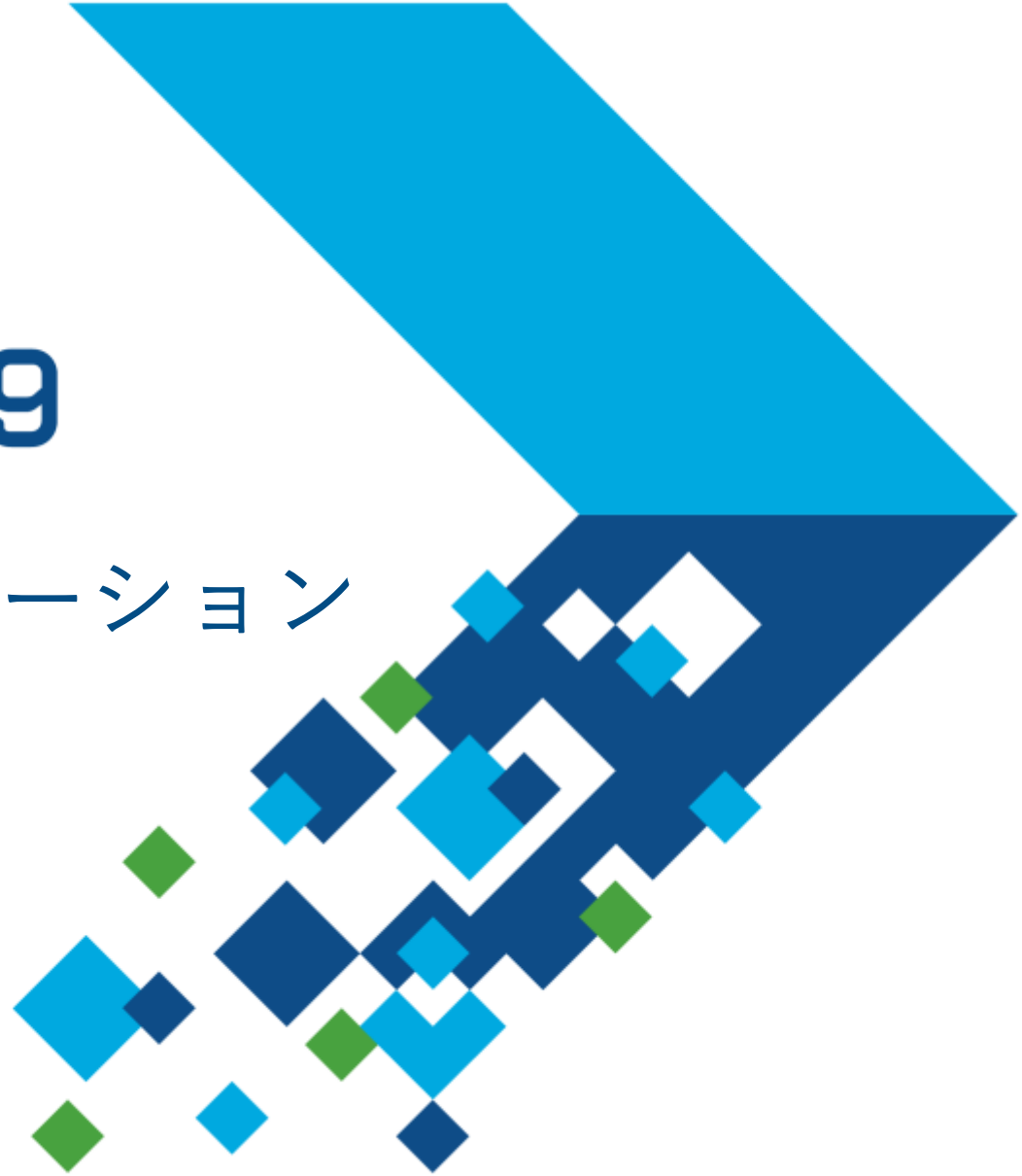
MATLAB EXPO 2019

ディープラーニングアプリケーションの組み込みGPU/CPU実装

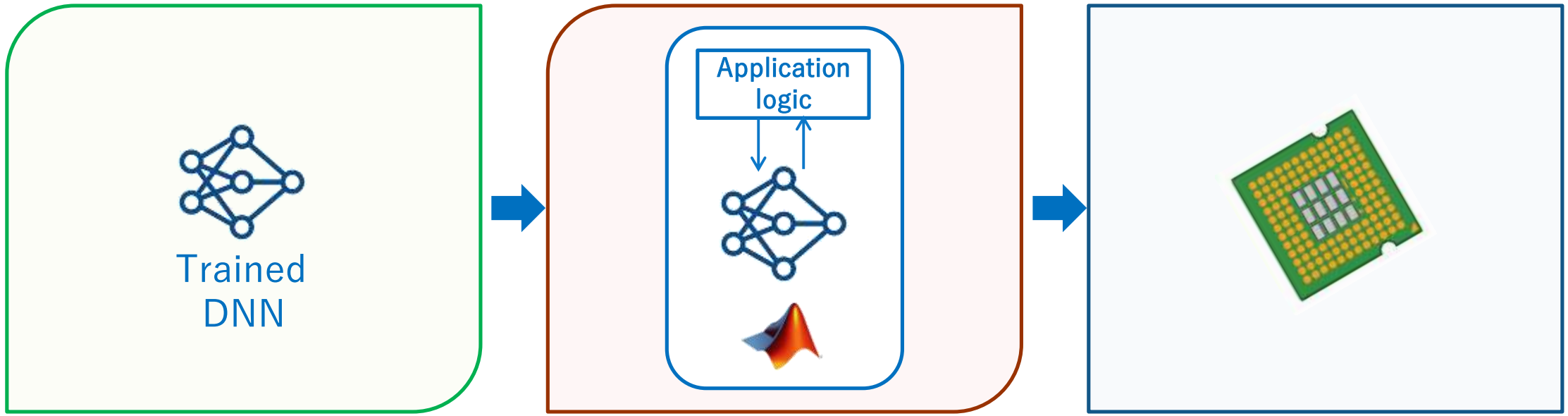
アプリケーション エンジニアリング部
町田　和也



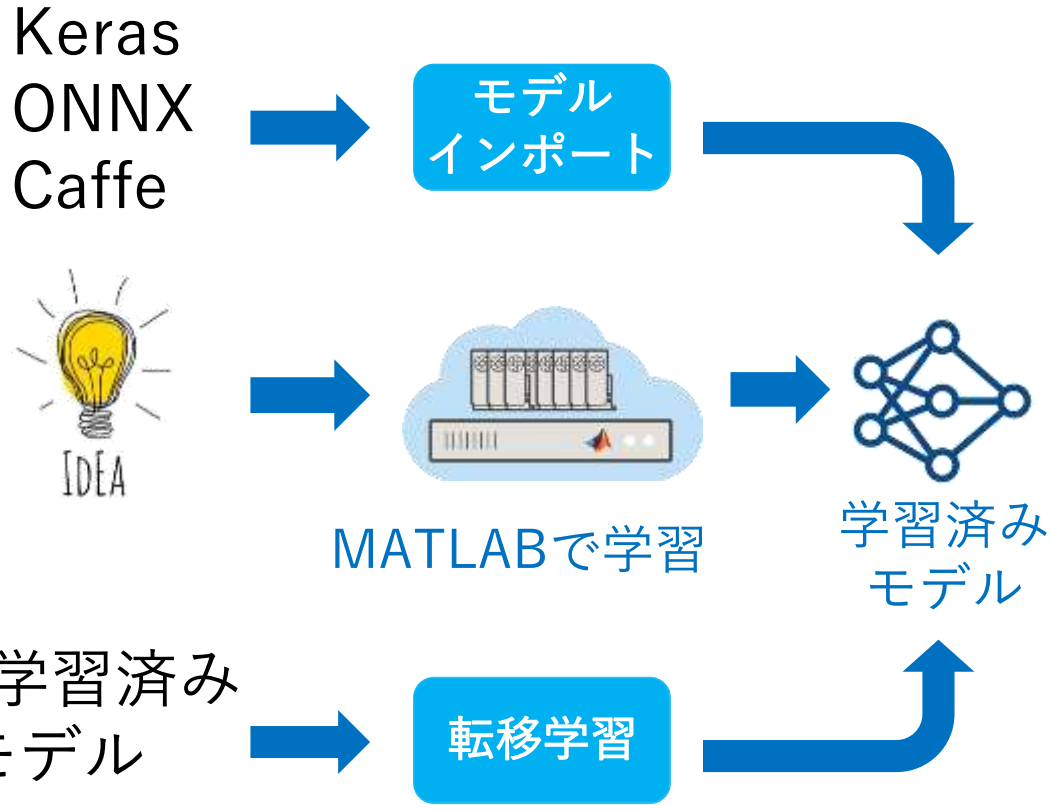
アジェンダ

- MATLAB Coder/GPU Coderの概要
- ディープニューラルネットワークの組み込み実装ワークフロー
- パフォーマンスに関して
- まとめ

ディープラーニングワークフローのおさらい

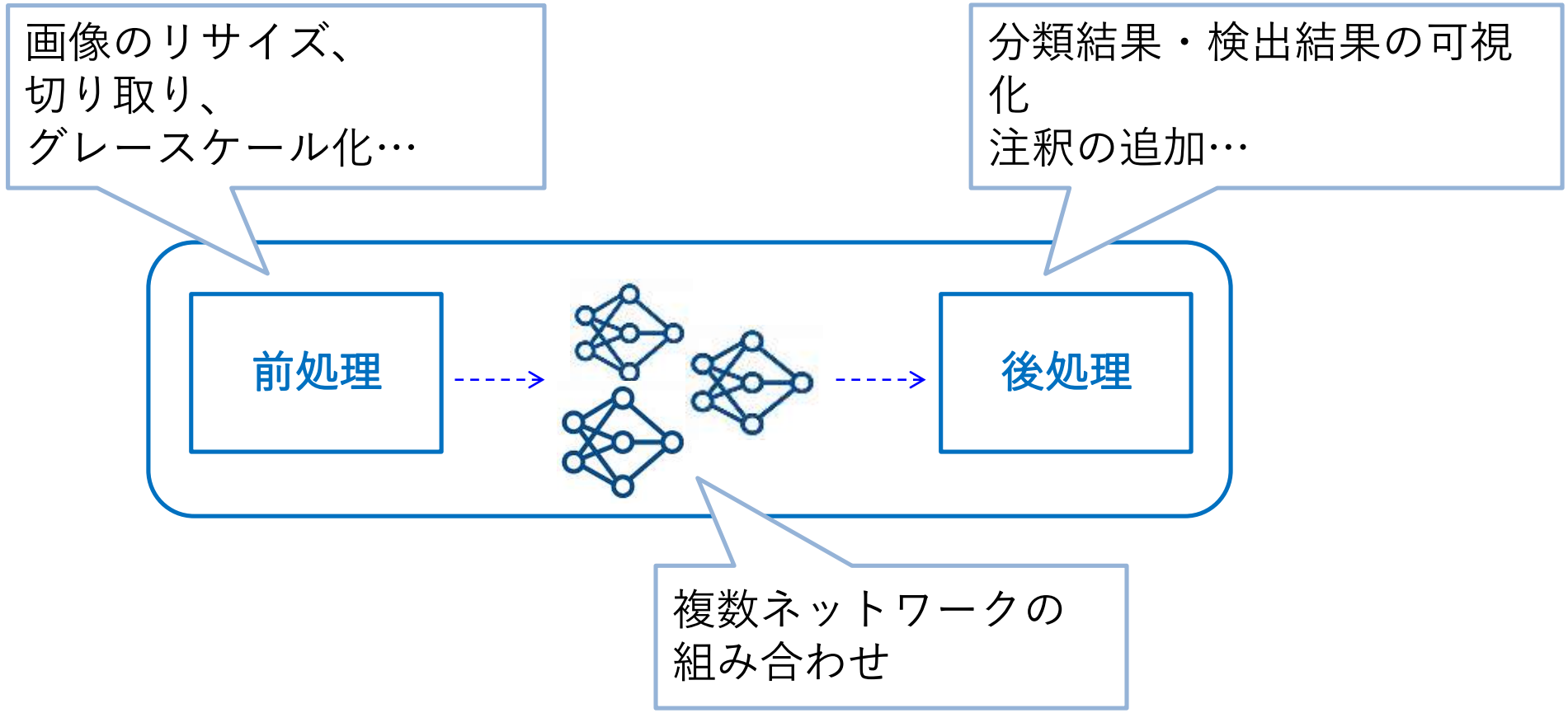


ディープラーニングワークフローのおさらい
ネットワークの構築と学習

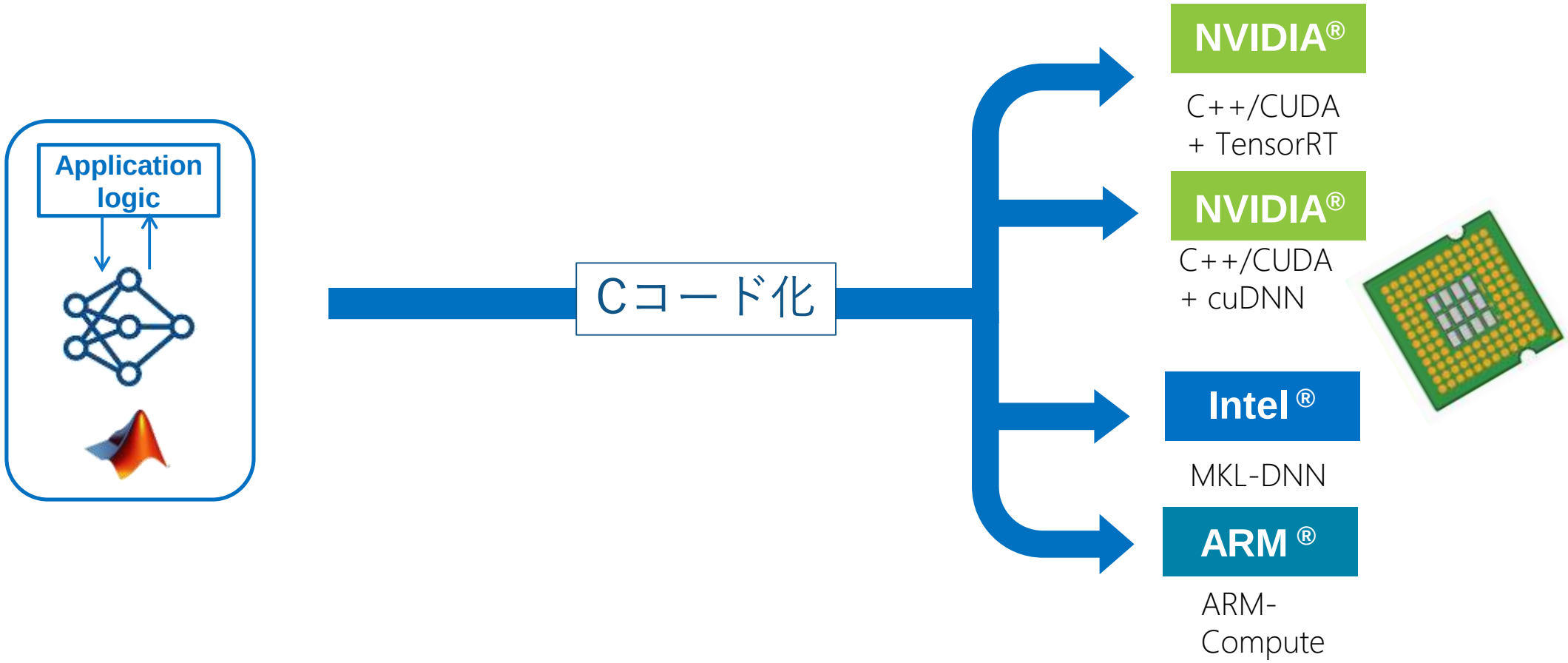


- **MATLABによる構築**
  - 多量のデータの取り扱い
  - ラベリングの自動化
  - 様々なモデルへのアクセス
- **MATLABによる学習**
  - GPUによる学習の高速化
  - クラスターによる分散処理

ディープラーニングワークフローのおさらい
アプリケーション化



ディープラーニングワークフローのおさらい
マルチプラットフォーム環境への実装



Cコード化のハードル

Vectorized MATLAB

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

Cコード化

CUDA Cコード

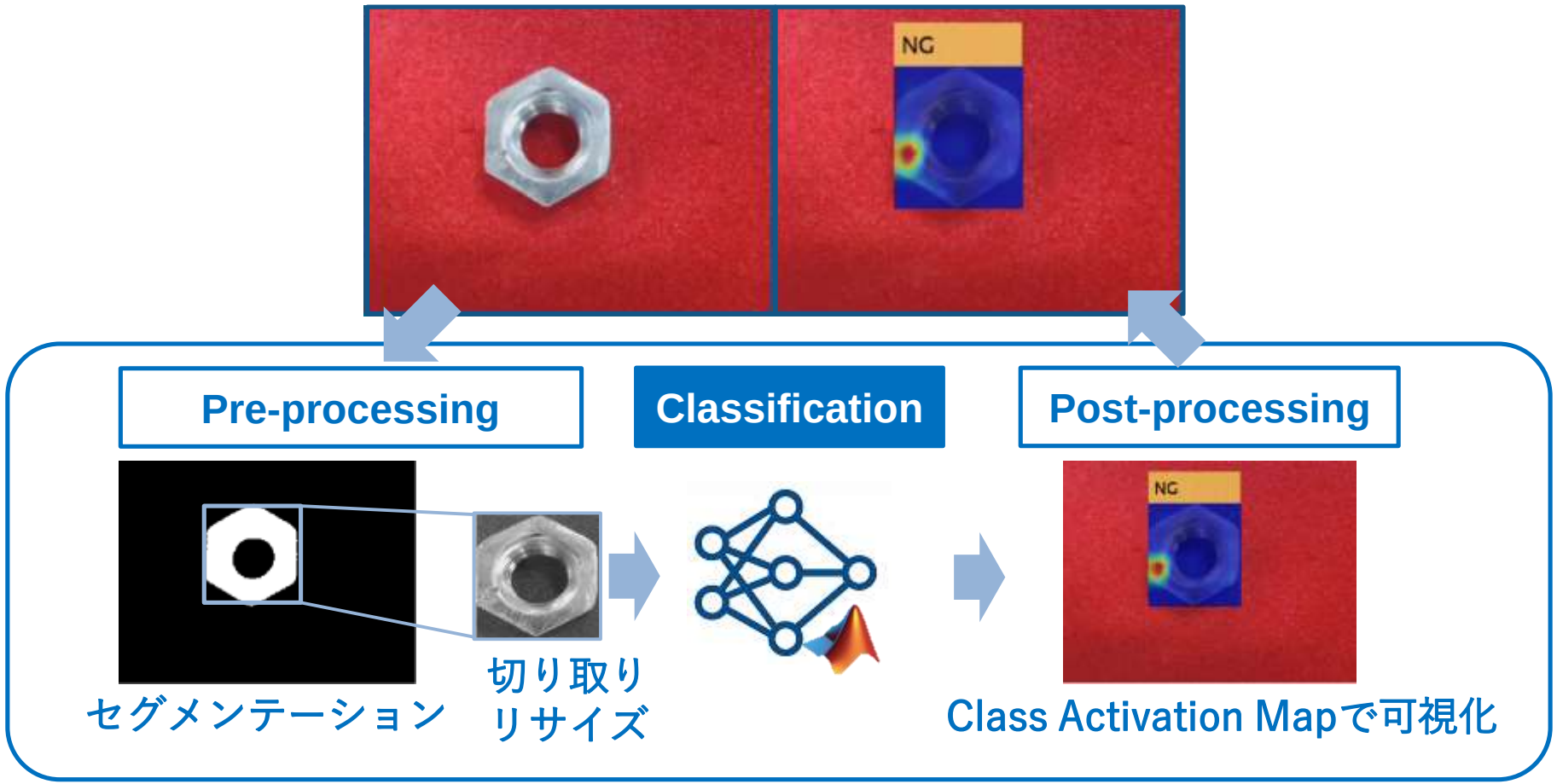
```
static __global__ __launch_bounds__(512, 1) void saxpy_kernel1(const real32_T *y,
  const real32_T *x, real32_T a, real_T *z)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (!(i >= 1048576)) {
    z[i] = (real_T)(a * x[i] + y[i]);
  }
}

void saxpy(real32_T a, const real32_T x[1048576], const real32_T y[1048576],
           real_T z[1048576])
{
  real32_T *gpu_y;
  real32_T *gpu_x;
  real_T *gpu_z;
  cudaMalloc(&gpu_z, 8388608UL);
  cudaMalloc(&gpu_x, 4194304UL);
  cudaMalloc(&gpu_y, 4194304UL);
  cudaMemcpy((void *)gpu_y, (void *)&y[0], 4194304UL, cudaMemcpyHostToDevice);
  cudaMemcpy((void *)gpu_x, (void *)&x[0], 4194304UL, cudaMemcpyHostToDevice);
  saxpy_kernel1<<<dim3(2048U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_y, gpu_x, a,
    gpu_z);
  cudaMemcpy((void *)&z[0], (void *)gpu_z, 8388608UL, cudaMemcpyDeviceToHost);
  cudaFree(gpu_y);
  cudaFree(gpu_x);
  cudaFree(gpu_z);
}
```

手書きによるバグ混入
等価性の維持が困難

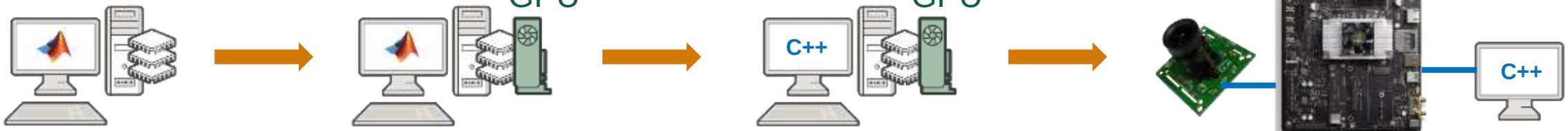
GPUは非常に強力なハードウェアですが、…
プログラミングは専門の知識が必要となります。

Cコード化のハードル

推論以外の処理もコード化する必要があります



組み込み実装の課題



1 機能確認

2 単体検証

高級言語
深層学習フレームワーク
大規模で複雑な
　ソフトウェアスタック

3 統合検証

C/C++
低水準API
特定用途のライブラリ

4 リアルタイム検証

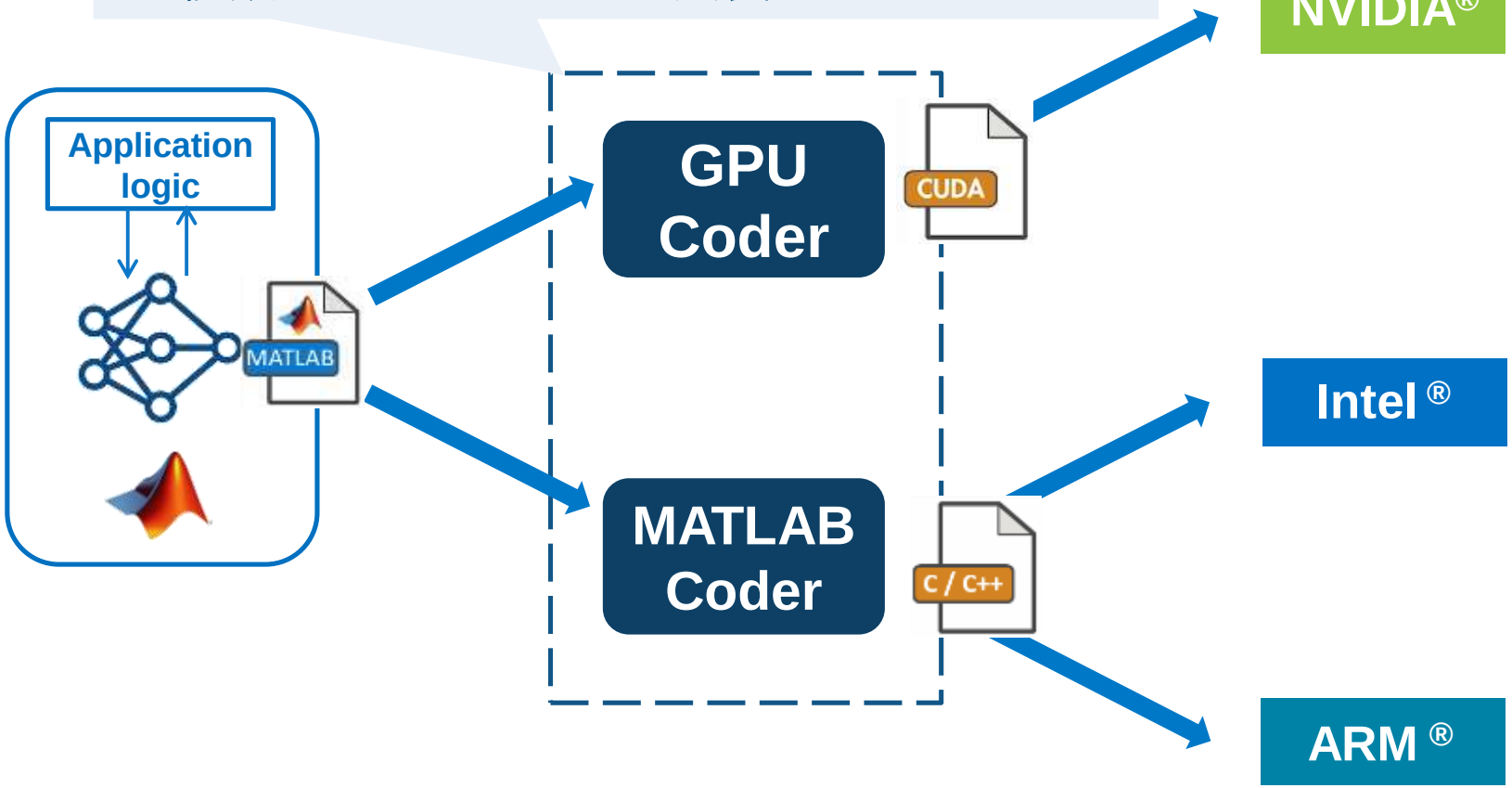
C/C++
ターゲットの最適化ライブラリ
メモリと処理速度の最適化

課題

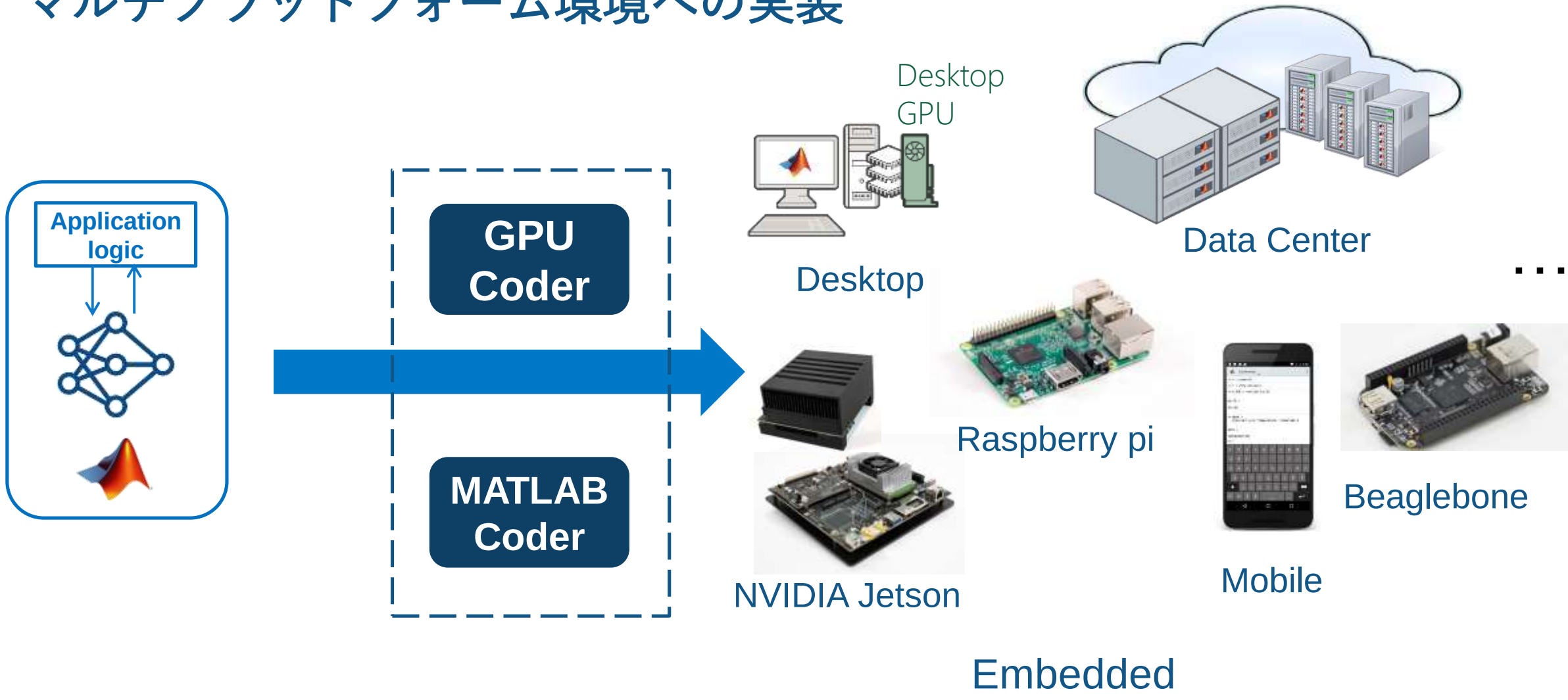
- 複数のライブラリとパッケージの統合
- 複数の実装の検証と維持
- アルゴリズムとベンダーロックインの検討

MATLAB Coder & GPU Coderによる実装ソリューション

- 前処理・後処理を含めたコードの自動生成
- 生成されるコードの最適化
- 複数のターゲットへの実装



MATLAB Coder & GPU Coderによる実装ソリューション
マルチプラットフォーム環境への実装



MATLAB Coder & GPU Coderによる組み込み実装

MATLAB アルゴリズム
(functional reference)

GPU Coder
MATLAB Coder

複数の実装形式に対応

Build type

MATLAB上で、
生成したCUDA
コードを直接Call

手書きしたmain関数
(CC++)から生成した
CUDAコードをCall

手書きしたmain関数
(CC++)から生成した
CUDAコードをCall

.mex

.lib

Cross-compiled
.lib

Desktop
GPU

Desktop
GPU

Embedded GPU

C++

C++

1 機能確認

2 単体検証

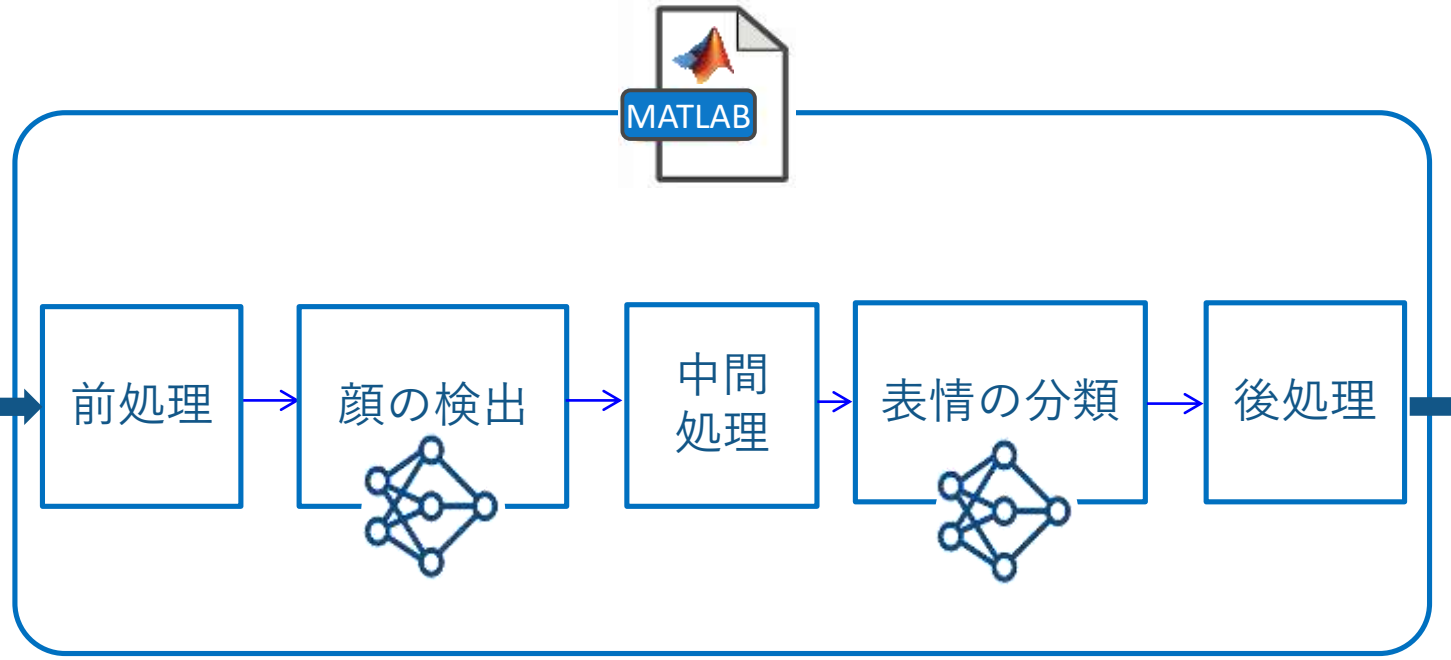
3 統合検証

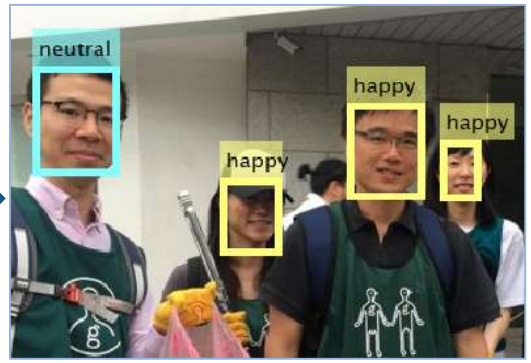
4 リアルタイム検証

アジェンダ

- MATLAB Coder/GPU Coderの概要
- ディープニューラルネットワークの組み込み実装ワークフロー
- パフォーマンスに関して
- まとめ

【Demo】顔検出と表情分類アプリケーションのJetson実装

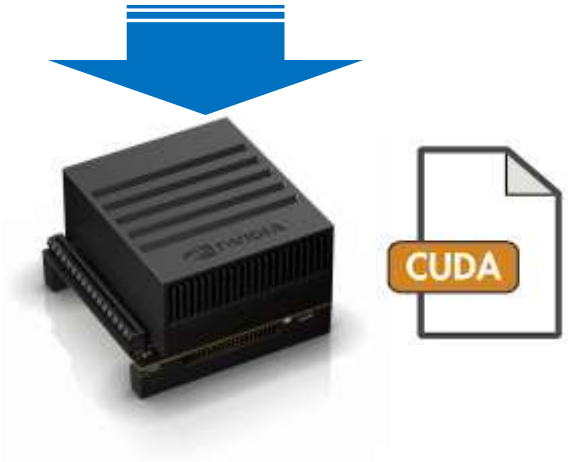




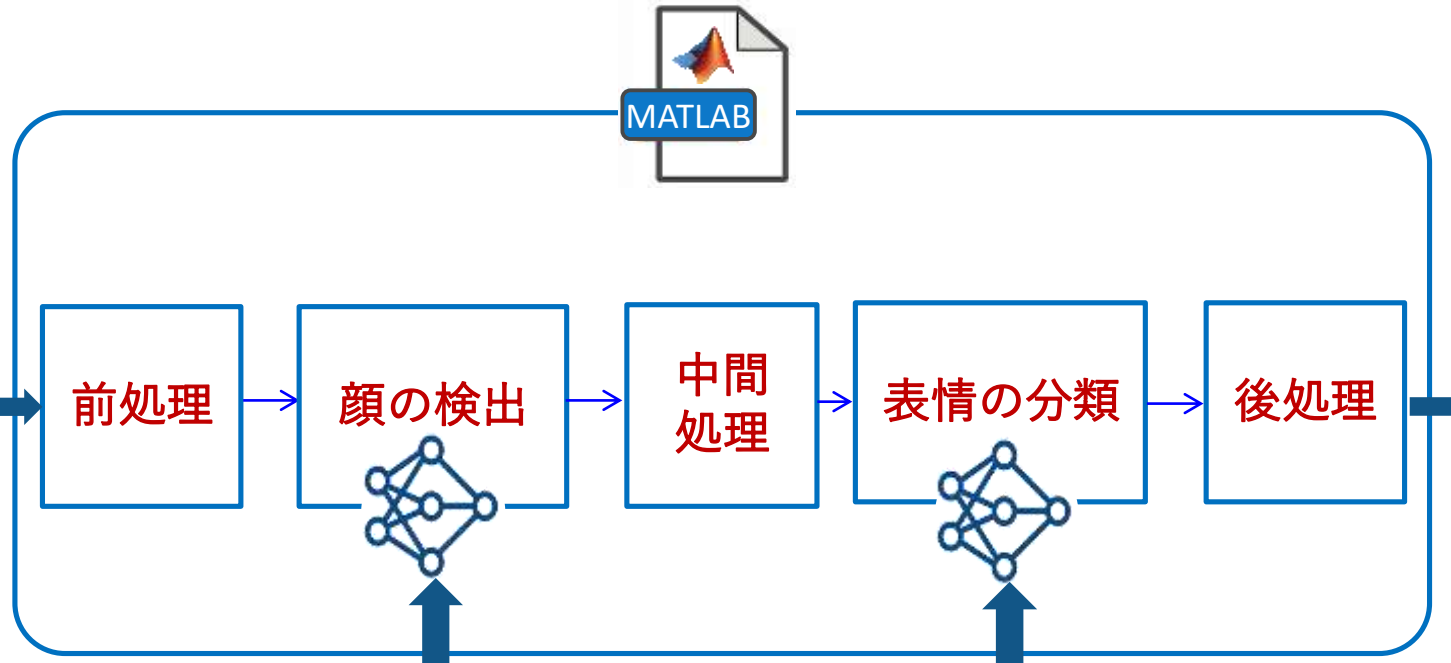
実装ワークフロー
①MATLABで検証

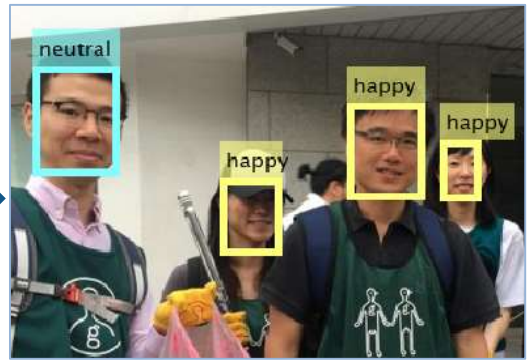
②コード生成→ホスト環境での検証

③コード生成→組み込み環境での検証



①MATLABで検証





機能検証
- アルゴリズムの動作検証

カスタム
YOLO v2

MATLABで学習させた
YOLOv2モデル
R2019a

Keras
TensorFlow

他フレームワークからの
モデルのインポート

①MATLABで検証

顔の検出

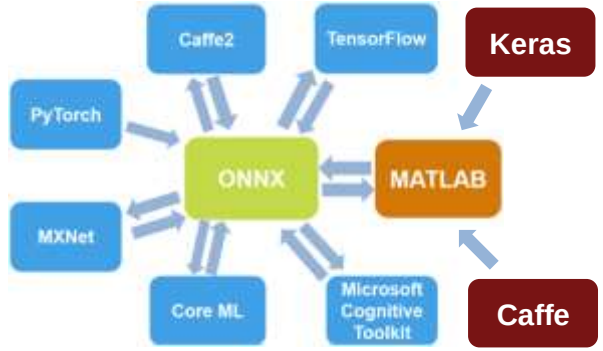
R2019a

- YOLO v2による物体検出
  - YOLO(You Only Look Once)とは？
    - リアルタイム向けの物体検出モデル
      (Faster R-CNNの100倍程度高速)
  - 学習からCUDAコード生成まで対応
    - 自身のデータセットに合わせたカスタム
      YOLOv2モデルの学習から実装までサポート


```
detector =
    trainYOLOv2ObjectDetector(trainingData,lgraph,options)
```

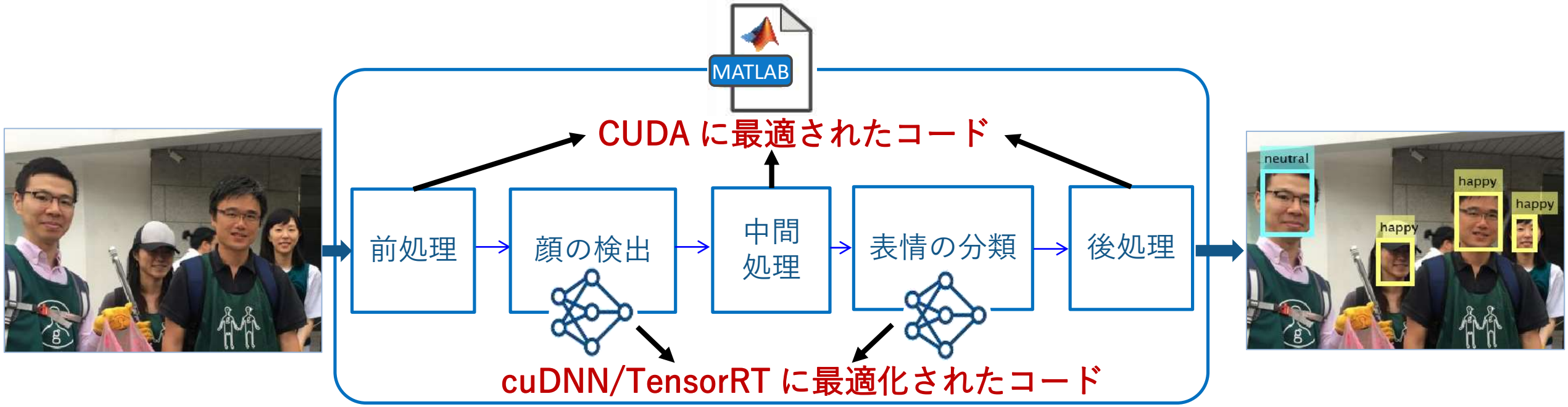
表情の分類

- 他フレームワークとの連携
  - Keras, Caffeモデルの取り込み
  - ONNXフォーマットを介したモデル
    のやり取り




```
net = importKerasNetwork(modelfile)
net = importCaffeNetwork(protofile,datafile)
net = importONNXNetwork(modelfile, ...
                        'OutputLayerType',outputtype)
```

②コード生成→ホスト環境での検証



単体検証

- コード生成可能か確認

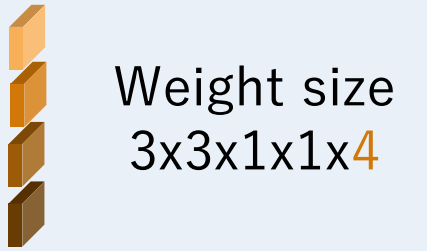
→生成不可の場合は生成可能な形に修正

②コード生成→ホスト環境での検証
コード修正が必要な例

グループ化畳み込み層を通常の畳み込み層に変換

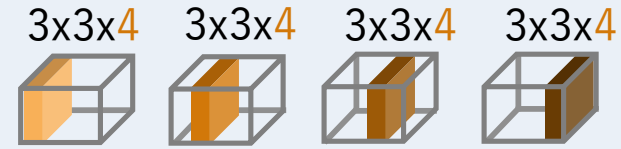
R2019aでは
コード生成未対応

groupedConvolution2dLayer



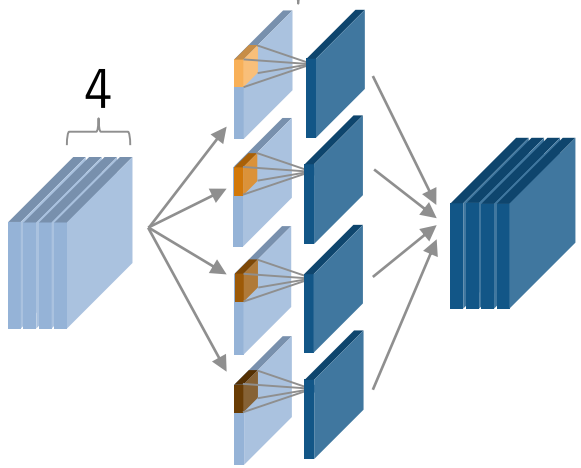
convolution2dLayer

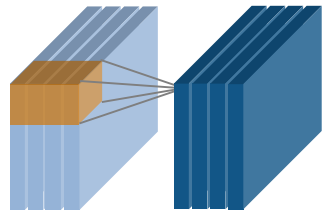
コード生成
対応



各々0埋めして拡張

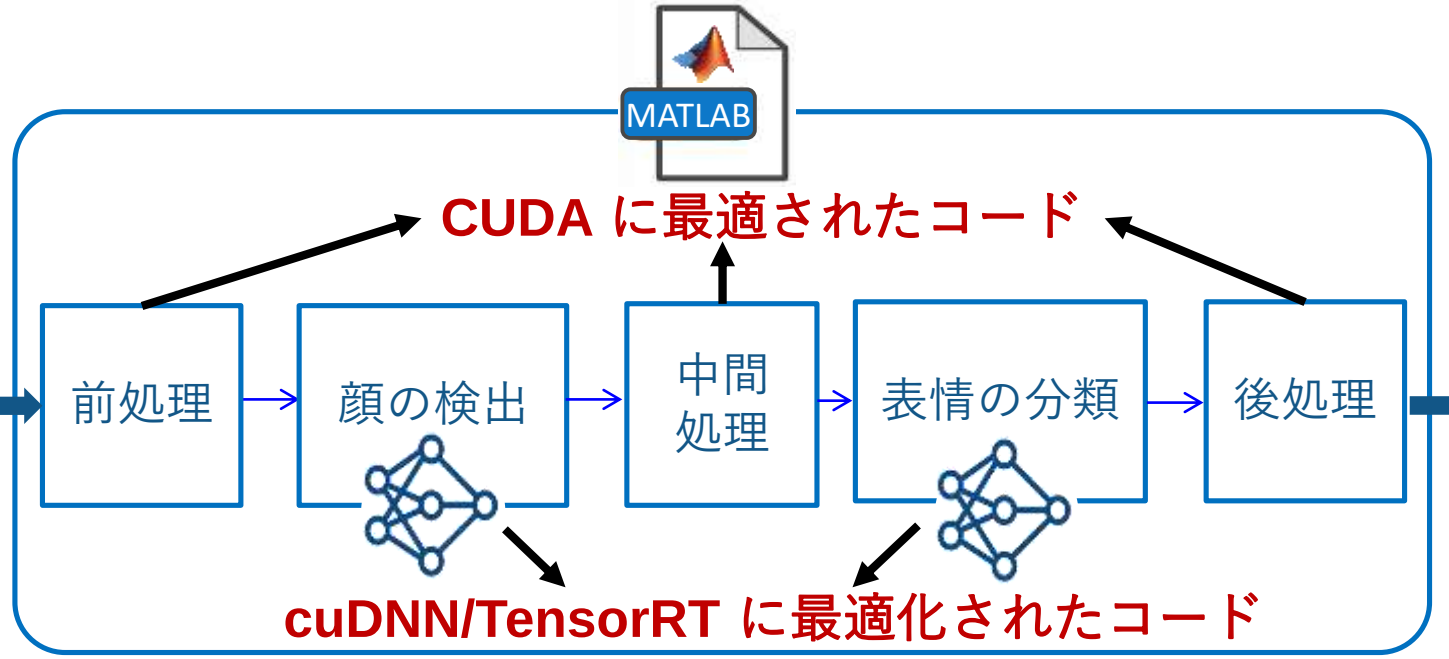
例：
4チャネルの入力を
1チャネル毎に
グループ化した場合

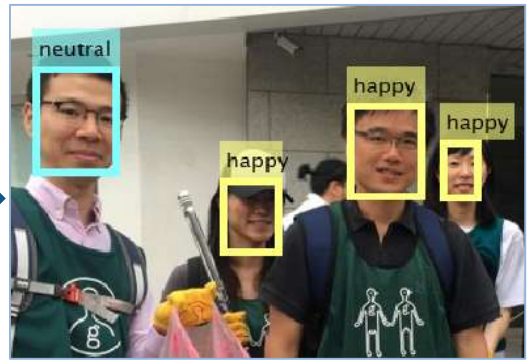




計算結果が等価になるように
通常の畳み込み層に重みとバイアスを移植

③コード生成→組み込み環境での検証





リアルタイム検証

- ターゲット環境で
  パフォーマンス確認

NVIDIA GPUでの実装例

Application logic

MATLAB

GPU Coder

Target Libraries

NVIDIA®

NVIDIA TensorRT & cuDNN Libraries

Intel®

Intel MKL-DNN Library

ARM®

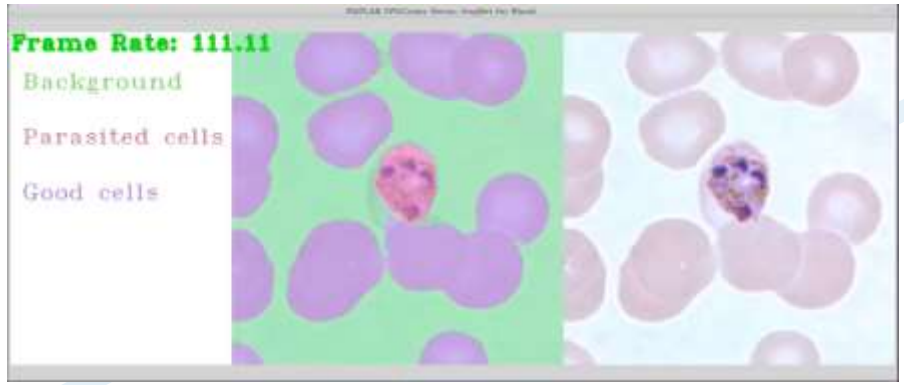
ARM Compute Library

Semantic Segmentation

Intel CPUでの実装例

Target Libraries

Blood Smear Segmentation



Application logic

MATLAB

MATLAB Coder

NVIDIA®

NVIDIA TensorRT & cuDNN Libraries

Intel®

Intel MKL-DNN Library

ARM®

ARM Compute Library

ARMプロセッサでの実装例

Target Libraries

Pedestrian Detection

Application logic

MATLAB

MATLAB Coder

NVIDIA®

NVIDIA TensorRT & cuDNN Libraries

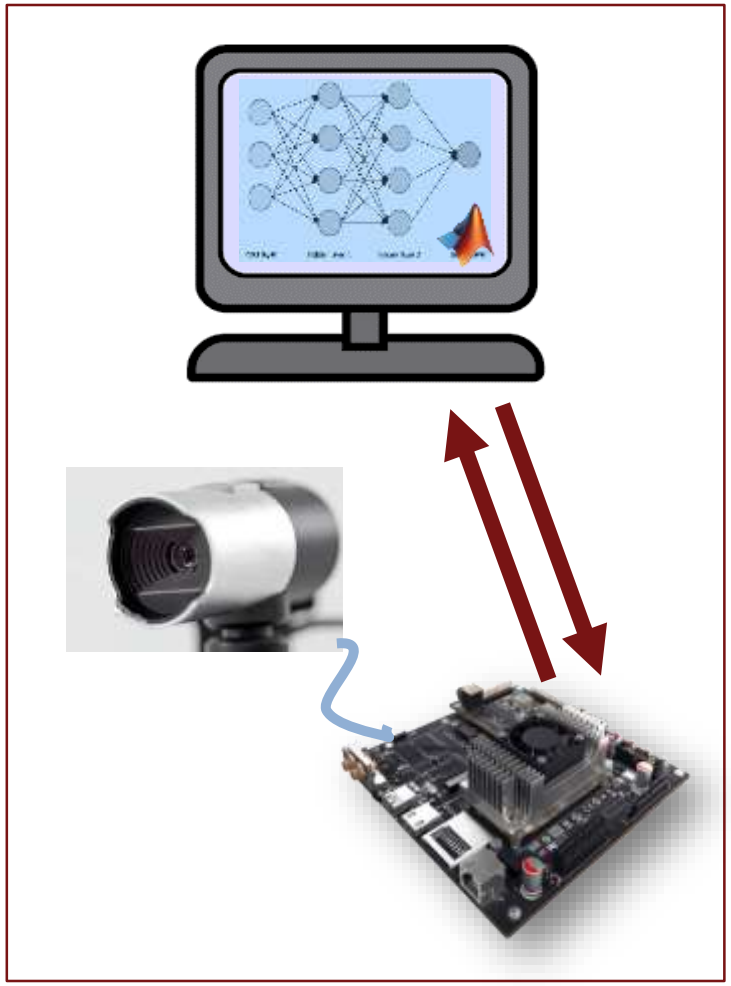
Intel®

Intel MKL-DNN Library

ARM®

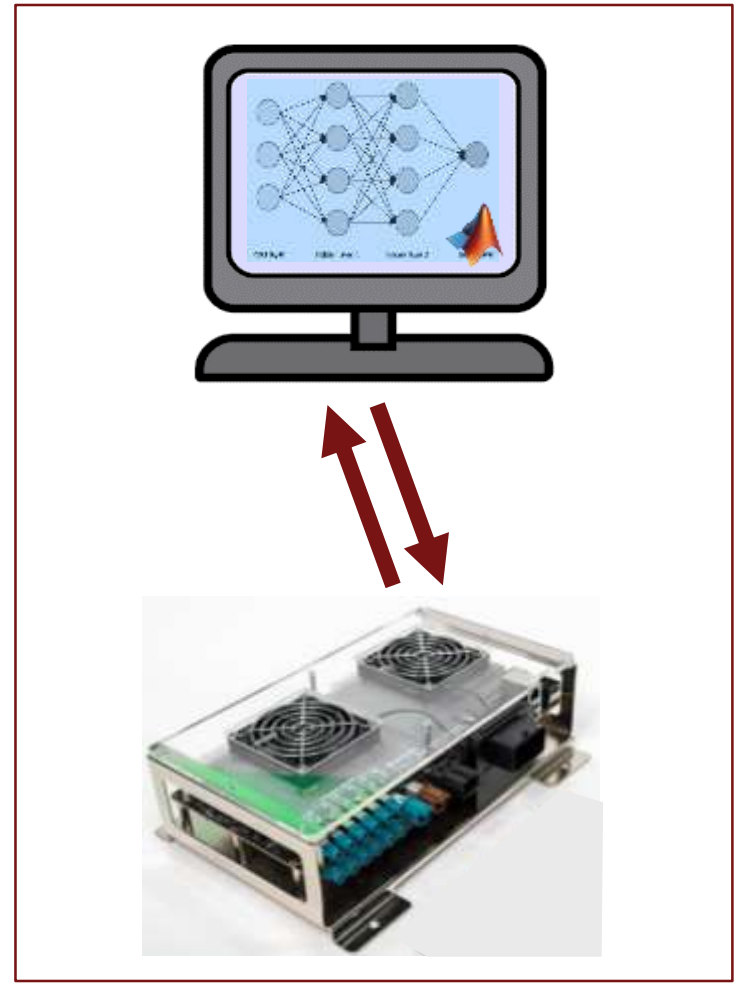
ARM Compute Library

ハードウェアへのアクセス



MATLABから、
周辺機器にアクセス

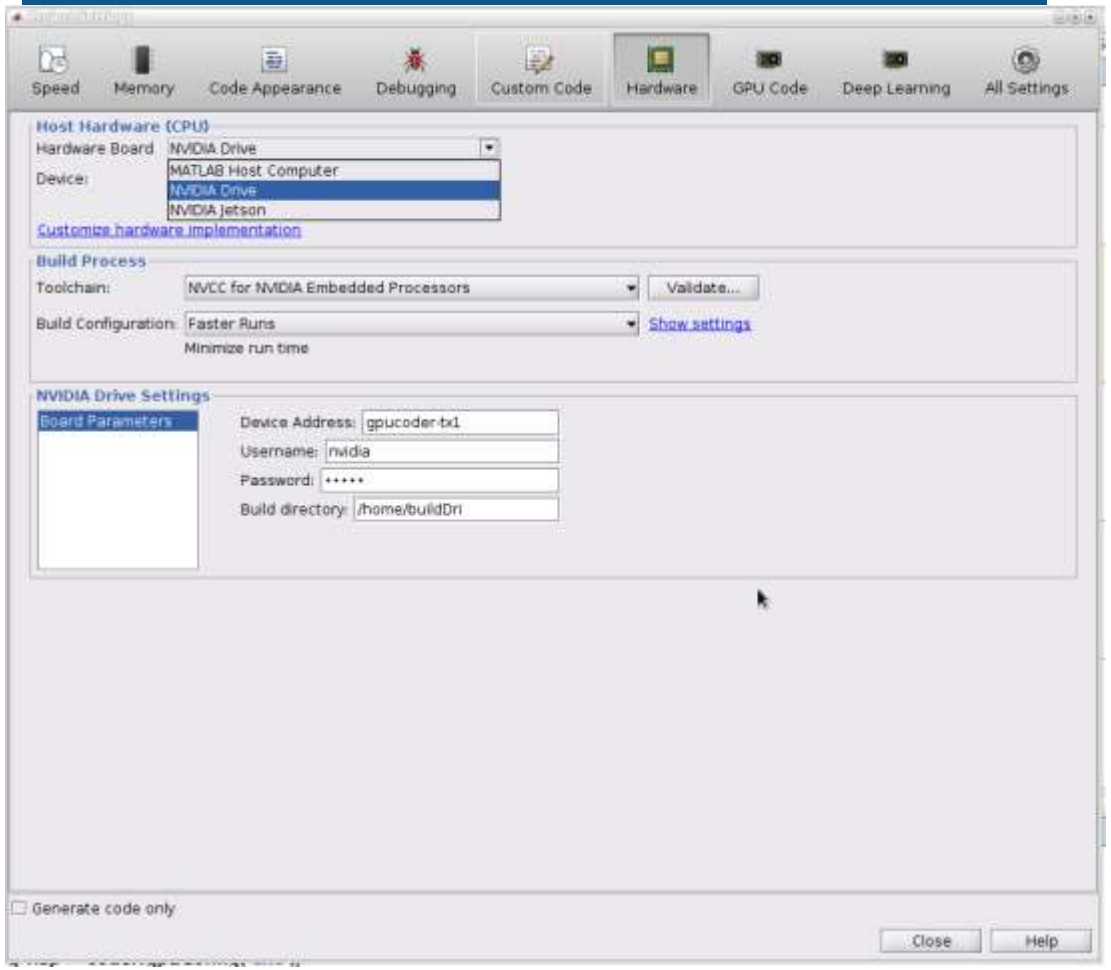
スタンドアローン
アプリケーションとして実装



Processor-in-Loop 検証

ターゲットへの実装方法

専用アプリによるコード生成と実装



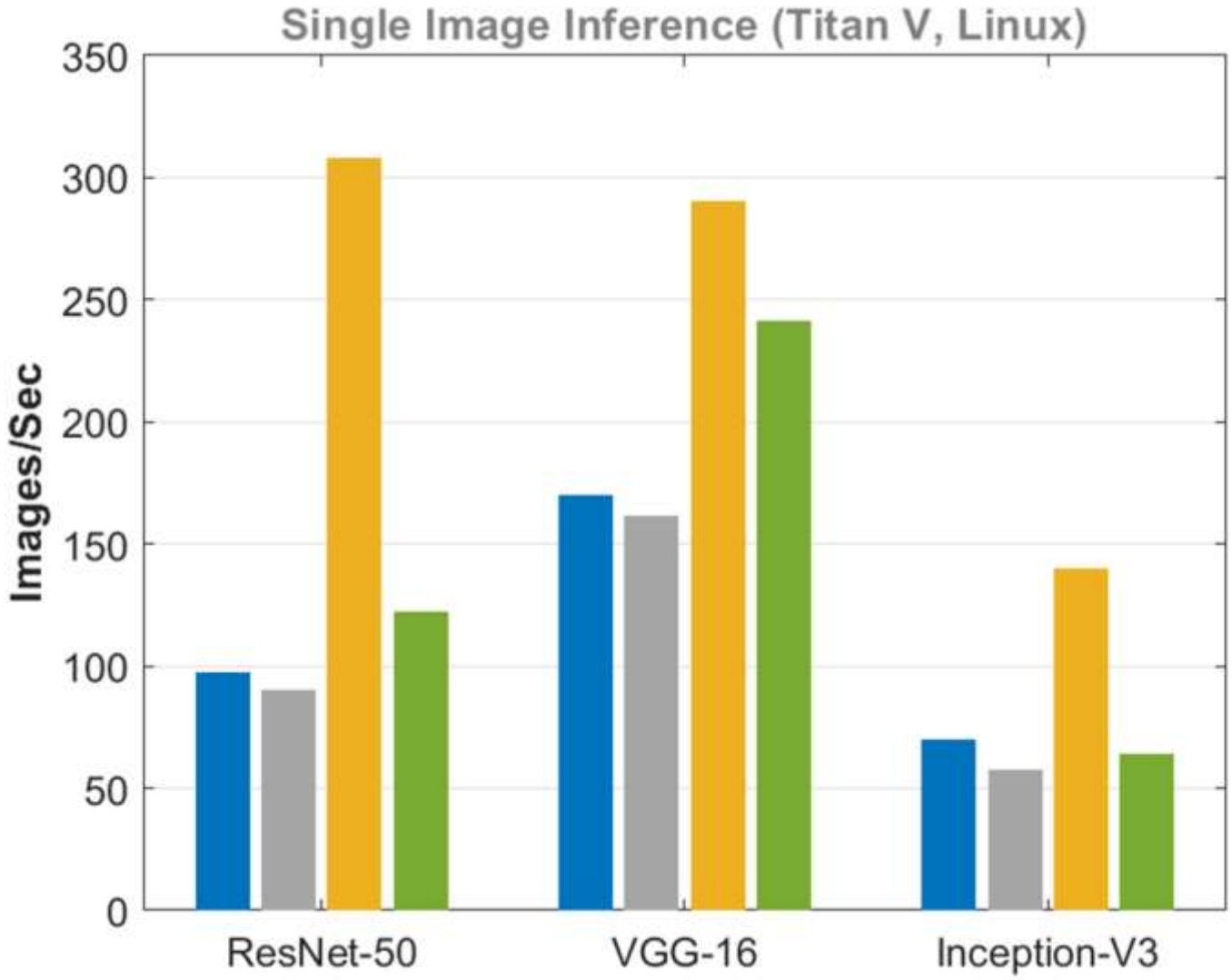
コマンドによるコード生成と実装

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

アジェンダ

- MATLAB Coder/GPU Coderの概要
- ディープニューラルネットワークの組み込み実装ワークフロー
- パフォーマンスに関して
- まとめ

画像一枚に対する推論速度の比較



TensorFlow (1.13.0)

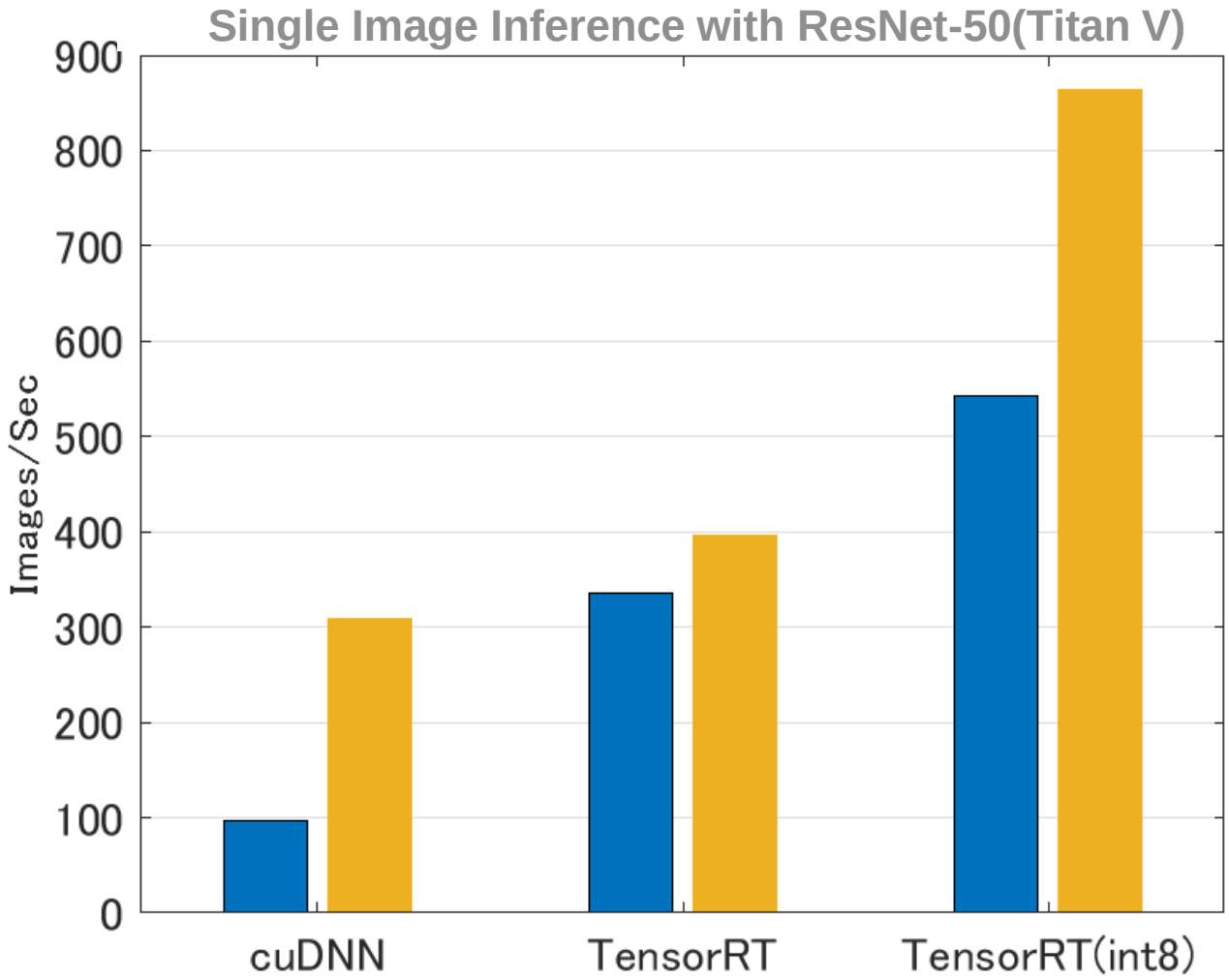
MXNet (1.4.0)

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

TensorRTによる高速化

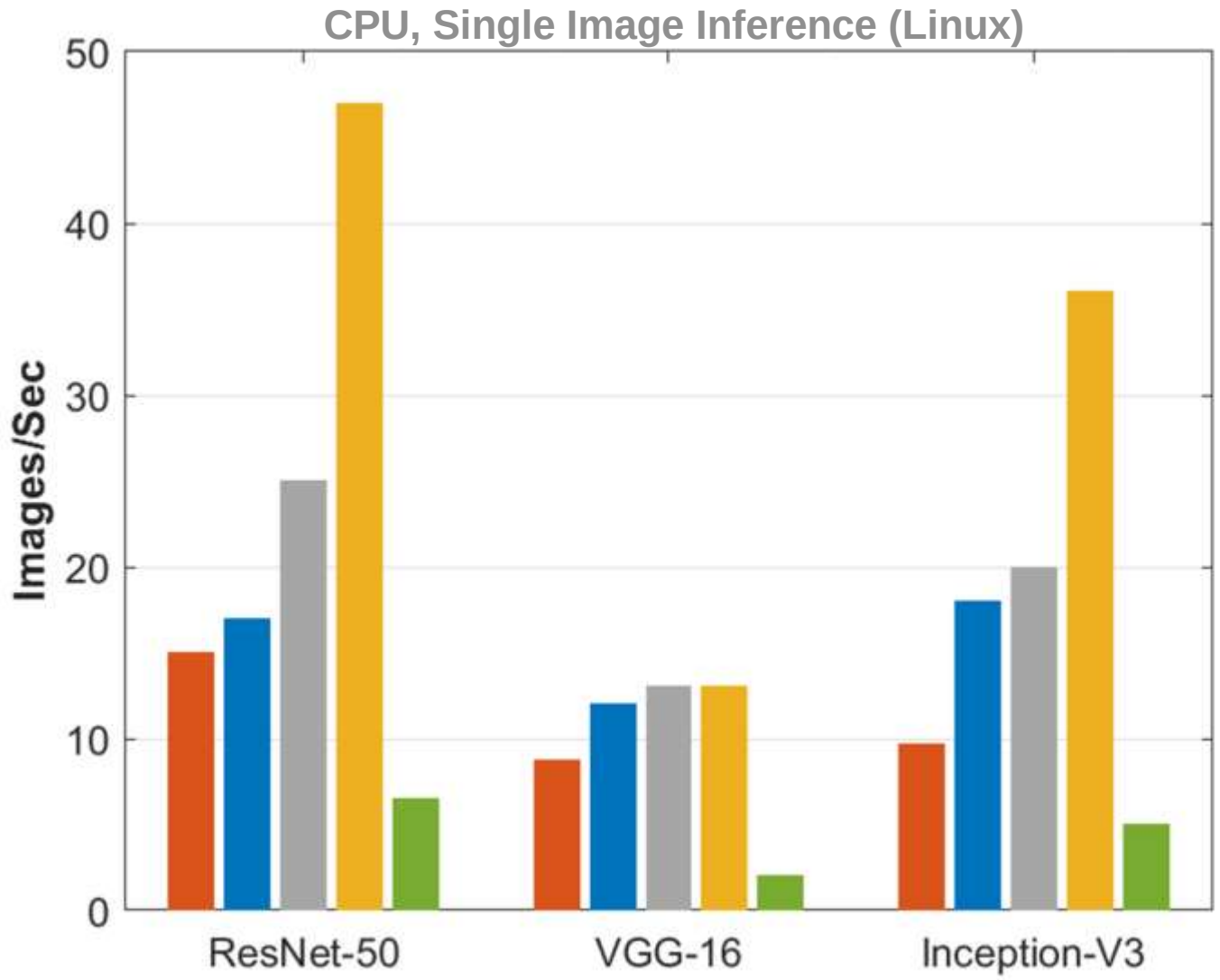


TensorFlow (1.13.0)

GPU Coder (R2019a)

Intel® Xeon® CPU 3.6 GHz - NVIDIA libraries: CUDA10 - cuDNN 7 – Tensor RT 5.0.2.6. Frameworks: TensorFlow 1.13.0

CPUでの推論速度の比較



MATLAB

TensorFlow

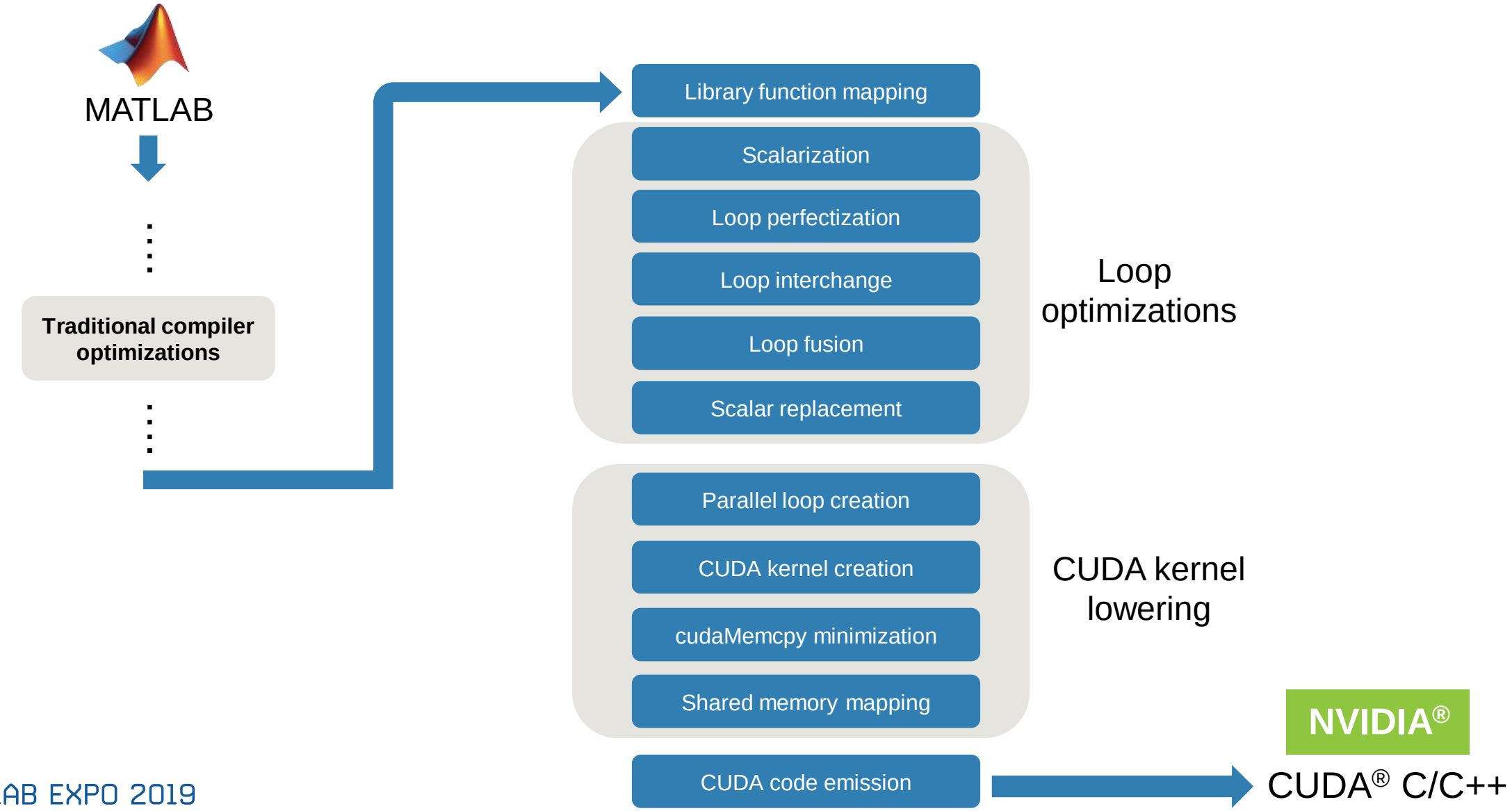
MXNet

MATLAB Coder

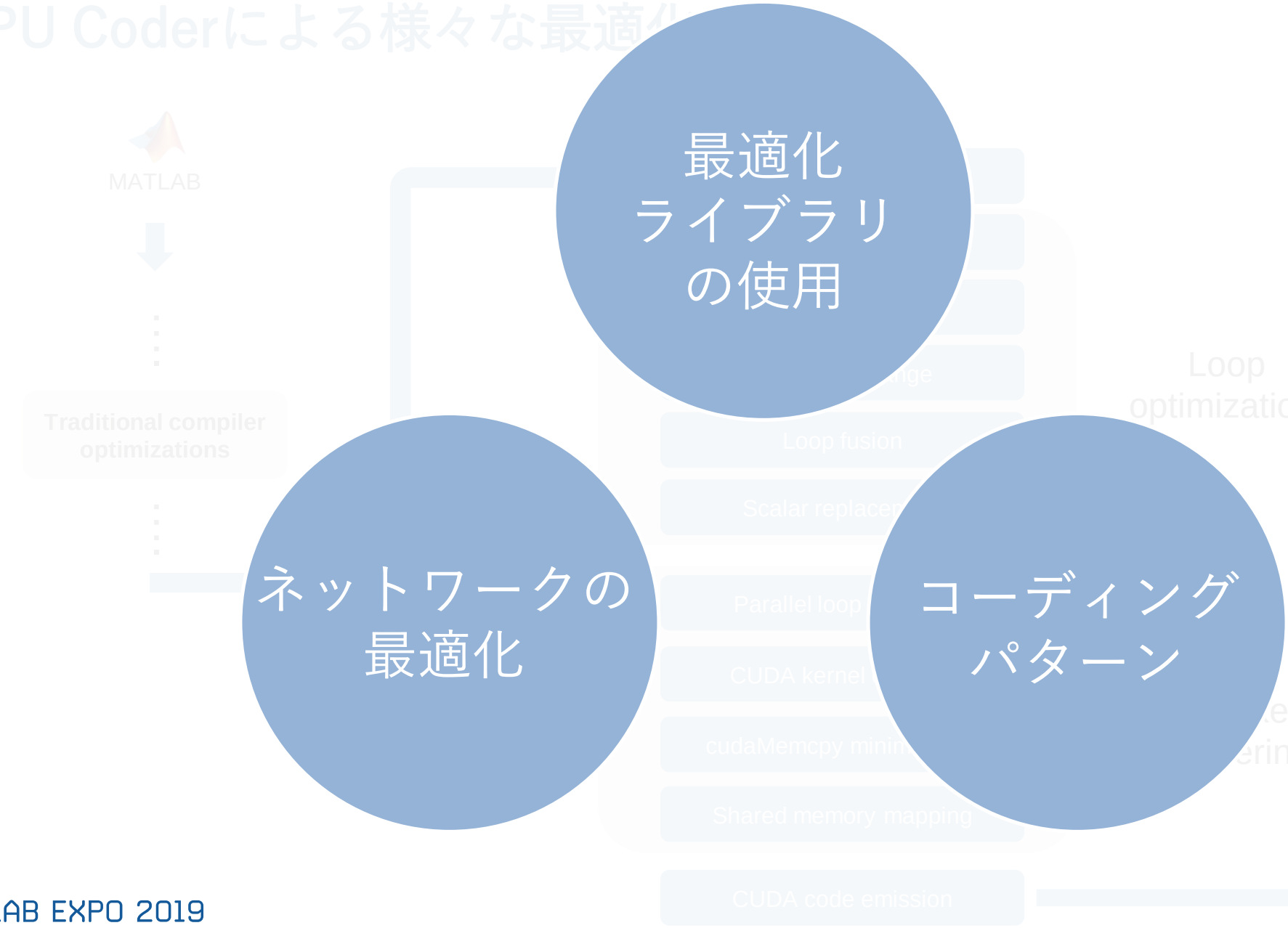
PyTorch

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

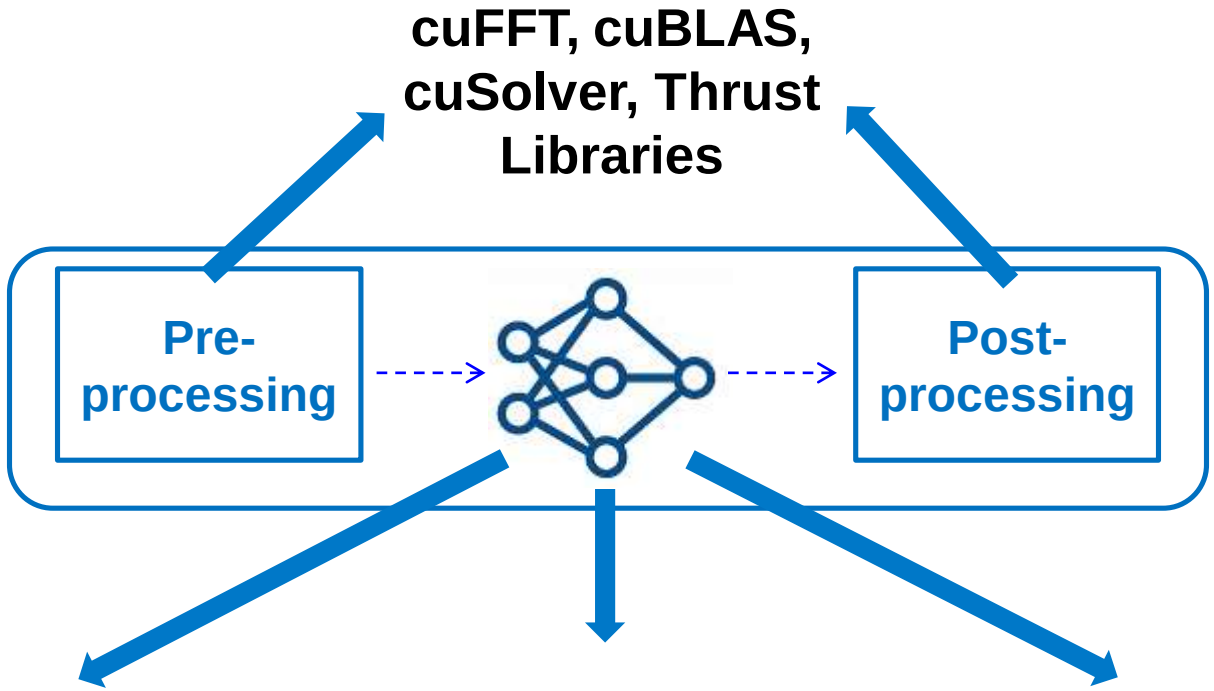
GPU Coderによる様々な最適化



GPU Coderによる様々な最適化



最適化ライブラリを用いたコード生成



Intel ®

Intel MKL-DNN Library

NVIDIA®

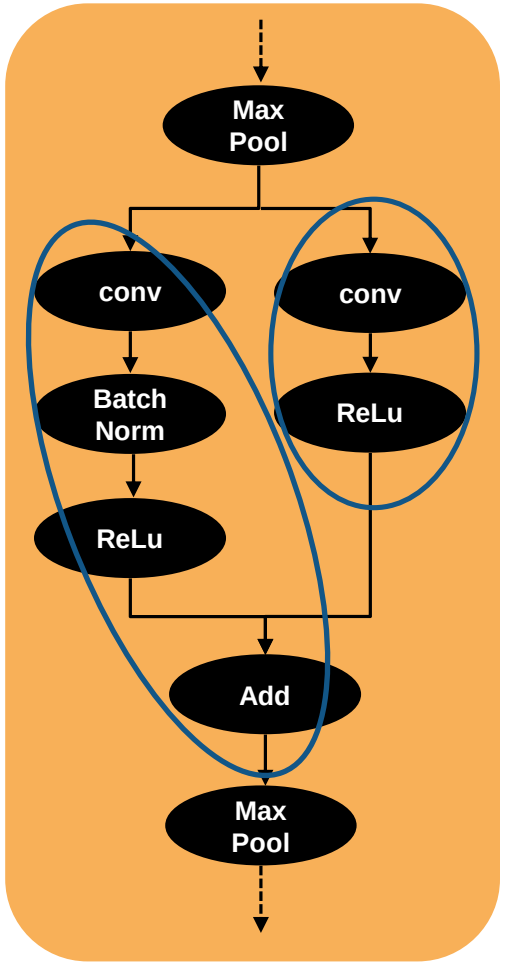
NVIDIA TensorRT & cuDNN Libraries

ARM ®

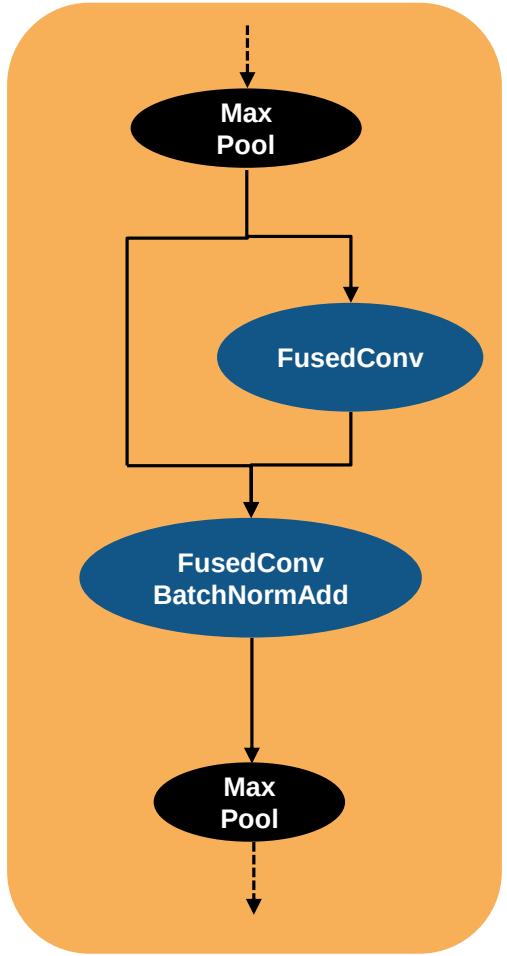
ARM Compute Library

ディープニューラルネットワークの最適化

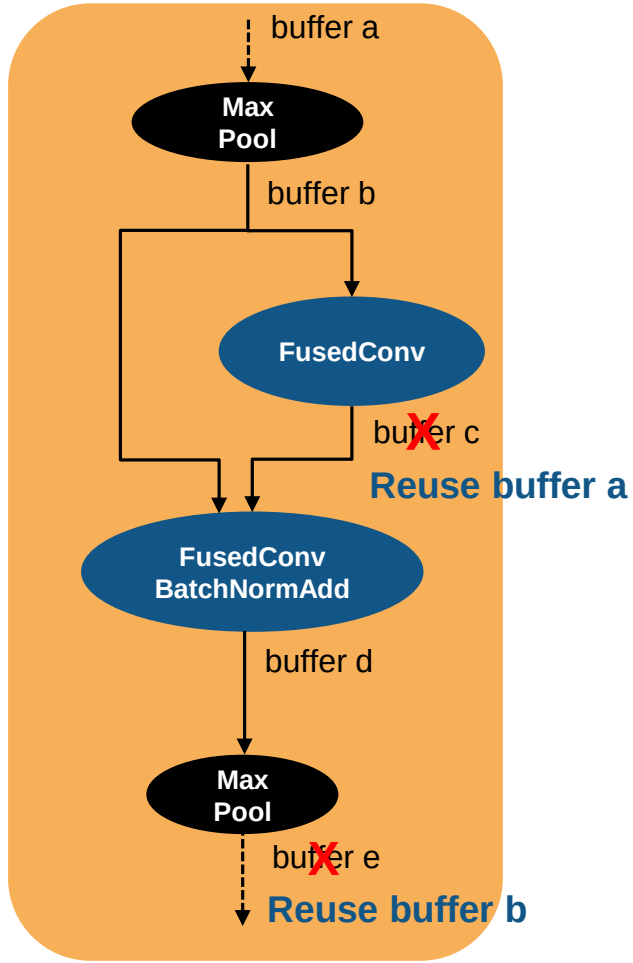
Performance
1. 最適化ライブラリの使用
2. ネットワークの最適化
3. コーディングパターン



Network



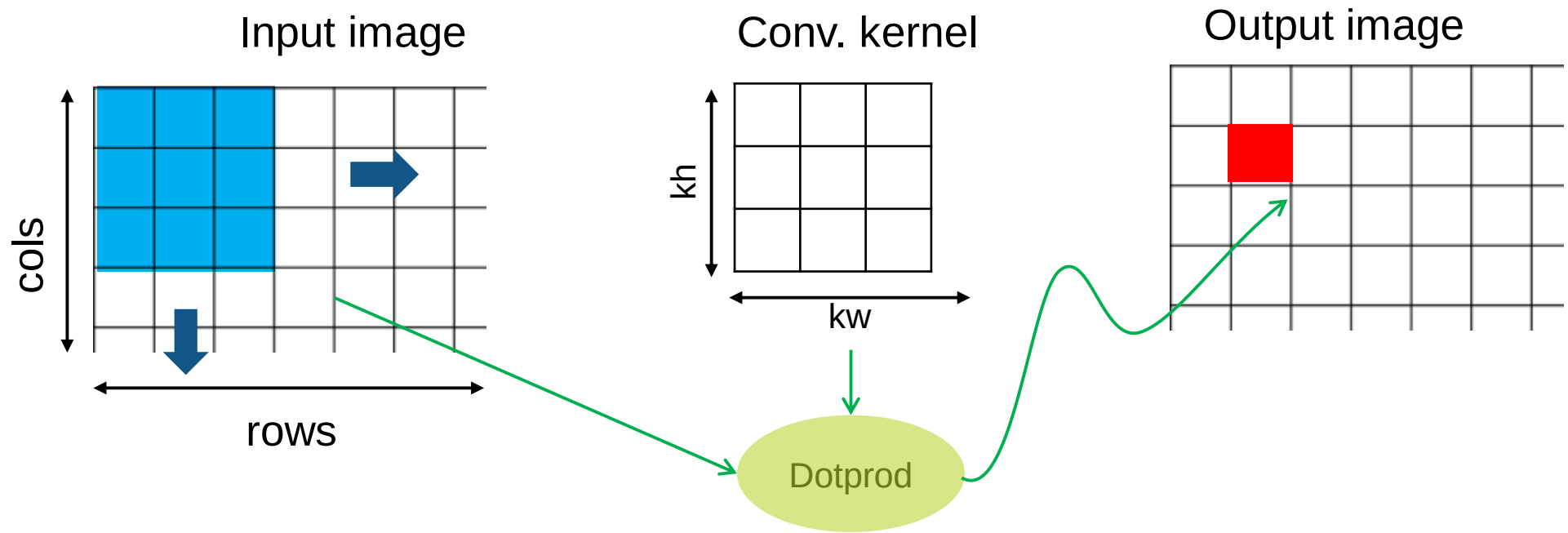
レイヤーフュージョン



バッファの最小化

コーディングパターン：ステンシルカーネル

- 画像処理系関数にはGPUによるステンシル演算が自動適用
  - 関数例： imfilter, imerode, imdilate, conv2, …
- マニュアルでの指定には *gpucoder.stencilKernel()*



コード生成でサポートされるレイヤはリリースごとに増加中

		GPU Coder™		MATLAB Coder™	
	Layer Name	cuDNN	TensorRT	Intel MKL-DNN	ARM Compute
	AdditionLayer	✔	✔	✔	✔
	AveragePooling2DLayer	✔	✔	✔	✔
	BatchNormalizationLayer	✔	✔	✔	✔
	ClassificationOutputLayer	✔	✔	✔	✔
	ClippedReLULayer	✔		✔	✔ (R2019a)
	Convolution2DLayer	✔	✔	✔	✔
R2019a	Crop2DLayer	✔ (R2019a)	✔ (R2019a)	✔ (R2019a)	
	CrossChannelNormalizationLayer	✔	✔	✔	✔
	DepthConcatenationLayer	✔	✔ (R2019a)	✔	✔
	DropoutLayer	✔	✔	✔	✔
	FullyConnectedLayer	✔	✔	✔	✔
	ImageInputLayer	✔	✔	✔	✔
	LeakyReLULayer	✔	✔	✔	✔ (R2019a)
	MaxPooling2DLayer	✔	✔	✔	✔
	MaxUnpooling2DLayer	✔	✔	✔ (R2019a)	
	PixelClassificationLayer	✔	✔	✔	✔
	ReLULayer	✔	✔	✔	✔
	RegressionOutputLayer	✔	✔	✔	✔
	SoftmaxLayer	✔	✔	✔	✔
	TransposedConvolution2DLayer	✔	✔	✔	
R2019a	YOLOv2OutputLayer	✔ (R2019a)	✔ (R2019a)	✔ (R2019a)	✔ (R2019a)
R2019a	YOLOv2ReorgLayer	✔ (R2019a)	✔ (R2019a)		
R2019a	YOLOv2TransformLayer	✔ (R2019a)	✔ (R2019a)		
	(Keras Layer)FlattenCStyleLayer	✔	✔		
	(Keras Layer)GlobalAveragePooling2dLayer	✔	✔	✔	✔
	(Keras Layer)SigmoidLayer	✔	✔		
	(Keras Layer)TanhLayer	✔	✔		
	(Keras Layer)ZeroPadding2dLayer	✔	✔		

R2019a

✔: R2019aからコード生成可能なレイヤー

サポートされるネットワーク(GPU編)

R2019a

		CuDNN	TensorRT
Image Classification		VGG, ResNet, SqueezeNet, GoogLeNet	VGG, ResNet SqueezeNet, GoogLeNet (R2019a)
Encoder/Decoder Semantic Segmentation		DenoiseNet, SegNet FCN (R2019a)	DenoiseNet, SegNet FCN (R2019a)
Object detection ONNX/Keras imported networks		YOLOv2 (R2019a) Keras (R2019a)	YOLOv2 (R2019a) Keras (R2019a)

サポートされるネットワーク(CPU編)

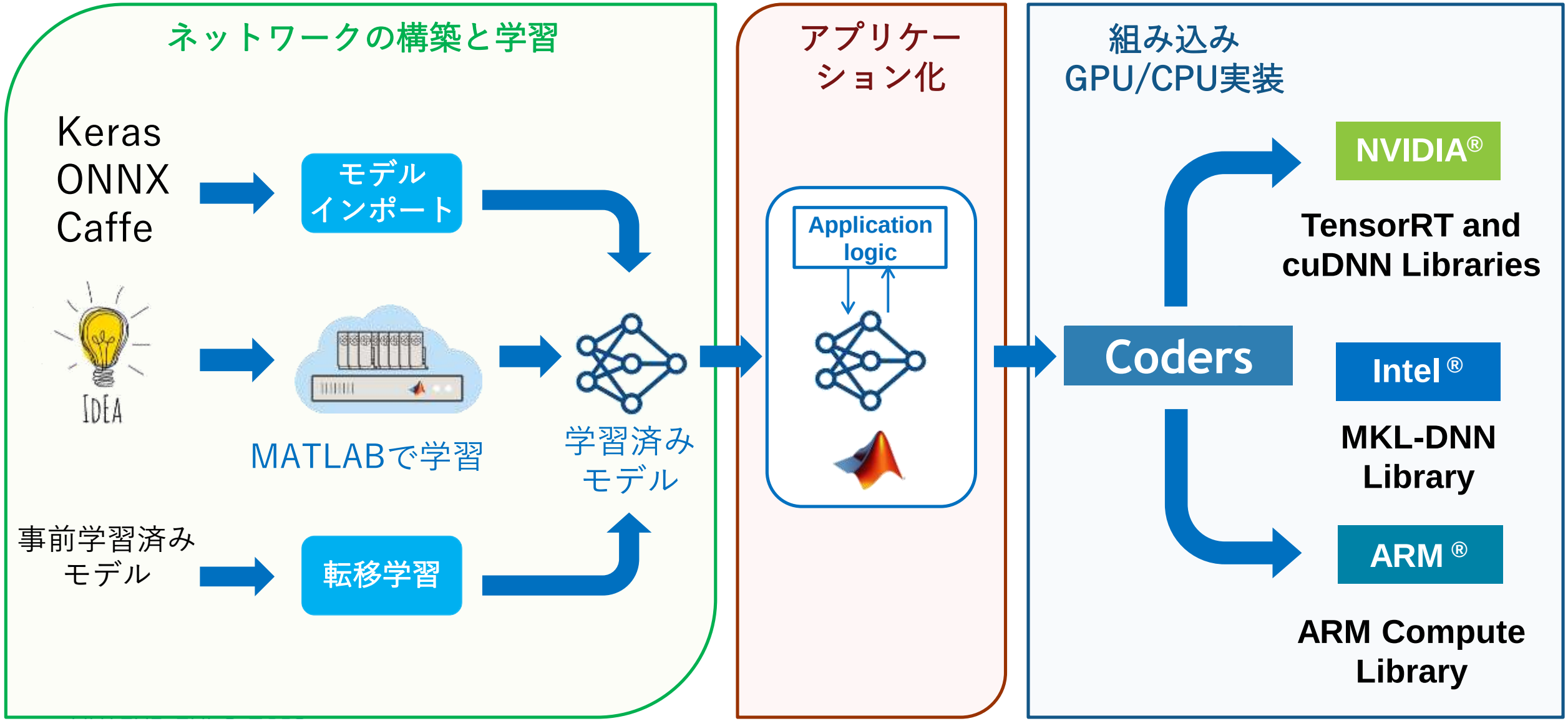
R2019a

		Intel MKLDNN	ARM CPUs
Image Classification		VGG, ResNet, SqueezeNet, GoogLeNet	VGG, ResNet SqueezeNet, GoogLeNet
Encoder/Decoder Semantic Segmentation		DenoiseNet Unet, SegNet (R2019a)	DenoiseNet

アジェンダ

- MATLAB Coder/GPU Coderの概要
- ディープニューラルネットワークの組み込み実装ワークフロー
- パフォーマンスに関して
- まとめ

まとめ


ネットワークの構築と学習
Keras
ONNX
Caffe
モデル
インポート
IDEA
MATLABで学習
学習済み
モデル
事前学習済み
モデル
転移学習
アプリケーション化
Application logic
組み込み
GPU/CPU実装
Coders
NVIDIA®
TensorRT and
cuDNN Libraries
Intel ®
MKL-DNN
Library
ARM ®
ARM Compute
Library