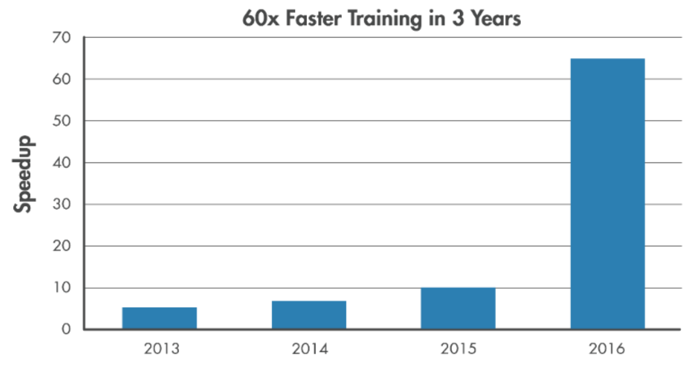
MATLAB EXPO 2018

Deploying Deep Learning Networks to Embedded GPUs and CPUs

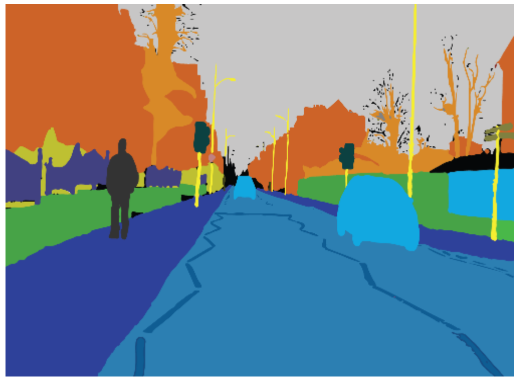
Pierre Nowodzienski

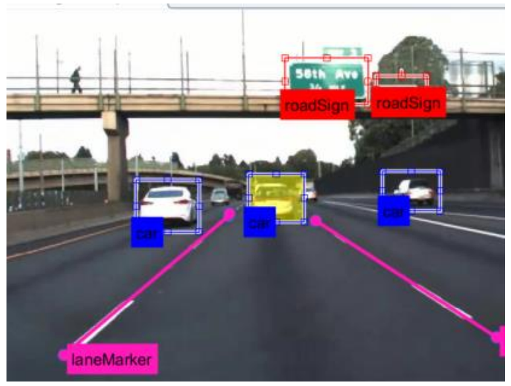
Deep Learning enablers

Increased GPU acceleration



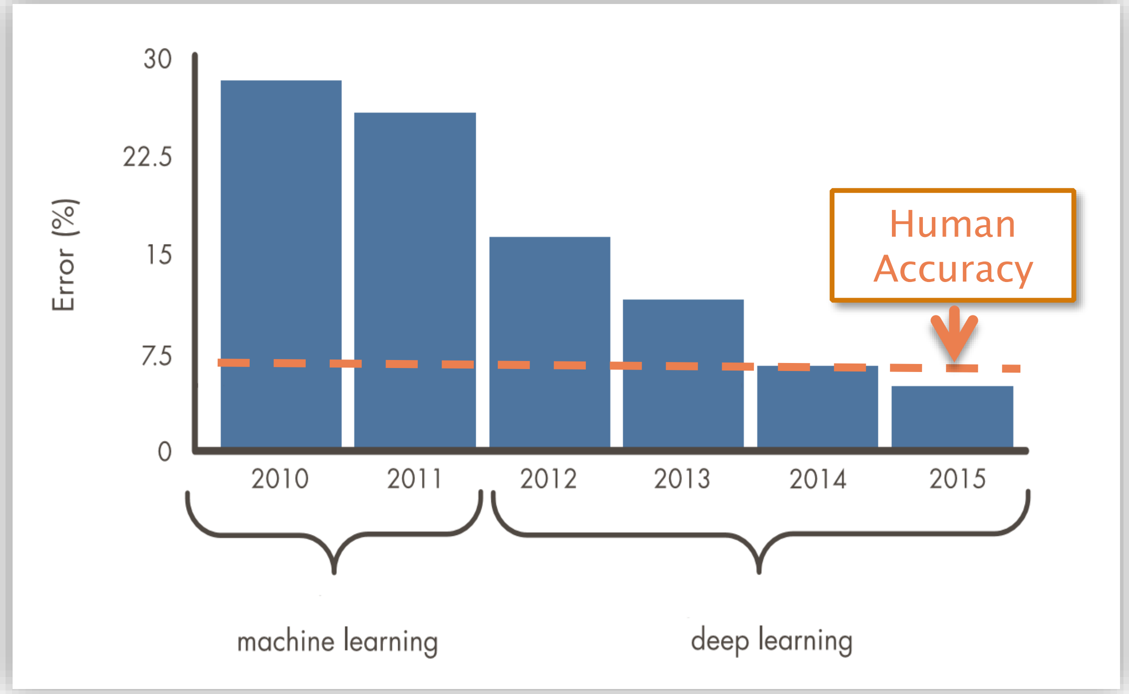
Labeled public datasets



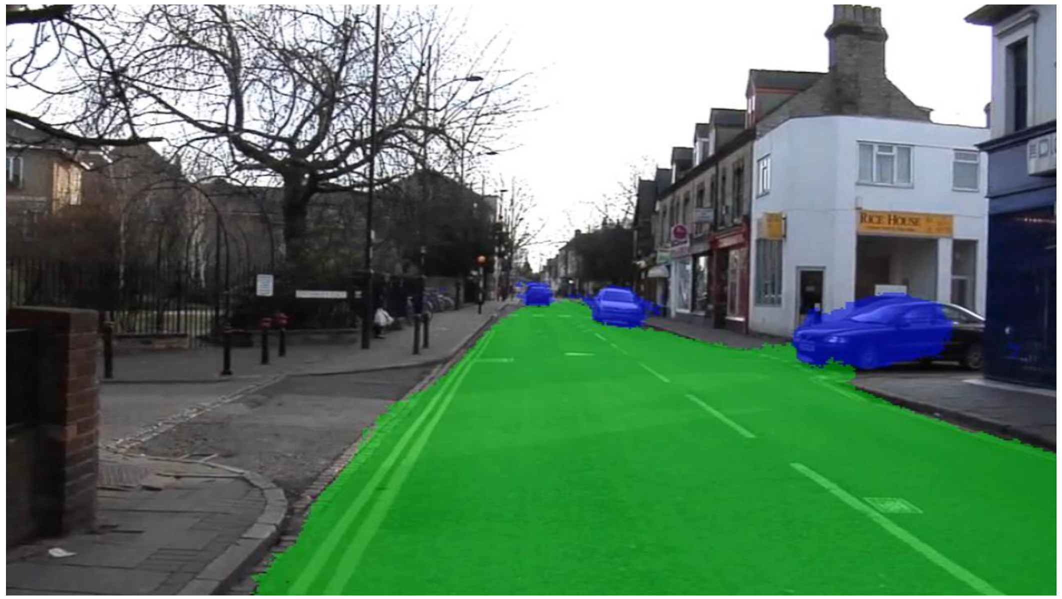


World-class models to be leveraged

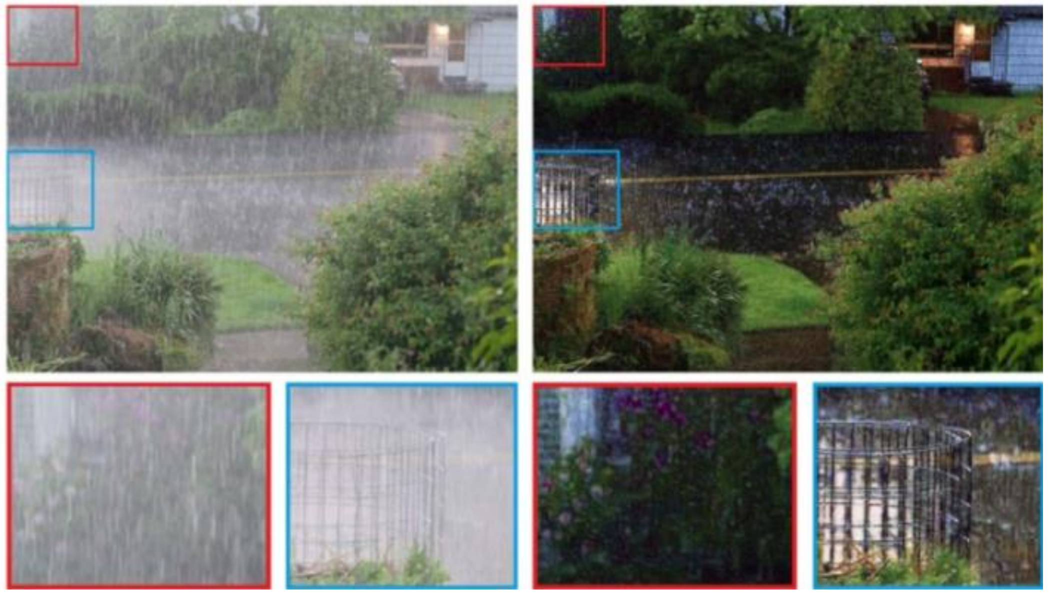
AlexNet PRETRAINED MODEL	VGG-16 PRETRAINED MODEL	ResNet PRETRAINED MODEL
Caffe MODELS	GoogLeNet PRETRAINED MODEL	TensorFlow/Keras MODELS



Deep Learning Applications:
Image classification, speech recognition, autonomous driving, etc…

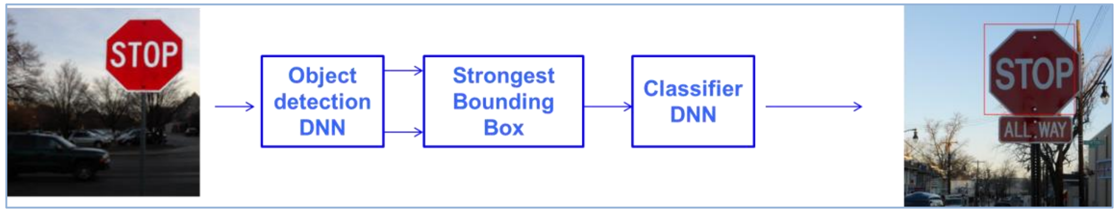


Detection of cars and road in autonomous driving systems



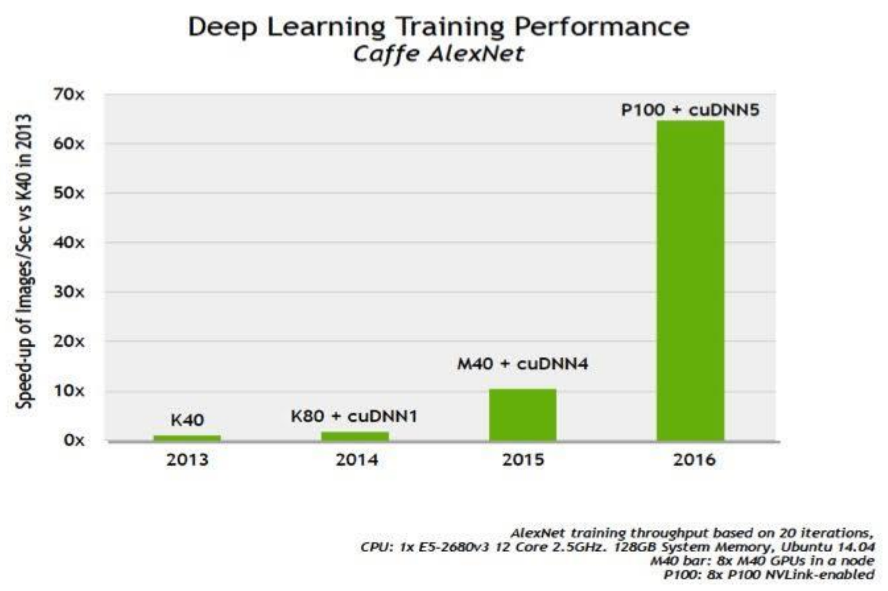
Rain Detection and Removal[1]

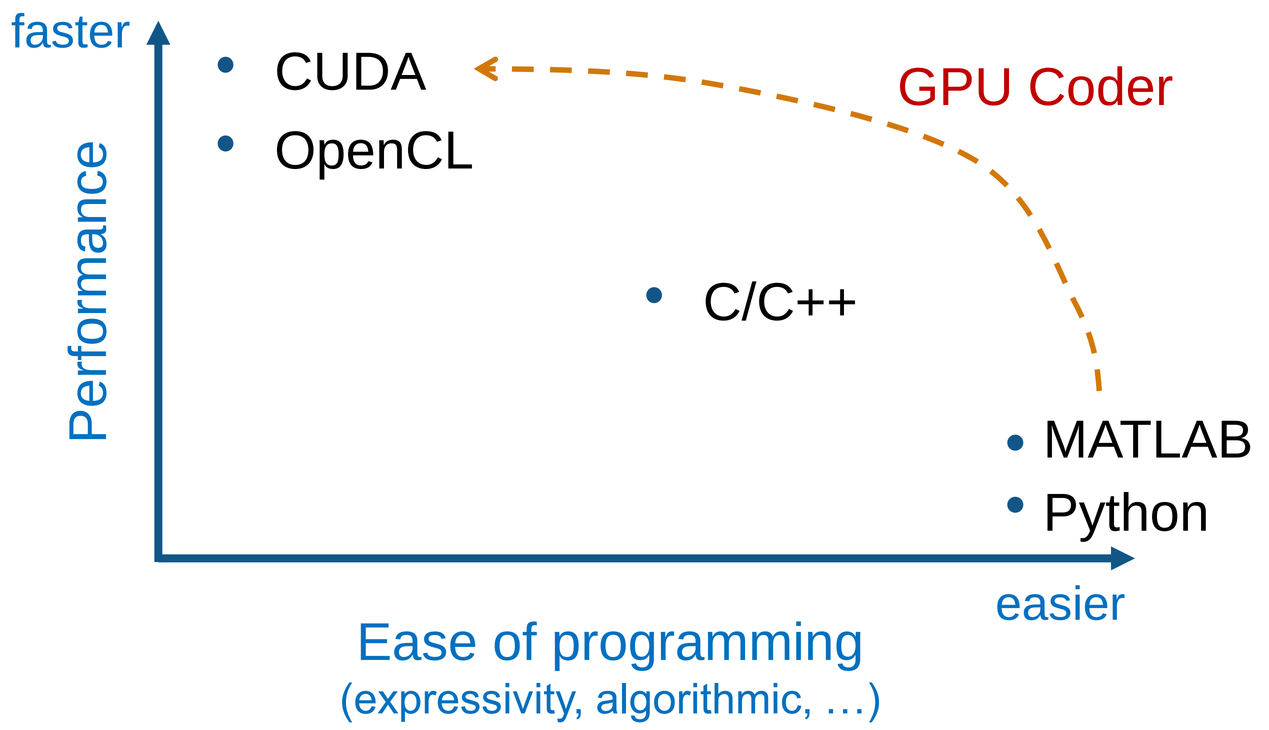
1. *Deep Joint Rain Detection and Removal from a Single Image*



Traffic Sign Recognition

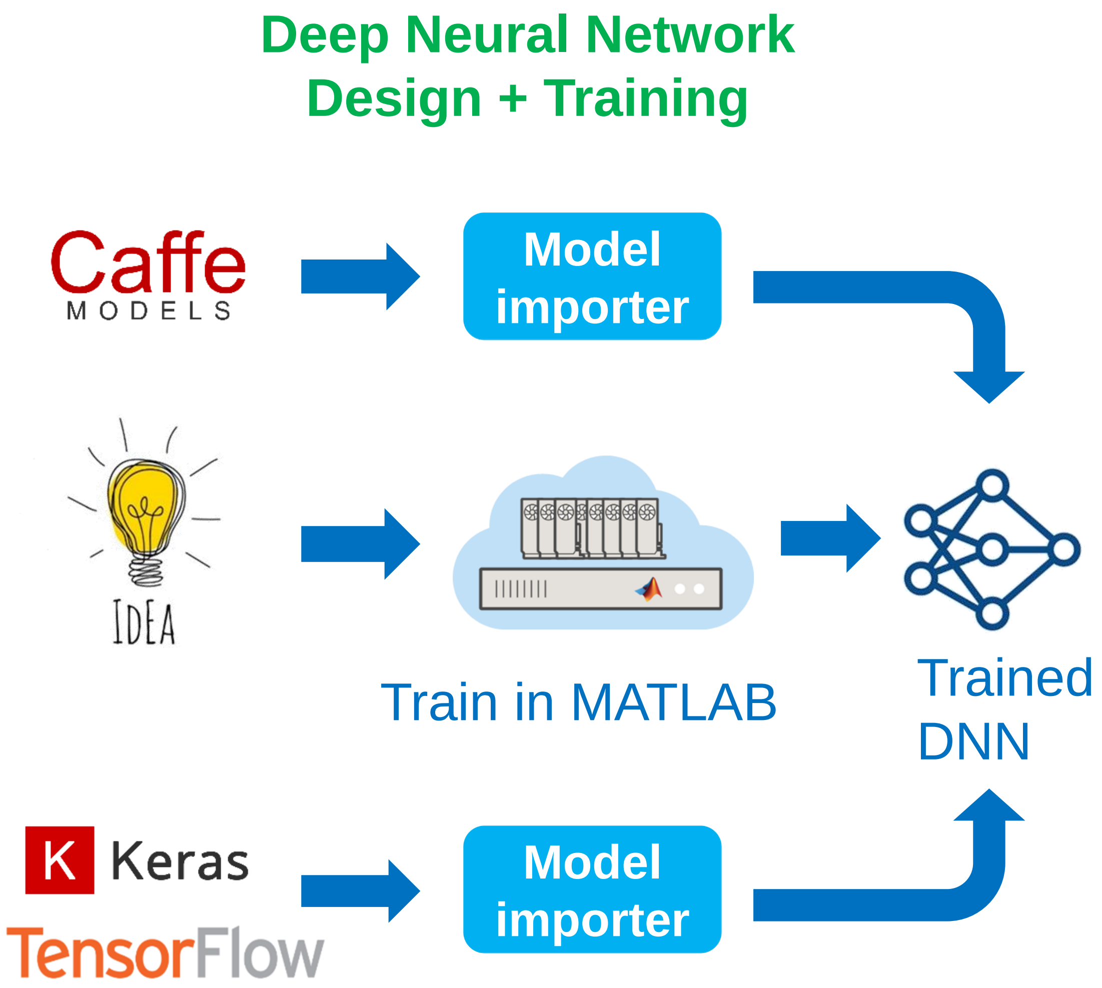
GPUs and CUDA programming





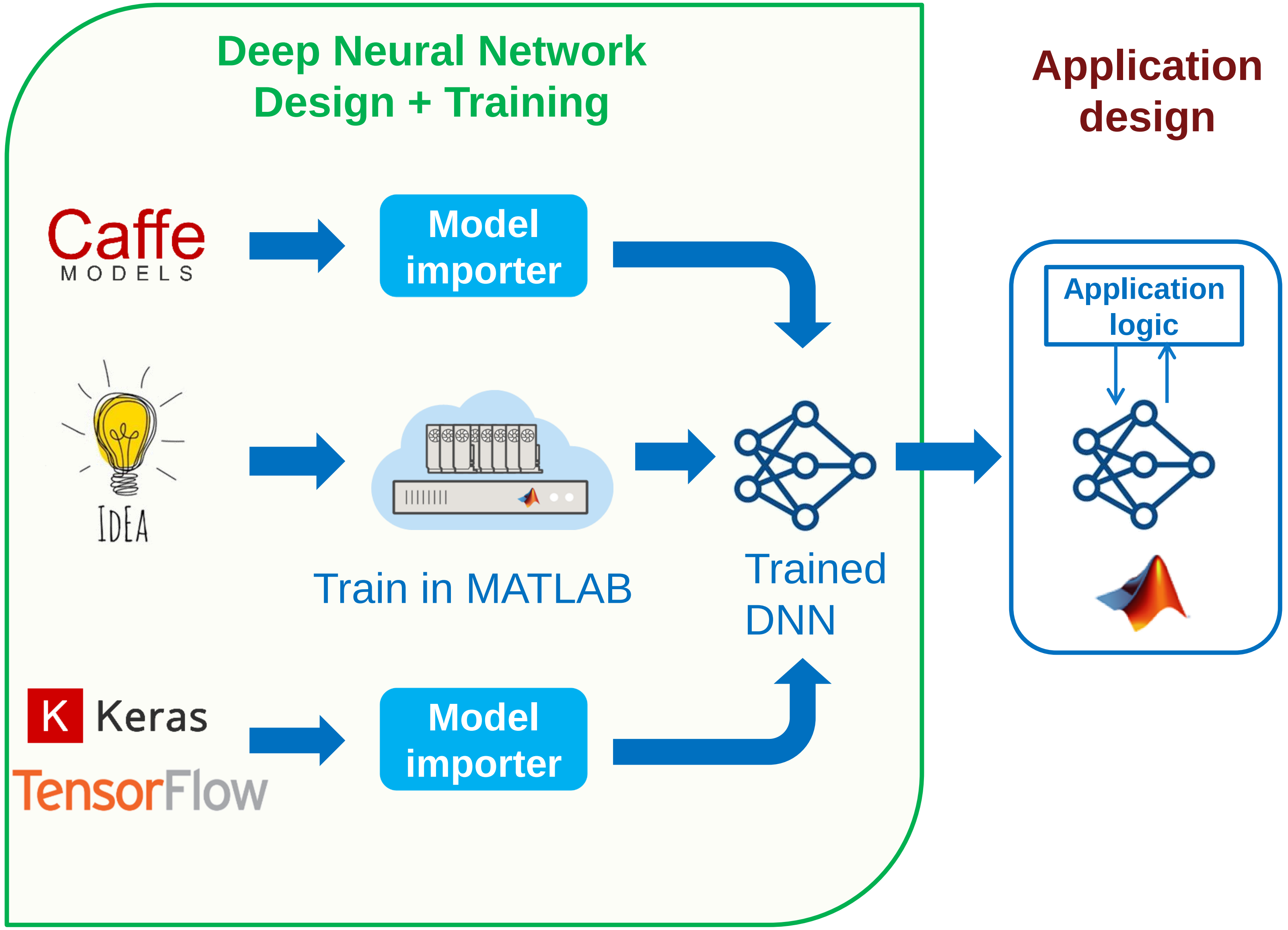
GPUs are "hardware on steroids", … but, programming them is hard

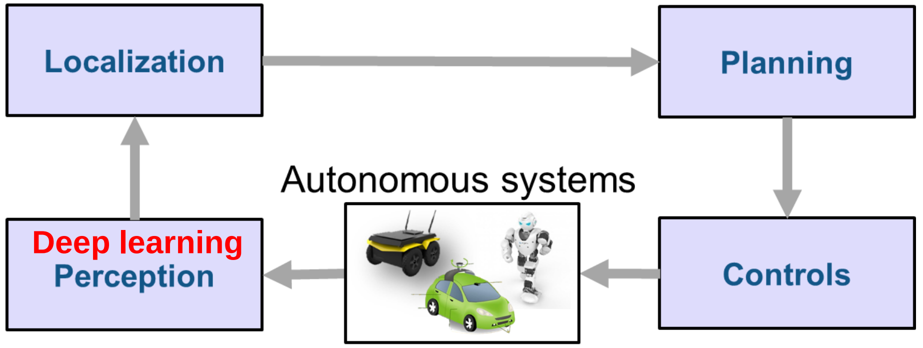
Deep learning workflow in MATLAB



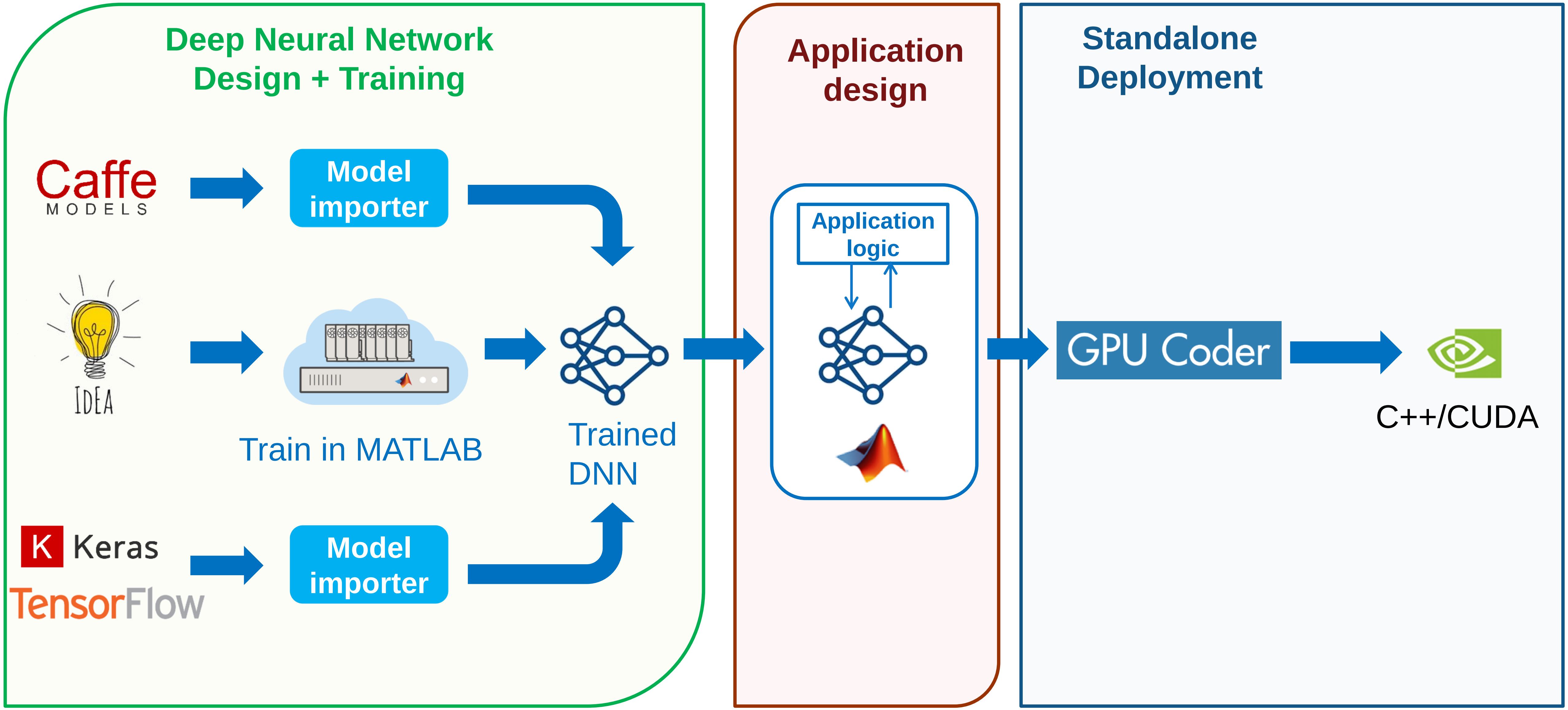
- **Design in MATLAB**
  - **Manage** large data sets
  - **Automate** data labeling
  - **Easy access** to models
- **Training in MATLAB**
  - **Acceleration** with GPU's
  - **Scale** to clusters

Deep learning workflow in MATLAB





Deep learning workflow in MATLAB



MathWorks

GPU Coder for Deployment

MATLAB

GPU Coder

Accelerated implementation of parallel algorithms on GPUs & CPUs

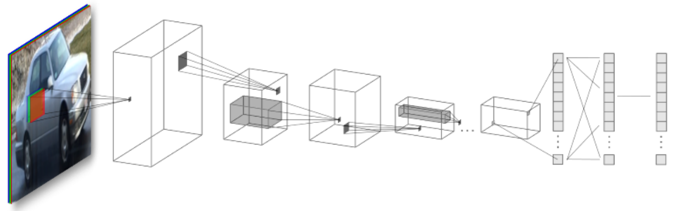
Intel MKL-DNN Library

NVIDIA CUDA C/C++

ARM Compute Library

Deep Neural Networks

Deep Learning, machine learning



5x faster than TensorFlow
2x faster than MXNet

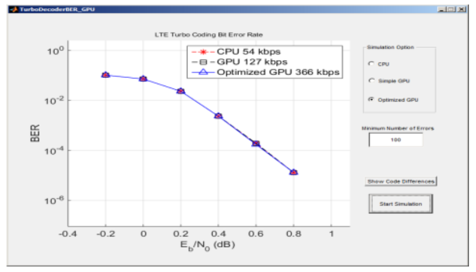
Image Processing and Computer Vision

Image filtering, feature detection/extraction

60x faster than CPUs for stereo disparity

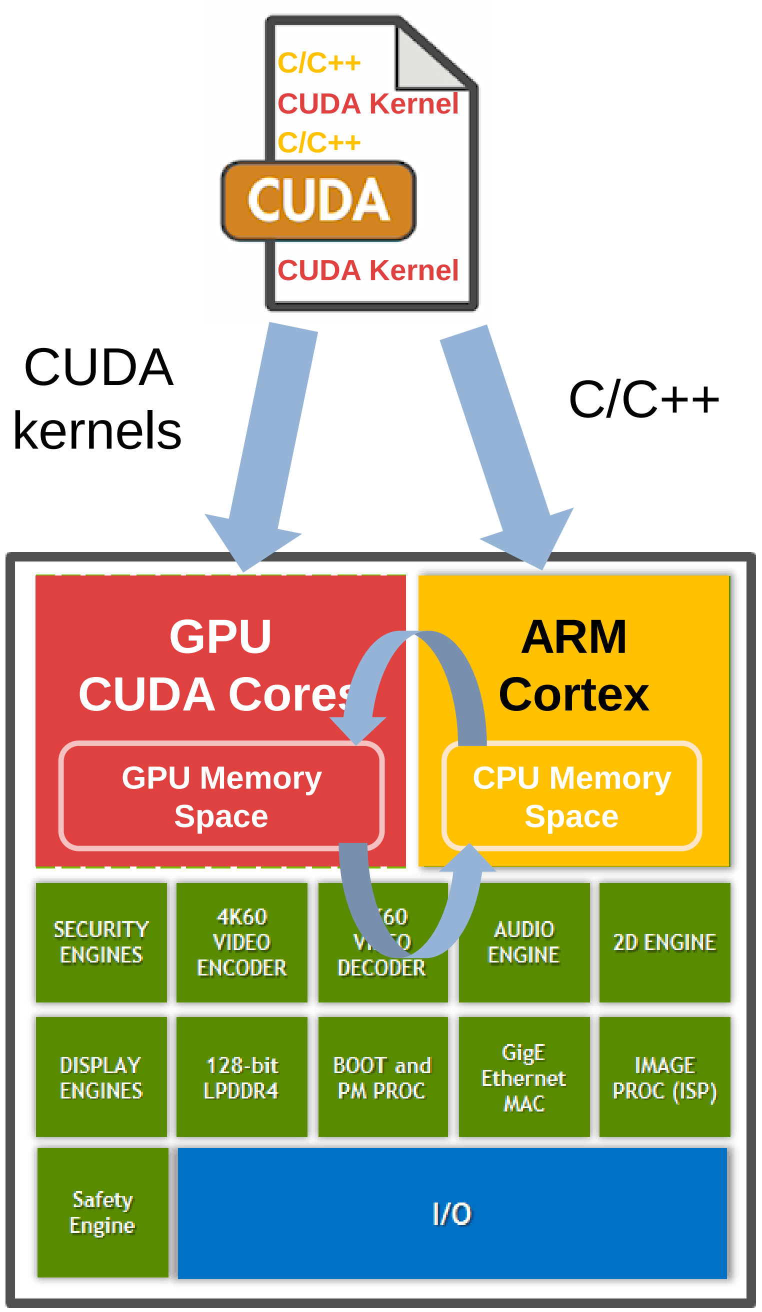
Signal Processing and Communications

FFT, filtering, cross correlation,



20x faster than CPUs for FFTs

GPUs and CUDA



Challenges of Programming in CUDA for GPUs

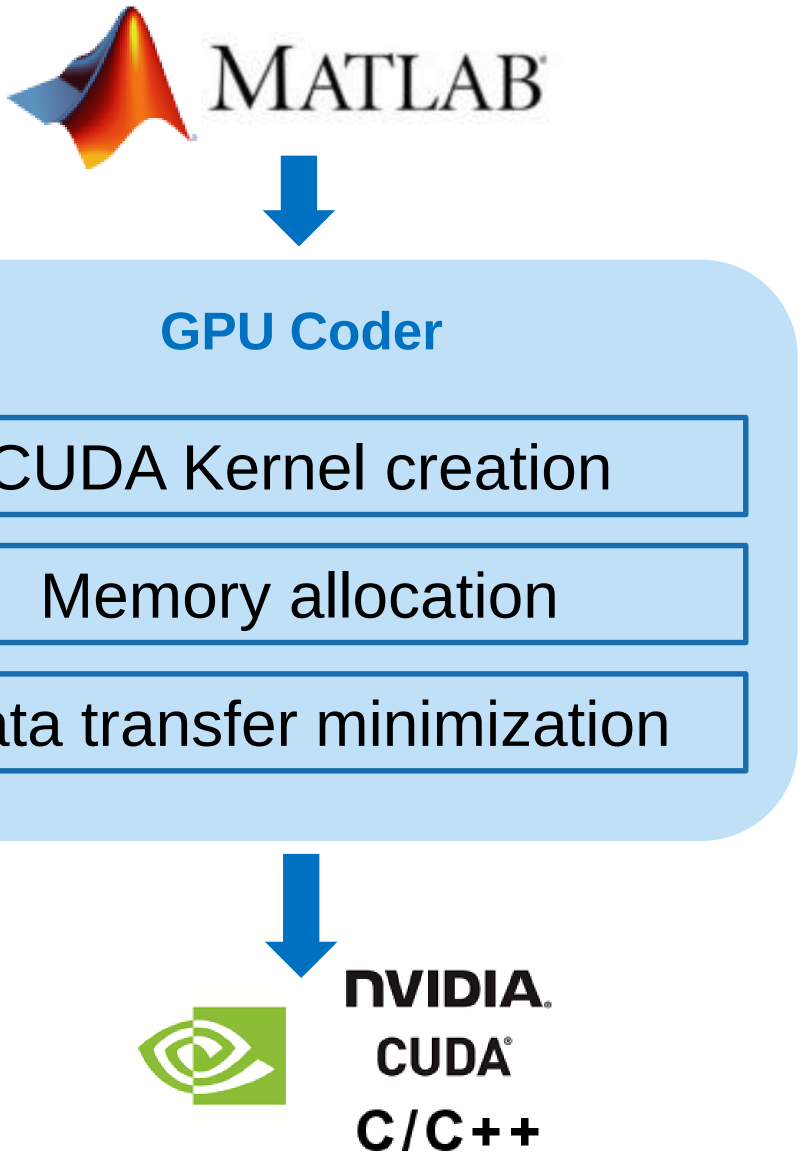
- Learning to program in CUDA
  - Need to rewrite algorithms for parallel processing paradigm
- Creating CUDA kernels
  - Need to analyze algorithms to create CUDA kernels that maximize parallel processing
- Allocating memory
  - Need to deal with memory allocation on both CPU and GPU memory spaces
- Minimizing data transfers
  - Need to minimize while ensuring required data transfers are done at the appropriate parts of your algorithm

GPU Coder Helps You Deploy to GPUs Faster

- Library function mapping
- Loop optimizations
- Dependence analysis

- Data locality analysis
- GPU memory allocation

- Data-dependence analysis
- Dynamic memcpy reduction



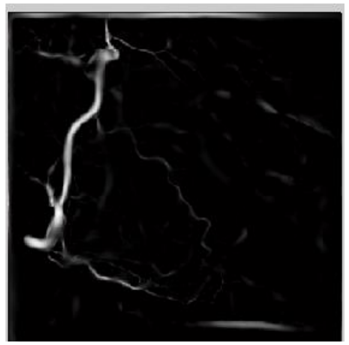
GPU Coder speeds up MATLAB for Image Processing and Computer Vision

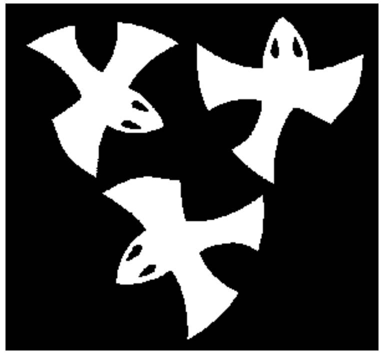
Fog removal

5x speedup

Frangi filter

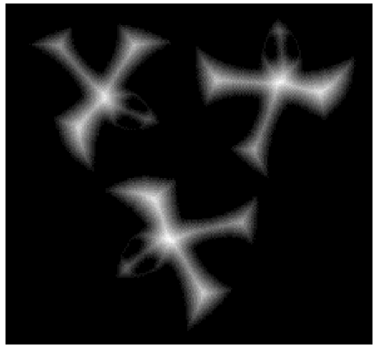
3x speedup





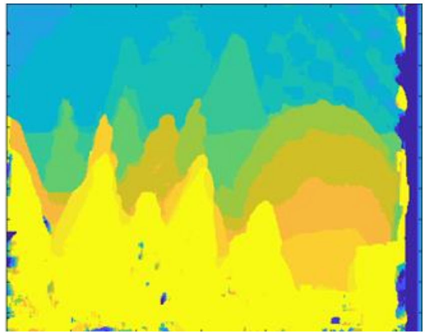
Distance transform

8x speedup



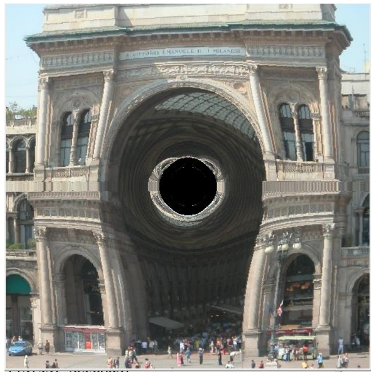
Stereo disparity

50x speedup



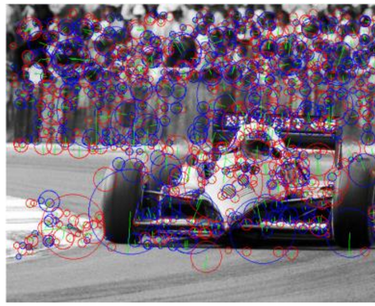
Ray tracing

18x speedup

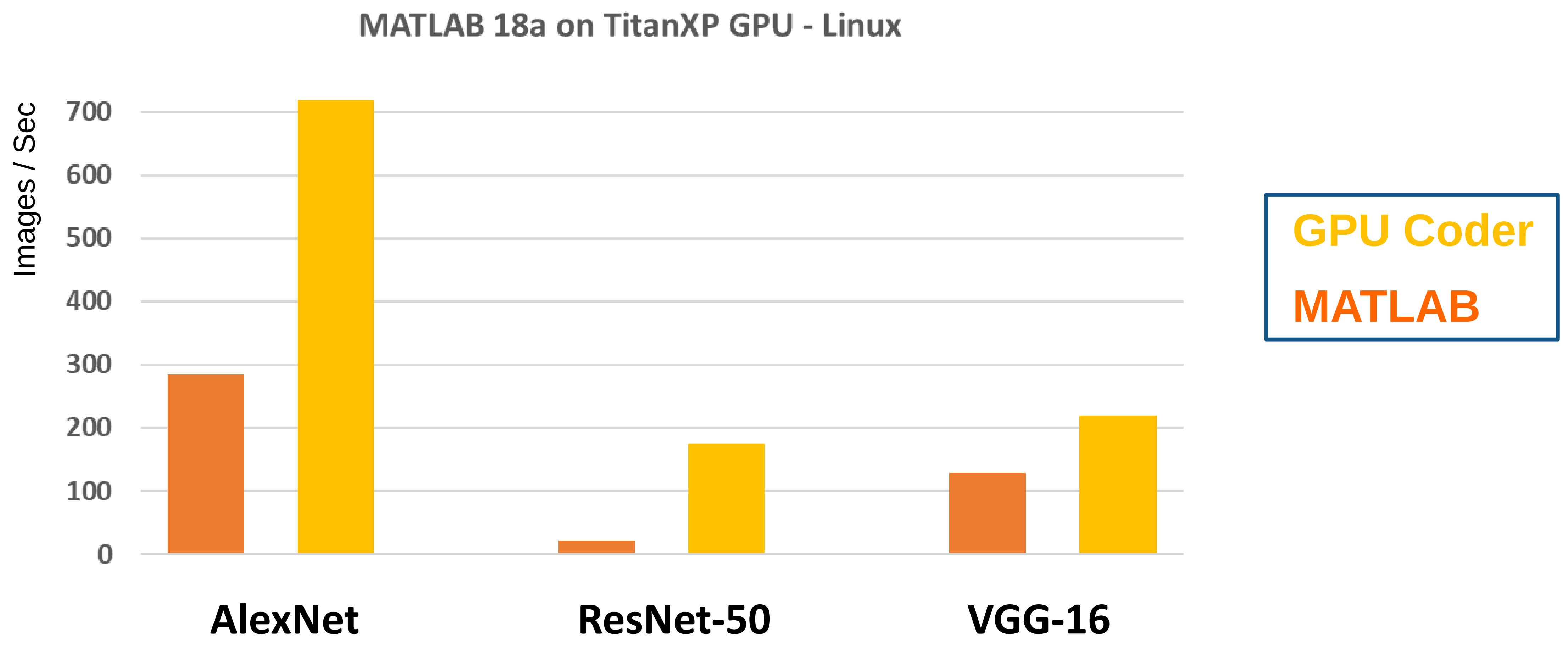


SURF feature extraction

700x speedup

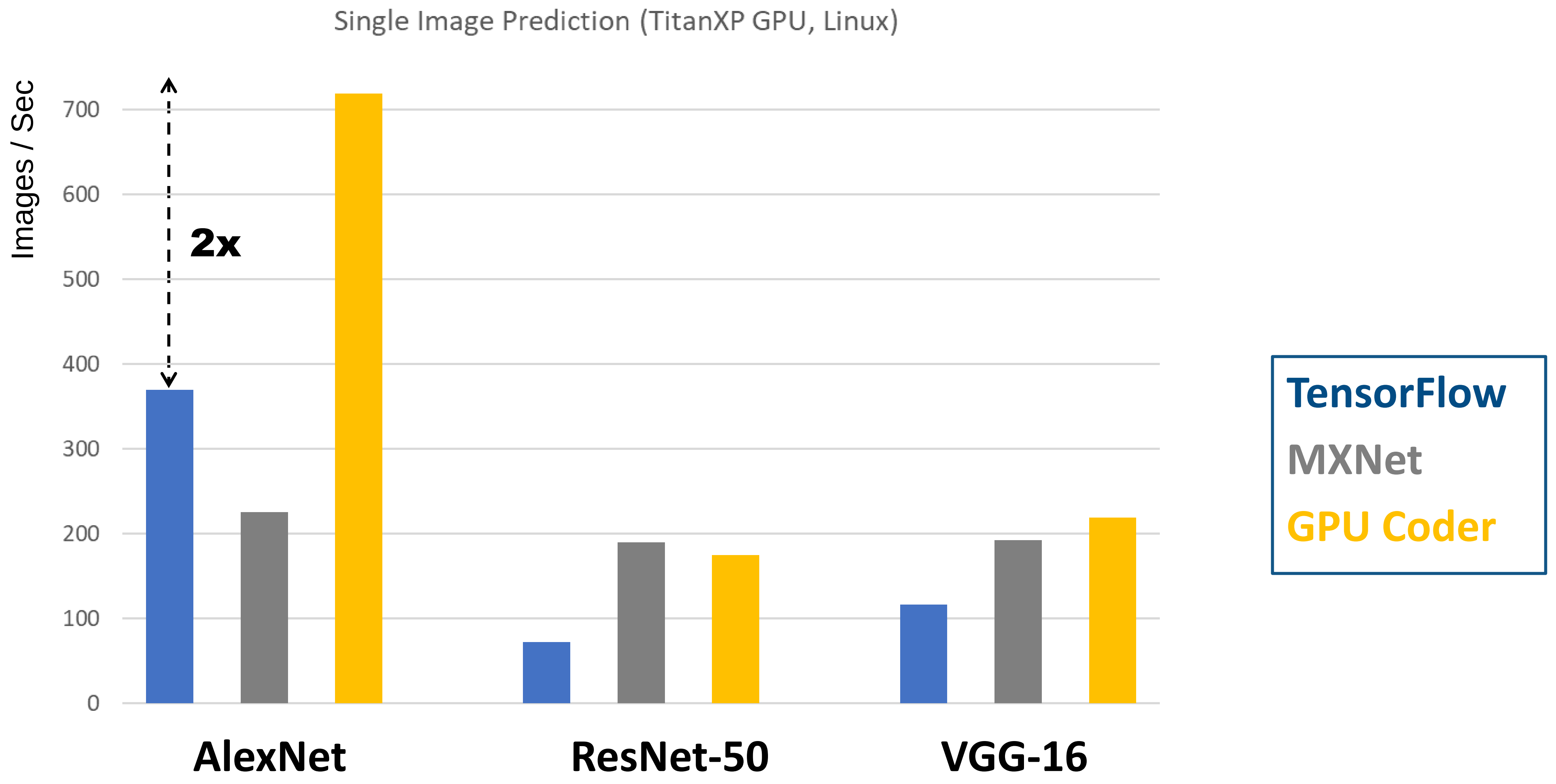


GPU Coder speeds up MATLAB at least 2x for inference



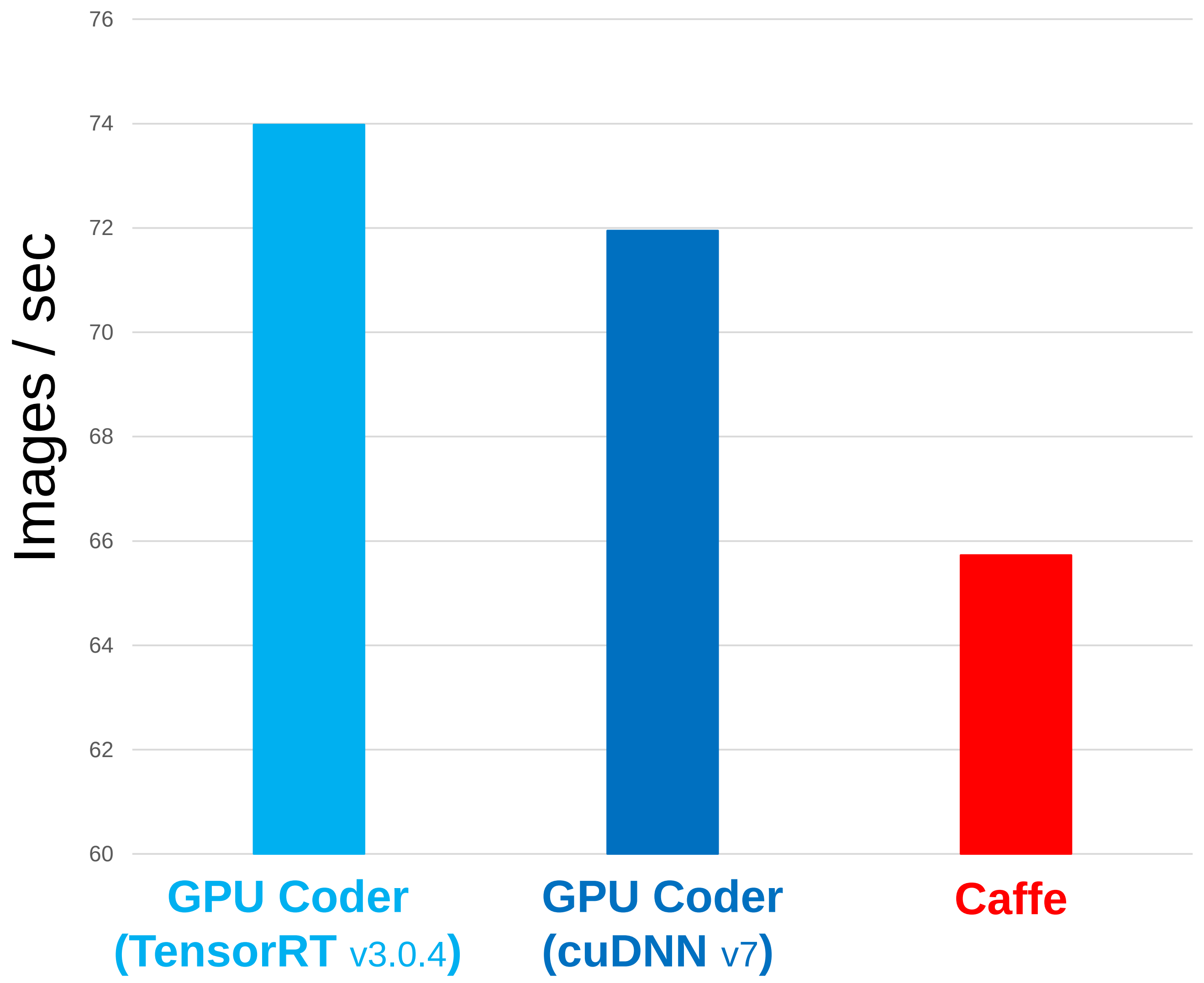
Single image prediction using Intel® Xeon® CPU - 3.6 GHz, NVIDIA libraries: CUDA8 - cuDNN 7
TensorFlow 1.6.0, MXNet 1.1.0, MATLAB 18a

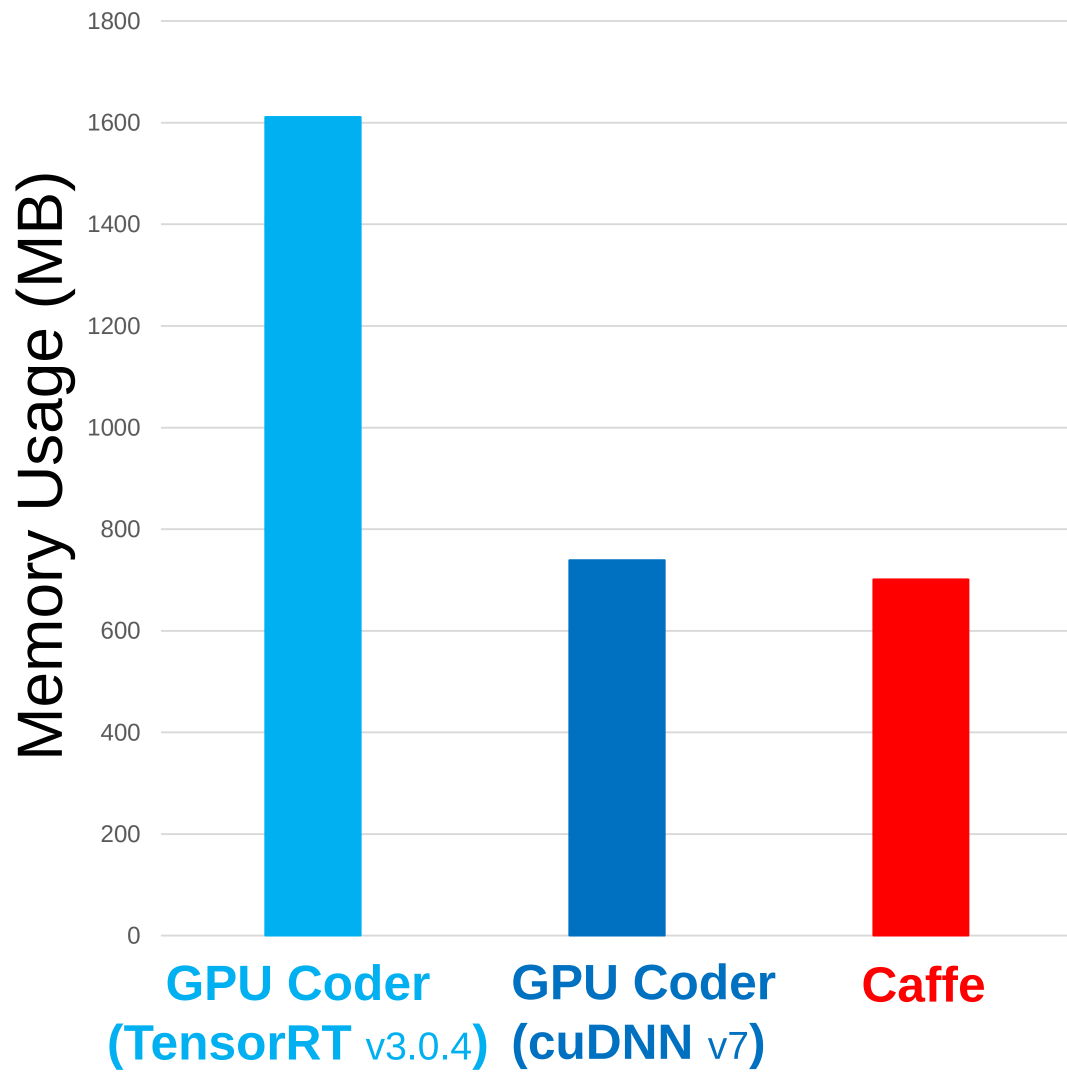
With GPU Coder, MATLAB is faster than other frameworks



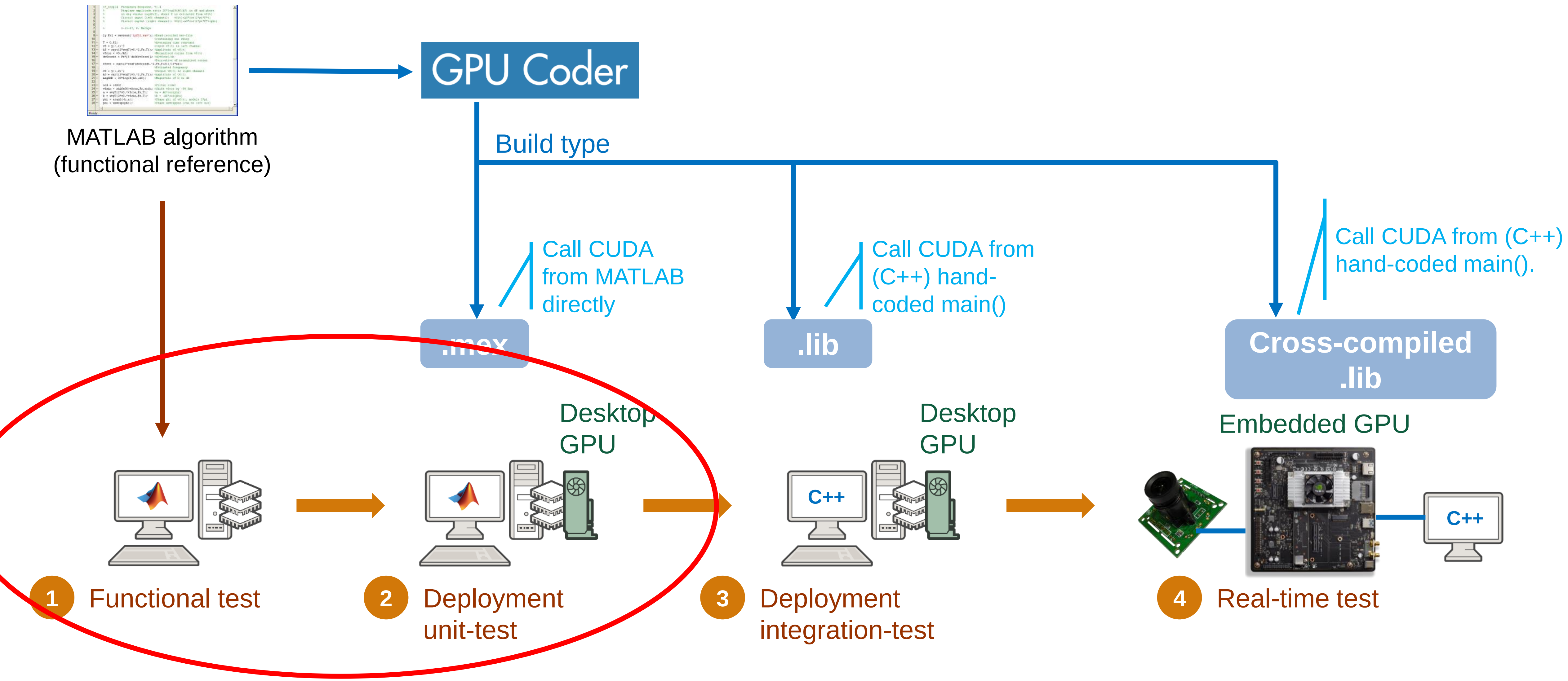
Single image prediction using Intel® Xeon® CPU - 3.6 GHz, NVIDIA libraries: CUDA8 - cuDNN 7
TensorFlow 1.6.0, MXNet 1.1.0, MATLAB 18a

Embedded GPU Benchmarking: Jetson TX2

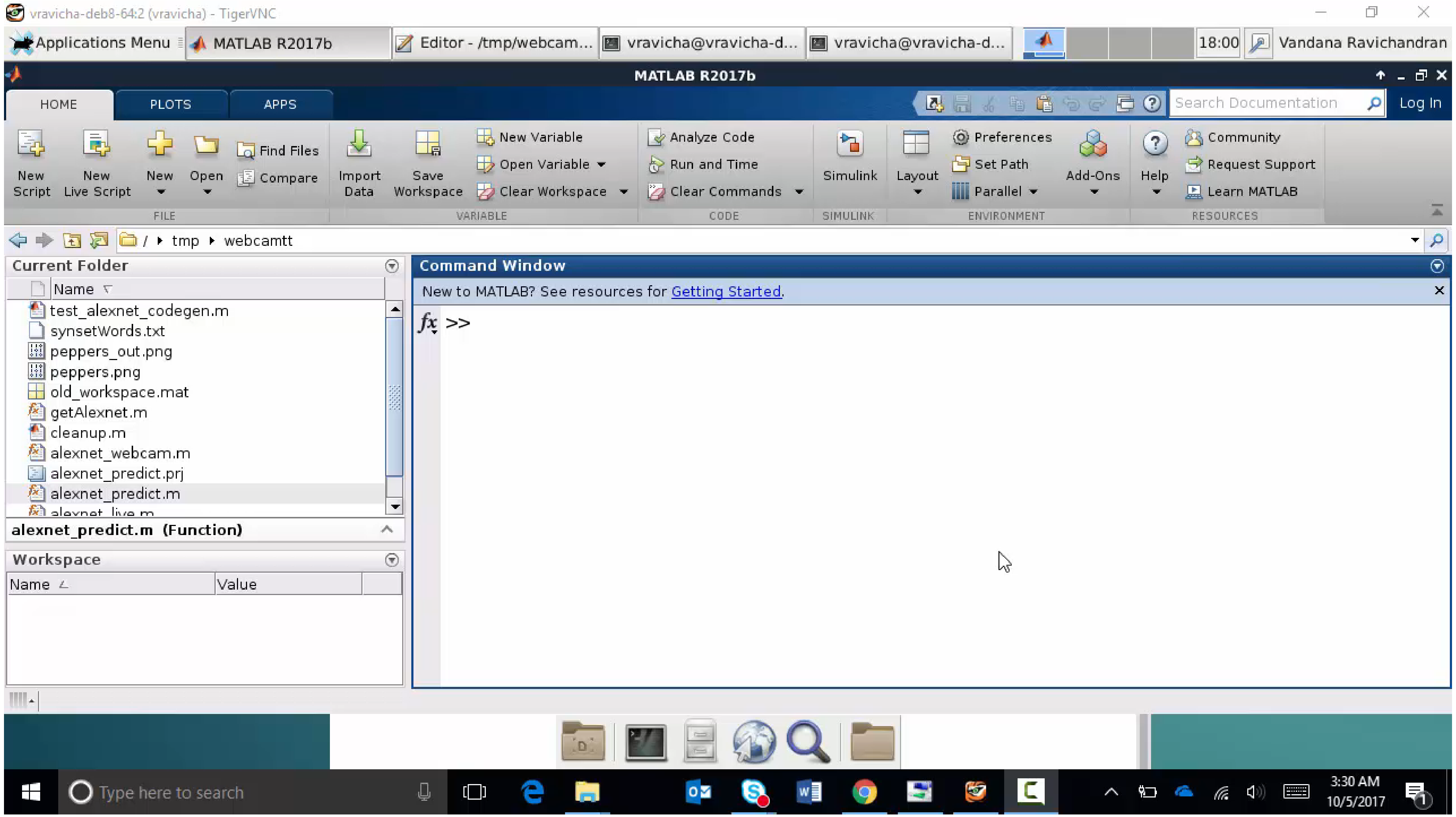




Algorithm Design to Embedded Deployment Workflow



Demo: Alexnet Deployment with 'mex' Code Generation



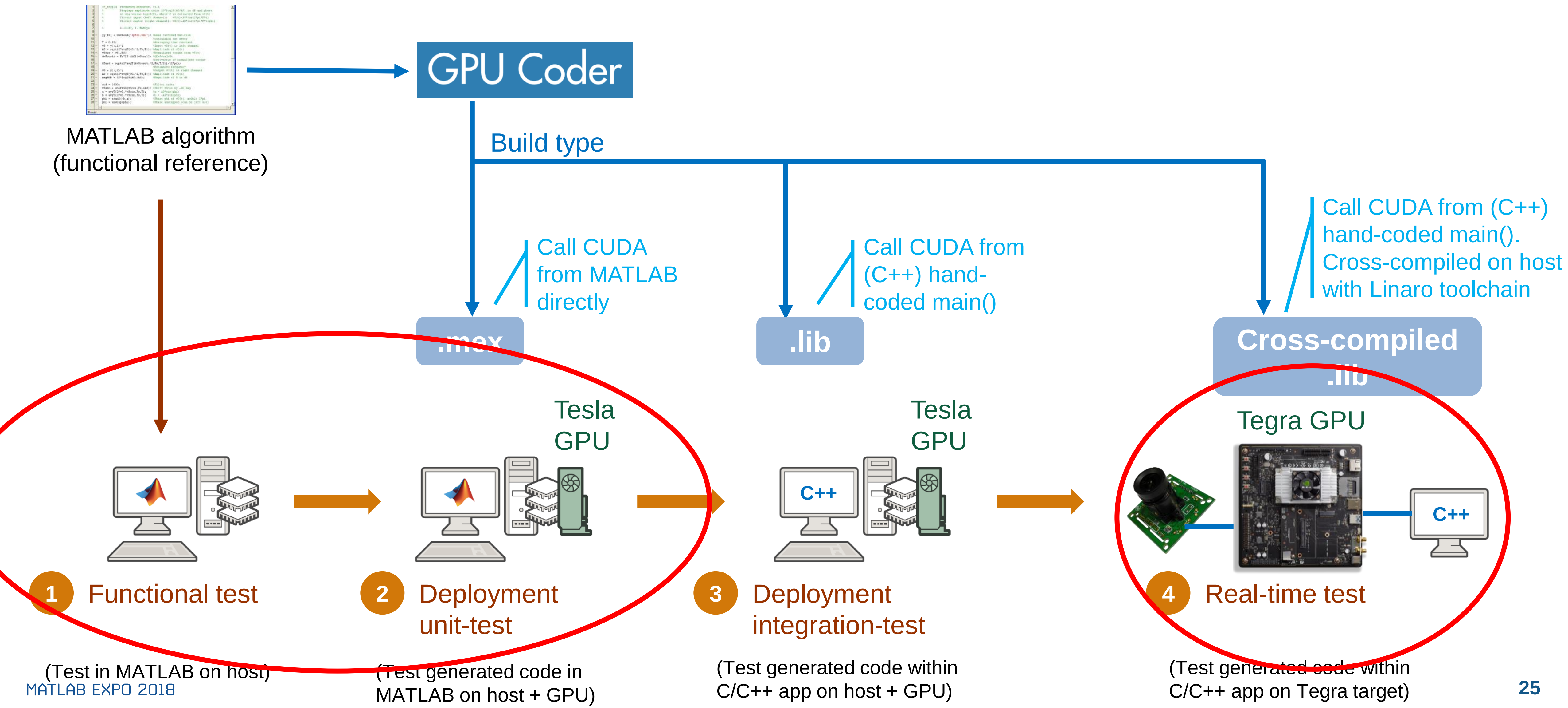
```
test_alexnet_codegen.m
synsetWords.txt
peppers_out.png
peppers.png
old_workspace.mat
getAlexnet.m
cleanup.m
alexnet_webcam.m
alexnet_predict.prj
alexnet_predict.m
alexnet_live.m
```

alexnet_predict.m (Function)

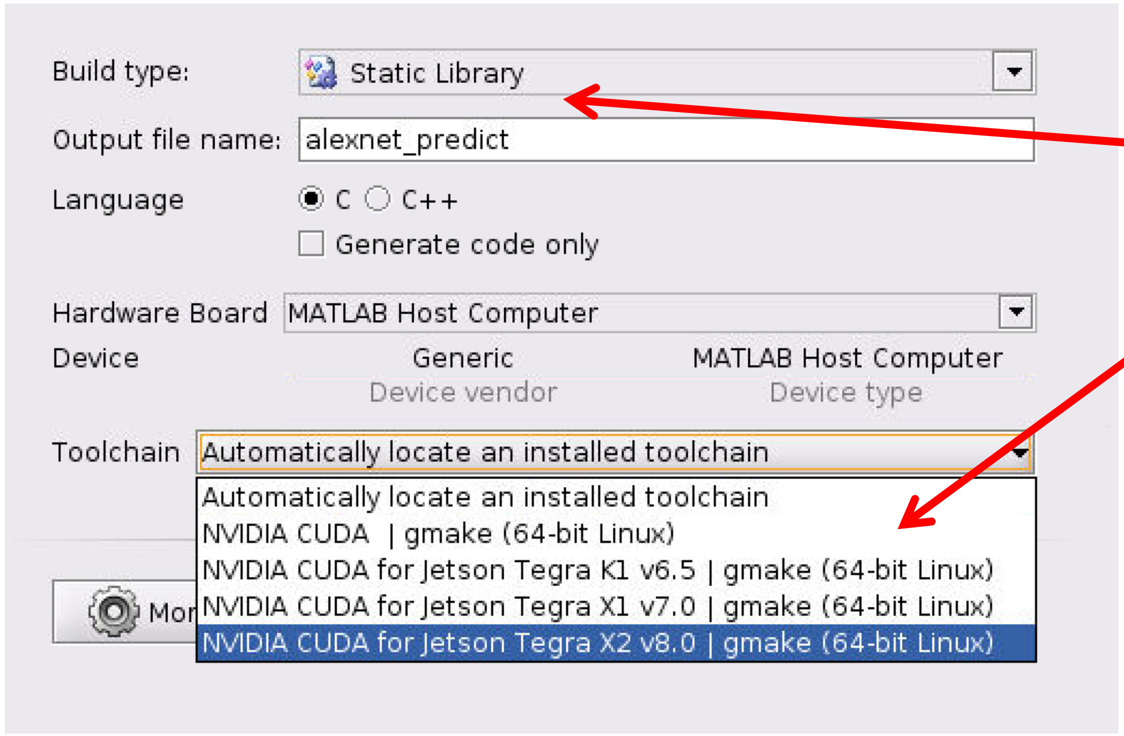
New to MATLAB? See resources for Getting Started.

```
>>
```

Algorithm Design to Embedded Deployment on Tegra GPU


MATLAB algorithm
(functional reference)
GPU Coder
Build type
Call CUDA from MATLAB directly
Call CUDA from (C++) hand-coded main()
Call CUDA from (C++) hand-coded main(). Cross-compiled on host with Linaro toolchain
.mex
.lib
Cross-compiled .lib
Tesla GPU
Tesla GPU
Tegra GPU
C++
C++
1 Functional test
2 Deployment unit-test
3 Deployment integration-test
4 Real-time test
(Test in MATLAB on host)
(Test generated code in MATLAB on host + GPU)
(Test generated code within C/C++ app on host + GPU)
(Test generated code within C/C++ app on Tegra target)

Alexnet Deployment to Tegra: Cross-Compiled with ‘lib’



Build type: Static Library

Output file name: alexnet_predict

Language [✓ C] C++

☐ Generate code only

Hardware Board: MATLAB Host Computer

Device	Generic	MATLAB Host Computer
	Device vendor	Device type

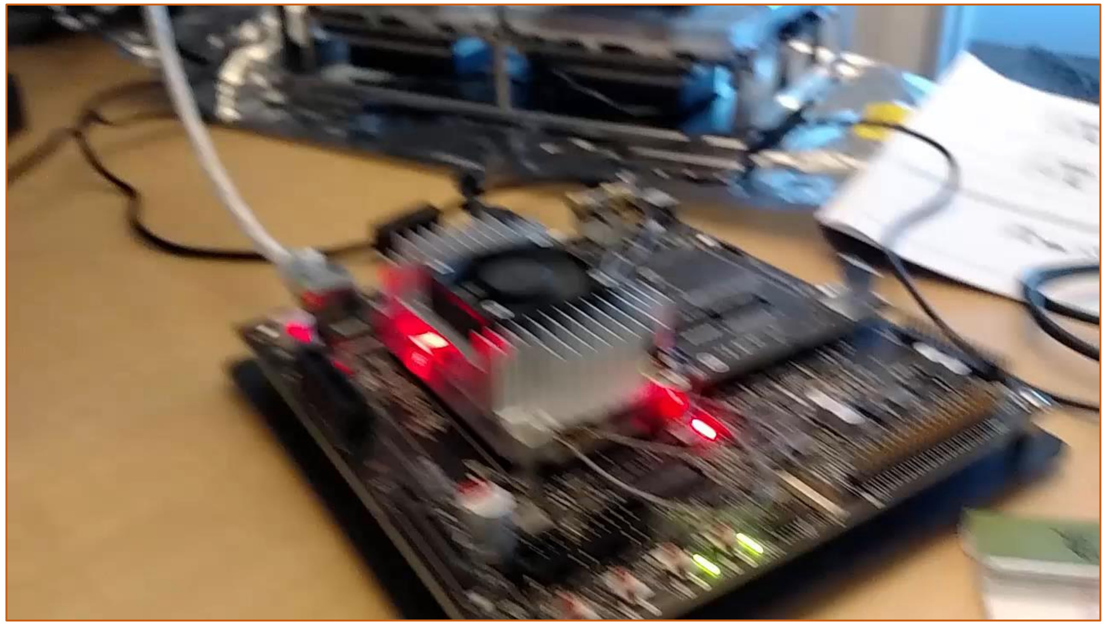
Toolchain: Automatically locate an installed toolchain

- Automatically locate an installed toolchain
- NVIDIA CUDA | gmake (64-bit Linux)
- NVIDIA CUDA for Jetson Tegra K1 v6.5 | gmake (64-bit Linux)
- NVIDIA CUDA for Jetson Tegra X1 v7.0 | gmake (64-bit Linux)
- [✓ NVIDIA CUDA for Jetson Tegra X2 v8.0 | gmake (64-bit Linux)]

Two small changes

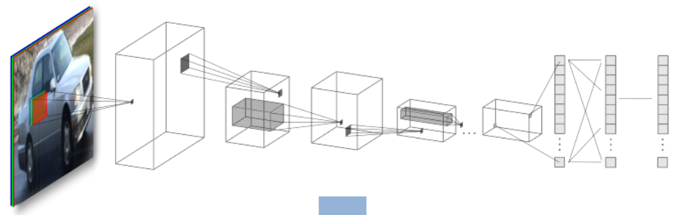
1. Change build-type to ‘lib’

2. Select cross-compile toolchain

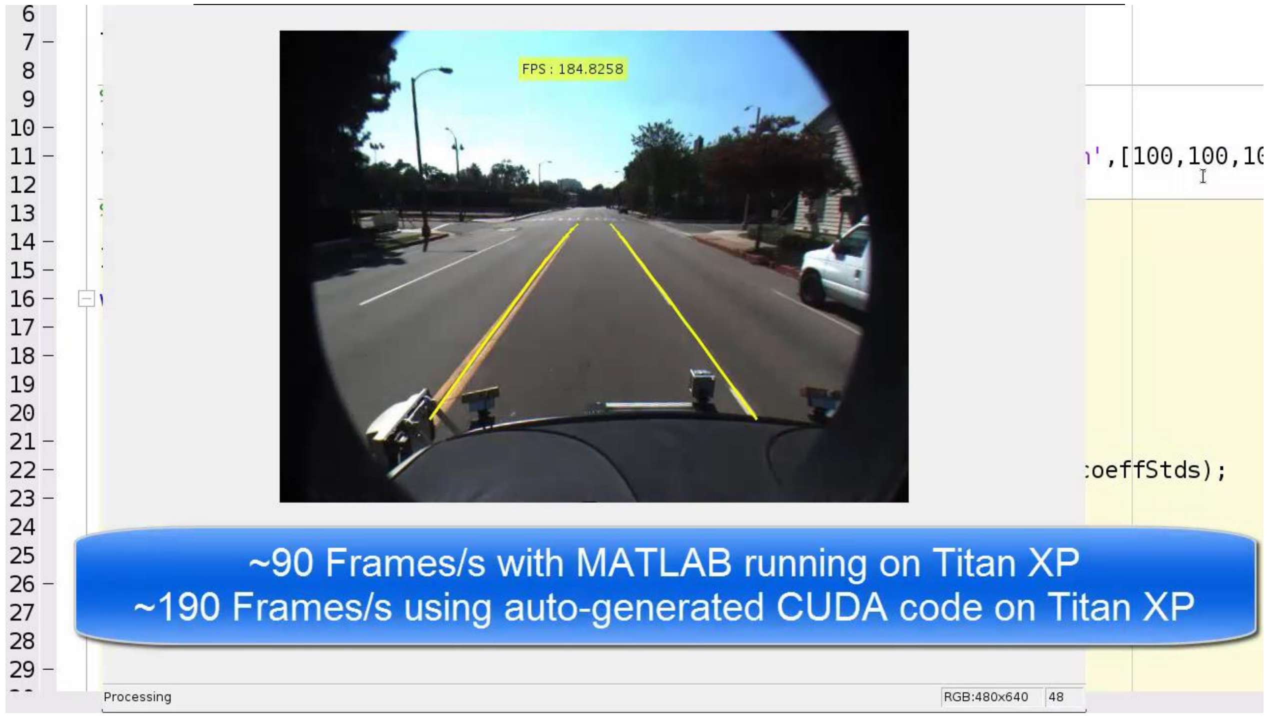


End-to-End Application: Lane Detection

Alexnet



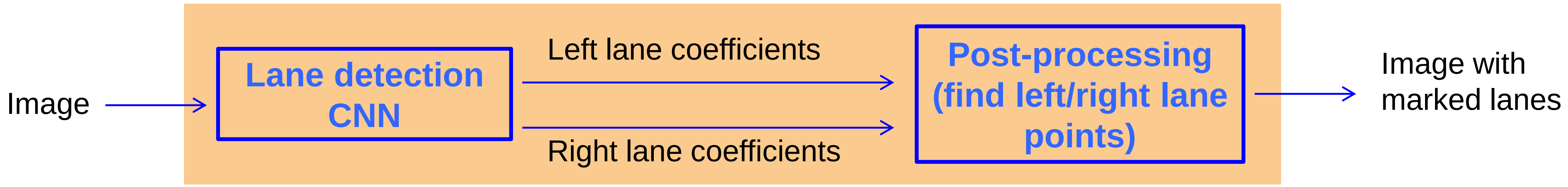
Transfer Learning



```
',[100,100,1
coeffStds);
```

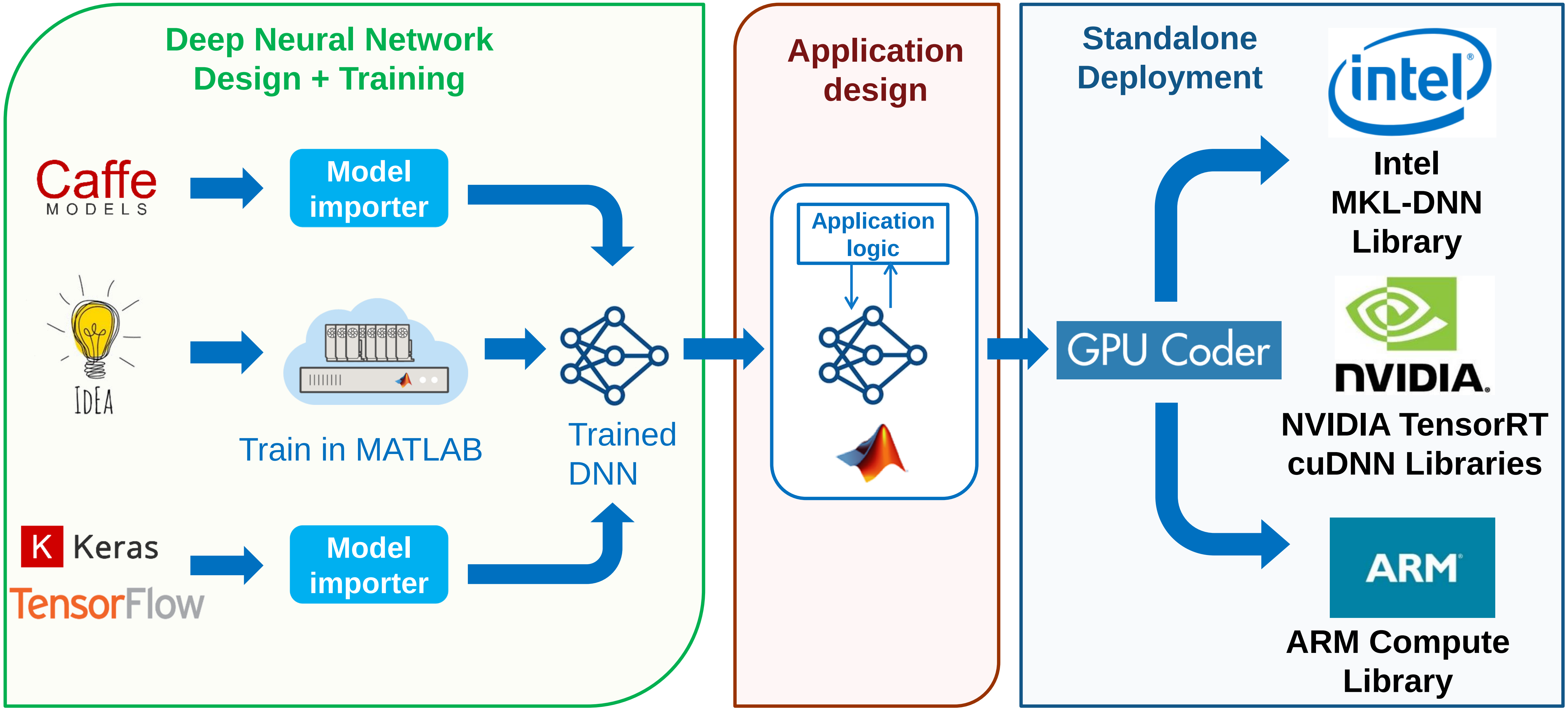
~90 Frames/s with MATLAB running on Titan XP
~190 Frames/s using auto-generated CUDA code on Titan XP

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



GPU coder generates code for whole application

Deep learning workflow in MATLAB


Deep Neural Network
Design + Training
Caffe MODELS
Model importer
IDEA
Train in MATLAB
Trained DNN
Keras
TensorFlow
Model importer
Application design
Application logic
Standalone Deployment
GPU Coder
intel
Intel MKL-DNN Library
NVIDIA
NVIDIA TensorRT cuDNN Libraries
ARM
ARM Compute Library