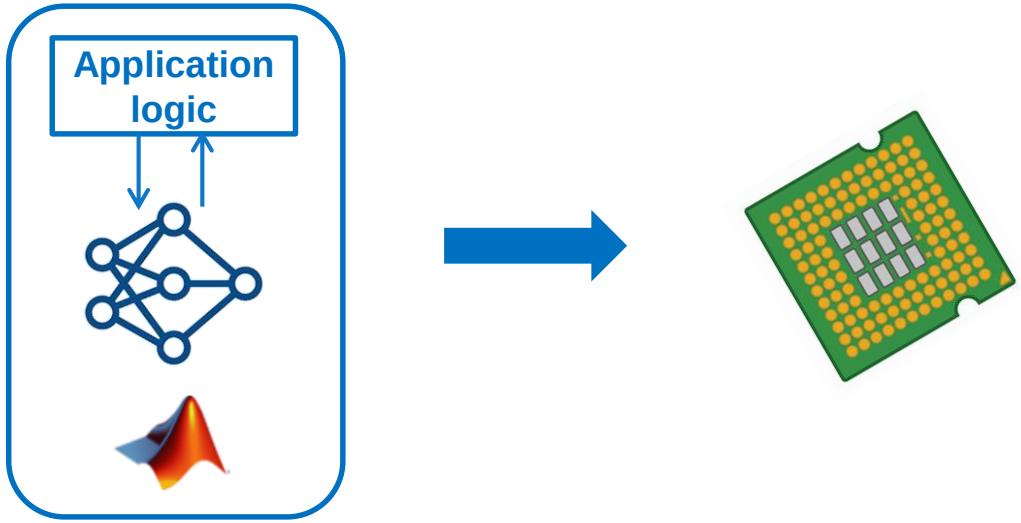


- **Design in MATLAB**
  - **Manage** large data sets
  - **Automate** data labeling
  - **Easy access** to models
- **Training in MATLAB**
  - **Acceleration** with GPU's
  - **Scale** to clusters

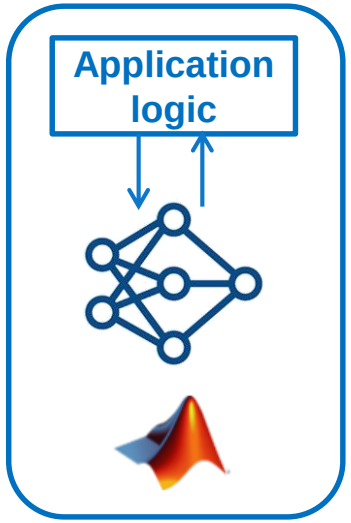
Application Design

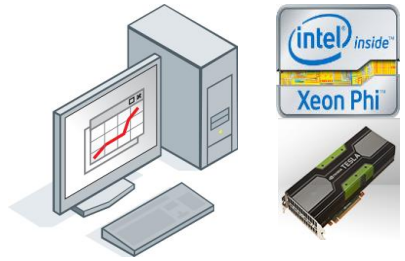


Multi-Platform Deep Learning Deployment



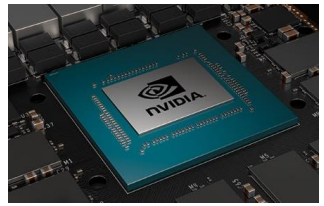
Multi-Platform Deep Learning Deployment



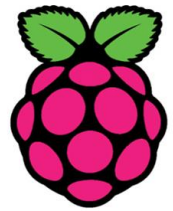


Desktop

Data Center

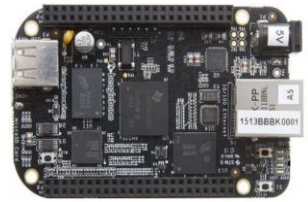


NVIDIA Jetson



Raspberry pi

Mobile

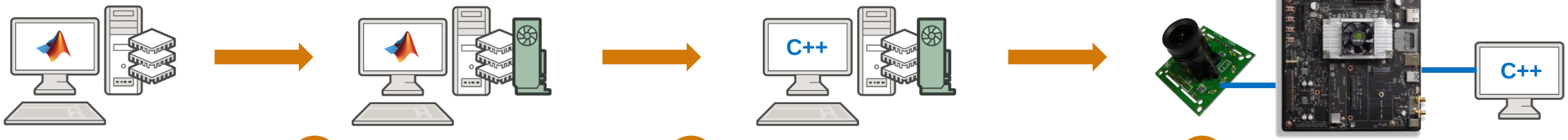


Beaglebone

. . .

Embedded

Algorithm Design to Embedded Deployment Workflow
Conventional Approach



1. Functional test

2. Deployment unit-test

   High-level language
   Deep learning framework
   Large, complex software stack

3. Deployment integration-test

   C/C++
   Low-level APIs
   Application-specific libraries

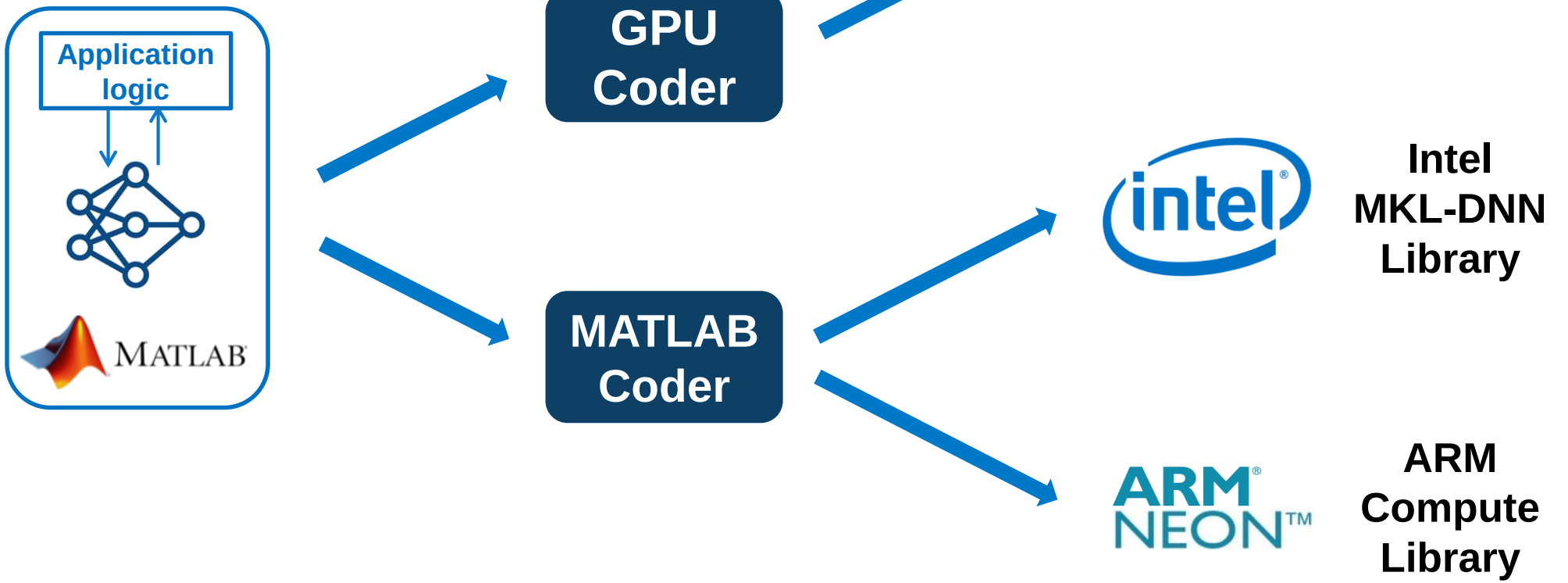
4. Real-time test

   C/C++
   Target-optimized libraries
   Optimize for memory & speed

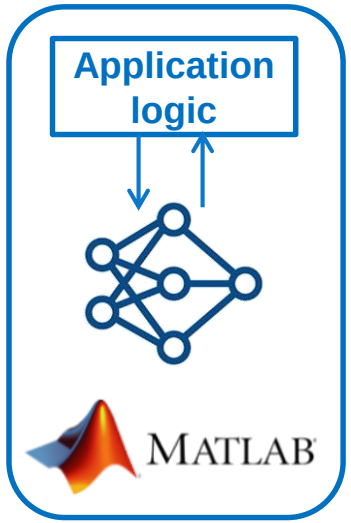
Challenges

- Integrating multiple libraries and packages
- Verifying and maintaining multiple implementations
- Algorithm & vendor lock-in

Solution: Use MATLAB Coder & GPU Coder for Deep Learning Deployment



Solution: Use MATLAB Coder & GPU Coder for Deep Learning Deployment



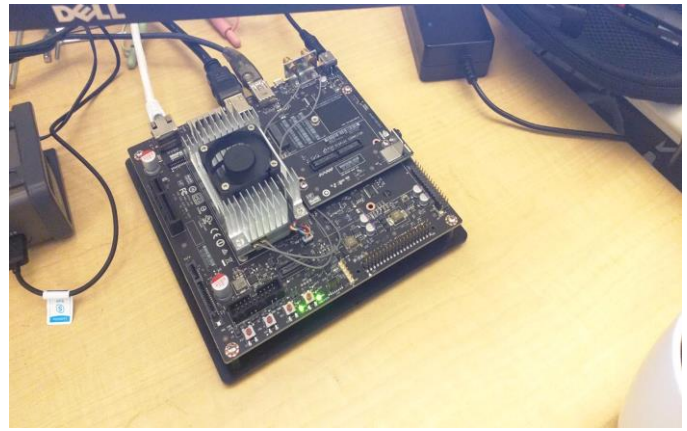
GPU Coder

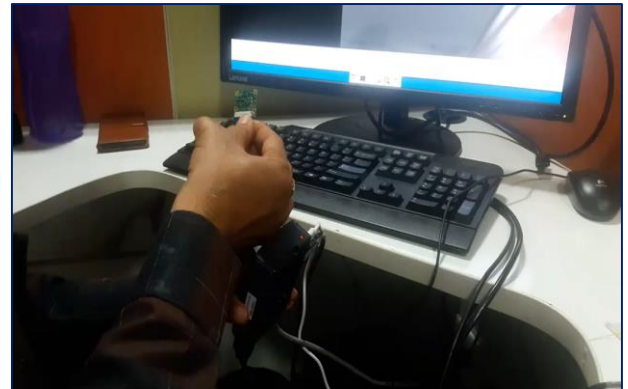
MATLAB Coder

NVIDIA

intel

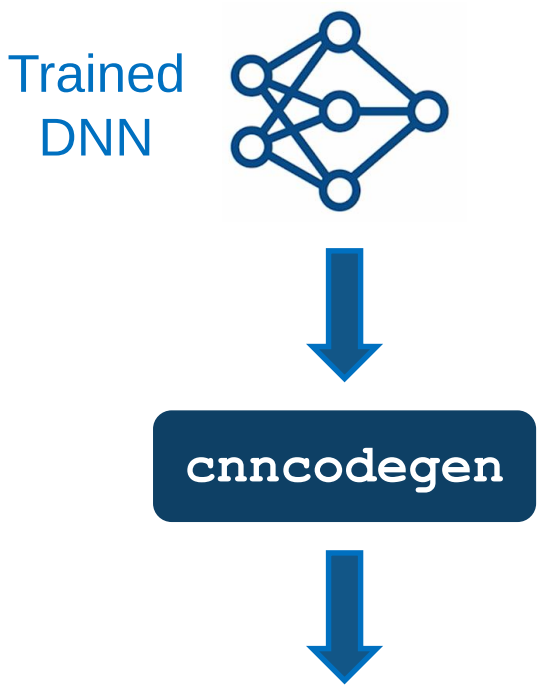
ARM NEON™



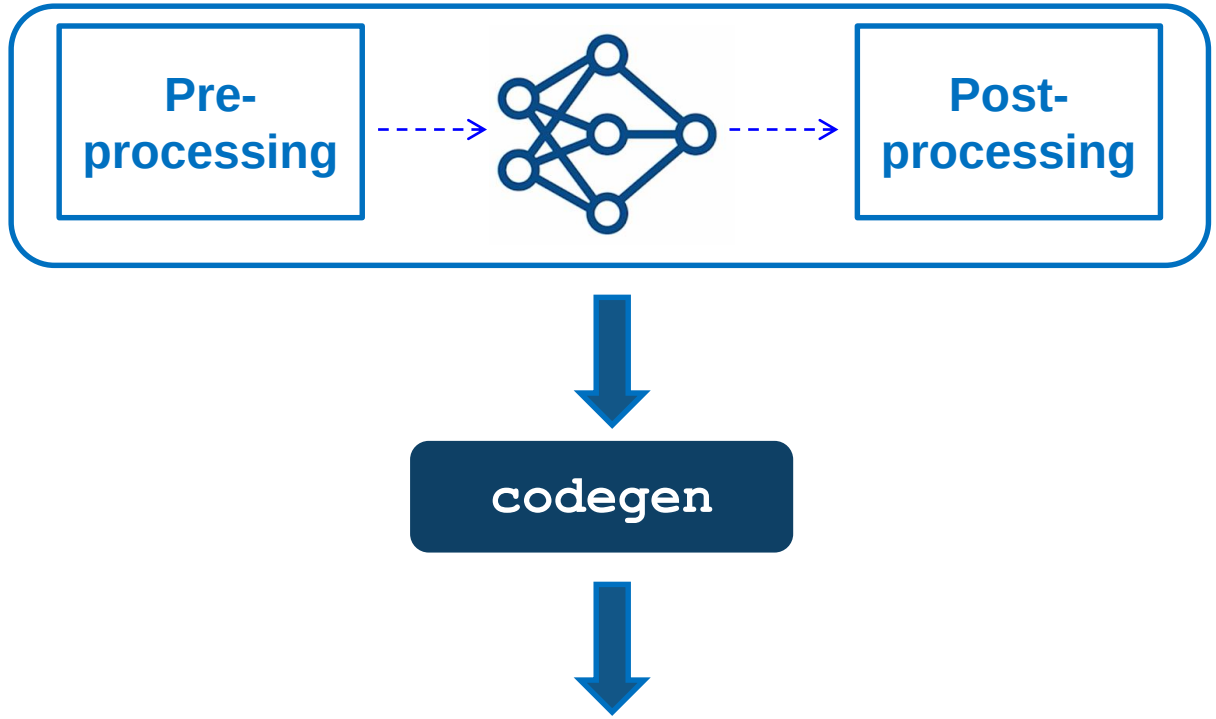


Deep Learning Deployment Workflows

INFERENCE ENGINE DEPLOYMENT

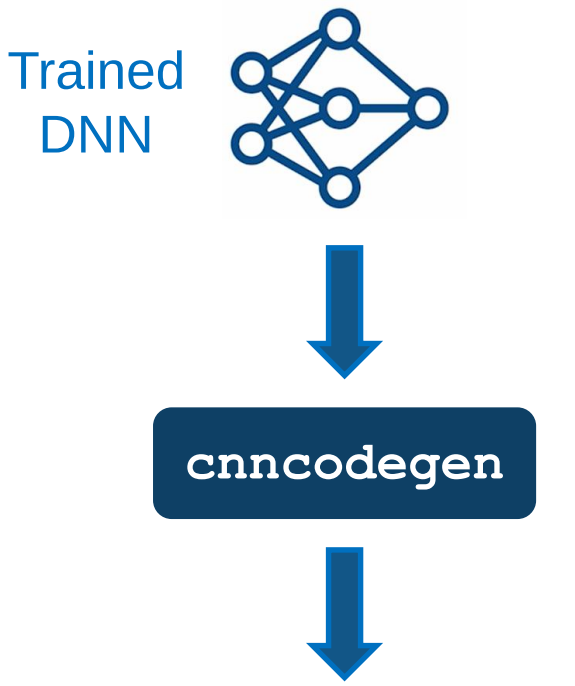


INTEGRATED APPLICATION DEPLOYMENT



Workflow for Inference Engine Deployment

INFERENCE ENGINE DEPLOYMENT



Steps for inference engine deployment

1. Generate the code for trained model

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

2. Copy the generated code onto target board

3. Build the code for the inference engine

```
>> make -C ./codegen -f …mk
```

4. Use hand written main function to call inference engine

5. Generate the exe and test the executable

```
>> make -C ./ ……
```

Deep Learning Inference Deployment

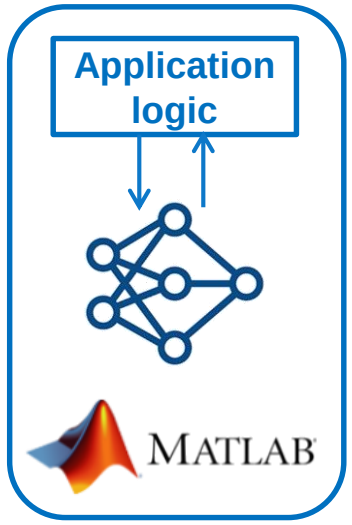
Target Libraries

Pedestrian Detection

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library

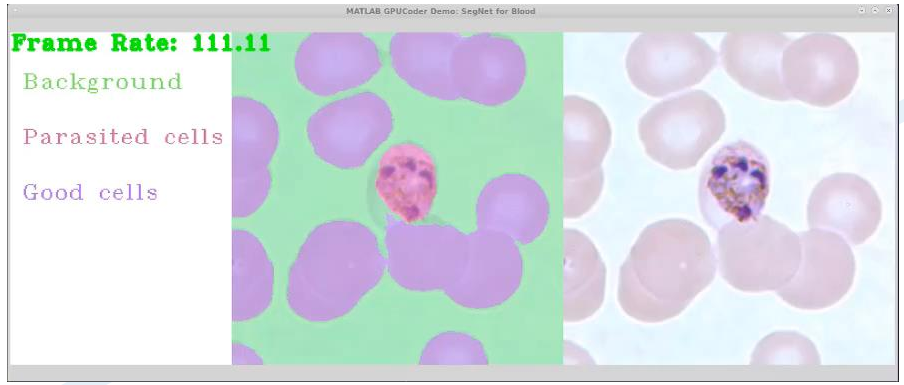


MATLAB Coder

Deep Learning Inference Deployment

Target Libraries

Blood Smear Segmentation



Application logic

MATLAB

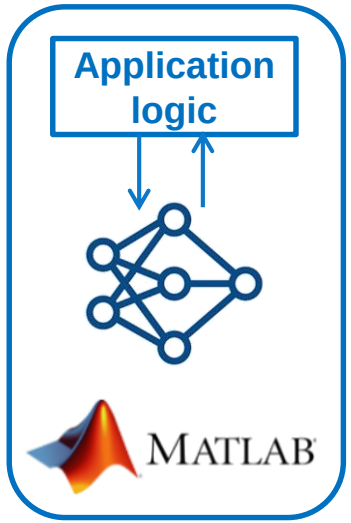
MATLAB Coder

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library

Deep Learning Inference Deployment



GPU Coder

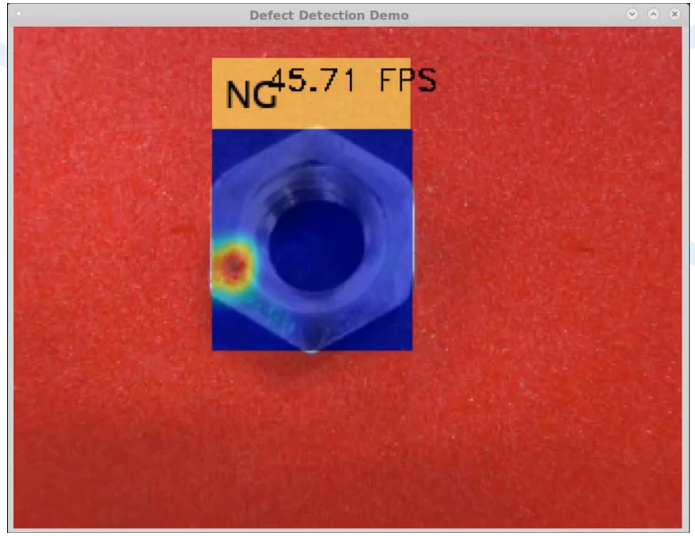
Target Libraries

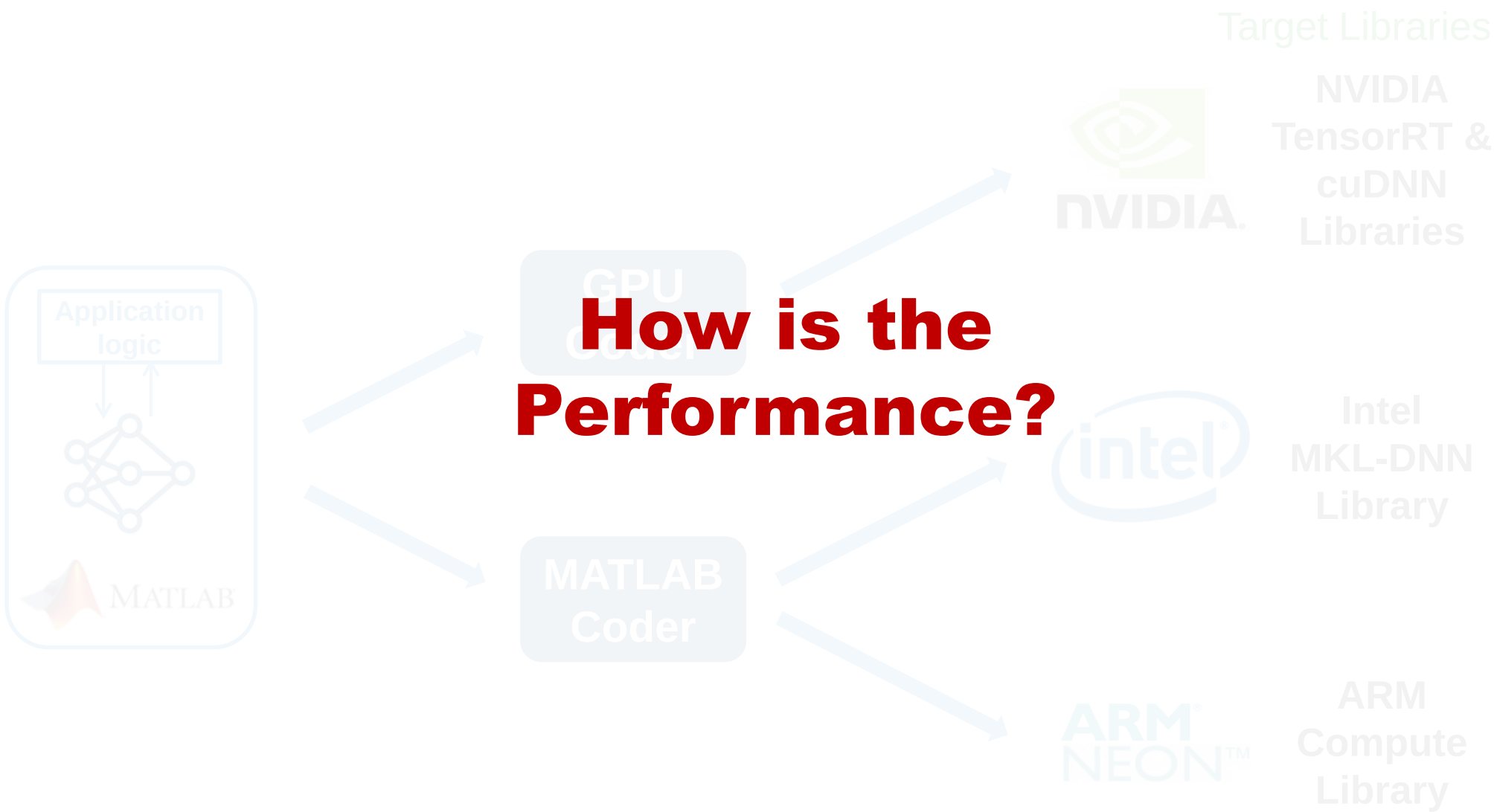
NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library

Defect Classification & Detection



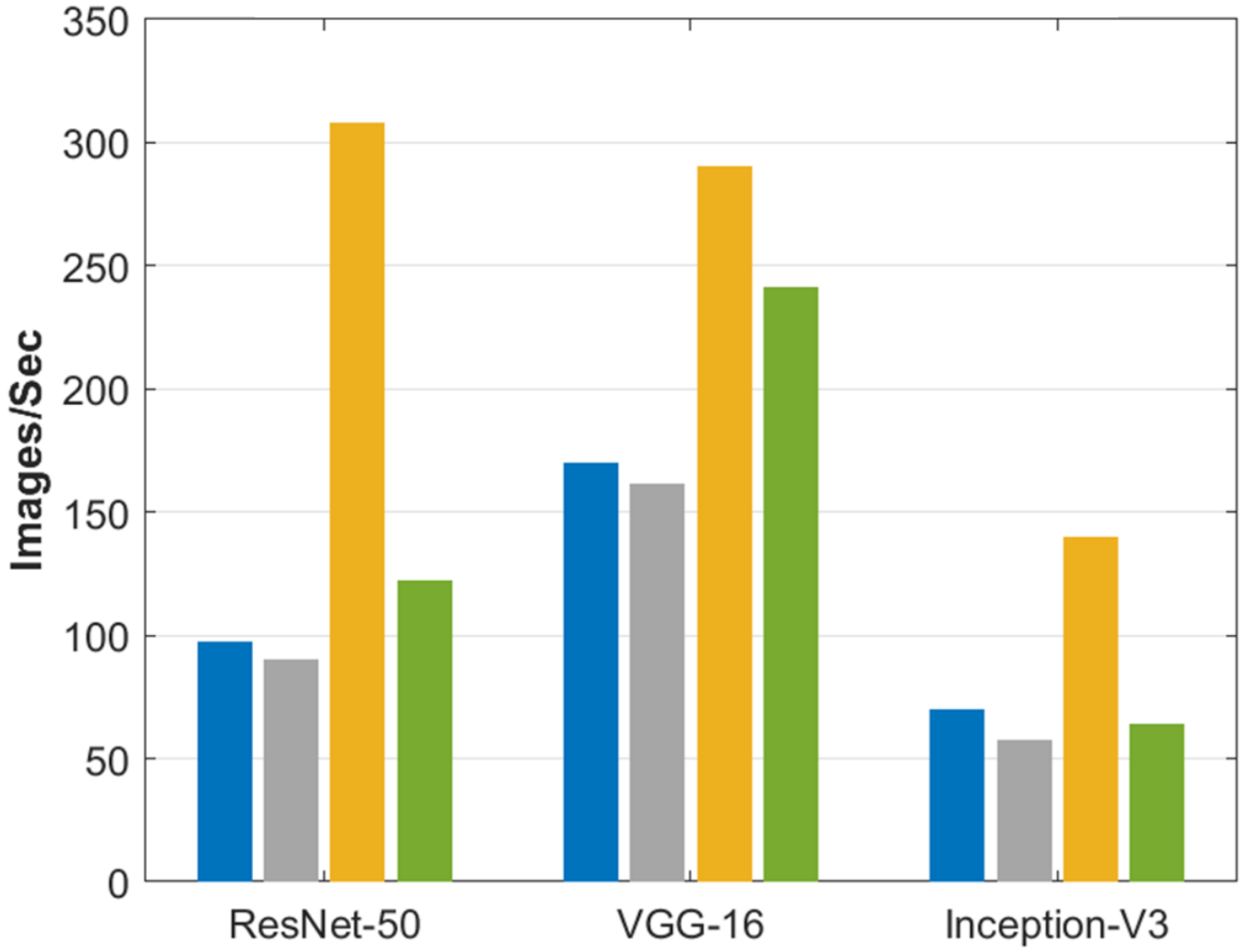


How is the Performance?

Performance of Generated Code

- CNN inference (ResNet-50, VGG-16, Inception V3) on Titan V GPU
- CNN inference (ResNet-50) on Jetson TX2
- CNN inference (ResNet-50 , VGG-16, Inception V3) on Intel Xeon CPU

Single Image Inference on Titan V using cuDNN



TensorFlow (1.13.0)

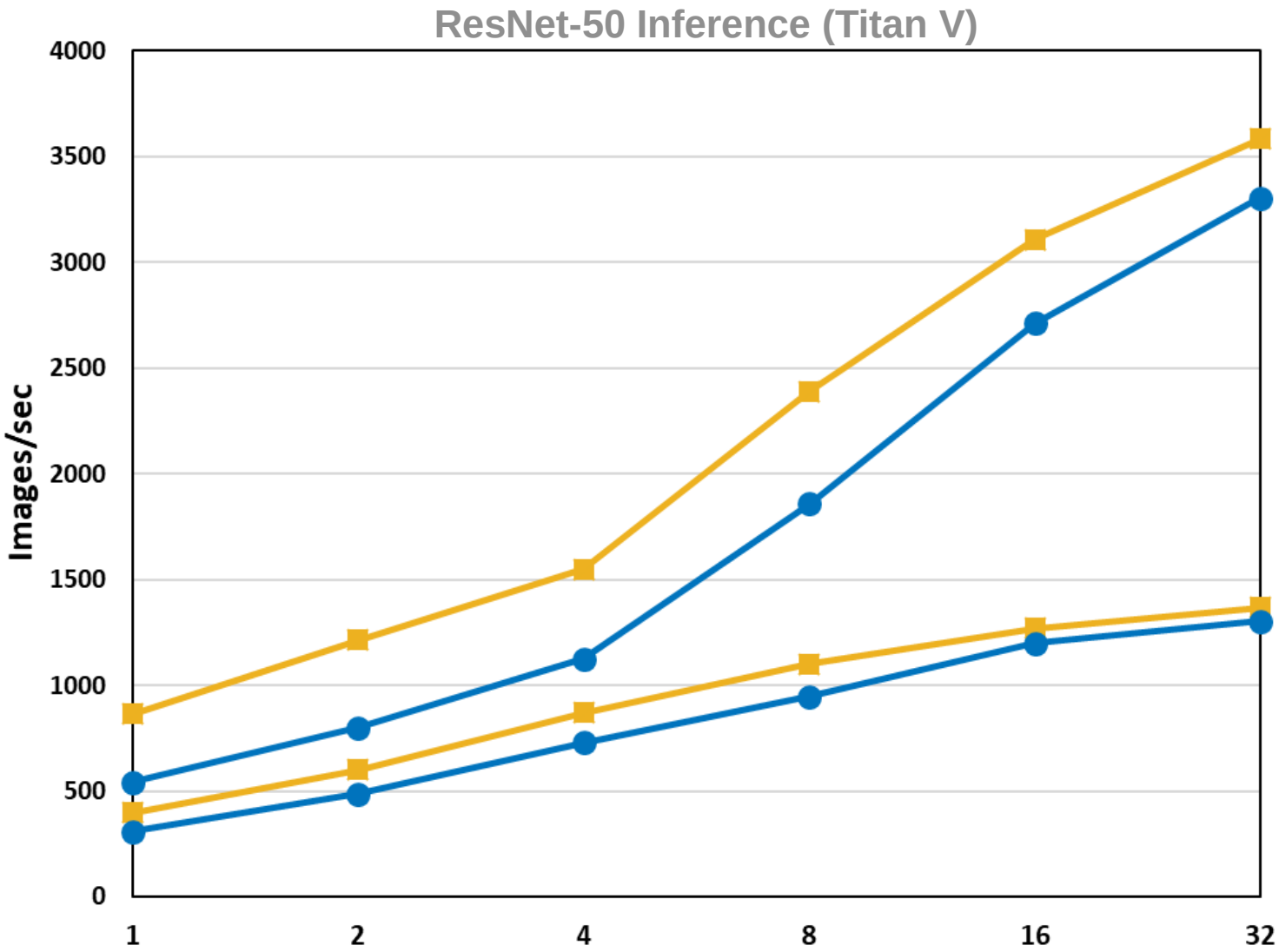
MXNet (1.4.0)

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

Even Stronger Performance with INT8 using TensorRT



GPU Coder + TensorRT (INT8)

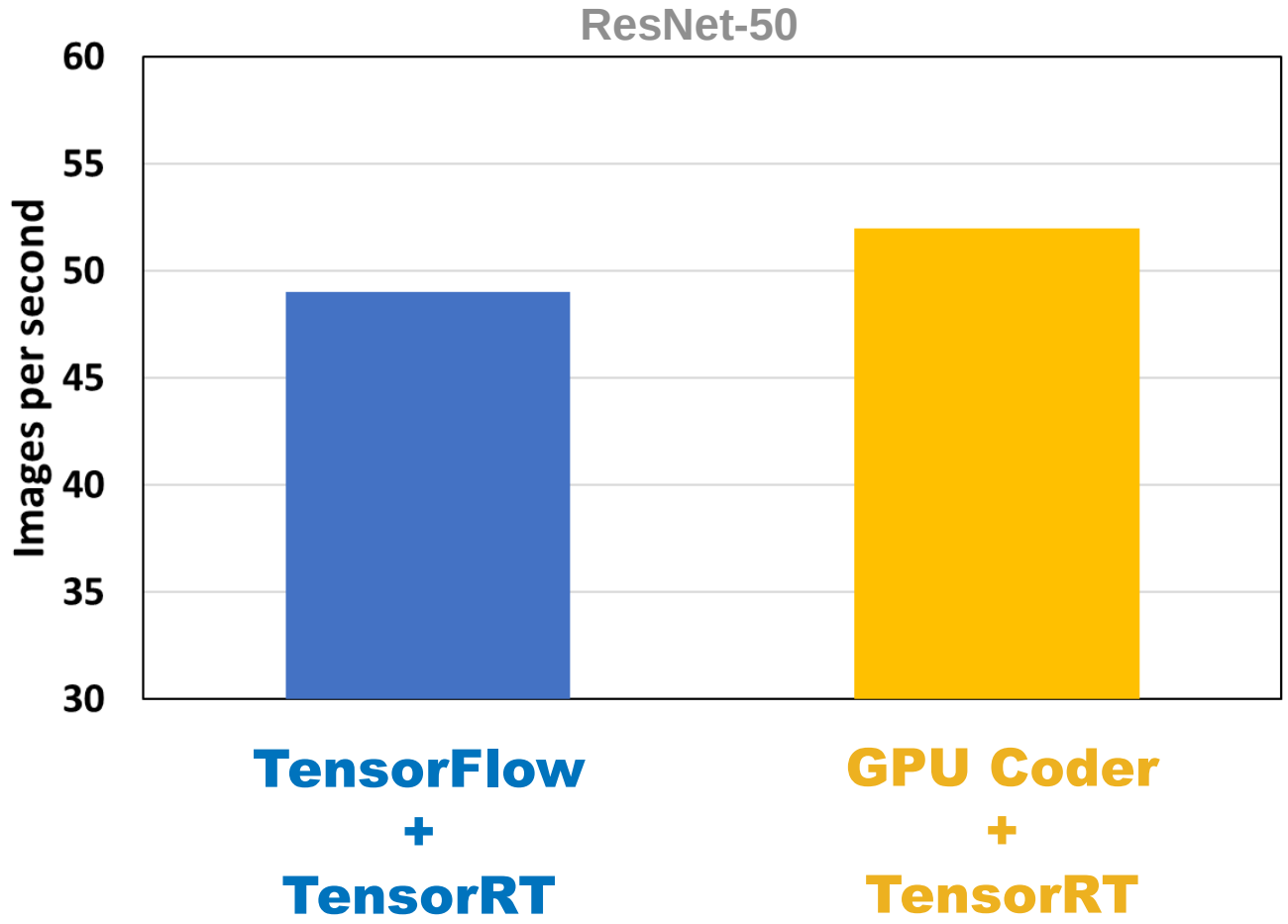
TensorFlow + TensorRT (INT8)

GPU Coder + TensorRT (FP32)

TensorFlow + TensorRT (FP32)

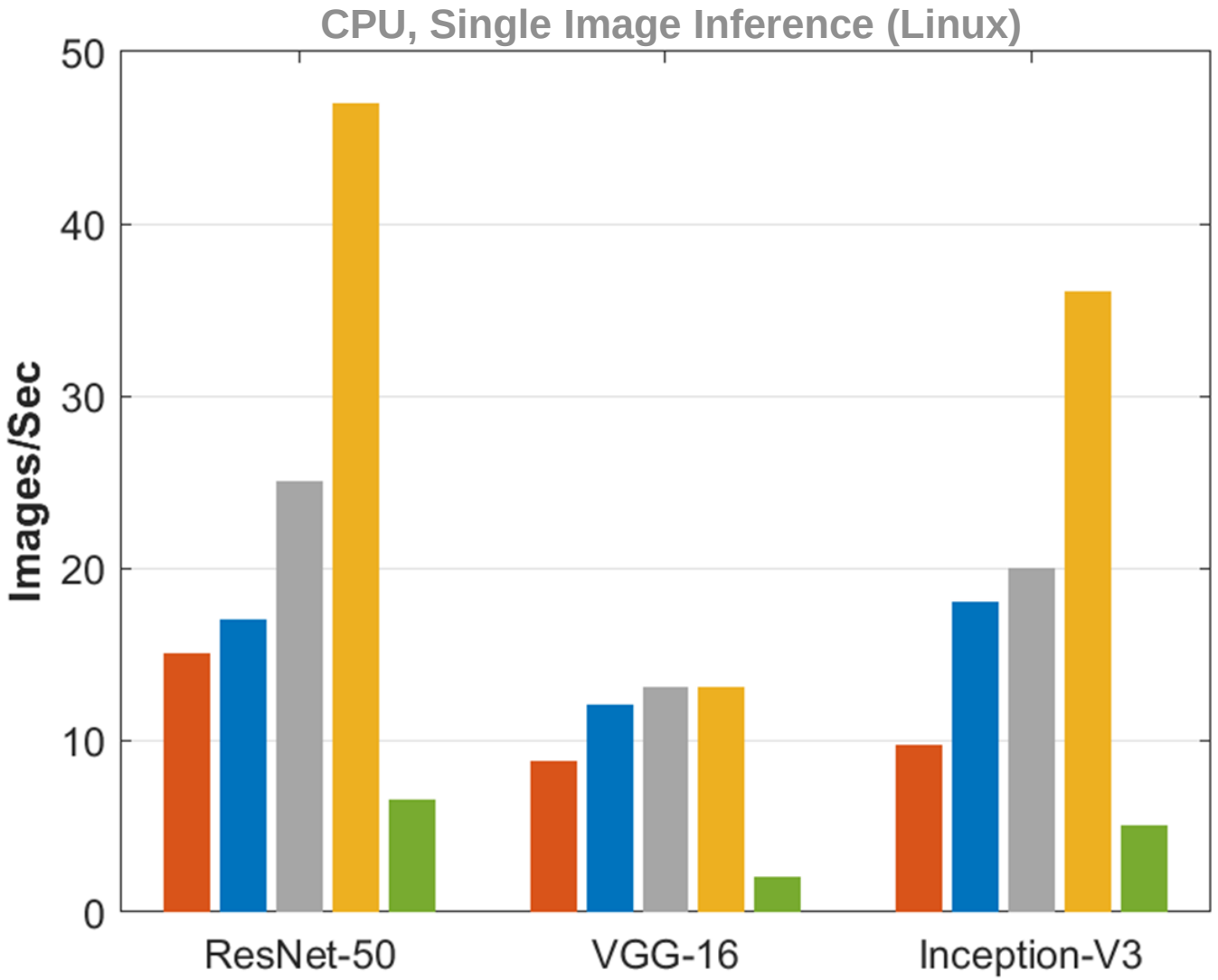
Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7.4.1 – TensorRT 5.0.2.6 - Frameworks: TensorFlow 1.13.0_rc0

Single Image Inference on Jetson TX2



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

CPU Performance



MATLAB

TensorFlow

MXNet

MATLAB Coder

PyTorch

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

Brief Summary

DNN libraries are great for inference, …

MATLAB Coder and GPU Coder generates code that takes advantage of:

NVIDIA® CUDA libraries, including TensorRT & cuDNN

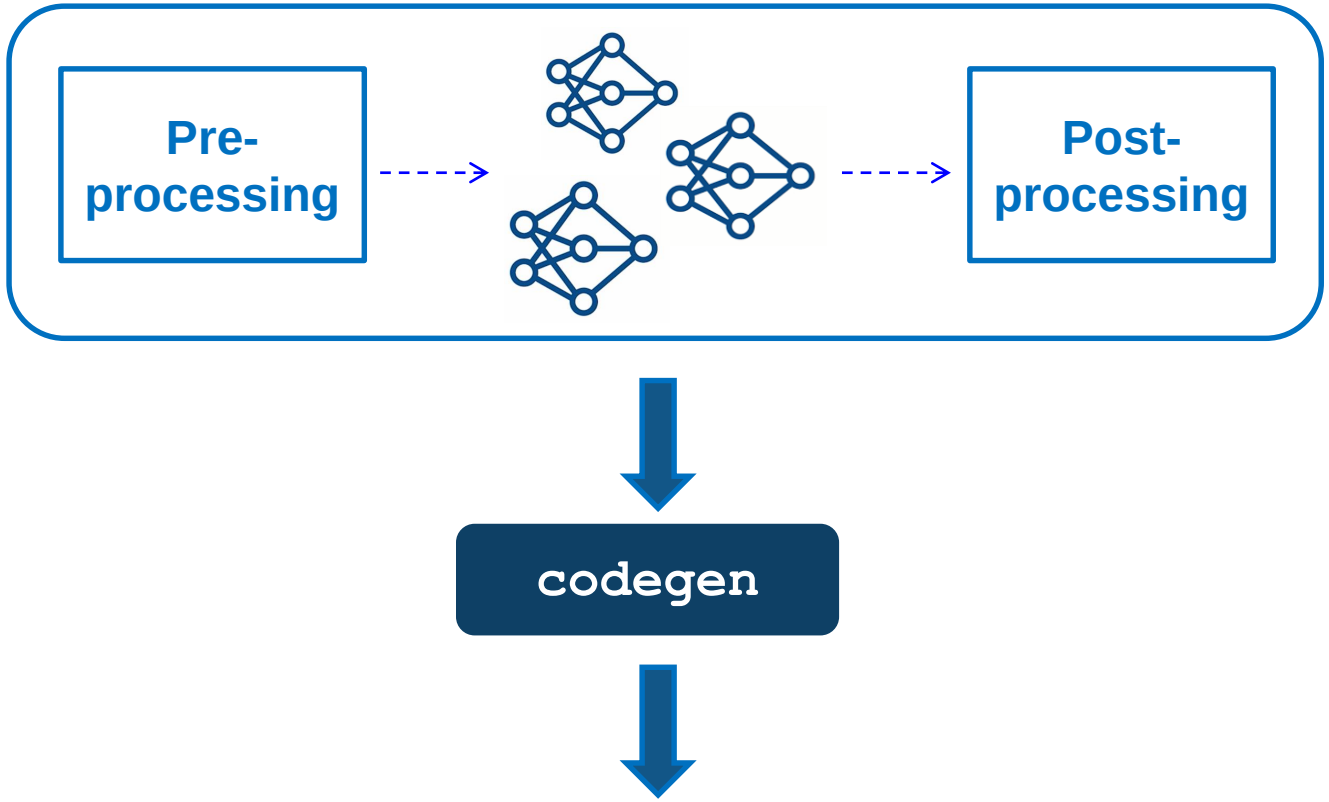
Intel® Math Kernel Library for Deep Neural Networks (MKL-DNN)

ARM® Compute libraries for mobile platforms

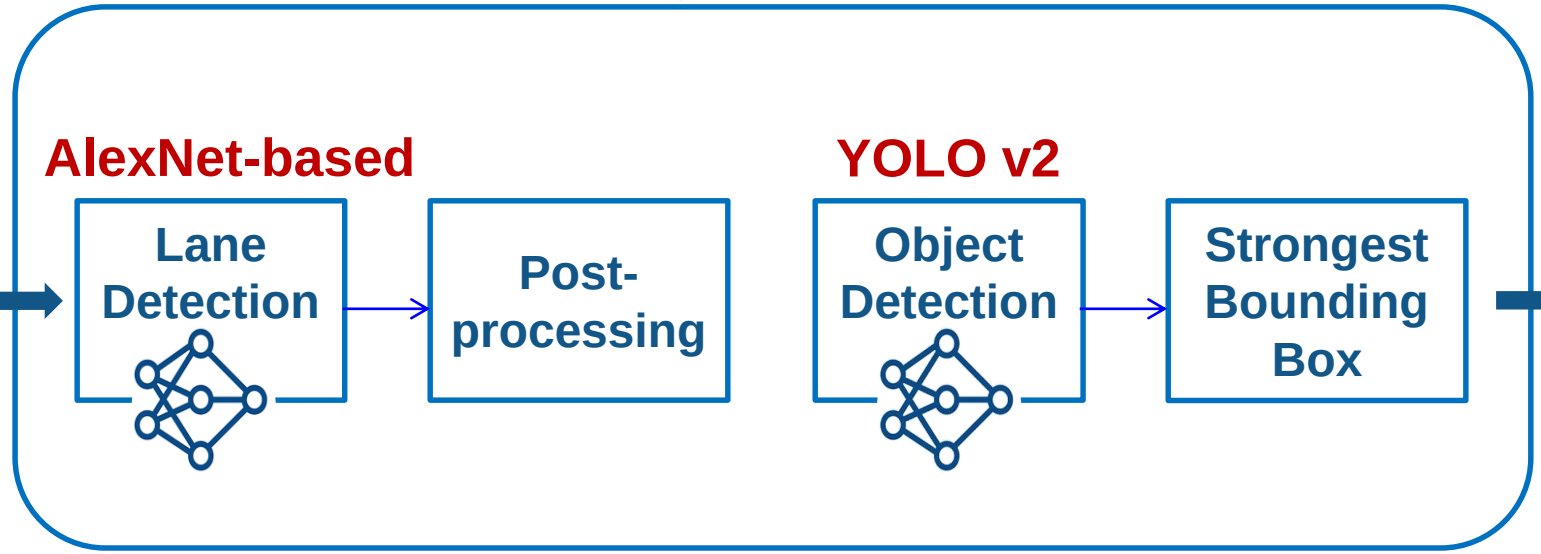
Brief Summary

But, Applications Require More than just Inference

Deep Learning Workflows: Integrated Application Deployment



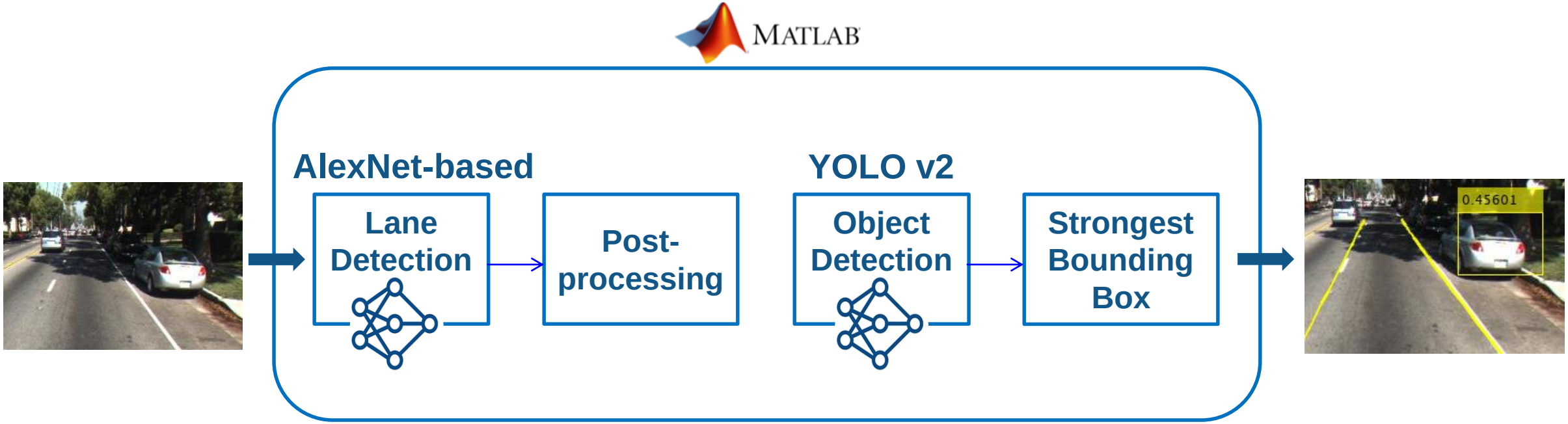
Lane and Object Detection using YOLO v2



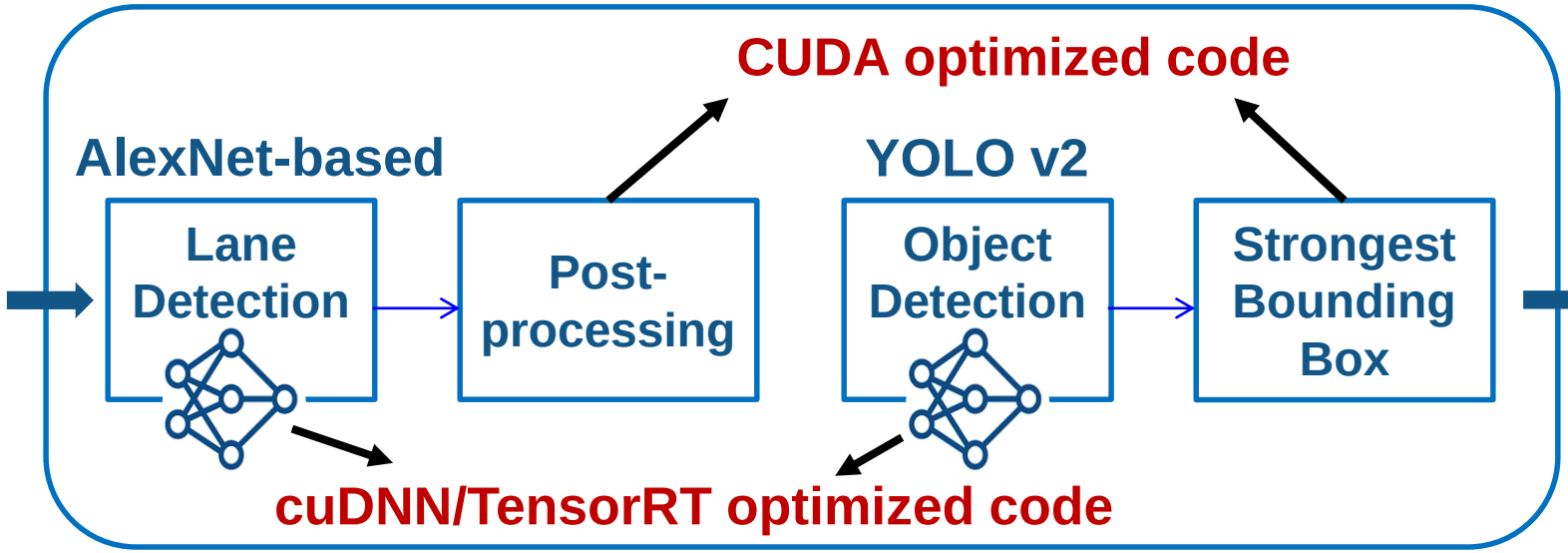
Workflow:

1) Test in MATLAB on CPU
2) Generate code and test on desktop GPU
3) Generate code and test on Jetson AGX Xavier GPU

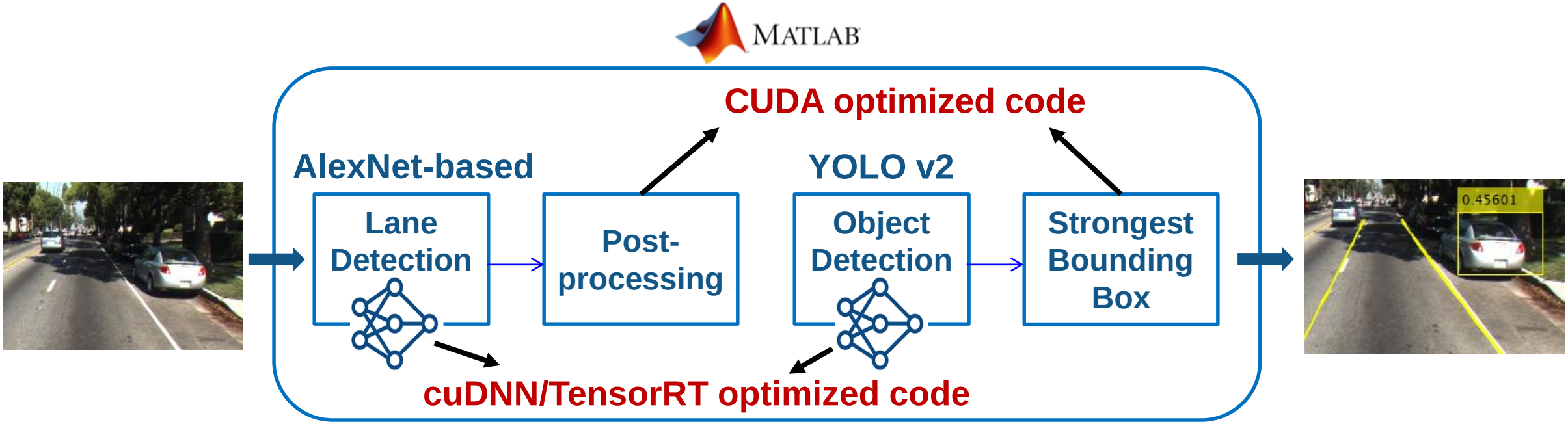
(1) Test in MATLAB on CPU



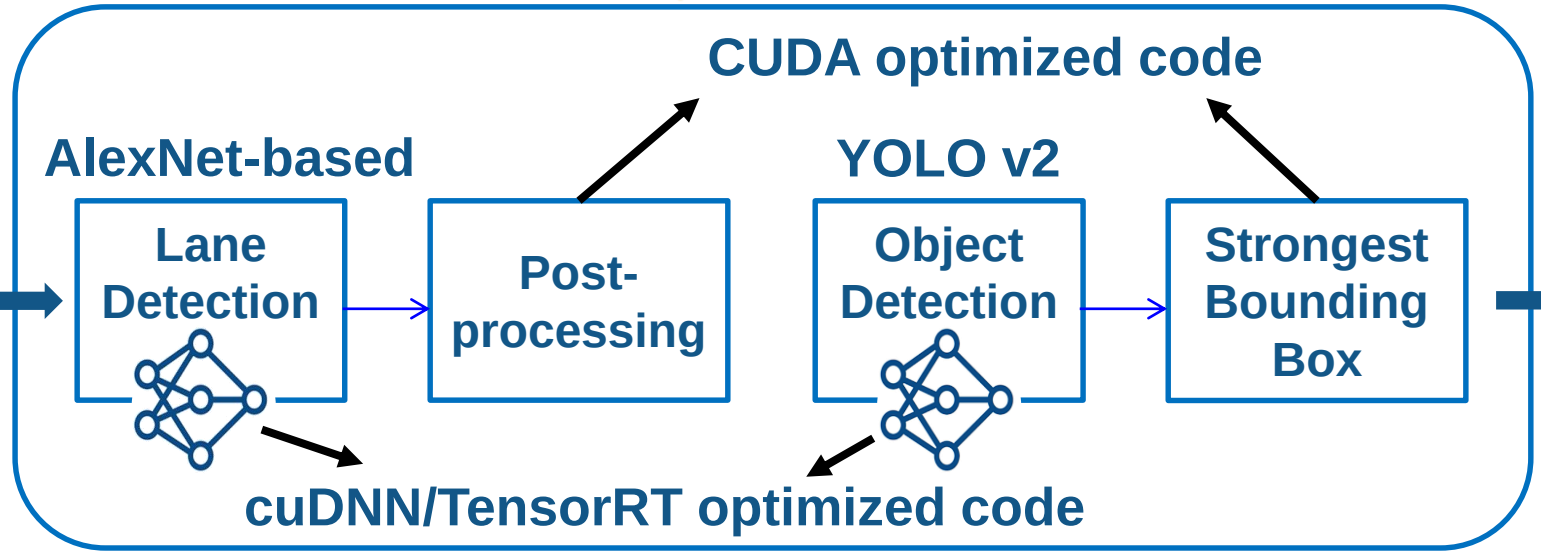
(2) Generate Code and Test on Desktop GPU

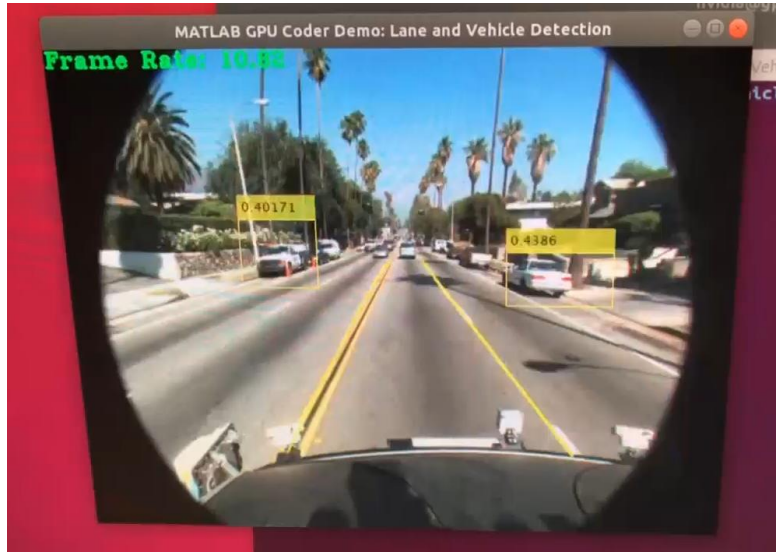


(3) Generate Code and Test on Jetson AGX Xavier GPU



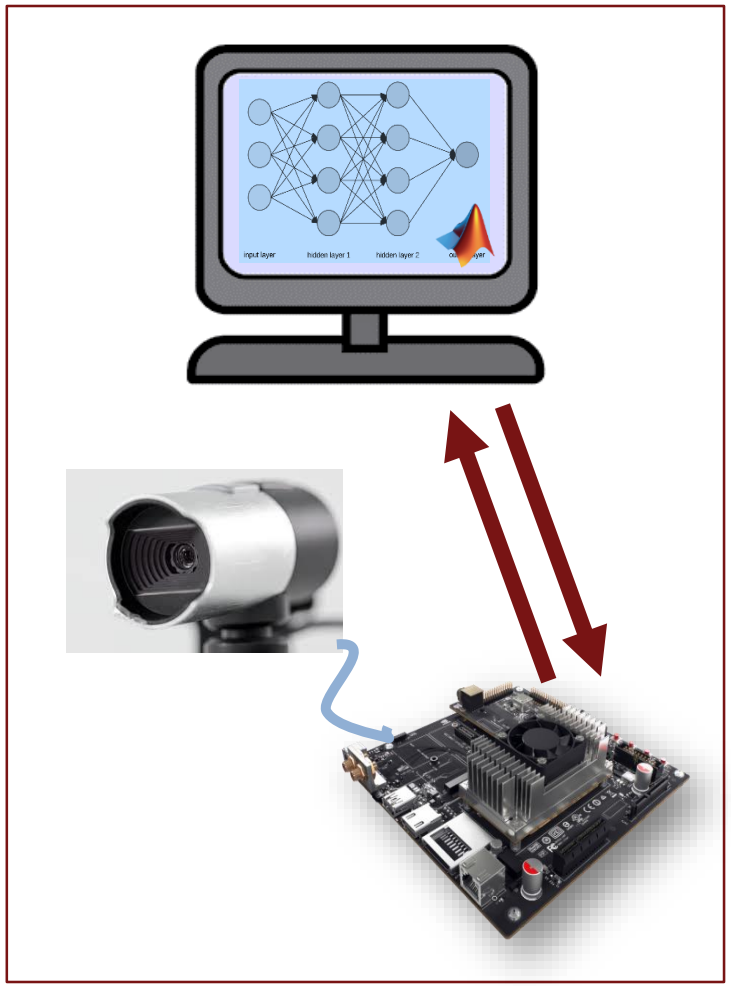
Lane and Object Detection using YOLO v2



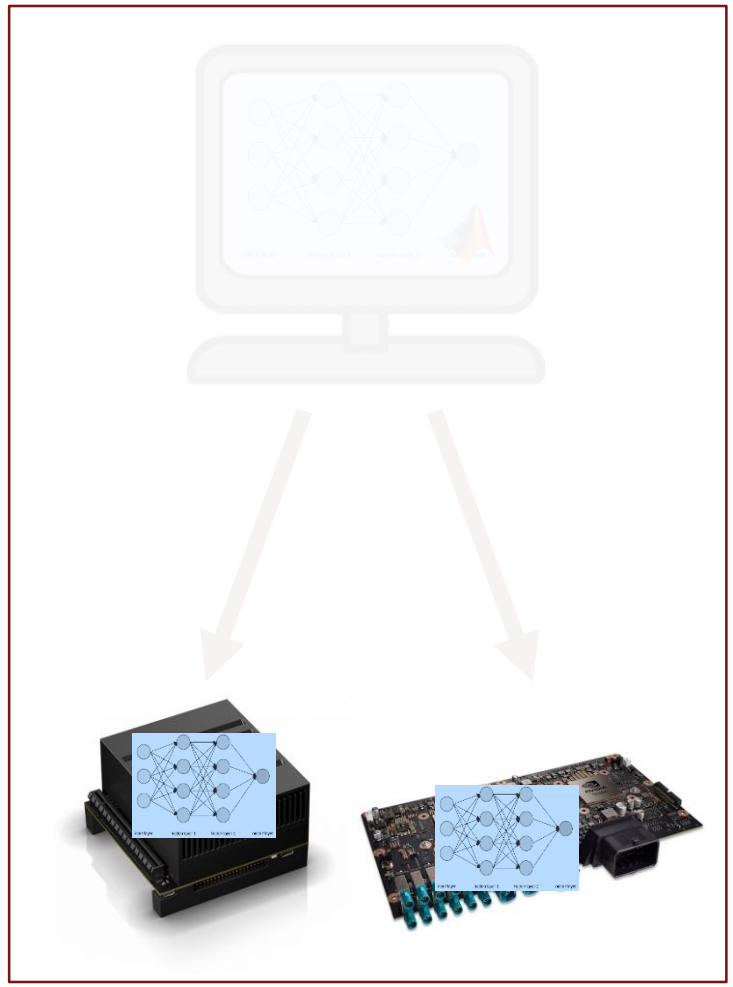


1) Running on CPU
2) 7X faster running generate code on desktop GPU
3) Generate code and test on Jetson AGX Xavier GPU

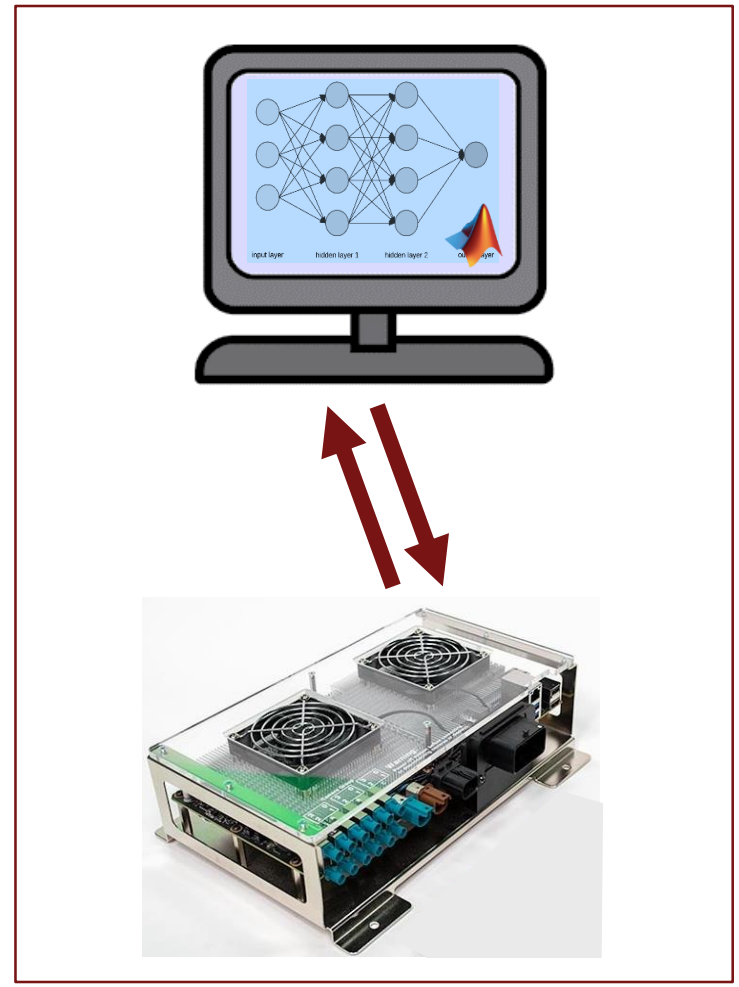
Accessing Hardware



Access Peripheral from MATLAB

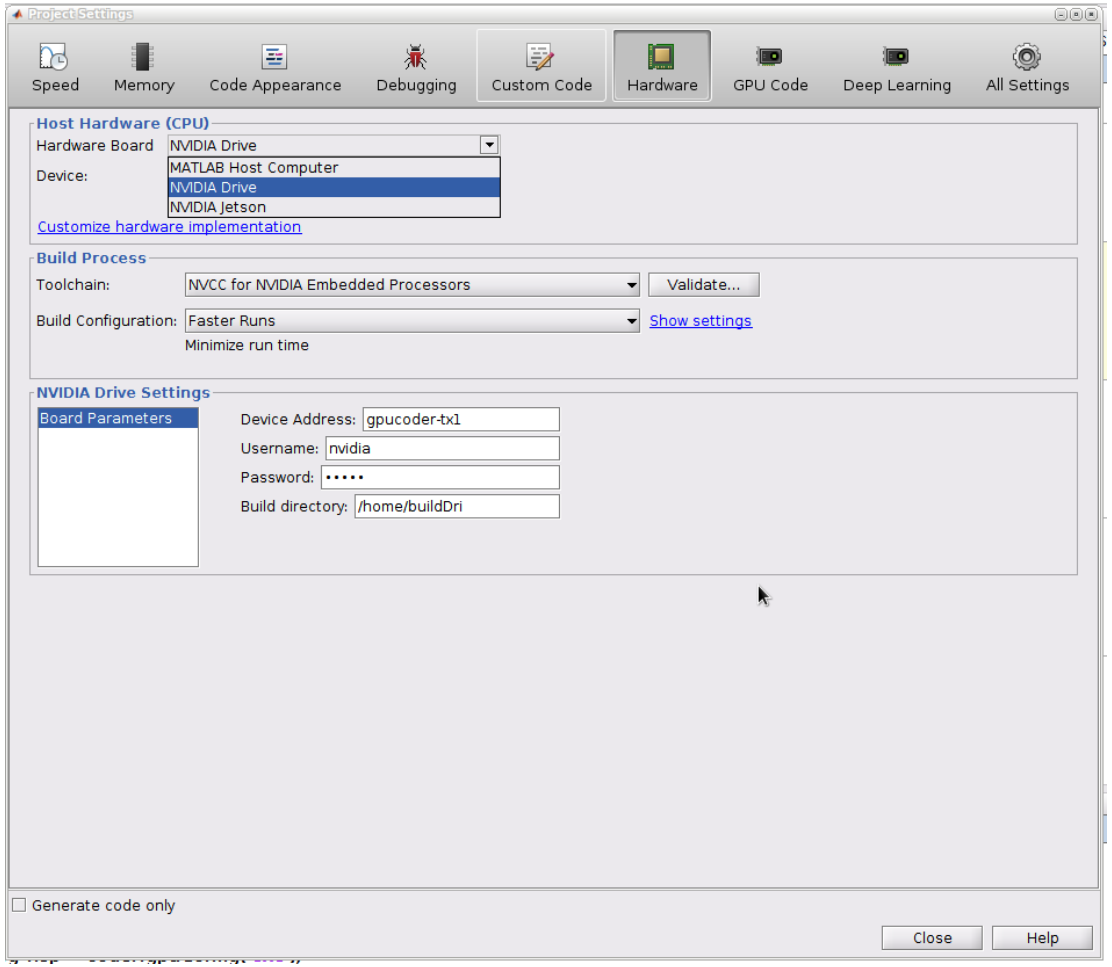


Deploy Standalone Application

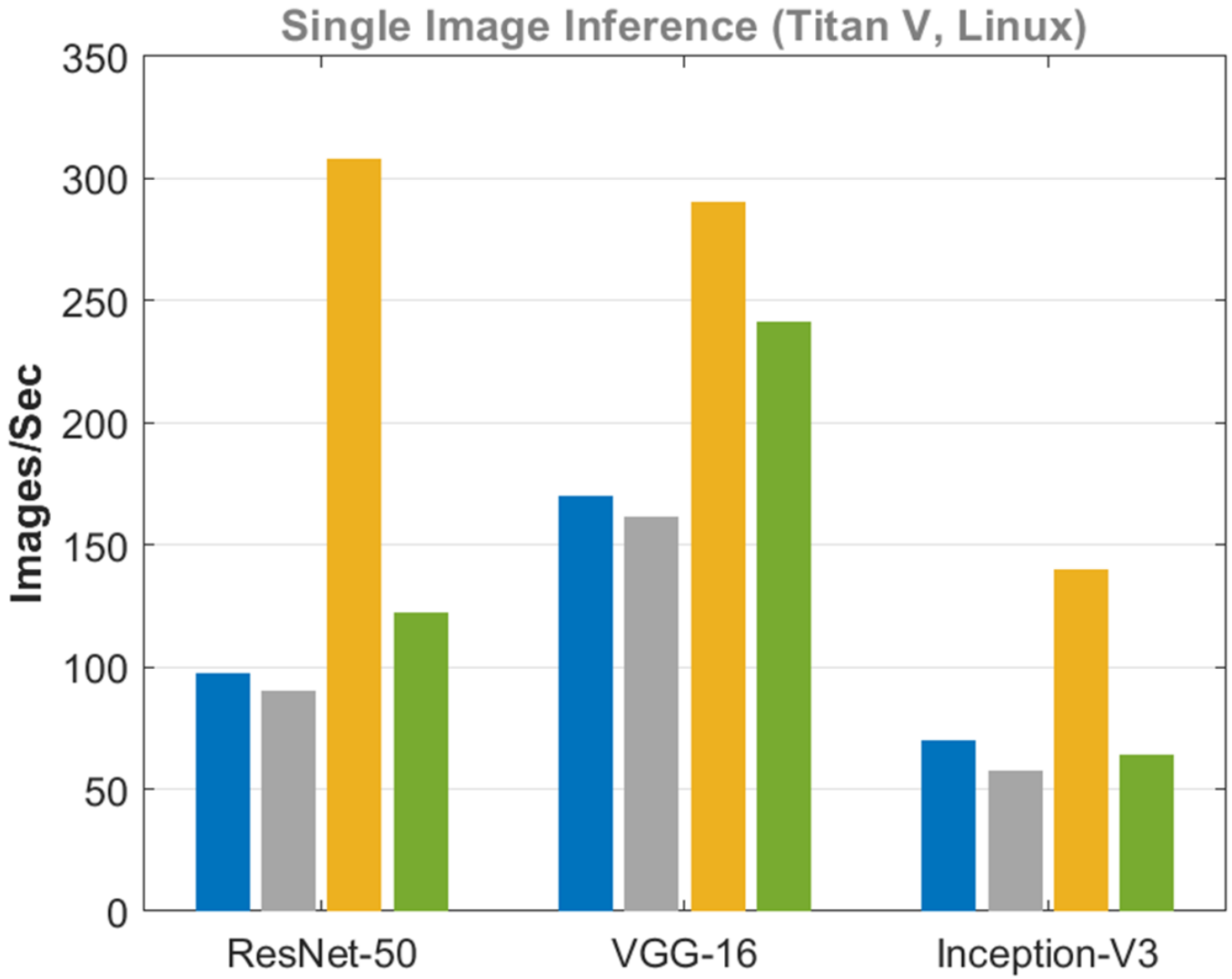


Processor-in-Loop Verification

Deploy to Target Hardware via Apps and Command Line




```
%% 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']);
```

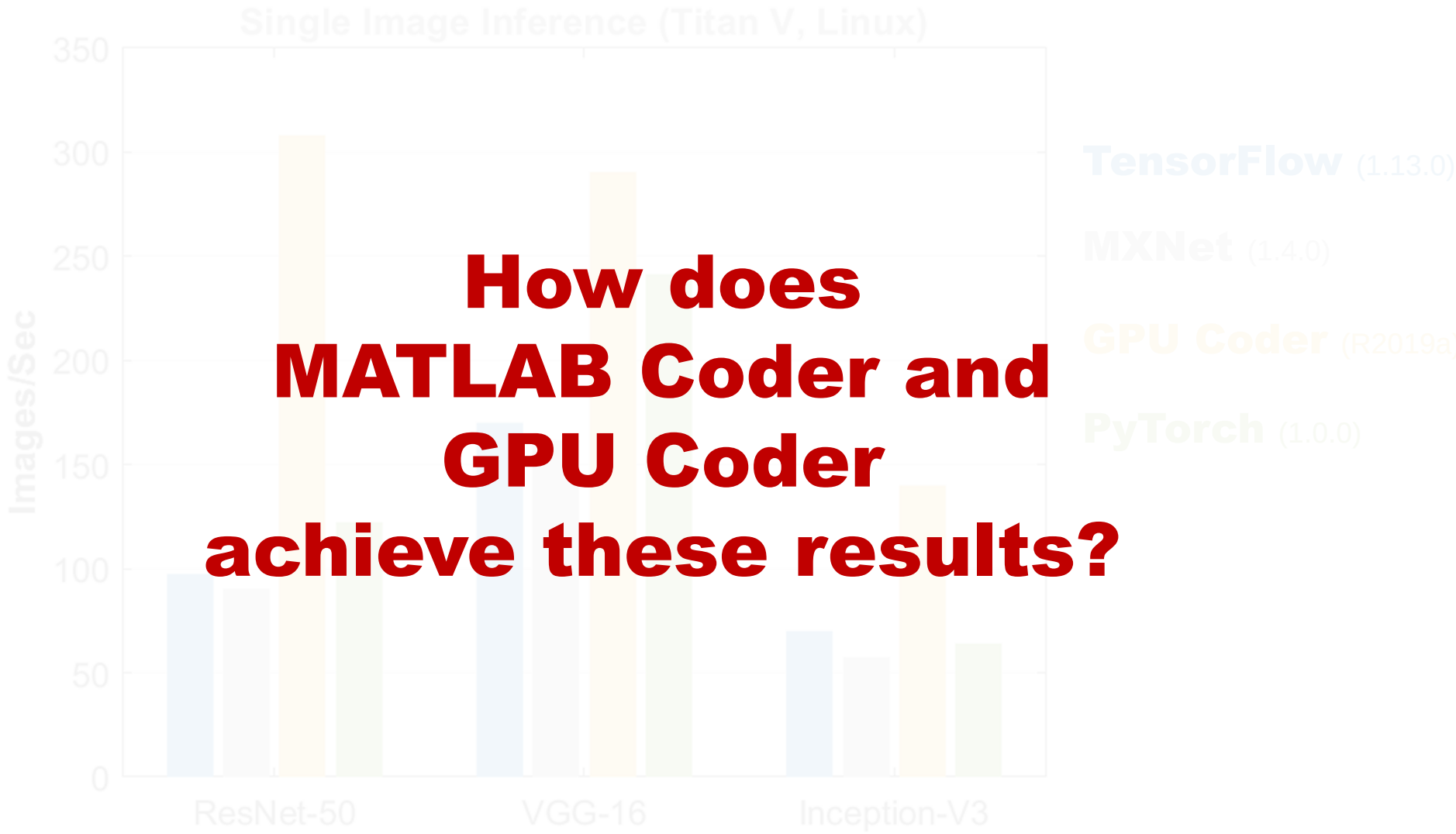


TensorFlow (1.13.0)

MXNet (1.4.0)

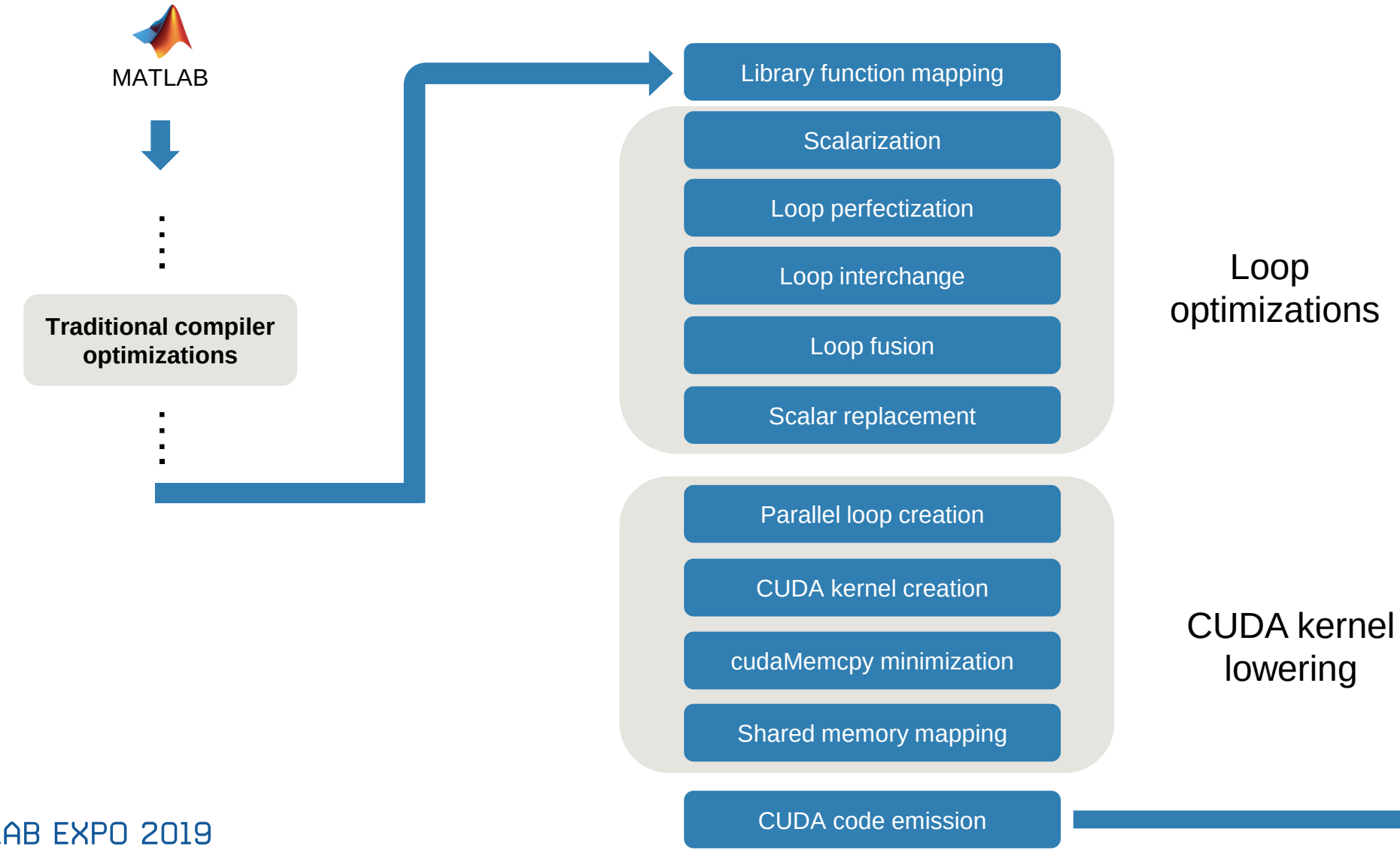
GPU Coder (R2019a)

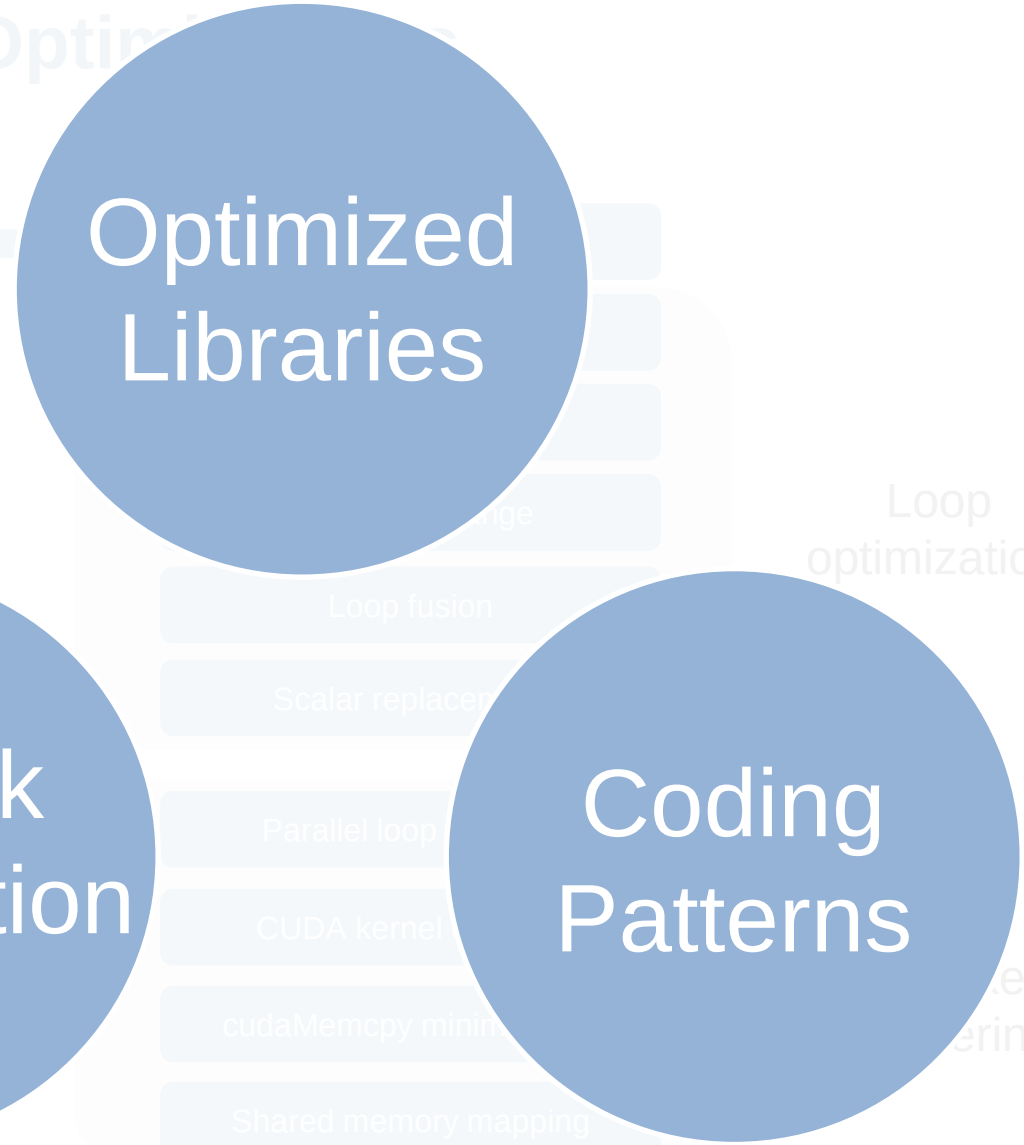
PyTorch (1.0.0)



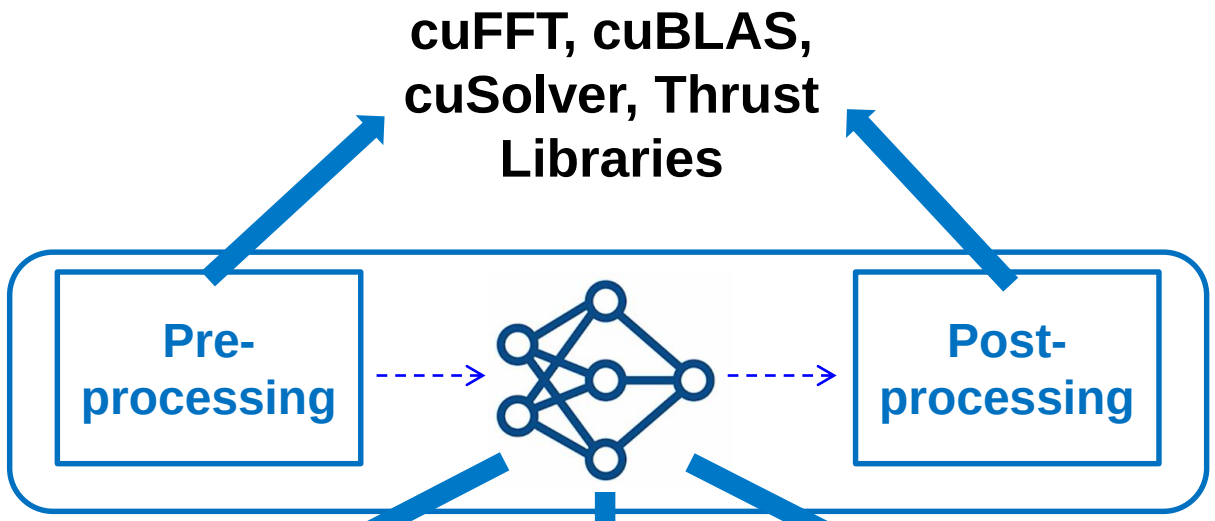
How does MATLAB Coder and GPU Coder achieve these results?

Coders Apply Various Optimizations




Optimized Libraries
Network Optimization
Coding Patterns

Generated Code Calls Optimized Libraries



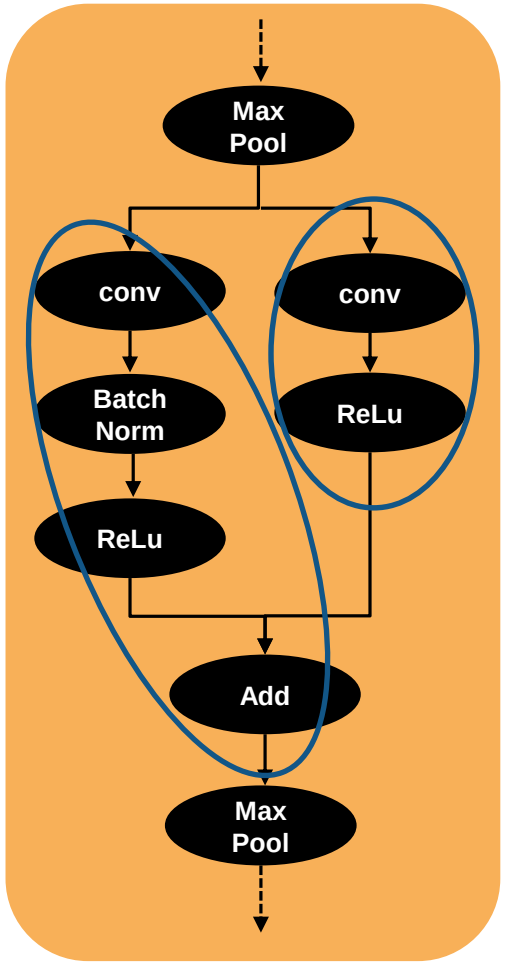
Intel MKL-DNN Library

NVIDIA TensorRT & cuDNN Libraries

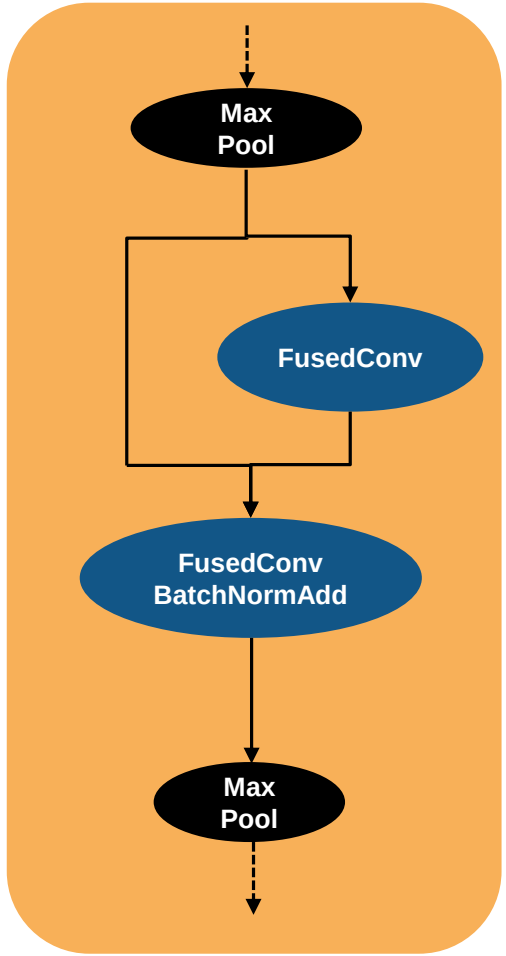
ARM Compute Library

Deep Learning Network Optimization

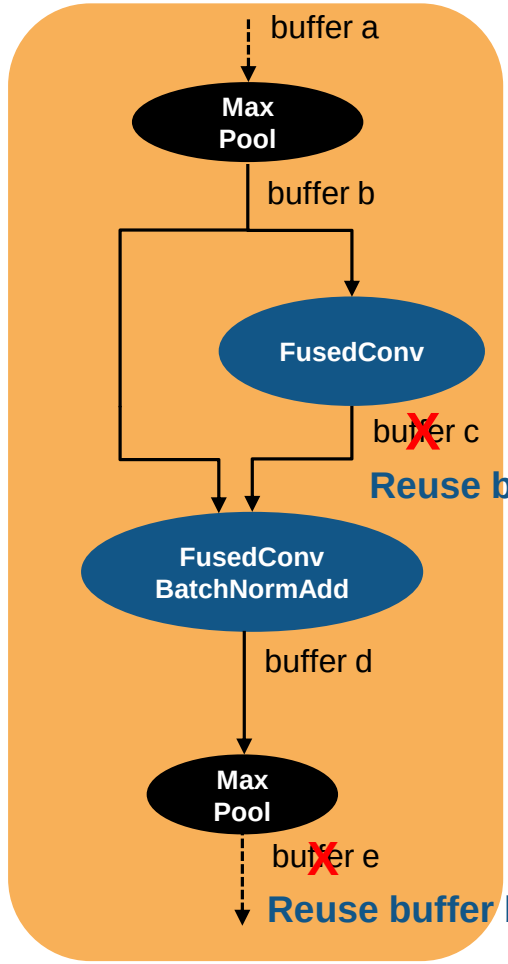
Performance
1. Optimized Libraries
2. **Network Optimizations**
3. Coding Patterns



Network



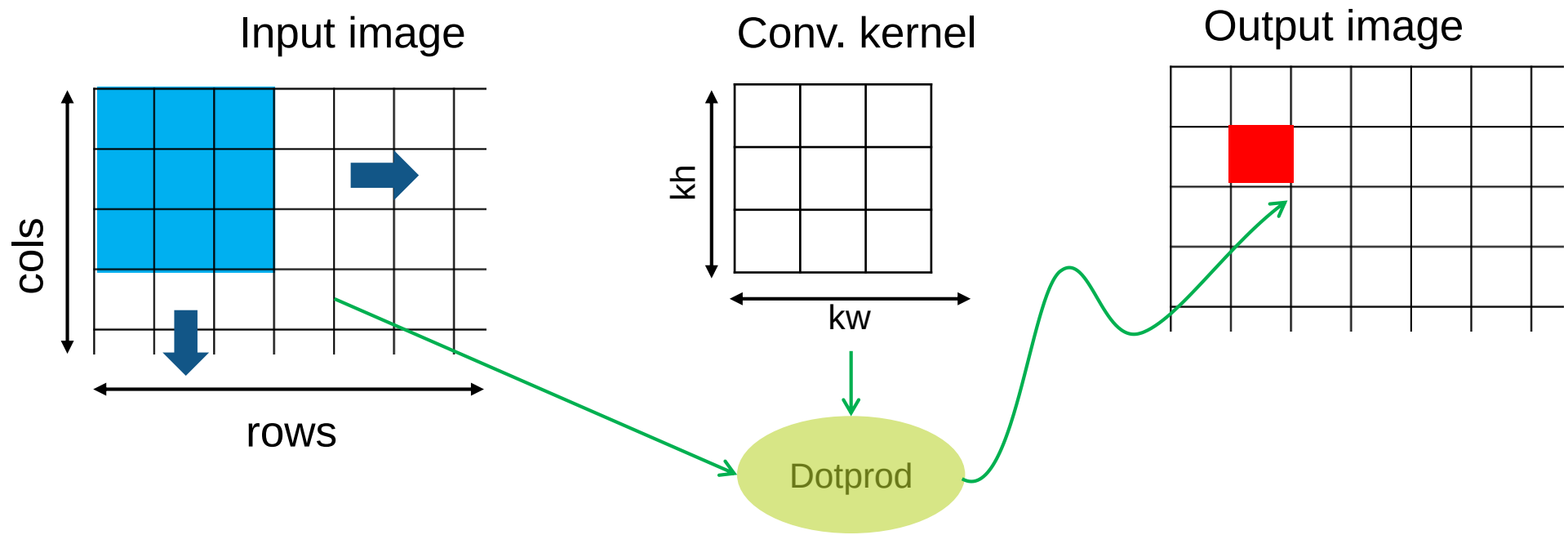
Layer fusion
Optimized computation



Buffer minimization
Optimized memory

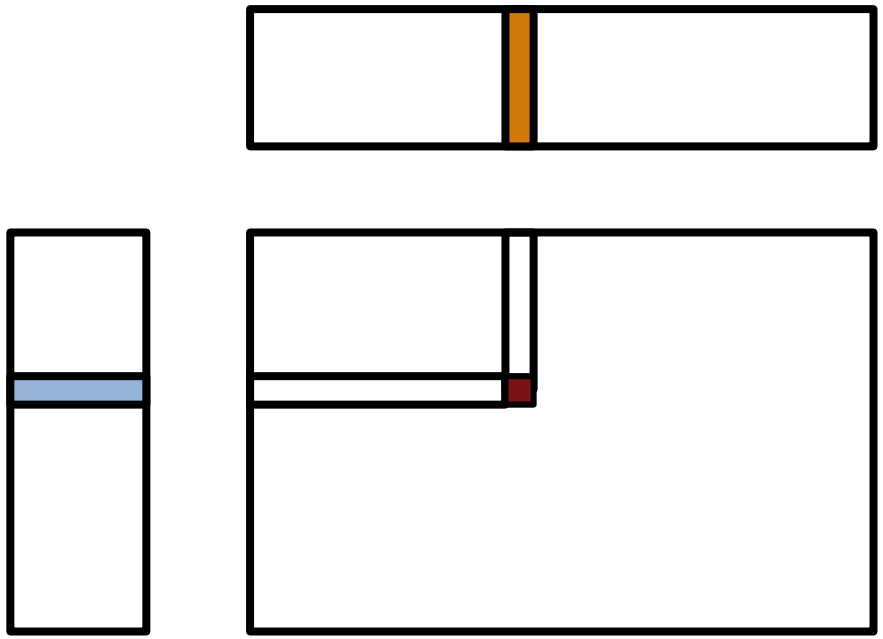
Coding Patterns: Stencil Kernels

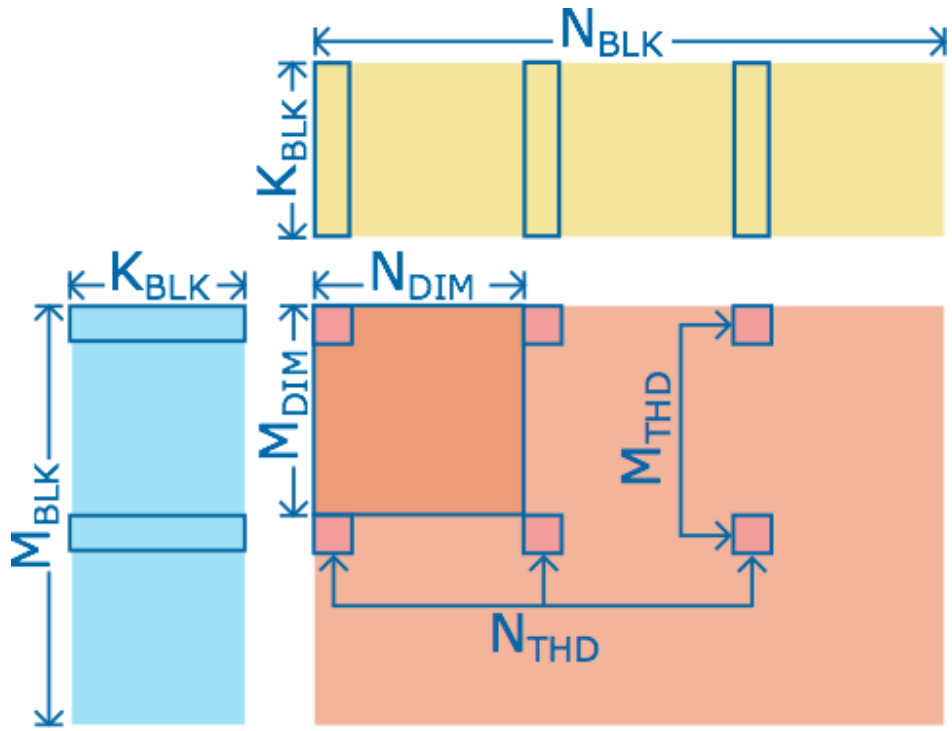
- Automatically applied for image processing functions (e.g. imfilter, imerode, imdilate, conv2, …)
- Manually apply using *gpucoder.stencilKernel()*



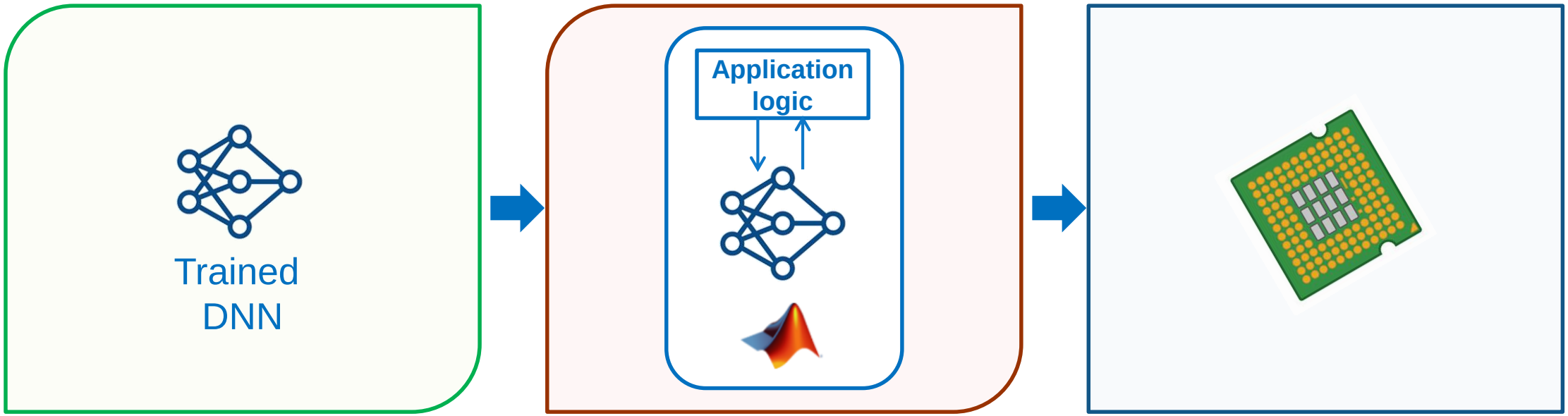
Coding Patterns: Matrix-Matrix Kernels

- Automatically applied for many MATLAB functions (e.g. matchFeatures SAD, SSD, pdist, …)
- Manually apply using *gpucoder.matrixMatrixKernel()*





Deep Learning Workflow in MATLAB



Deep Neural Network Design + Training

Application Design

Standalone Deployment

Deep Learning Workflow in MATLAB