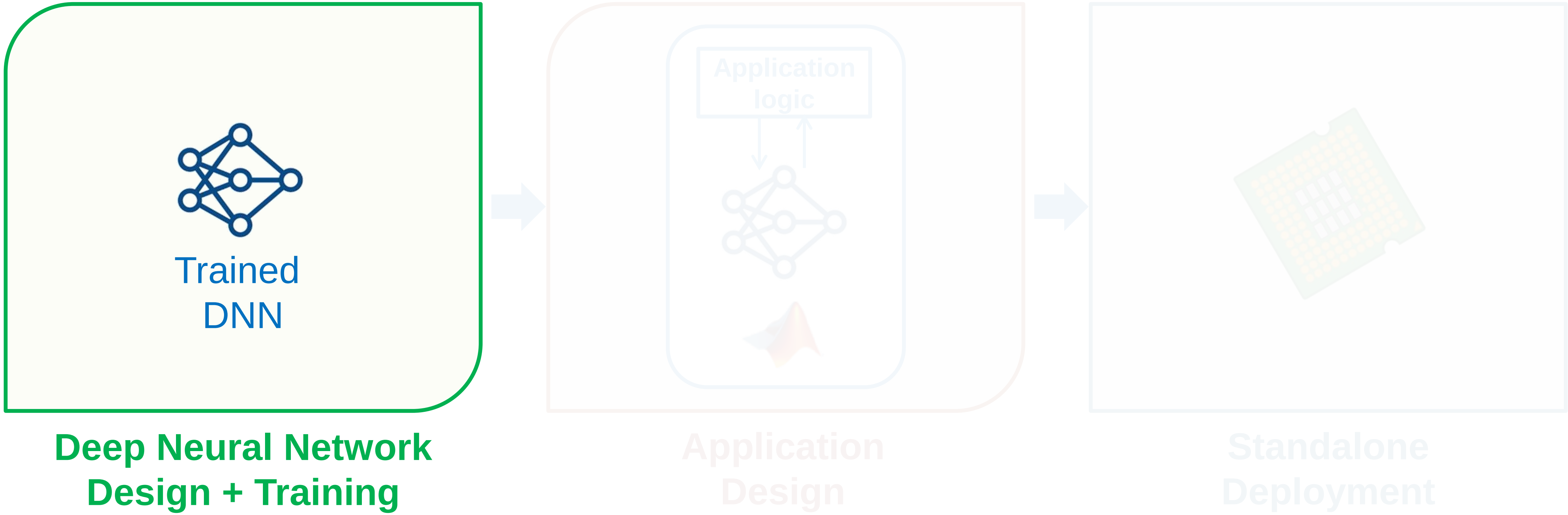
MATLAB EXPO 2019

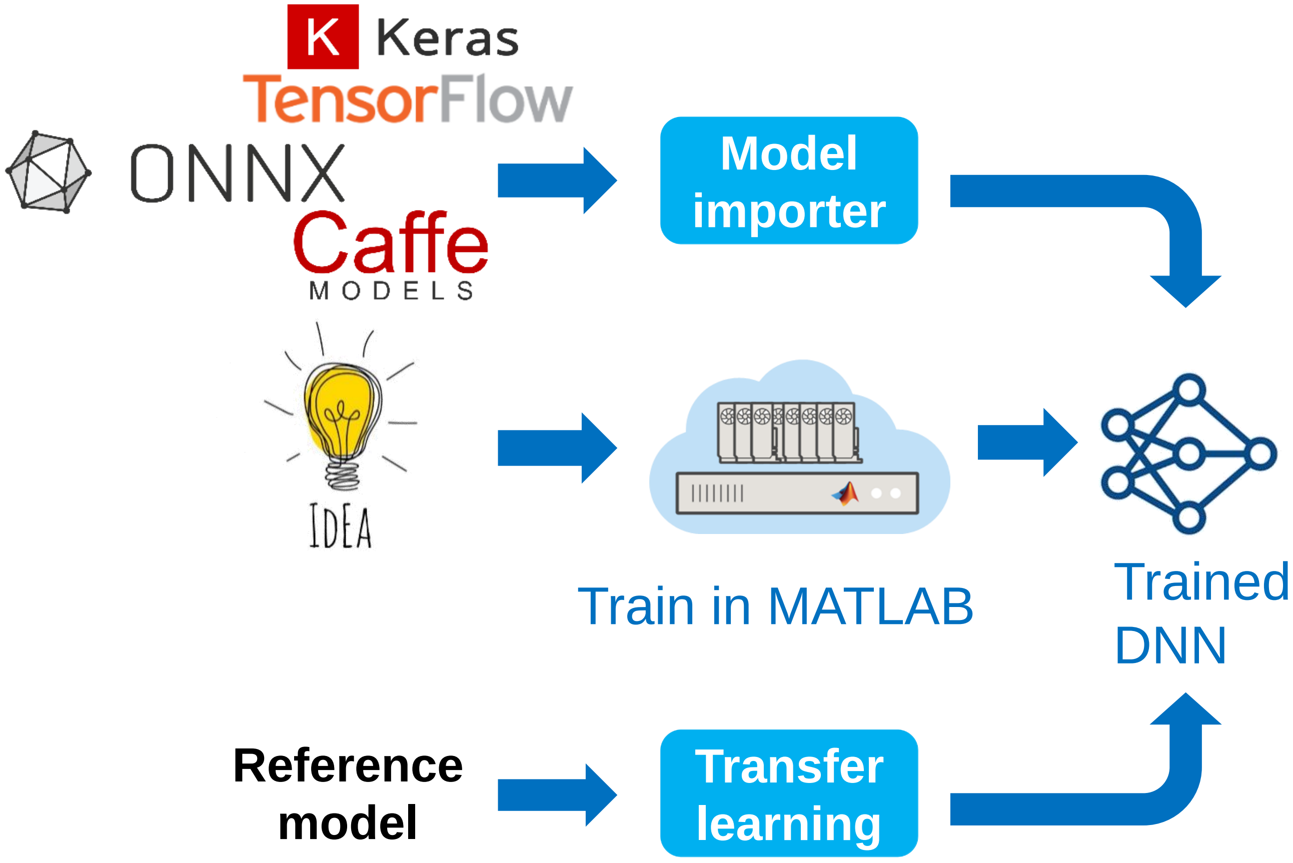
Deploying Deep Neural Networks to Embedded GPUs and CPUs

Dr Rishu Gupta
Senior Application Engineer

Deep Learning Workflow in MATLAB



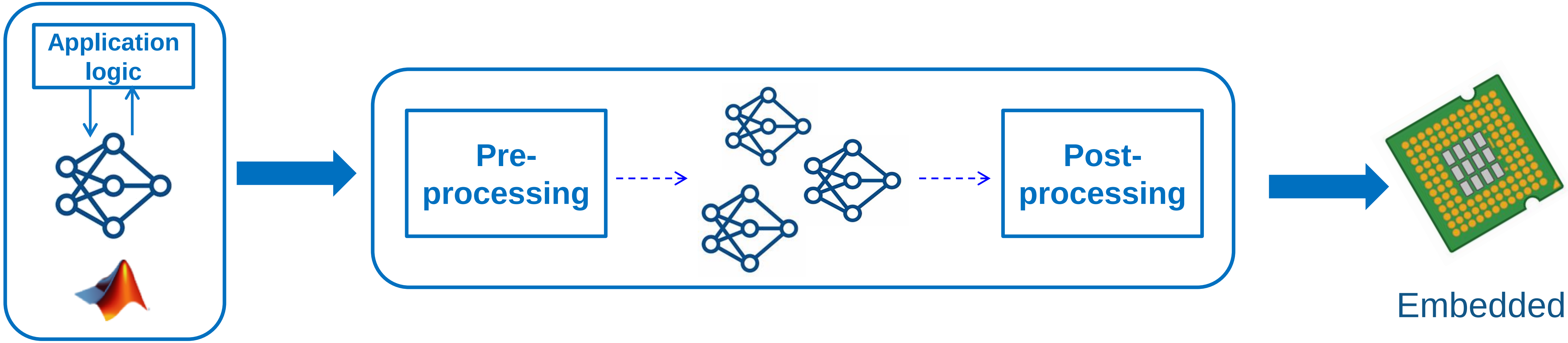
Deep Neural Network Design and Training



- **Design in MATLAB**
  - **Manage** large data sets
  - **Automate** data labeling
  - **Easy access** to models

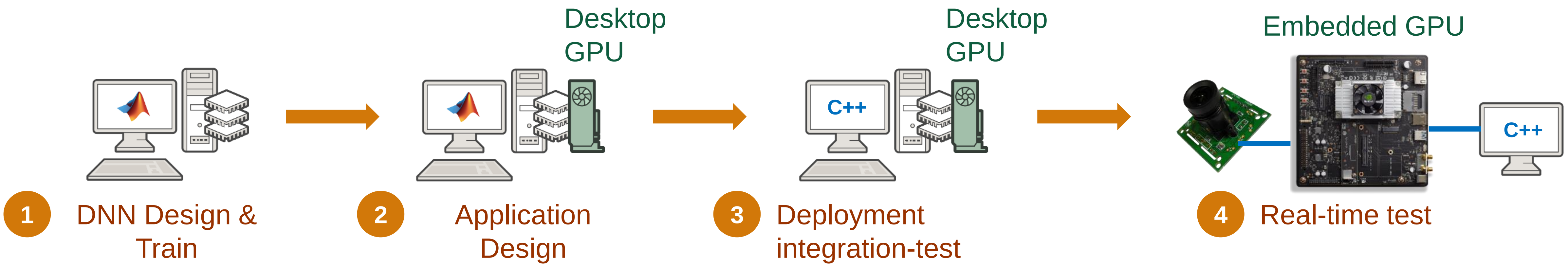
- **Training in MATLAB**
  - **Acceleration** with GPU's
  - **Scale** to clusters

Application Design



Multi-Platform Deep Learning Deployment

Algorithm Design to Embedded Deployment Workflow



1 DNN Design & Train

Desktop GPU

2 Application Design

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

Desktop GPU

3 Deployment integration-test

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

Embedded GPU

4 Real-time test

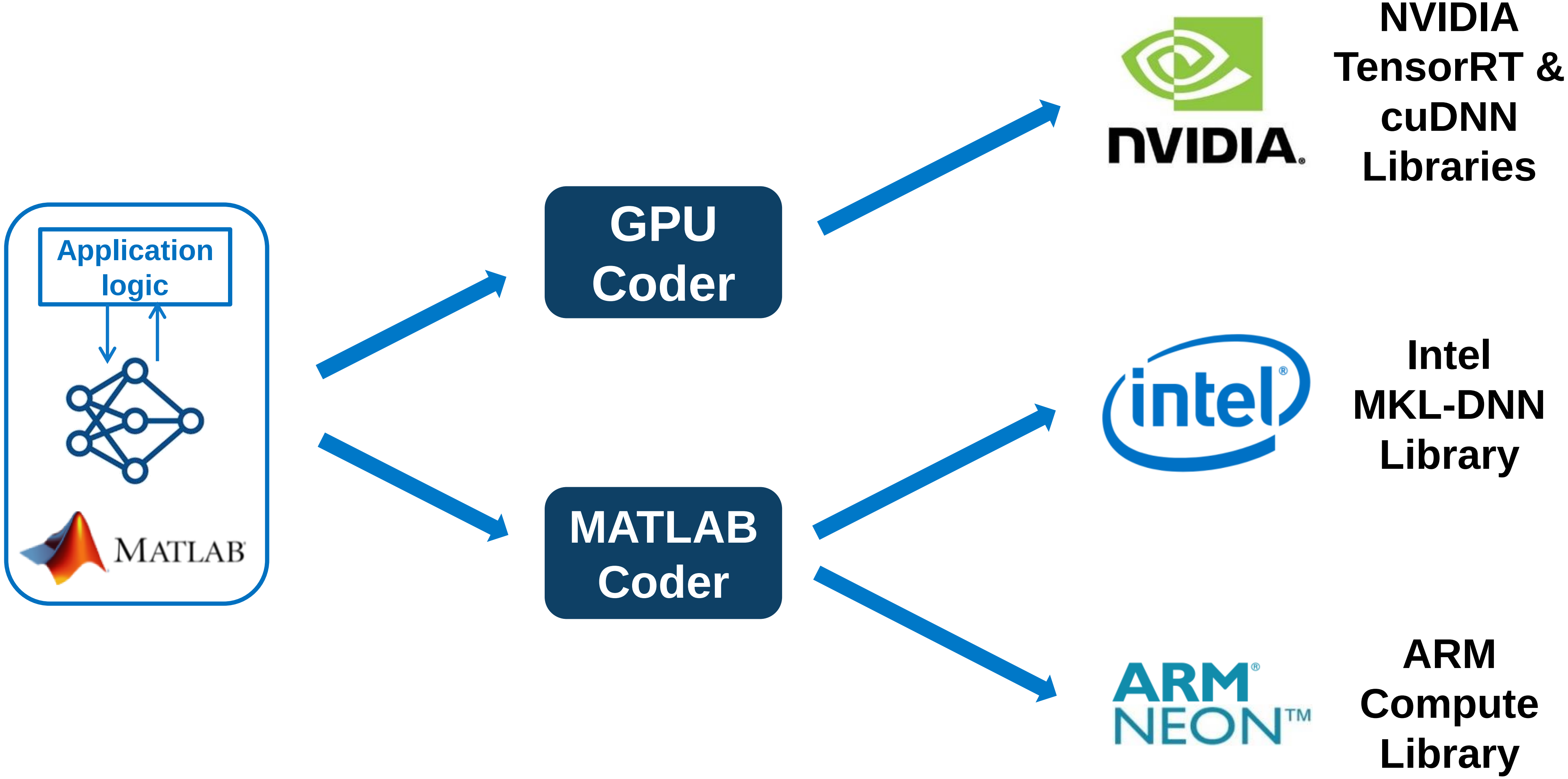
C/C++
Target-optimized libraries
Optimize for memory & speed

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

Target Libraries



Application logic

MATLAB

GPU Coder

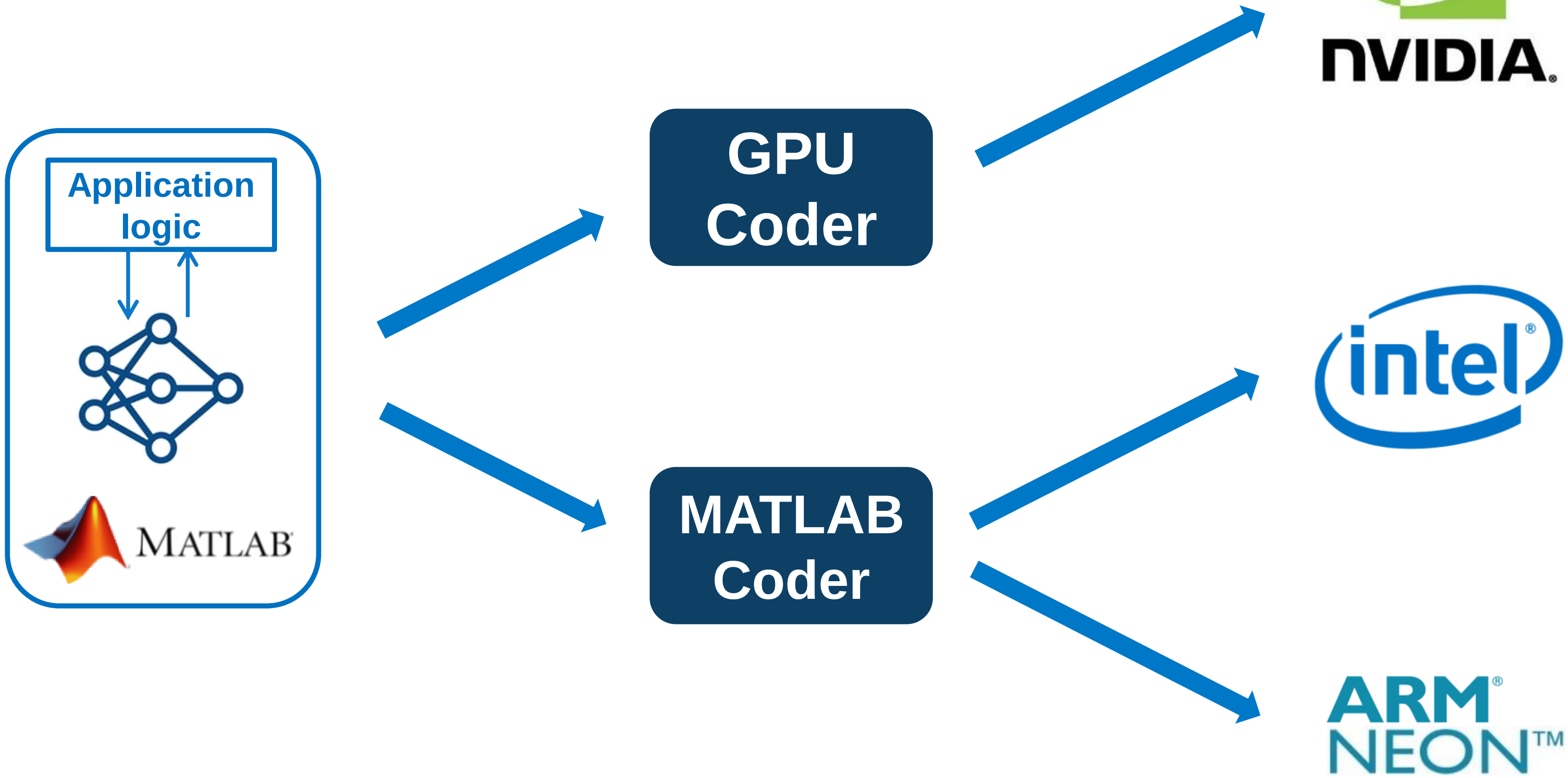
MATLAB Coder

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library

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



Application logic

MATLAB

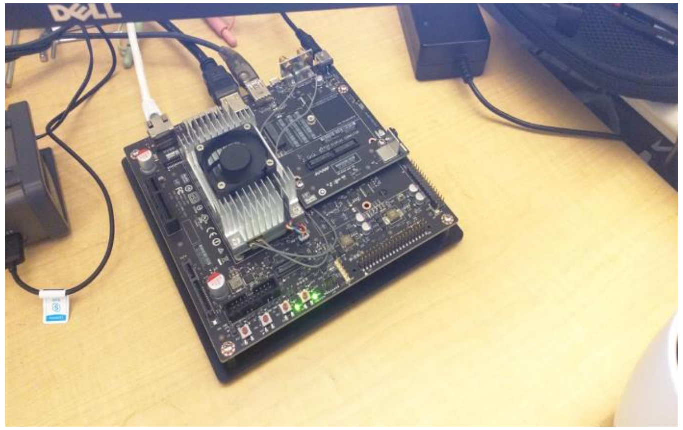
GPU Coder

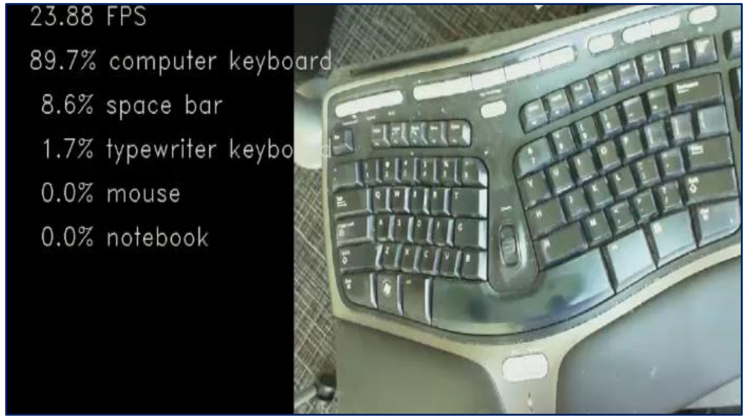
MATLAB Coder

NVIDIA

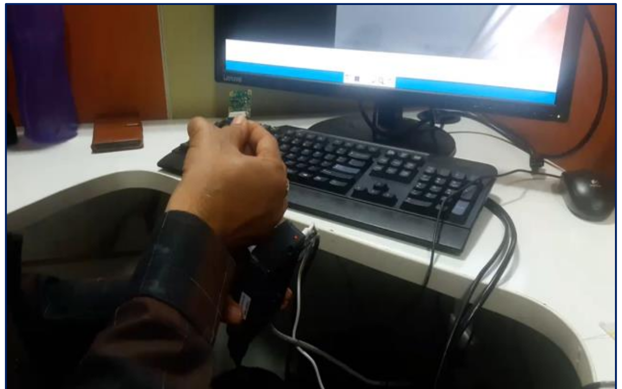
intel

ARM NEON



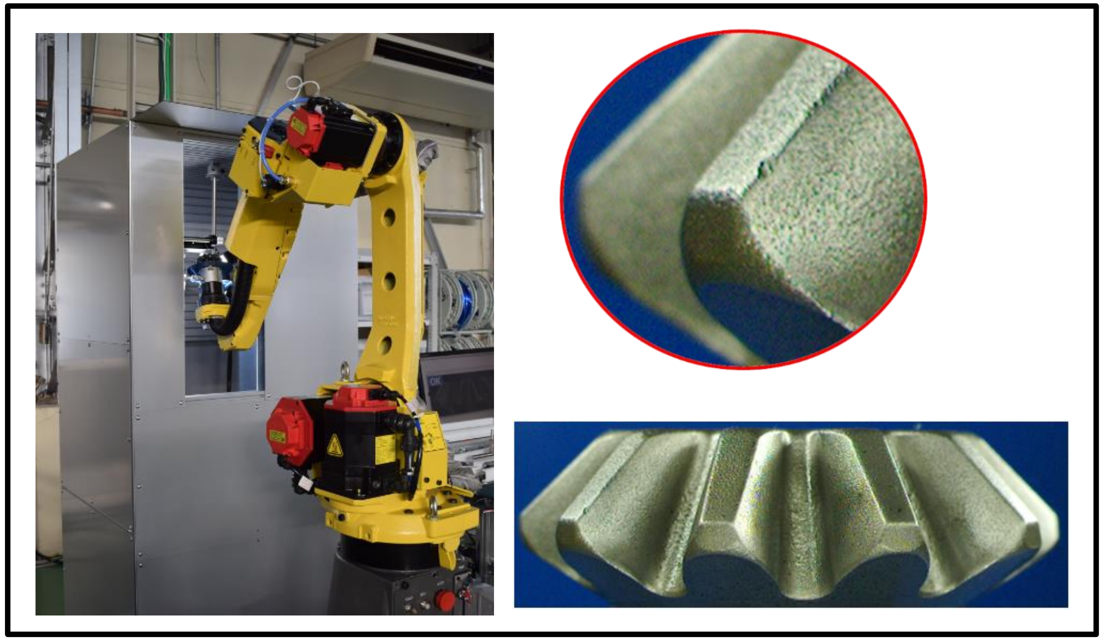


```
23.88 FPS
89.7% computer keyboard
 8.6% space bar
 1.7% typewriter keyboard
 0.0% mouse
 0.0% notebook
```



Musashi Seimitsu Industry Co.,Ltd.

Detect Abnormalities in Automotive Parts



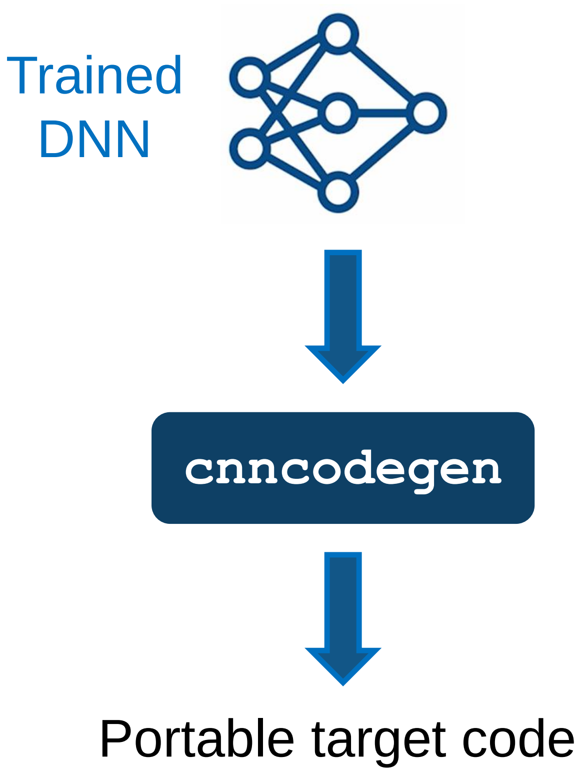
Automated visual inspection of 1.3 million bevel gear per month

MATLAB use in project:

- Preprocessing of captured images
- Image annotation for training
- Deep learning based analysis
  - Various transfer learning methods (Combinations of CNN models, Classifiers)
  - Estimation of defect area using Class Activation Map (CAM)
  - Abnormality/defect classification
- Deployment to NVIDIA Jetson using GPU Coder

Deep Learning Deployment Workflows

INFERENCE ENGINE DEPLOYMENT

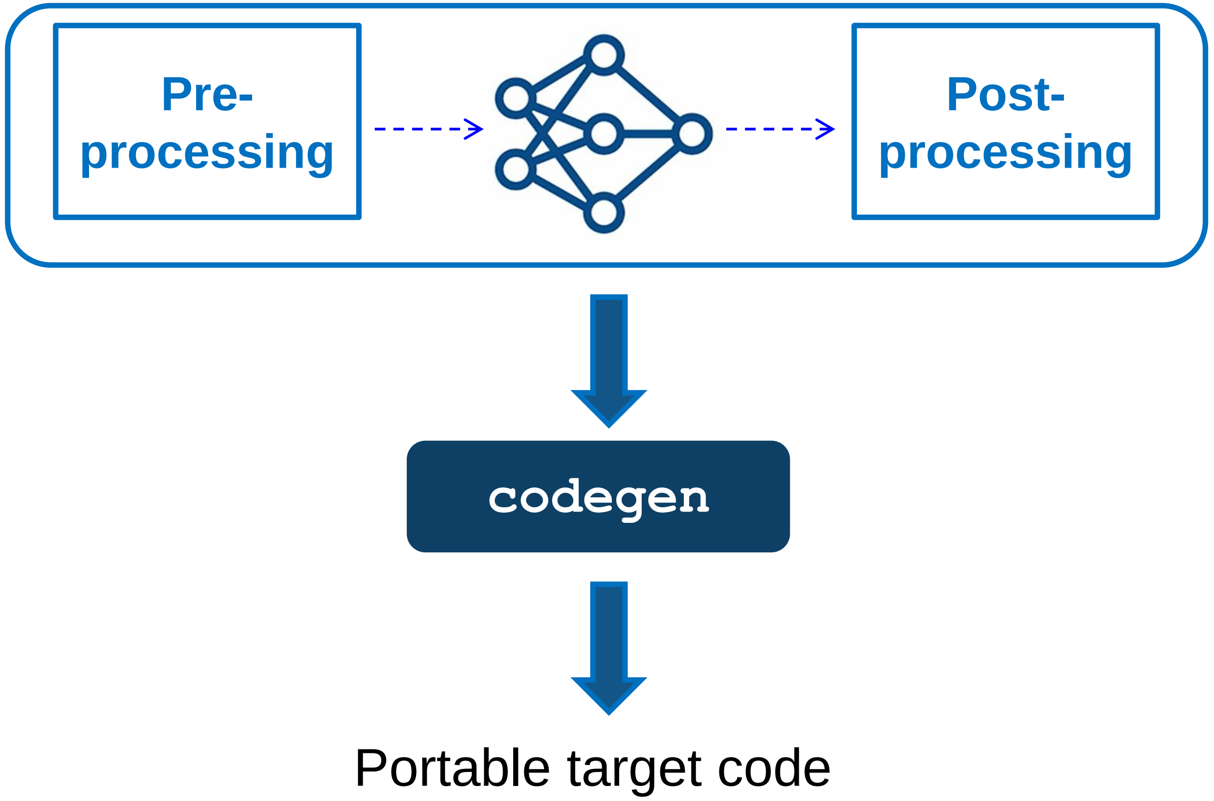


Trained DNN

`cnncodegen`

Portable target code

INTEGRATED APPLICATION DEPLOYMENT



Pre-processing

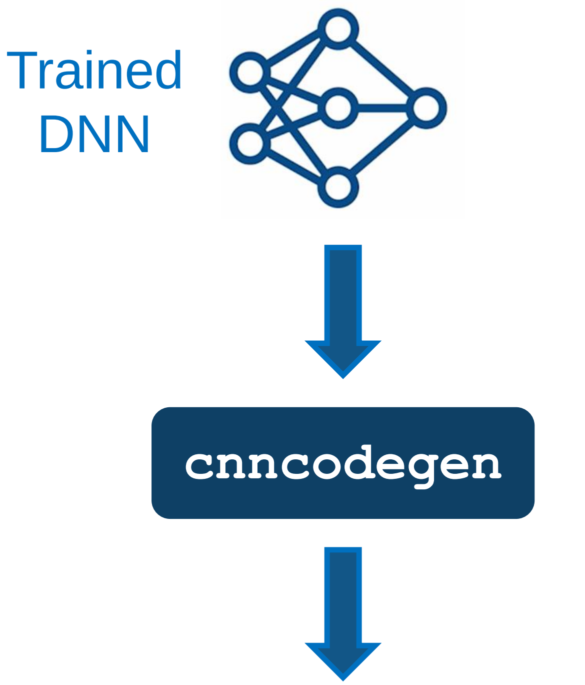
Post-processing

`codegen`

Portable target code

Workflow for Inference Engine Deployment

INFERENCE ENGINE DEPLOYMENT



Trained DNN

cnncodegen

Portable target code

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

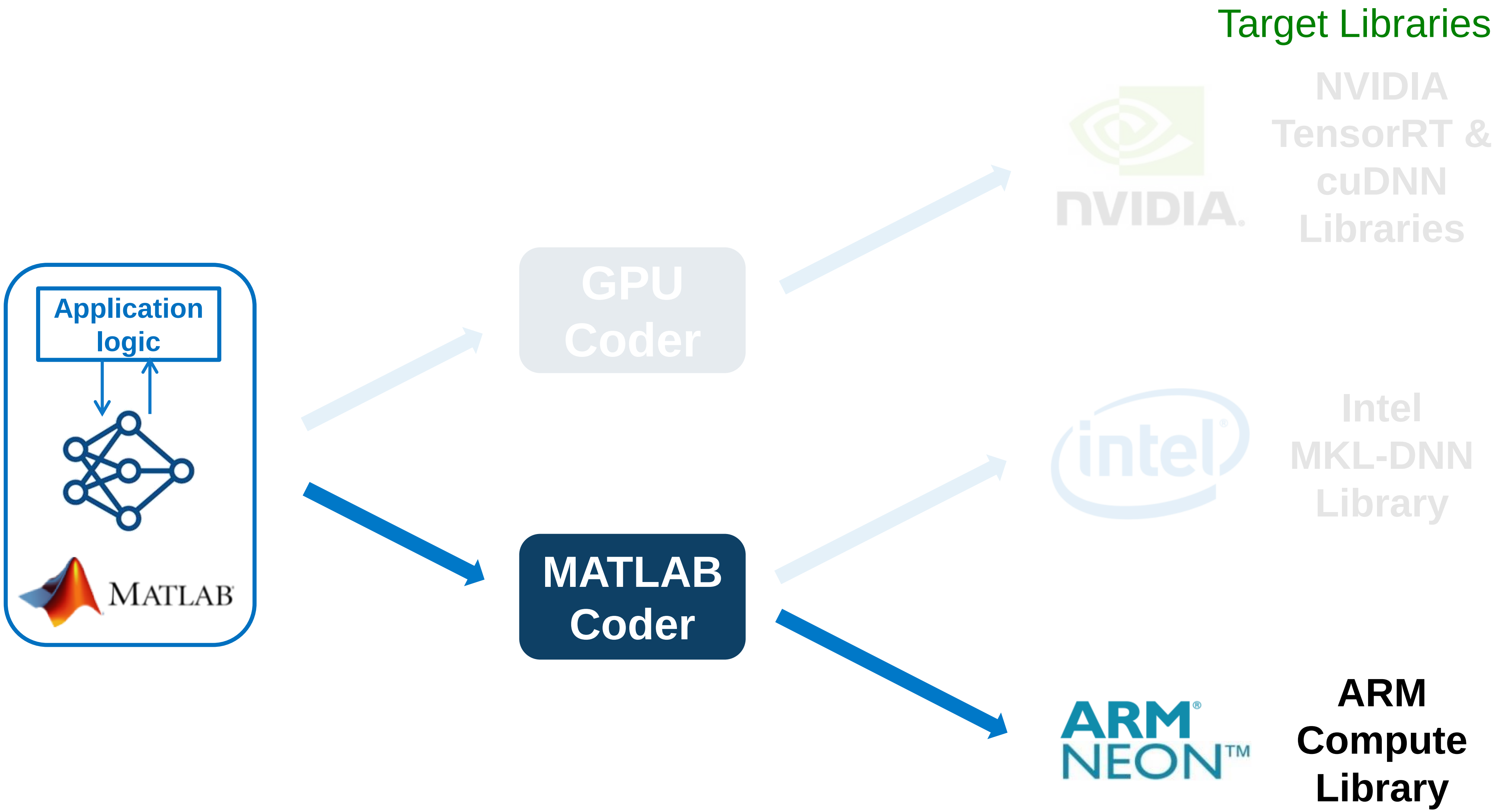
```
>> make -C ./codegen -f …mk
```

4. Use hand written main function to call inference engine

5. Generate the exe and test the executable

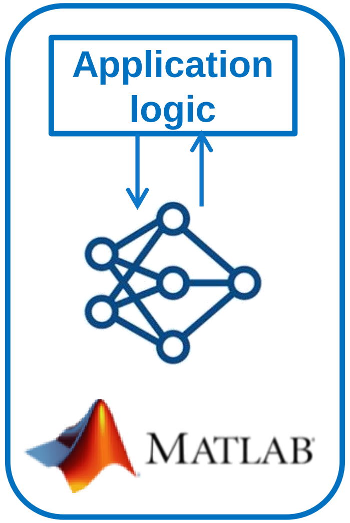
```
>> make -C ./ ……
```

Deep Learning Inference Deployment



Deep Learning Inference Deployment

Pedestrian Detection



MATLAB Coder

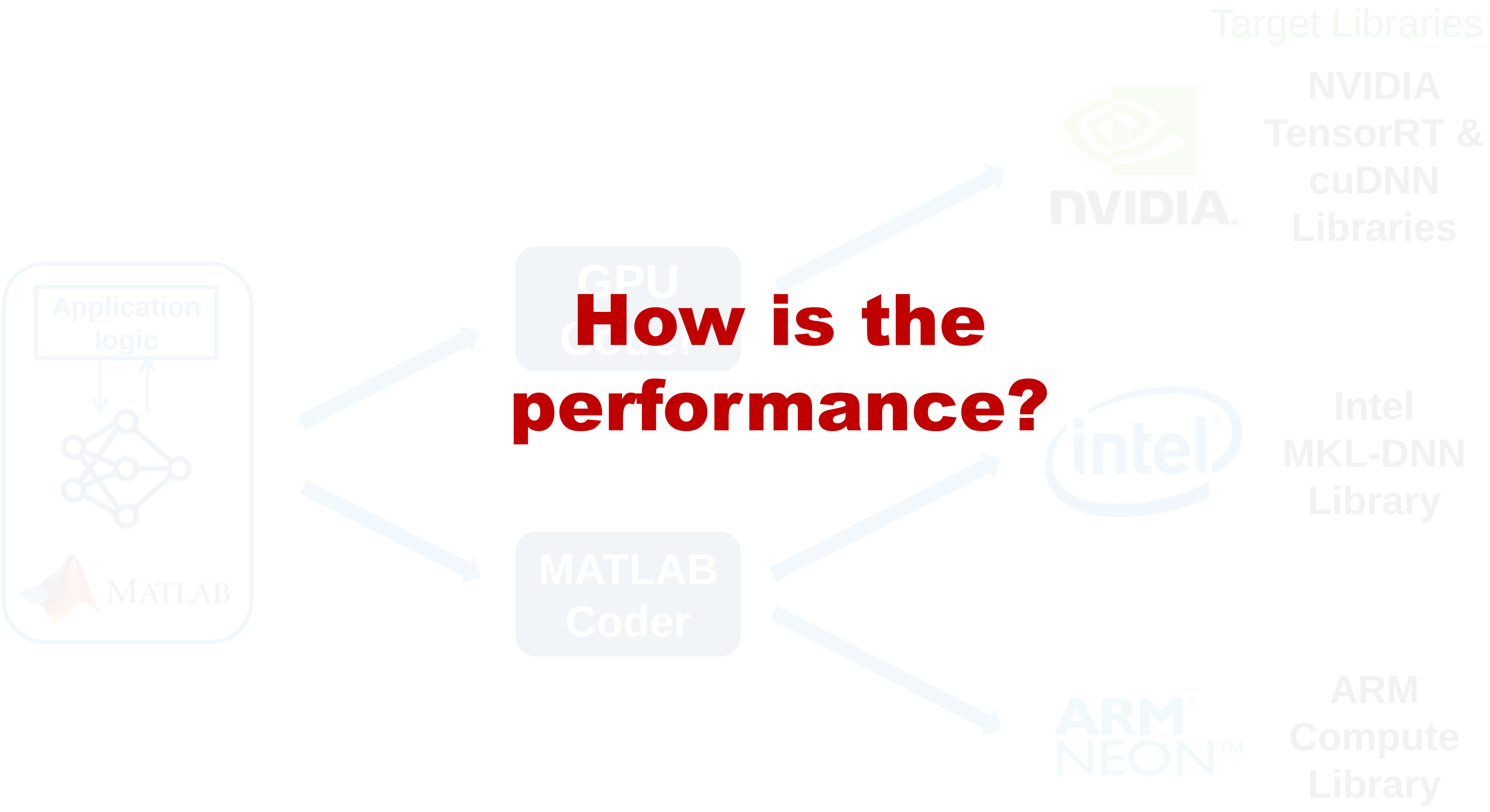
Target Libraries

NVIDIA TensorRT & cuDNN Libraries

Intel MKL-DNN Library

ARM Compute Library

How is the performance?



Target Libraries

Application logic

MATLAB

GPU Coder

MATLAB Coder

NVIDIA TensorRT & cuDNN Libraries

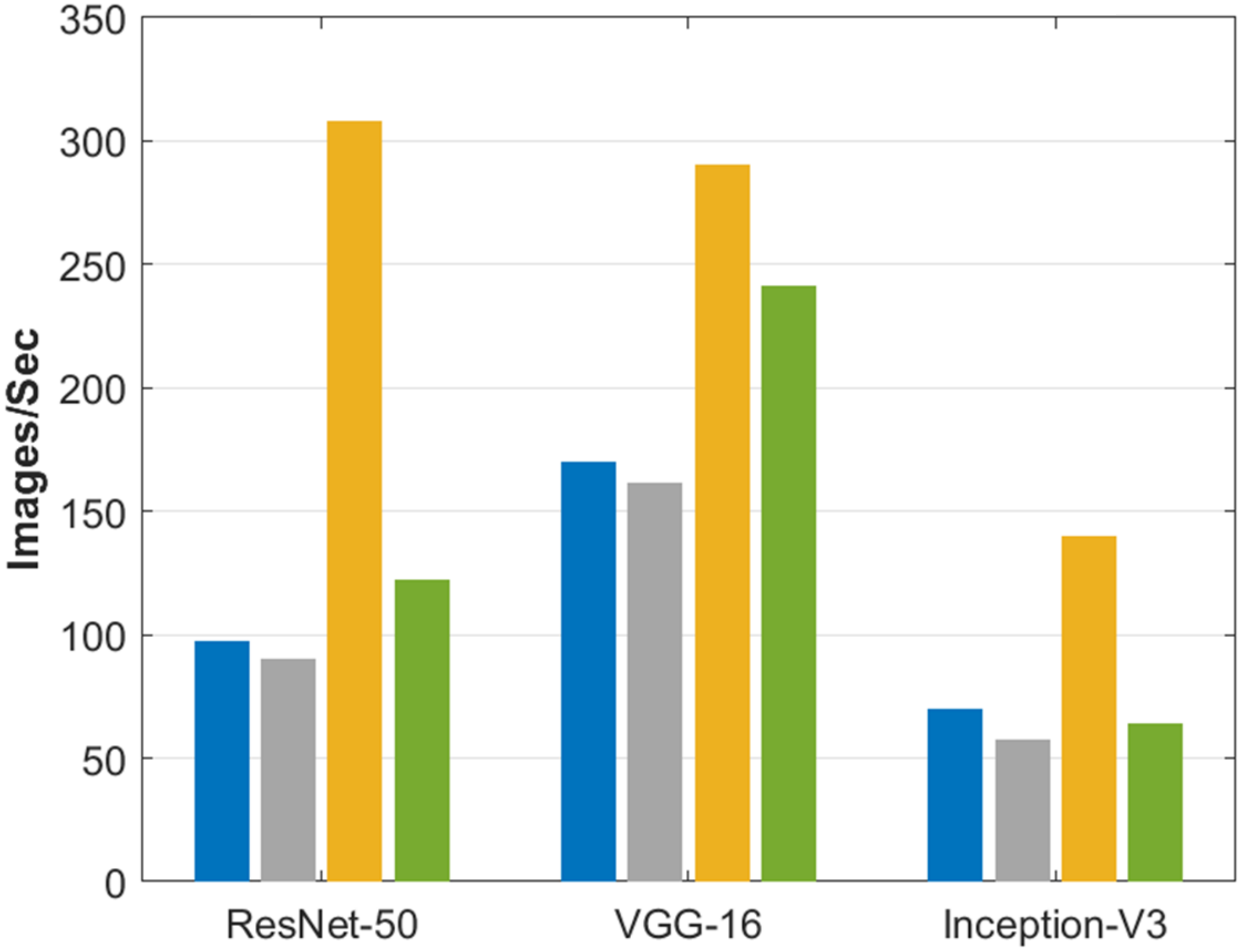
Intel MKL-DNN Library

ARM Compute Library

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)

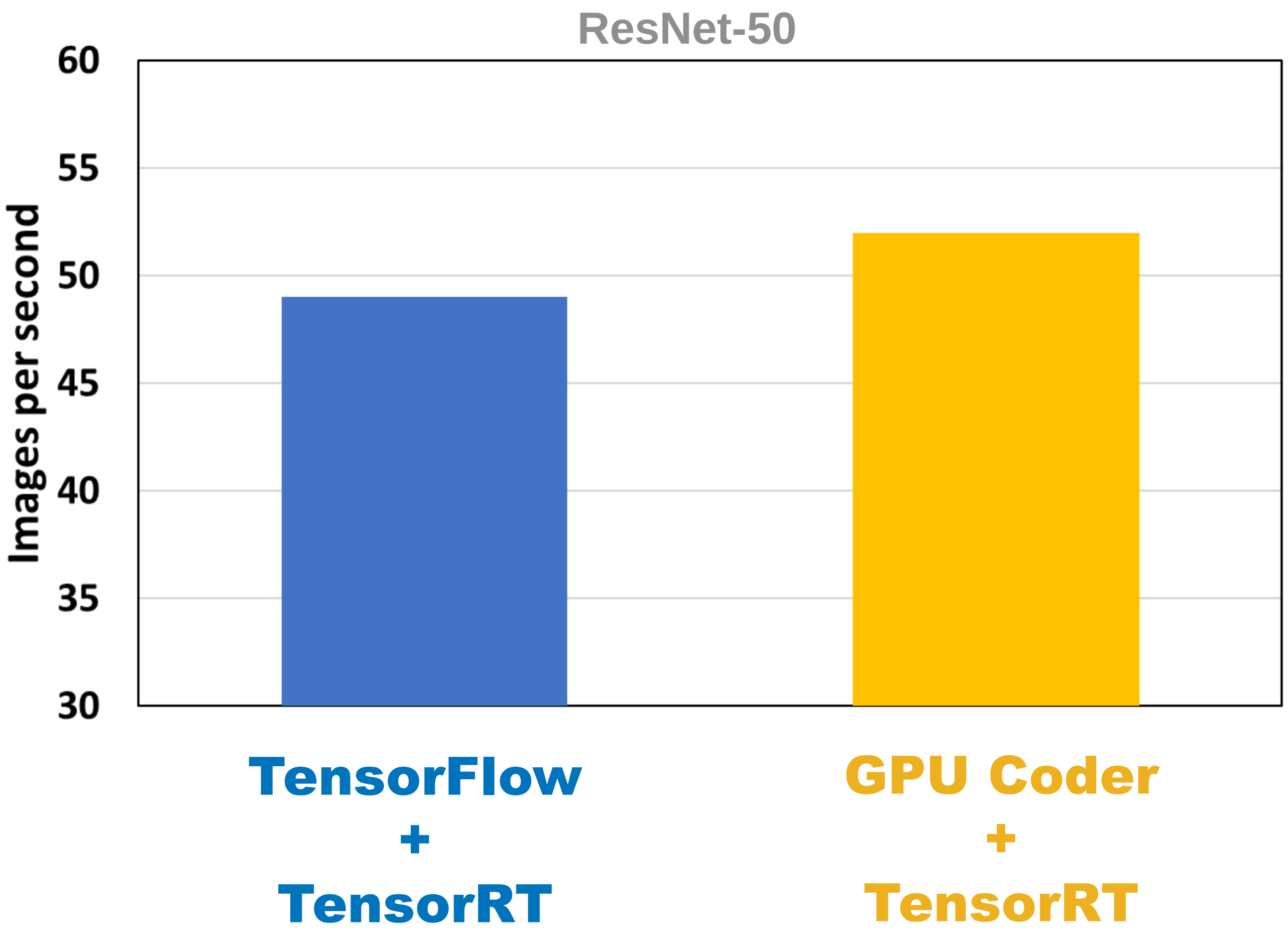
MXNet (1.4.0)

GPU Coder (R2019a)

PyTorch (1.0.0)

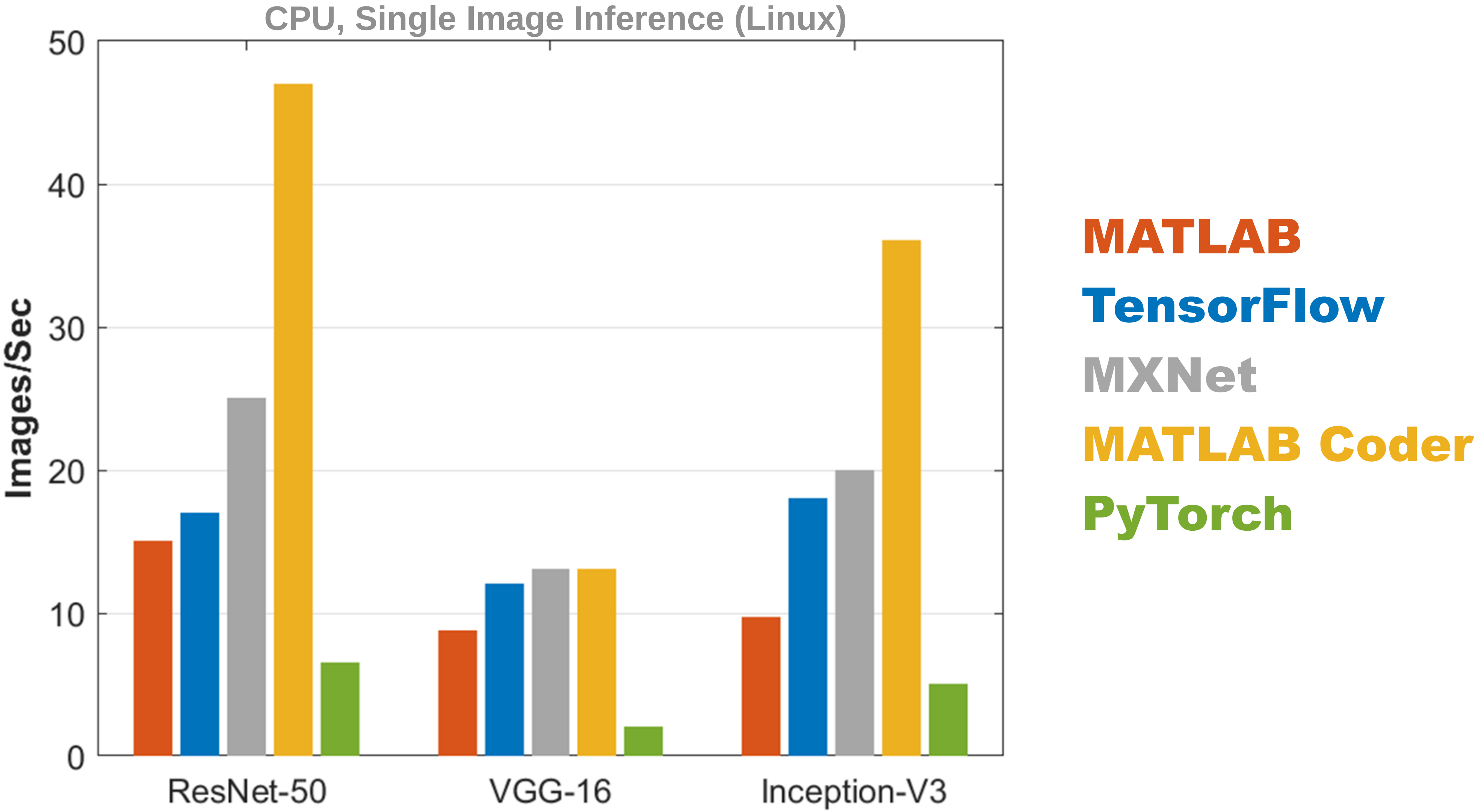
Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7 - Frameworks: TensorFlow 1.13.0, MXNet 1.4.0 PyTorch 1.0.0

Single Image Inference on Jetson TX2



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

CPU Performance



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

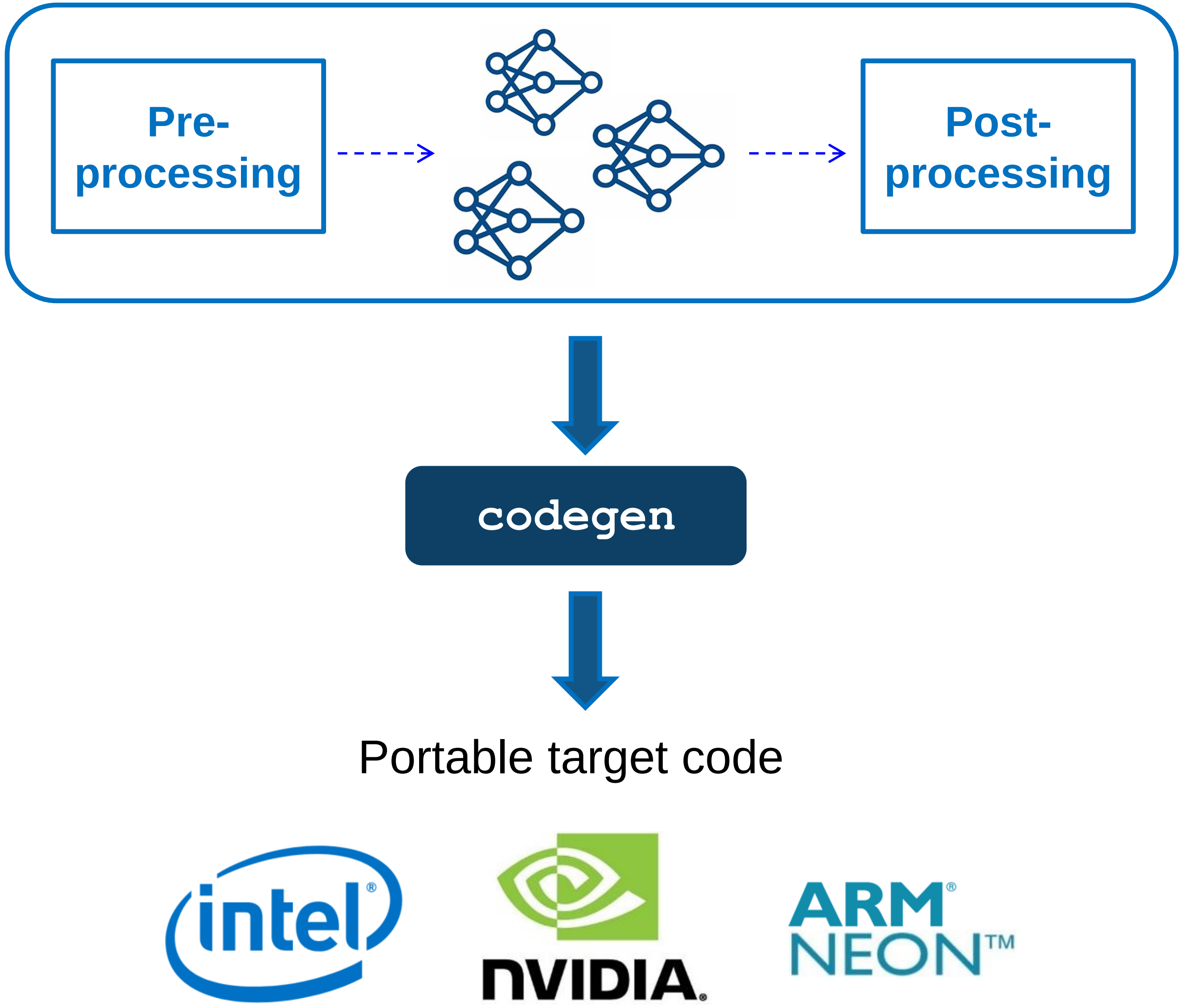
DNN libraries are great for inference, …

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

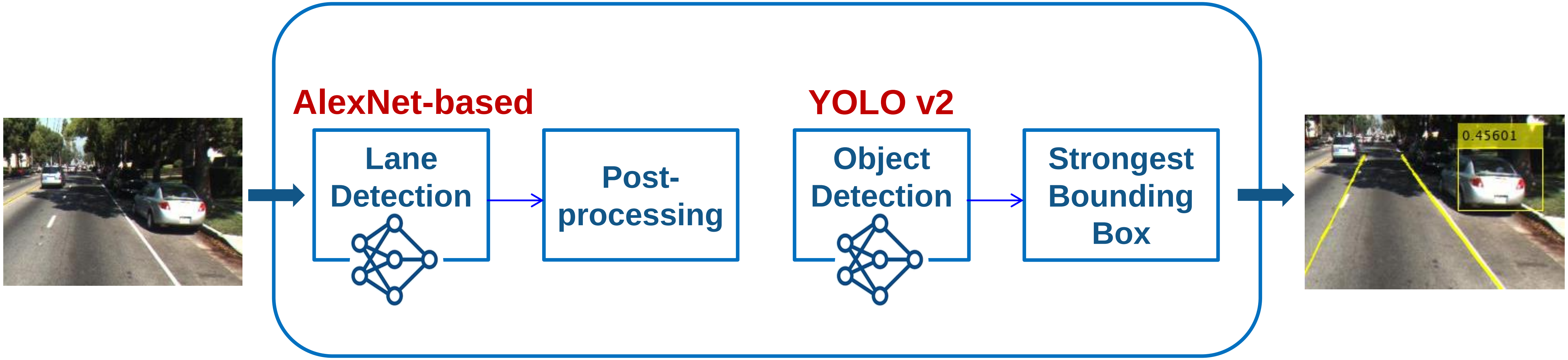
- NVIDIA® CUDA libraries, including TensorRT & cuDNN
- Intel® Math Kernel Library for Deep Neural Networks (MKL-DNN)
- ARM® Compute libraries for mobile platforms

But, applications require more than just inference

Deep Learning Workflows: Integrated Application Deployment



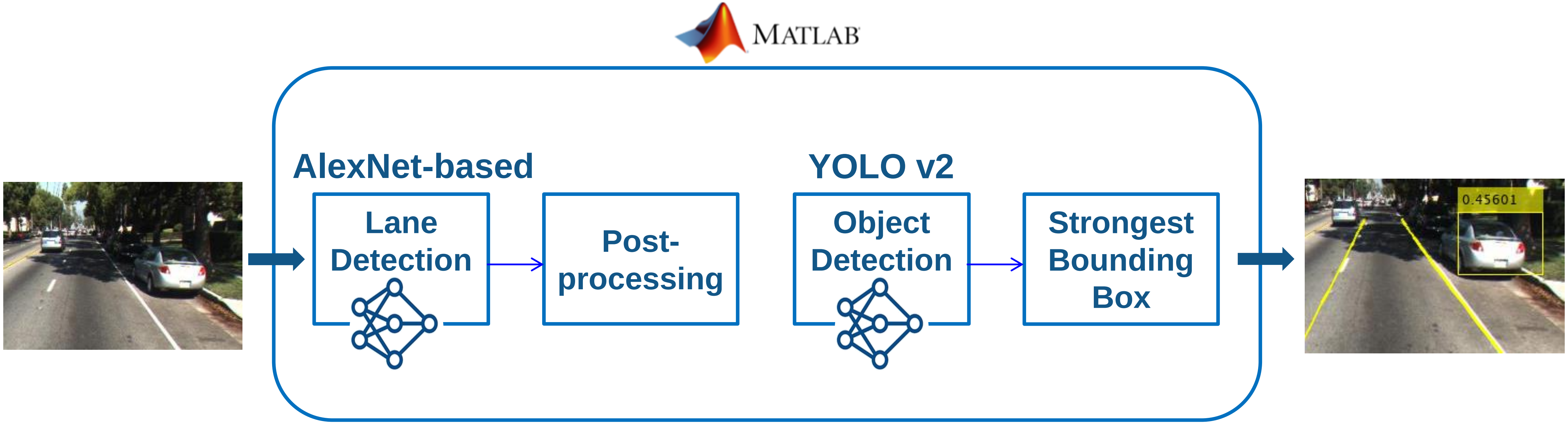
Lane and Object Detection using YOLO v2



Workflow:

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

(1) Test in MATLAB



MATLAB

AlexNet-based

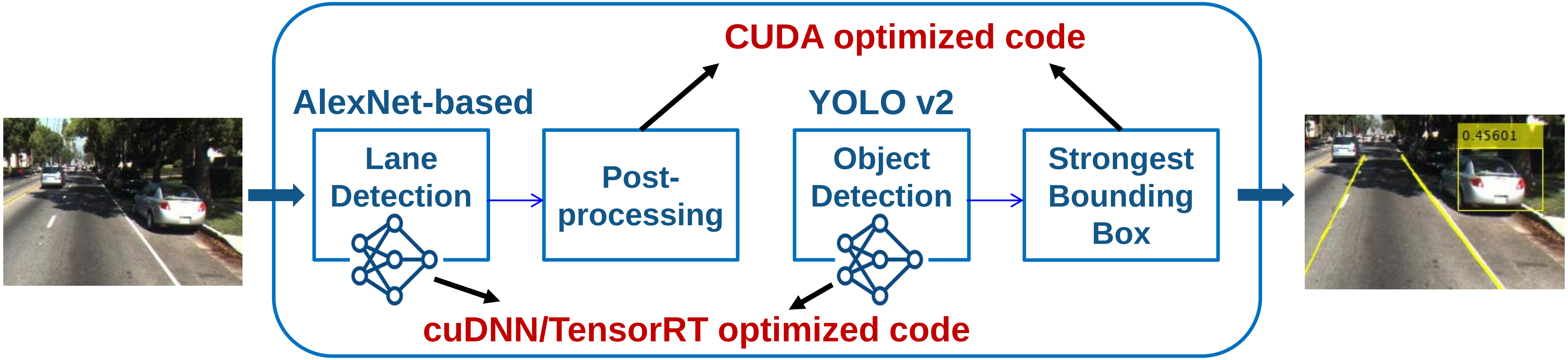
Lane Detection → Post-processing

YOLO v2

Object Detection → Strongest Bounding Box

(2) Generate Code and Test on Desktop GPU

MATLAB



CUDA optimized code

AlexNet-based

Lane Detection

Post-processing

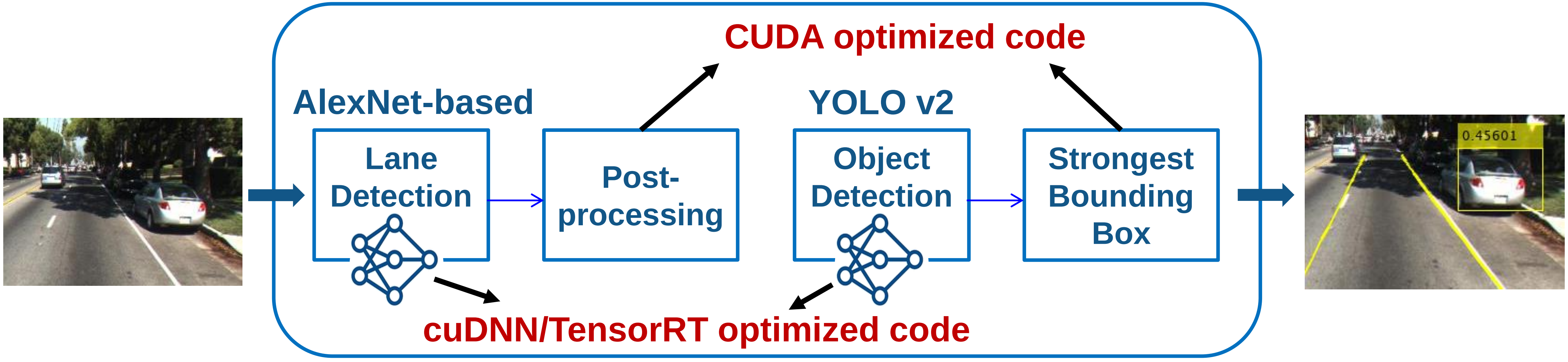
YOLO v2

Object Detection

Strongest Bounding Box

cuDNN/TensorRT optimized code

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



CUDA optimized code

AlexNet-based

Lane Detection

Post-processing

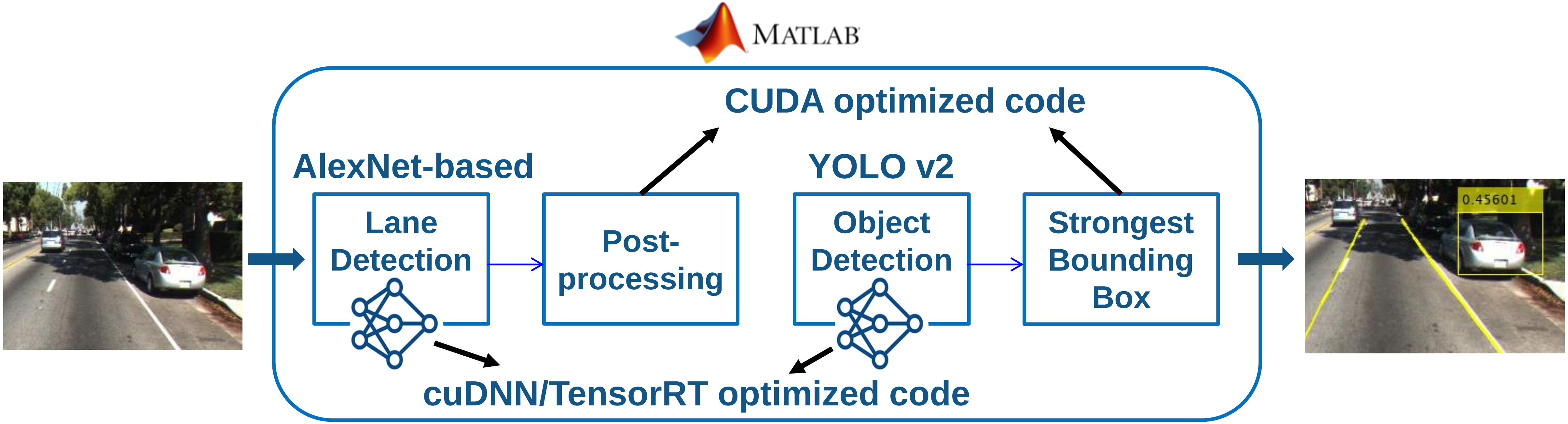
YOLO v2

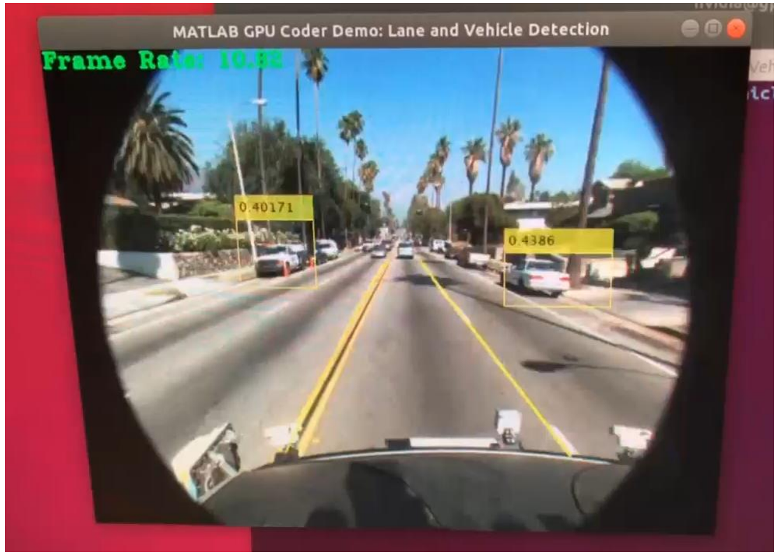
Object Detection

Strongest Bounding Box

cuDNN/TensorRT optimized code

Lane and Object Detection using YOLO v2

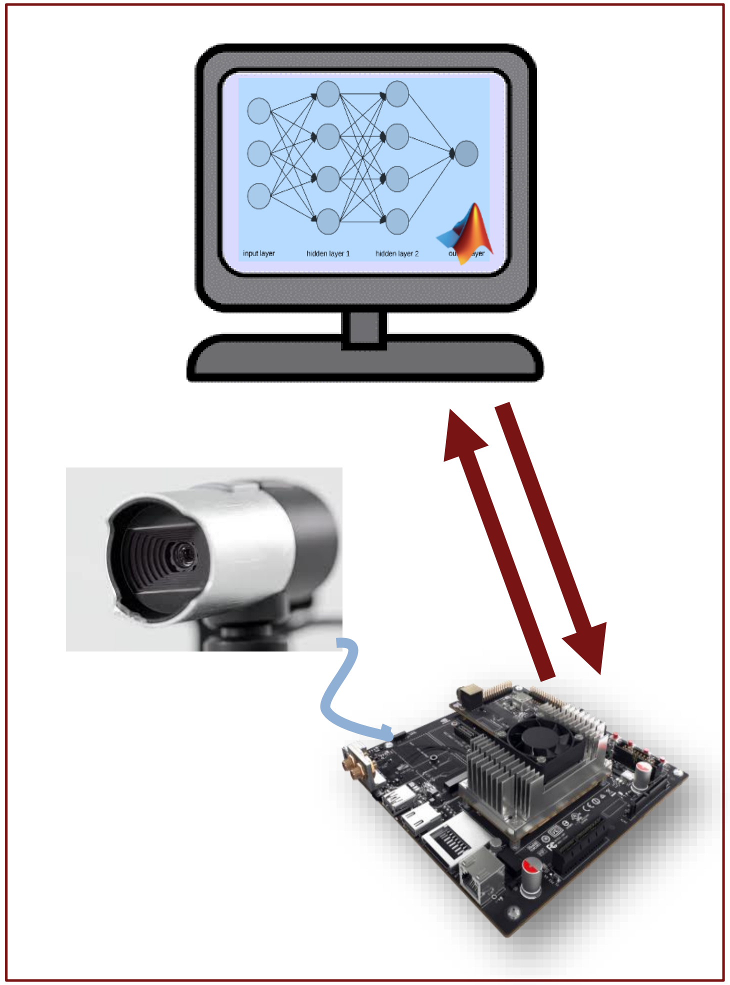



MATLAB GPU Coder Demo: Lane and Vehicle Detection

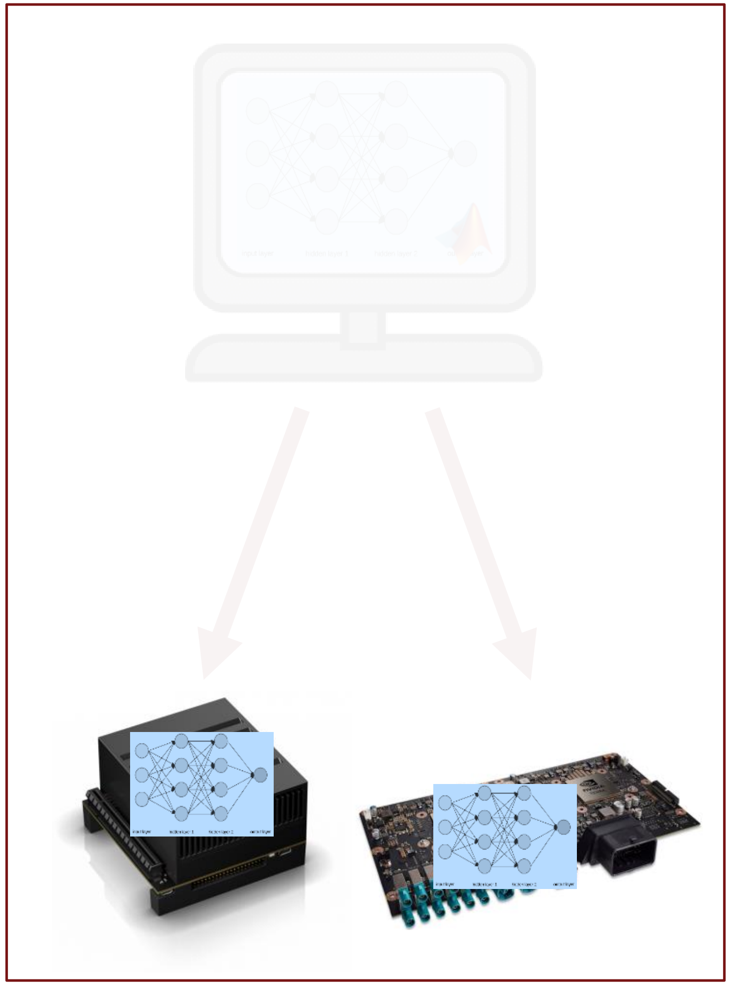
Workflow:

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

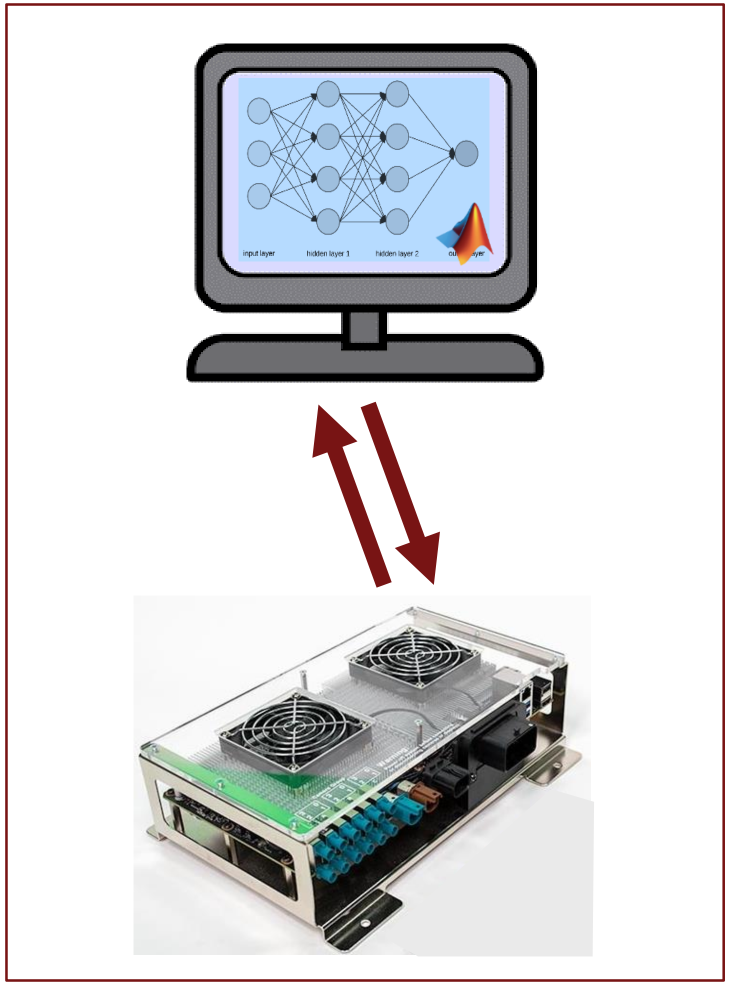
Accessing Hardware



Access Peripheral from MATLAB

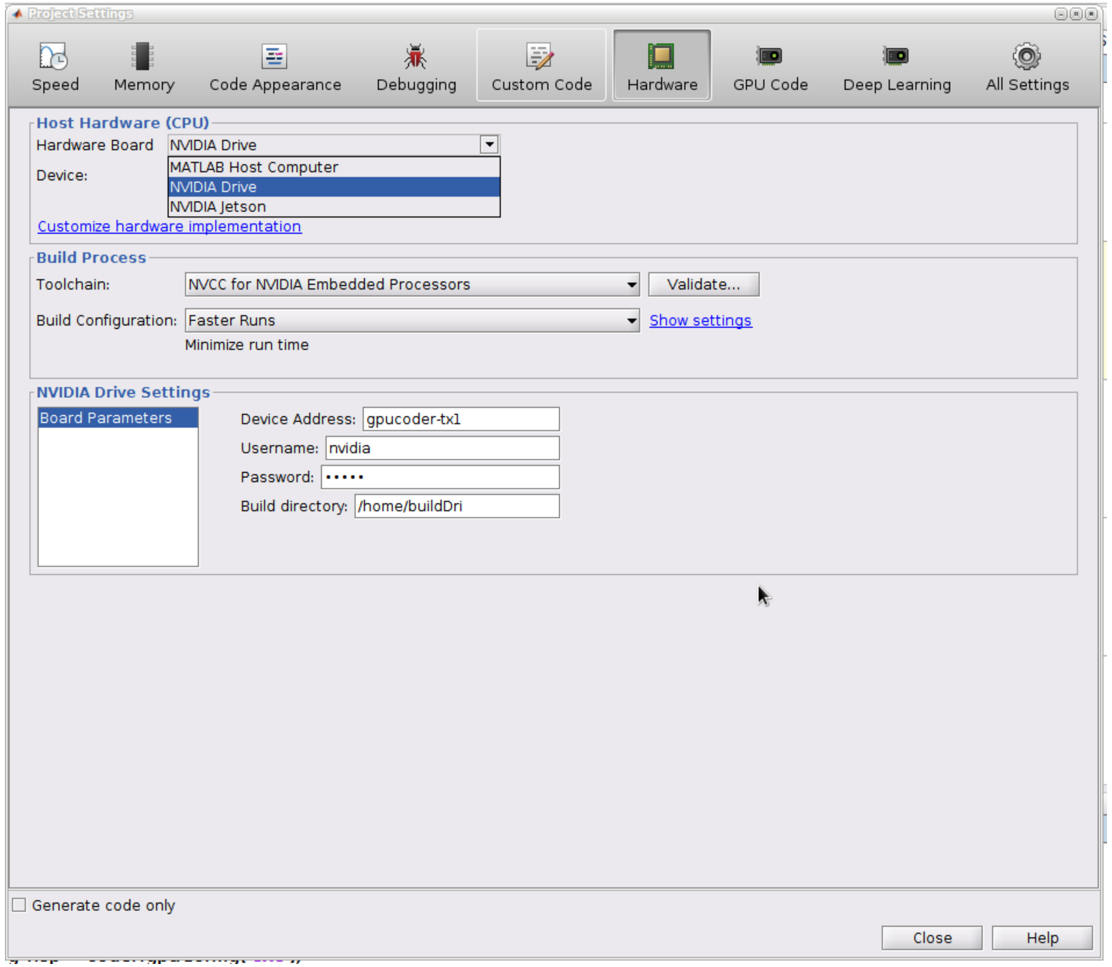


Deploy Standalone Application



Processor-in-Loop Verification

Deploy to Target Hardware via Apps and Command Line



Host Hardware (CPU)

Hardware Board: NVIDIA Drive

Device:

Customize hardware implementation

Build Process

Toolchain: NVCC for NVIDIA Embedded Processors

Build Configuration: Faster Runs — Show settings

Minimize run time

NVIDIA Drive Settings

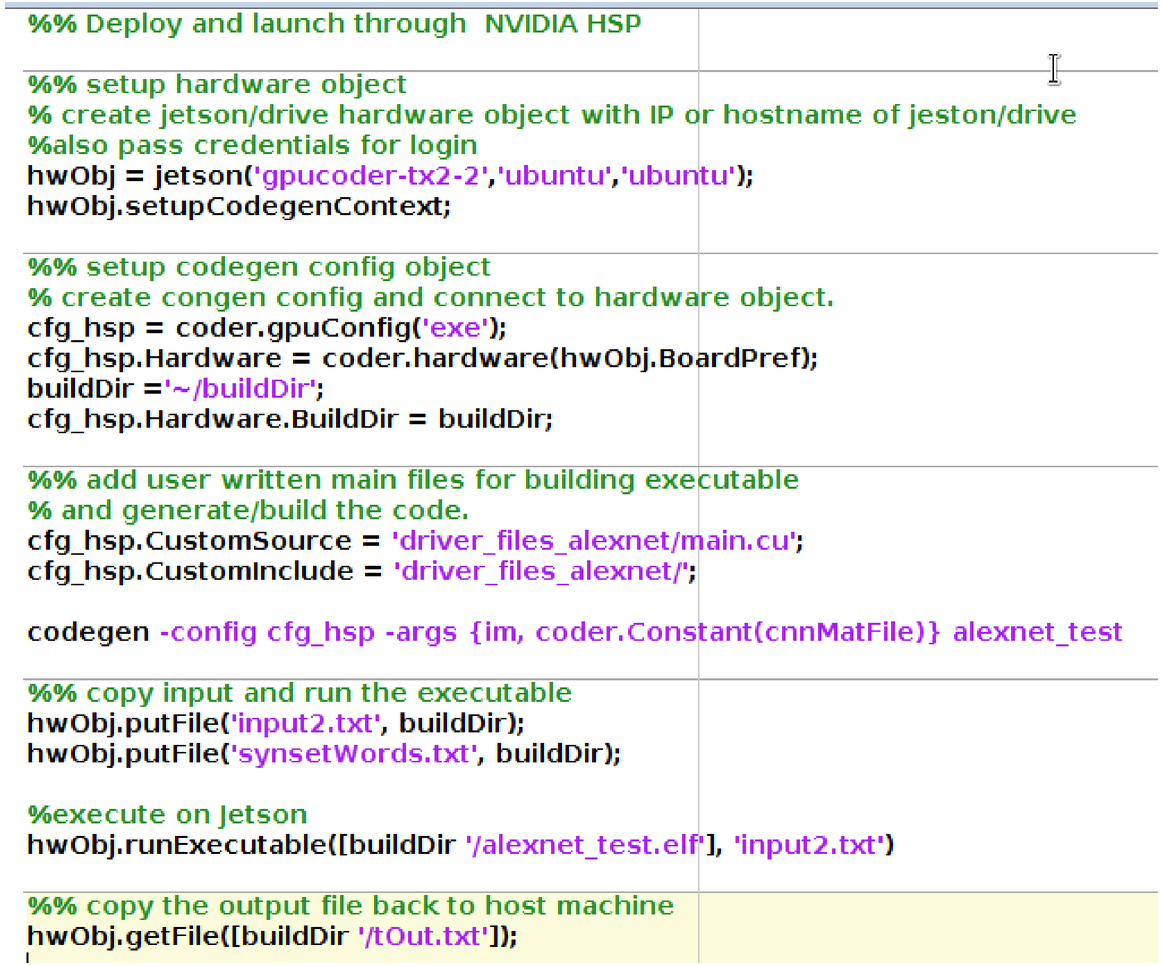
Device Address: gpucoder-tx1

Username: nvidia

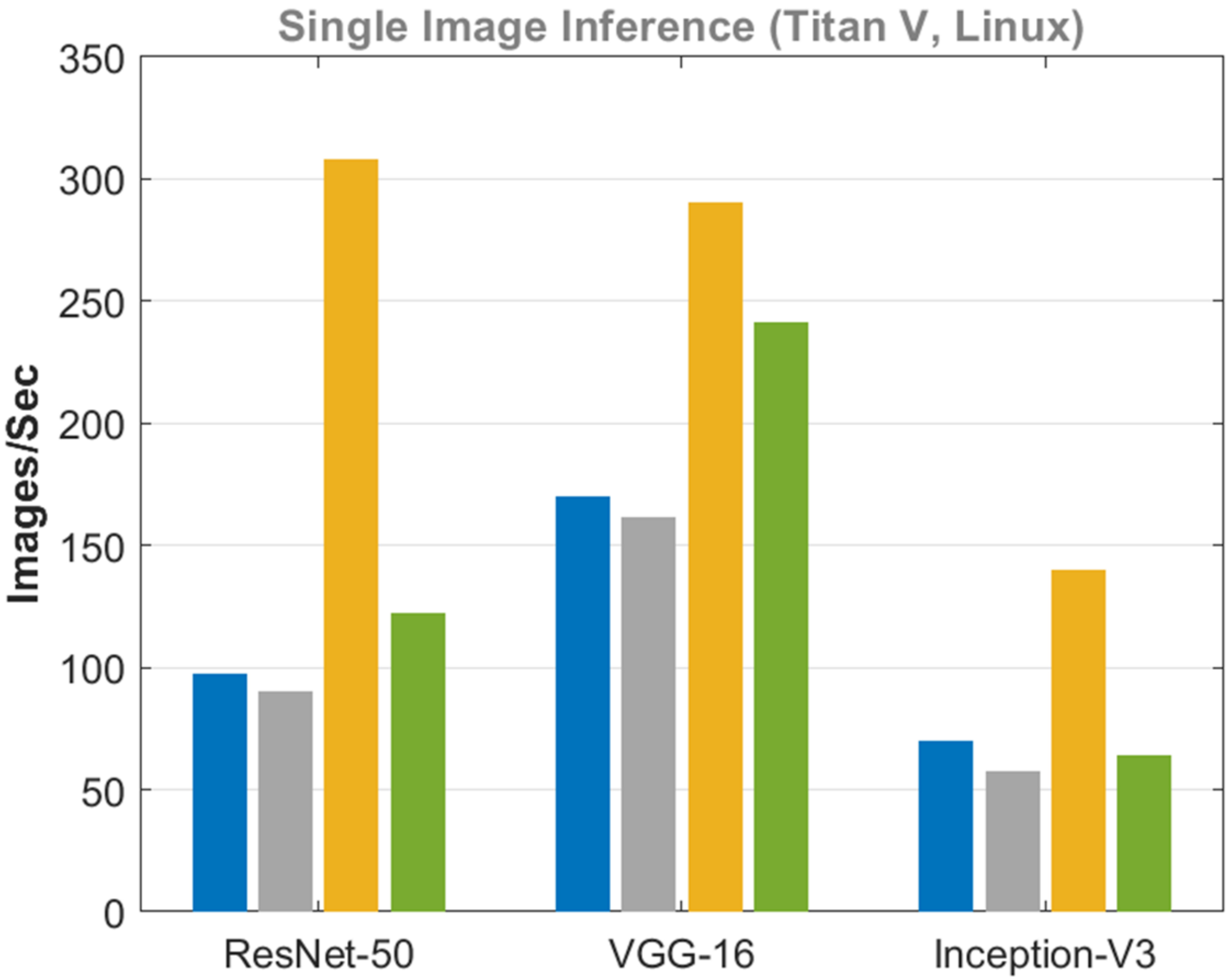
Password: •••••

Build directory: /home/buildDri

□ Generate code only



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

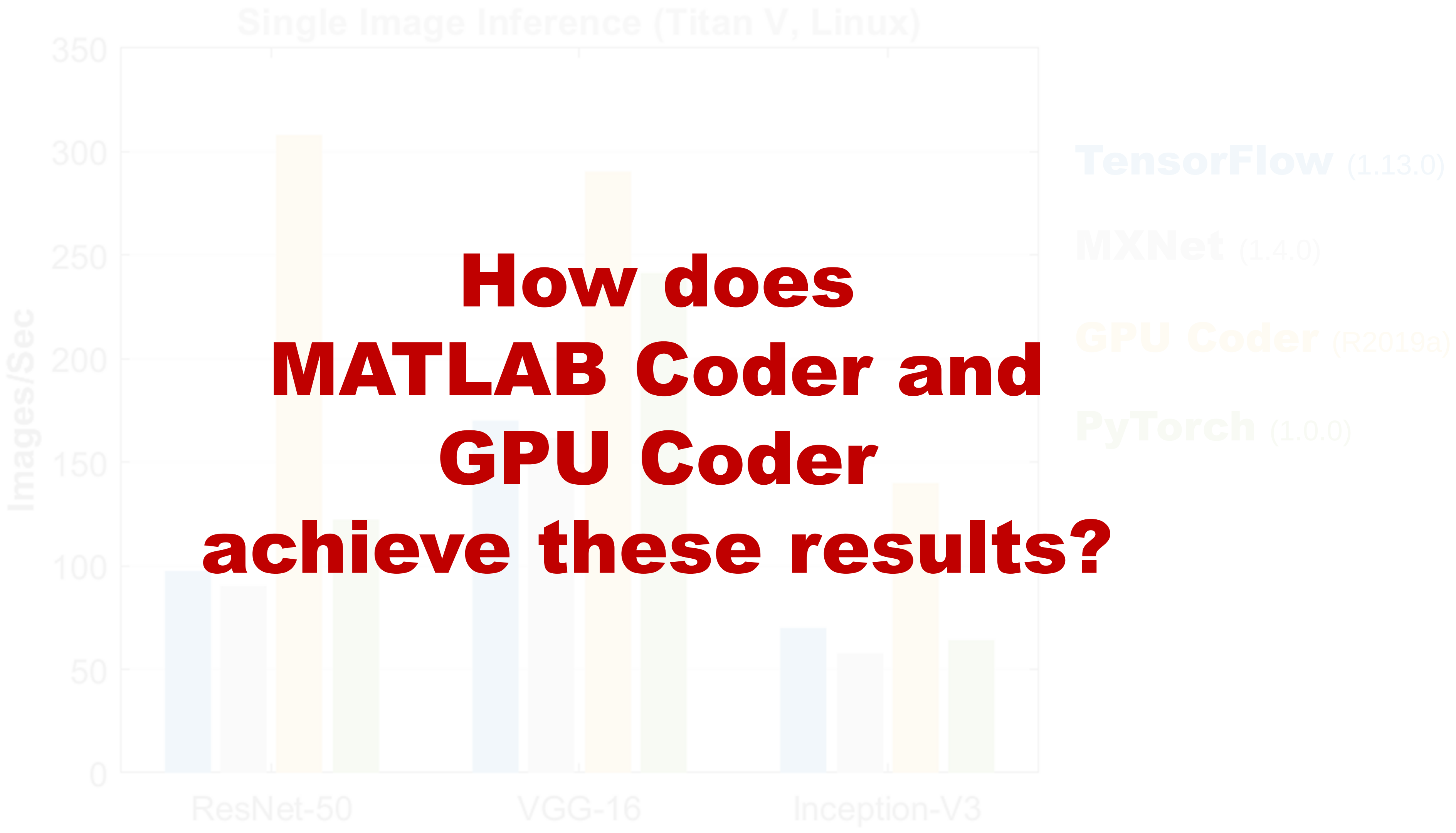


TensorFlow (1.13.0)

MXNet (1.4.0)

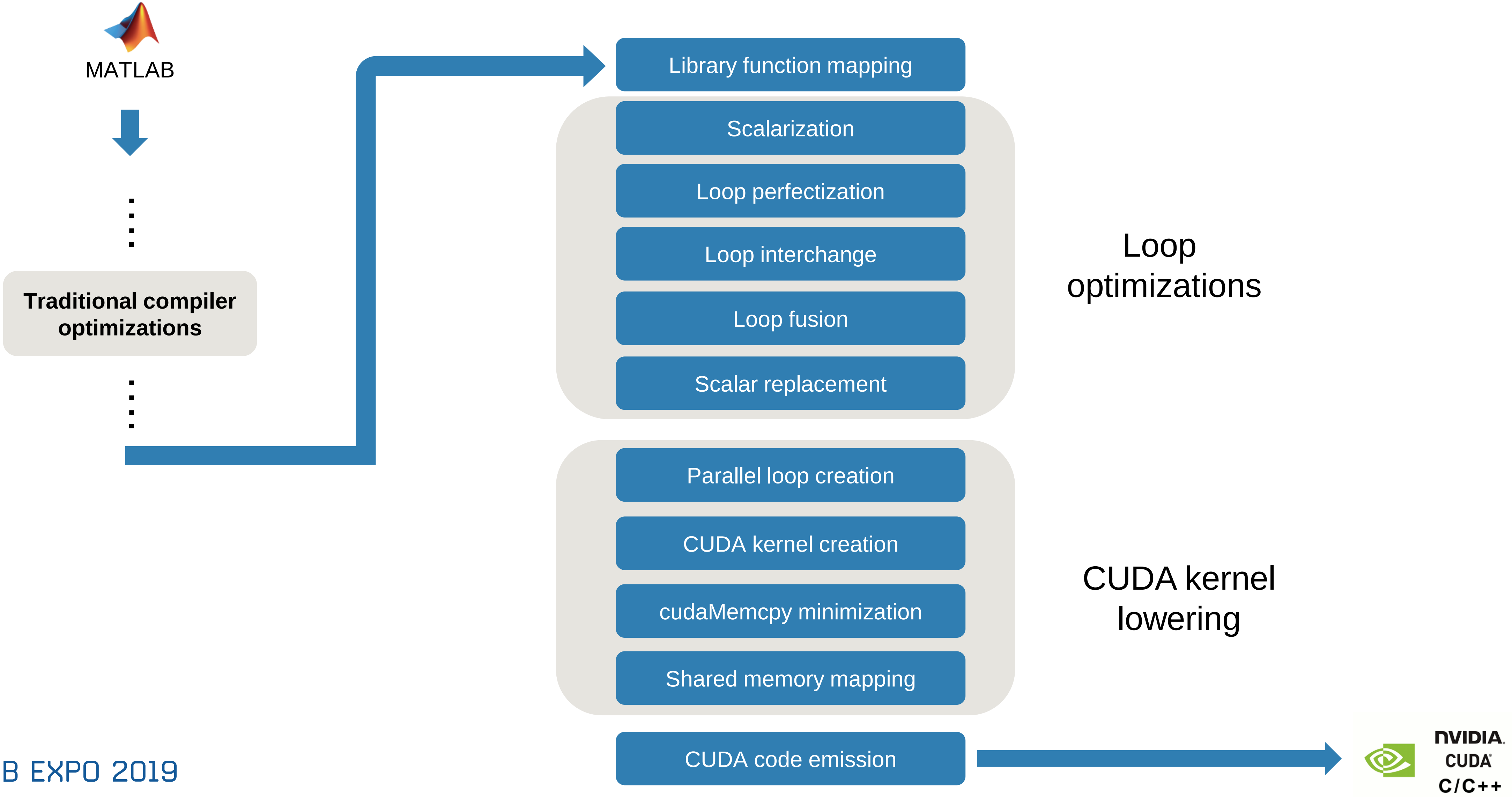
GPU Coder (R2019a)

PyTorch (1.0.0)



How does MATLAB Coder and GPU Coder achieve these results?

Coders Apply Various Optimizations



MATLAB

Traditional compiler optimizations

Library function mapping

Scalarization

Loop perfectization

Loop interchange

Loop fusion

Scalar replacement

Loop optimizations

Parallel loop creation

CUDA kernel creation

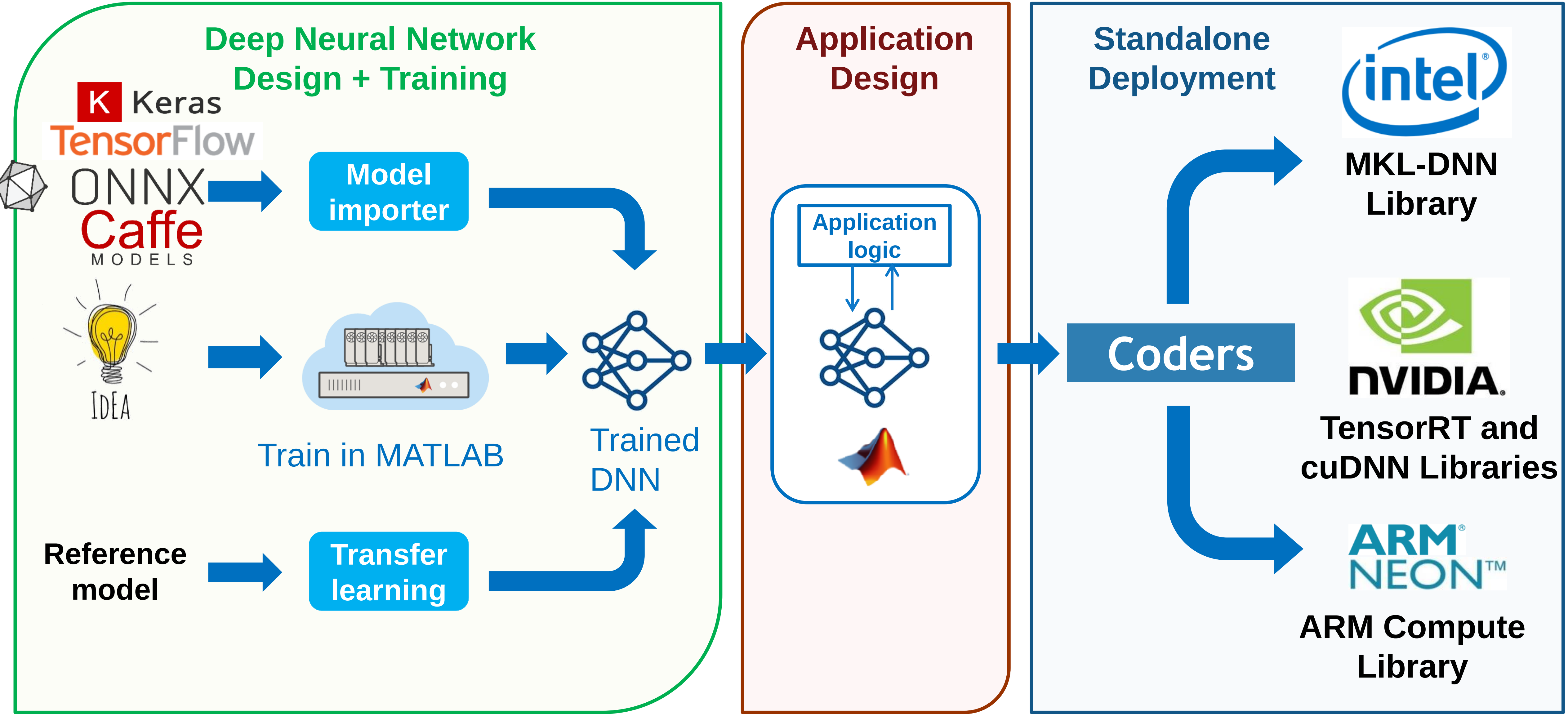
cudaMemcpy minimization

Shared memory mapping

CUDA kernel lowering

CUDA code emission

Deep Learning Workflow in MATLAB



MathWorks® | *Training Services*

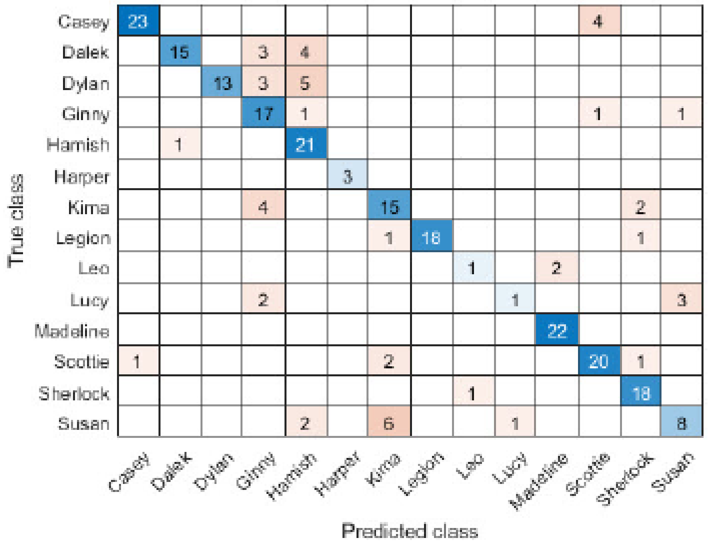
Deep Learning with MATLAB

This two-day course provides a comprehensive introduction to practical deep learning using MATLAB®.

Topics include:

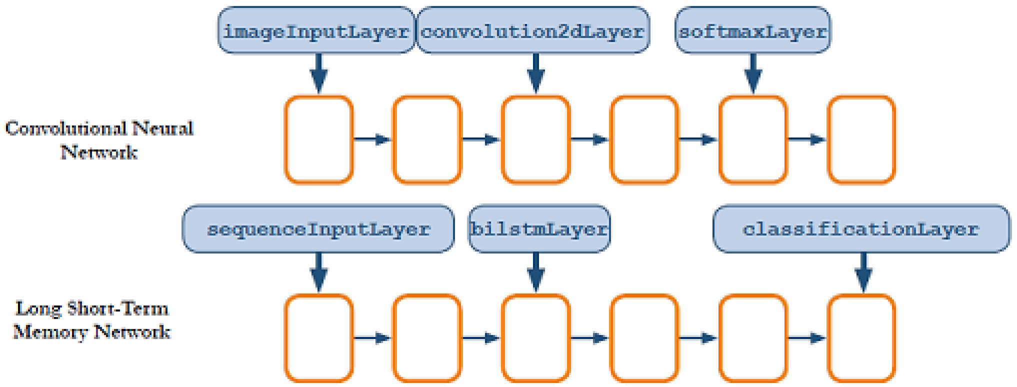
- Importing image and sequence data
- Using convolutional neural networks for image classification, regression, and object detection
- Using long short-term memory networks for sequence classification and forecasting
- Modifying common network architectures to solve custom problems
- Improving the performance of a network by modifying training options

Transfer Learning



	Casey	Dalek	Dylan	Ginny	Hamish	Harper	Kima	Legion	Leo	Lucy	Madeline	Scottie	Sherlock	Susan
Casey	23											4		
Dalek		15		3	4									
Dylan			13	3	5									
Ginny				17	1							1		1
Hamish		1			21									
Harper						3								
Kima				4			15						2	
Legion							1	18					1	
Leo									1		2			
Lucy				2						1				3
Madeline											22			
Scottie	1						2					20	1	
Sherlock									1				18	
Susan					2		6			1				8

Classifying Sequence Data



MATLAB EXPO 2019

Email: rishu.g@mathworks.com,

LinkedIn: https://www.linkedin.com/in/rishu-gupta-72148914/

Please provide feedback for this block of sessions

- Scan this QR Code or log onto link below

  (link also sent to your phone and email)
- http://bit.ly/expo19-feedback
- Enter the registration id number displayed on your badge
- Provide feedback for this session

Email: rishu.g@mathworks.com,

LinkedIn: https://www.linkedin.com/in/rishu-gupta-72148914/