

ディープラーニングの組み込み機器実装ソリューション ~GPU/CPU編~

MathWorks Japan
アプリケーションエンジニアリング部
大塚 慶太郎
Kei.Otsuka@mathworks.co.jp



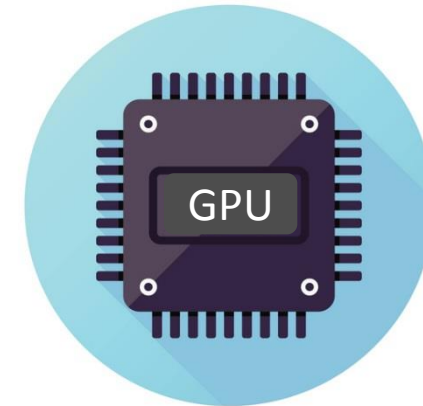
自動運転：車、歩行者等の物体認識、白線検出



モデル

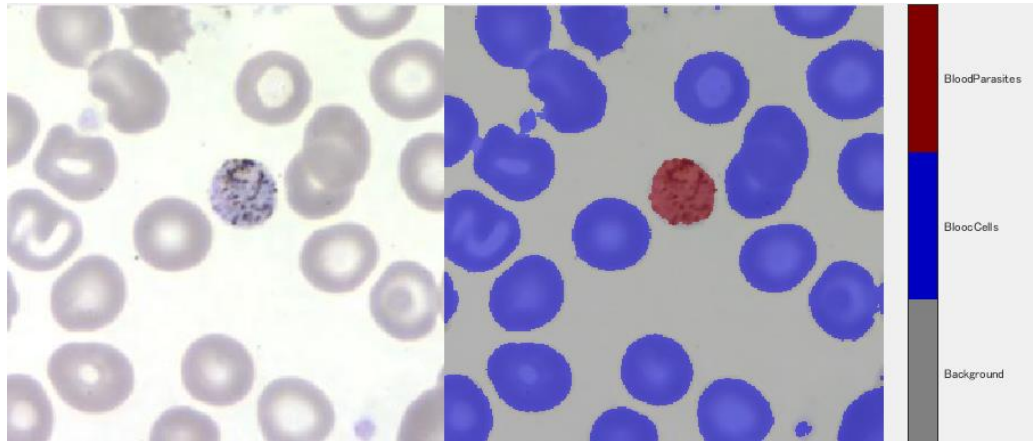


“組み込みGPUへの実装”



実装/配布

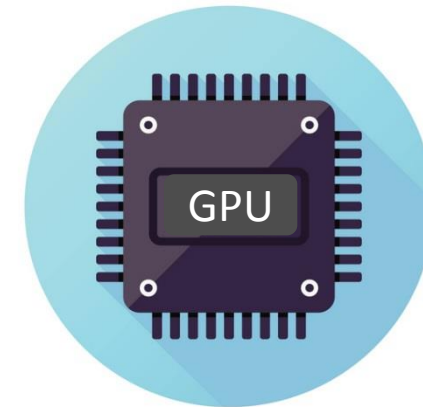
医用画像：腫瘍等、特定の部位の検出



モデル



“組み込みGPUへの実装”

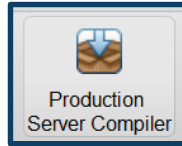
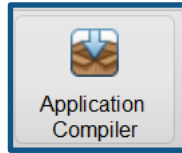
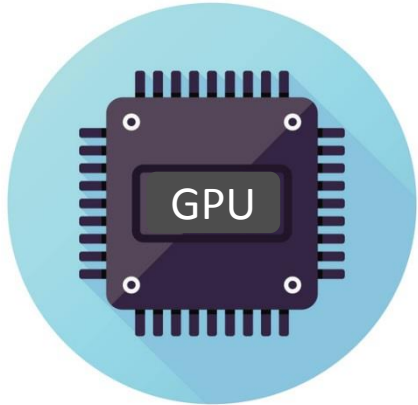


実装/配布

画像ソース：米国CDC DPDx Parasite Image Library
<https://www.cdc.gov/dpdx/malaria/index.html>

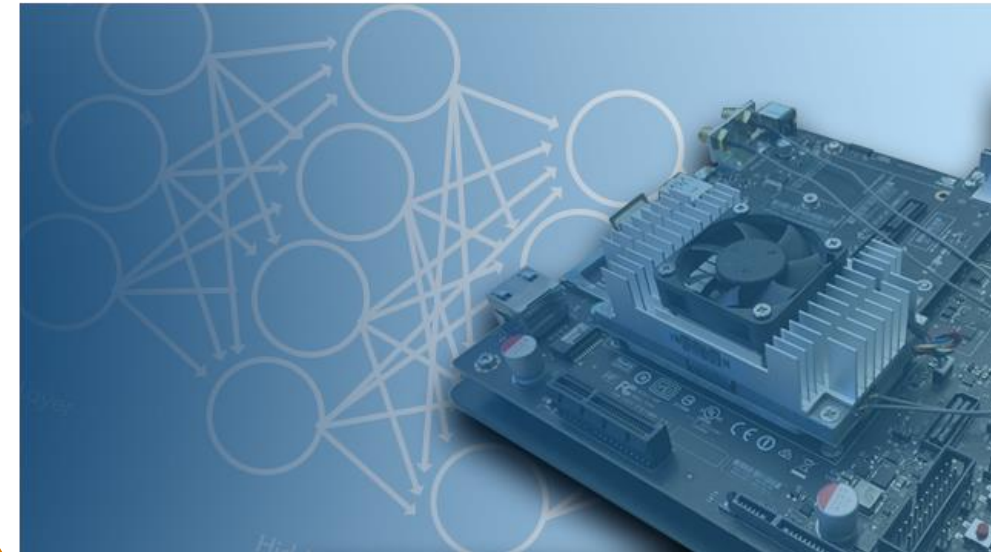
ディープラーニング 実装ソリューション

“組み込みGPU
への実装”



- デスクトップアプリケーション
- Web/エンタープライズアプリケーション

GPU Coder™



実装/配布

学習済みモデルのシェア
機器・デバイスへの実装



- NVIDIA® GPUs
- Intel Xeon® Processors
- ARM® CPUs



Agenda

- Introduction
- MATLAB上でのディープラーニング開発フロー
- GPU Coder™による効率的なGPU/CPU実装
- すぐに試せる例題集
- まとめ

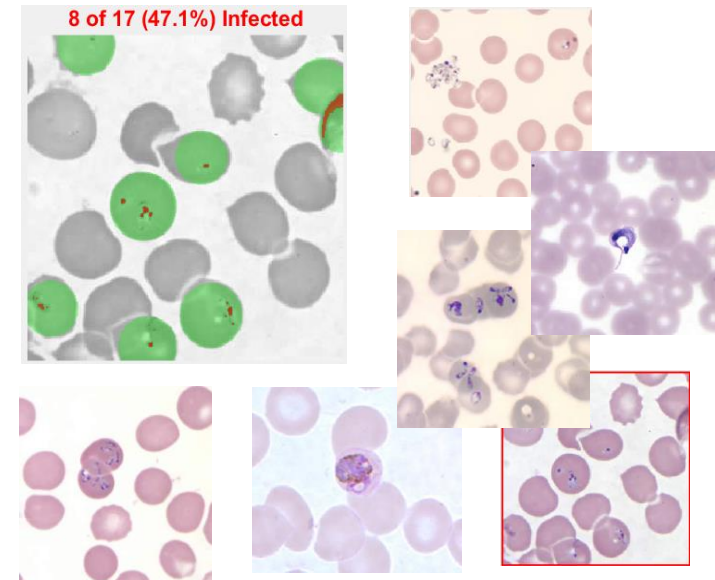
転移学習を使った画像分類

Deep Learning for Image Classification



Demo : 血液検査画像の分類

- 3つの寄生感染症を分類
 - バベシア
 - マラリア原虫
 - トリパノソーマ
- 従来手法(局所特徴+SVM)では~70%程度の分類精度



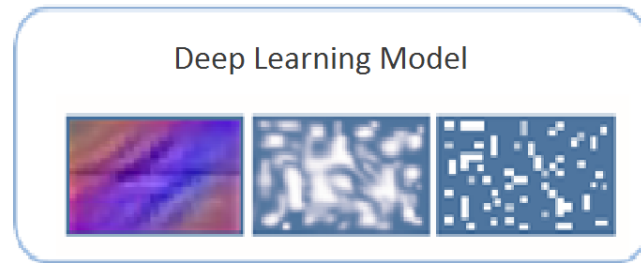
コンピュータビジョン向けディープラーニング ワークフロー

“膨大なデータの
取り扱い”



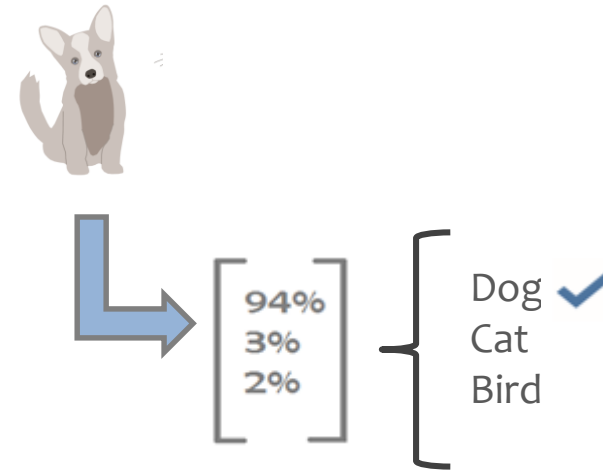
データアクセス

“学習済み
ネットワークの取り込み”



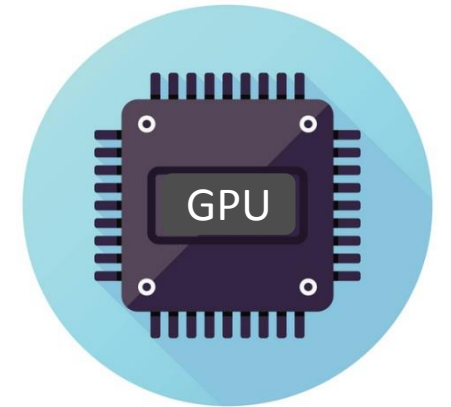
モデル

“マルチGPU、クラスタ環境を
使った効率的な学習”



学習

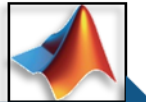
“組み込み機器
への実装”



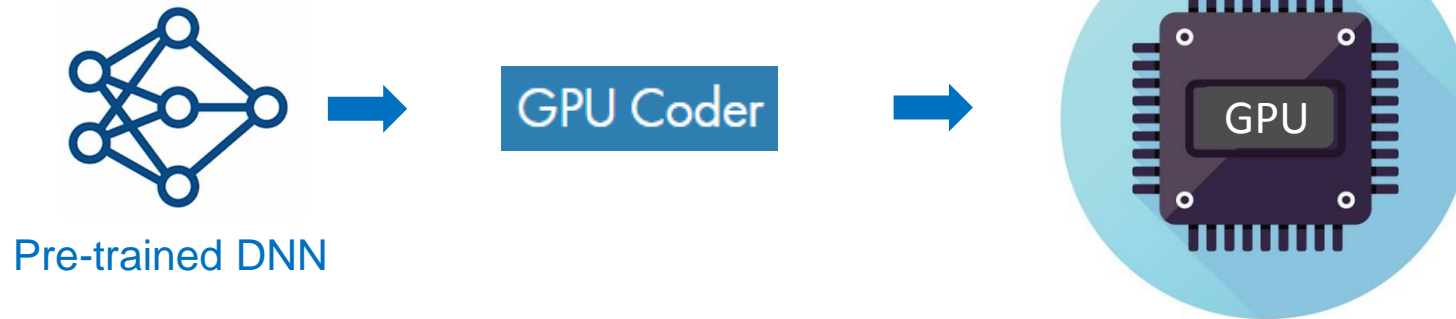
実装/配布

ワークフローの各ステップで
MATLABの使いやすさ、豊富な機能が開発者を強力にサポートします

コンピュータビジョン向けディープラーニング ワークフロー : 実装



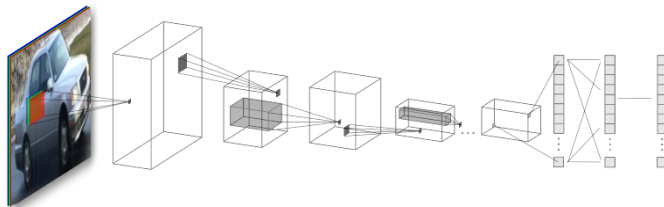
Demo



実装/配布

学習済みモデルのシェア
機器・デバイスへの実装

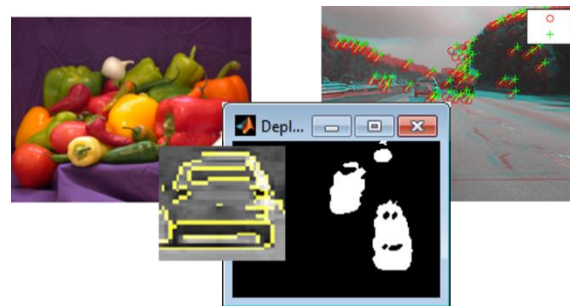
Deep Neural Networks
Deep Learning, machine learning



7x faster than state-of-art

Image Processing and Computer Vision

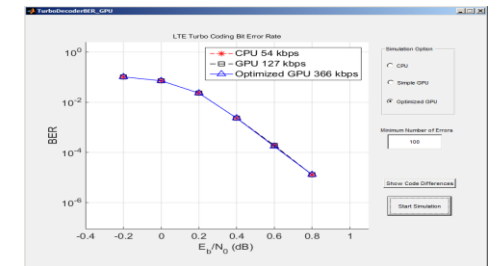
Image filtering, feature detection/extraction



700x faster than CPUs
for feature extraction

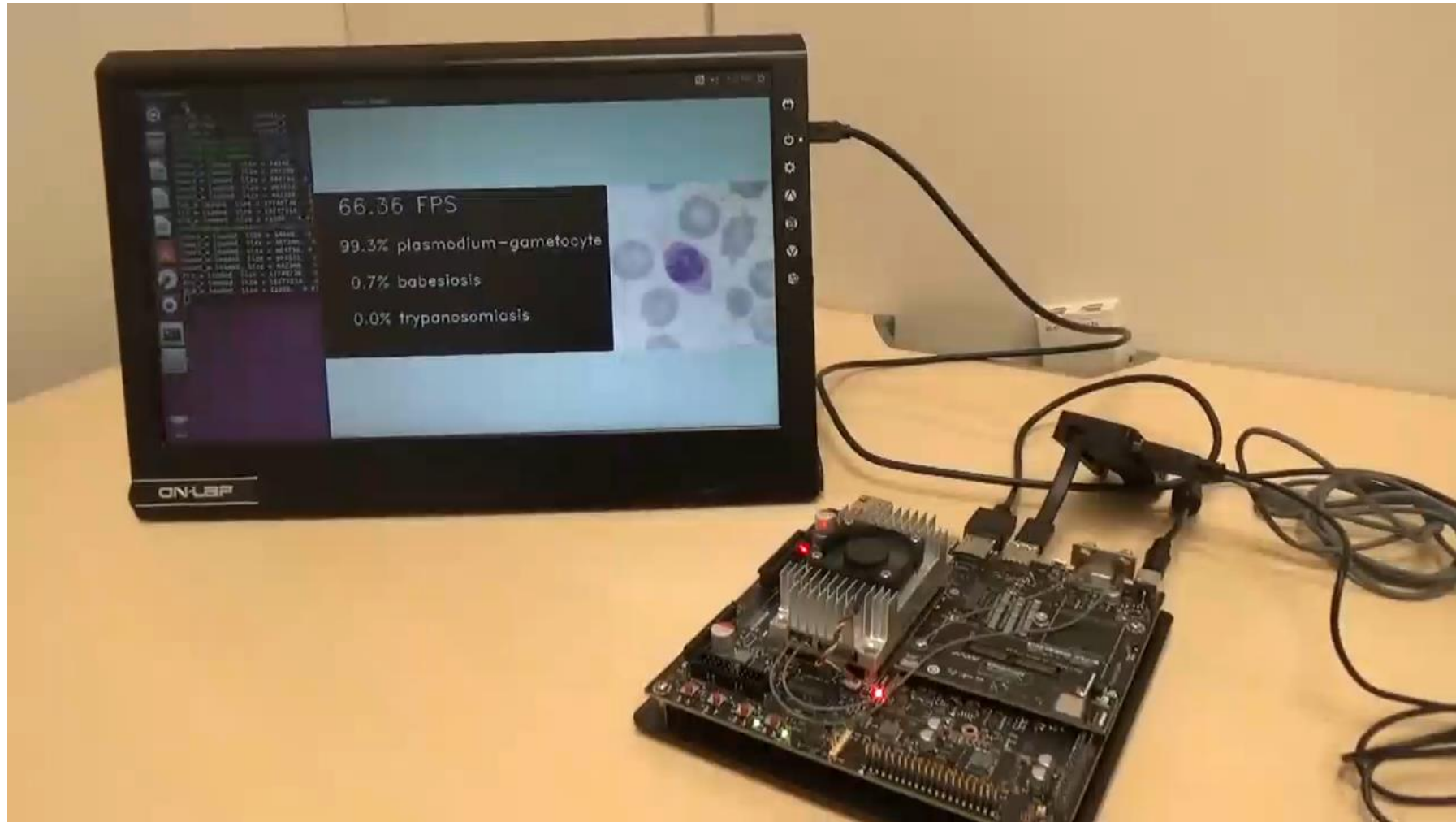
Signal Processing and Communications

FFT, filtering, cross correlation,



20x faster than
CPUs for FFTs

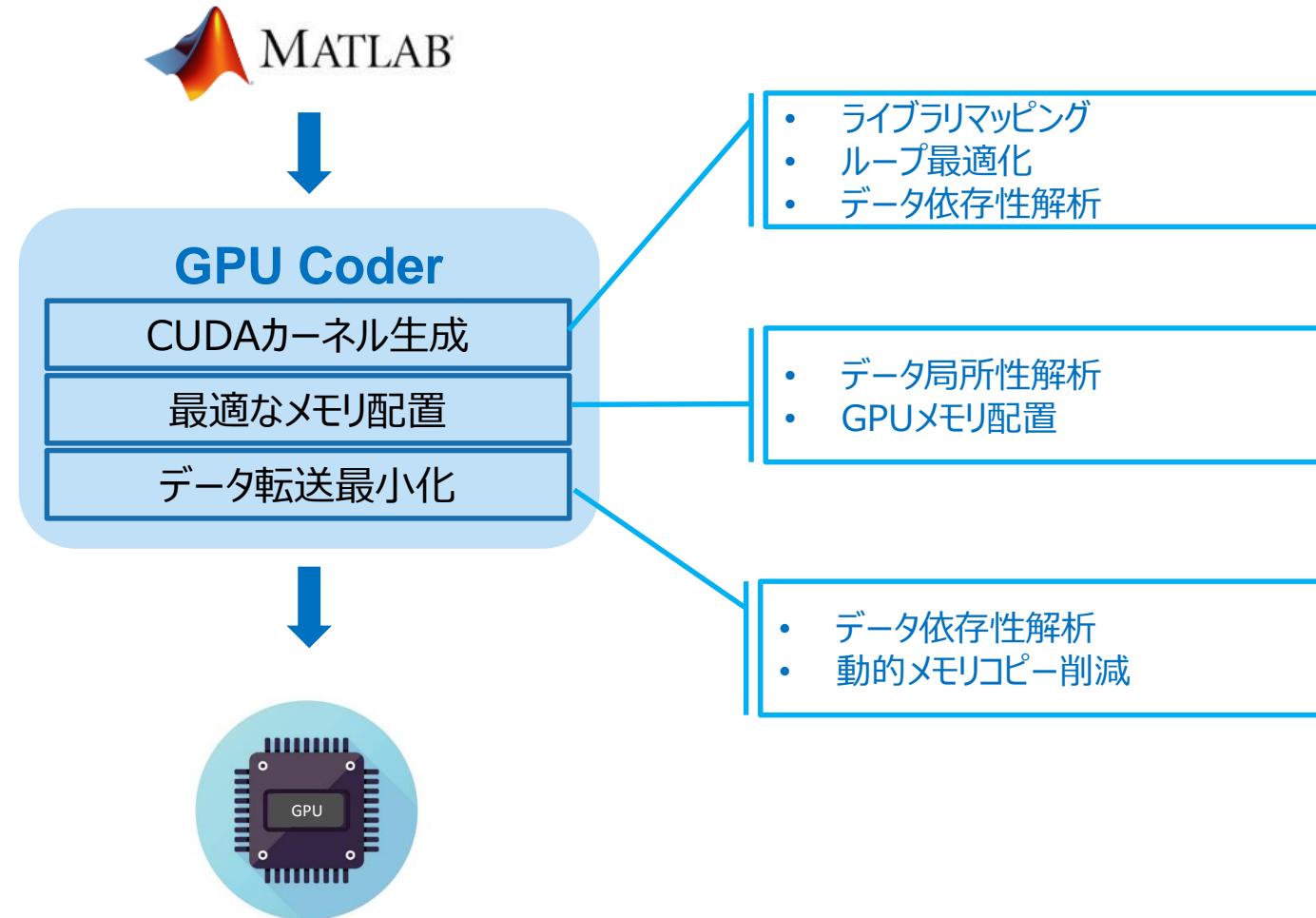
MATLAB®から組み込みGPUへの実装：血液検査画像の分類



Agenda

- Introduction
- MATLAB上でのディープラーニング開発フロー
- GPU Coder™による効率的なGPU/CPU実装
- すぐに試せる例題集
- まとめ

GPU Coder : 高度な関数解析機能が効率の良いコード生成を実現



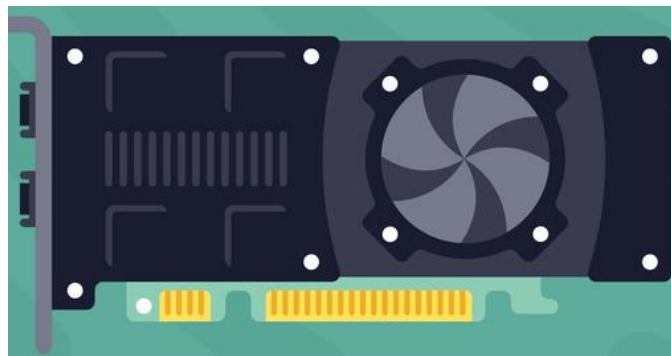
ホスト・カーネル両方のプログラムを生成可能

MATLAB Coder™のコード生成機能を利用



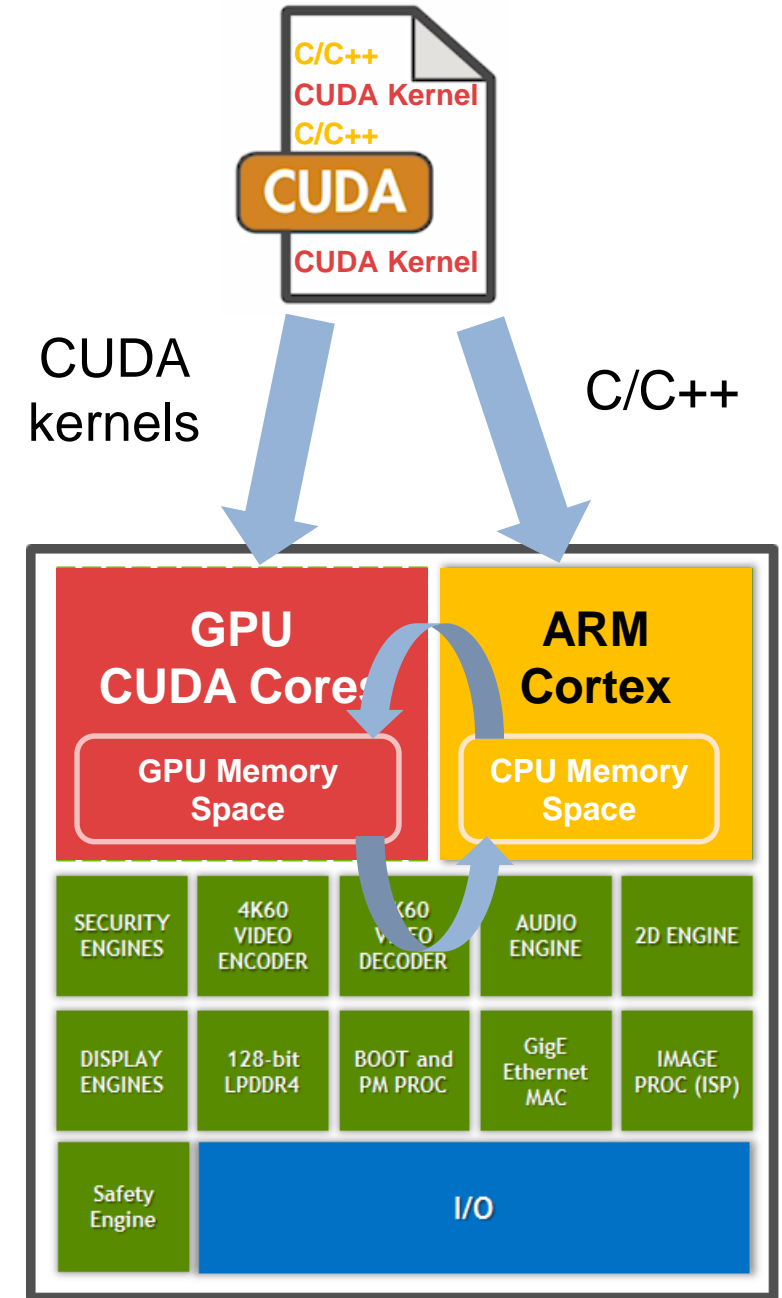
CPU

周波数：~4GHz
コア数：~24
シーケンシャルな処理が得意

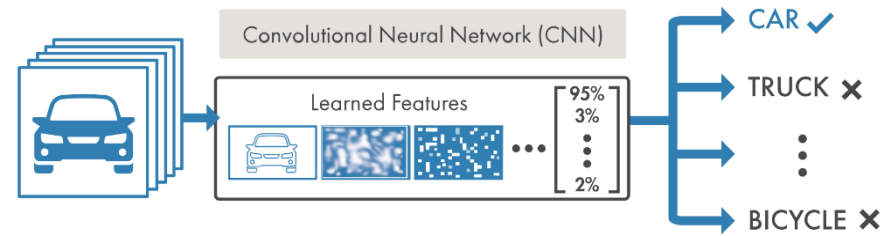


GPUアクセラレータ

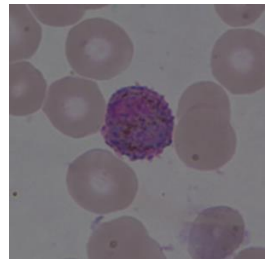
周波数：~1.5GHz
コア数：~6000
並列処理が得意



ディープラーニングを含む、ビジョン系アルゴリズムの実装が可能



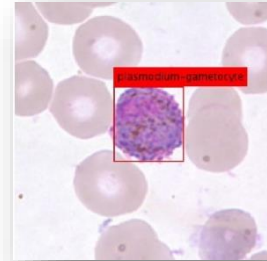
前処理
(コントラスト調整等)



学習済み
ネットワーク



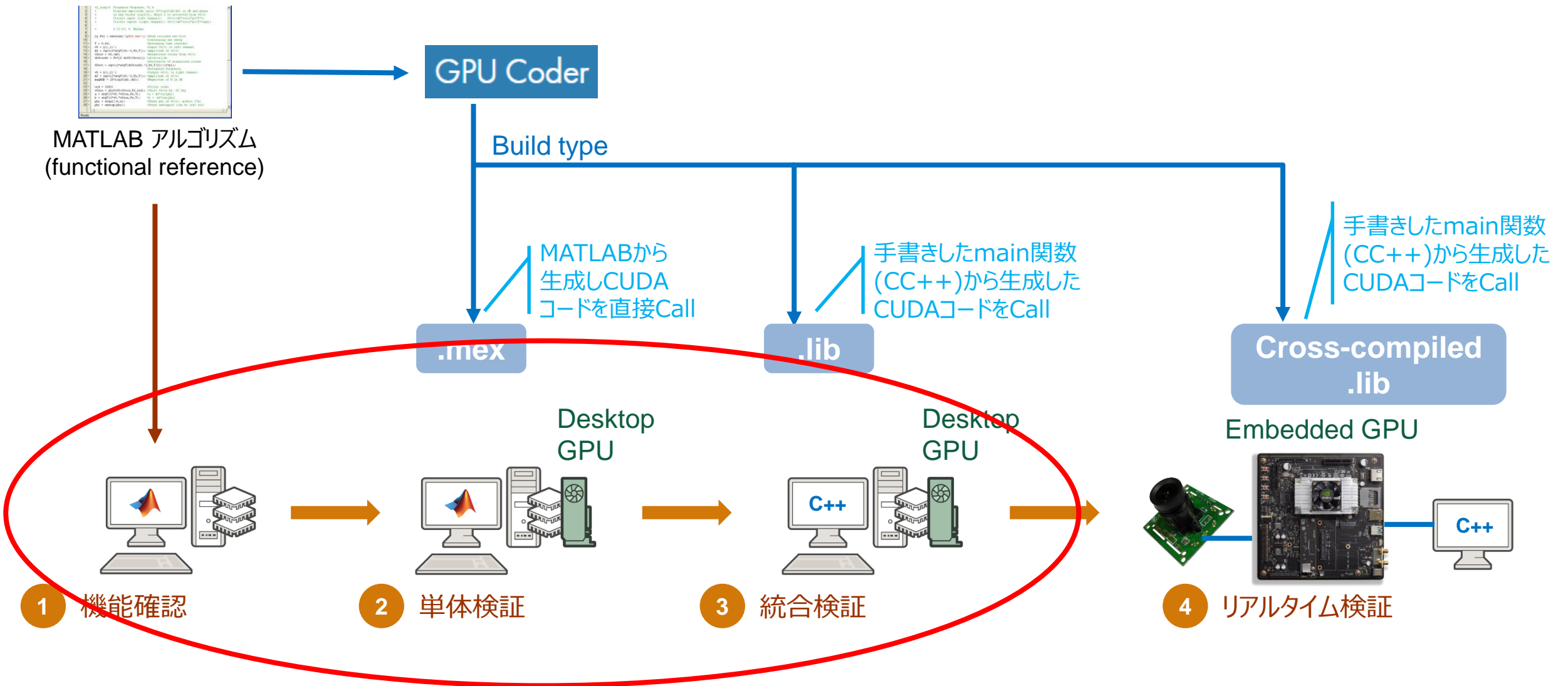
後処理
(ROI抽出等)



画像前処理・後処理+ディープラーニングで
コード生成可能!

MATLAB

組み込み機器への実装ワークフロー



GPU Coderの使い方 : コードの準備 - DNNを含まない場合

- 専用プラグマを挿入し、コード生成対象となる関数を明示

```
function s = vecSum(v)
    s = 0;
    coder.gpu.kernelfun();
    for i = 1:length(v)
        s = s + v(i);
    end
end
```

以下の関数でもカーネル生成が行われます

```
function s = vecSum(v)
    coder.gpu.kernelfun();
    s = sum(v);
end
```

GPU Coderの使い方 : コードの準備 - DNNに対してコード生成を行う場合

- 予めDNNモデルをmat形式で保存し、専用関数でロード

```
function out = myBsnet(in)

    persistent mynet;

    if isempty(mynet)
        mynet = coder.loadDeepLearningNetwork('netTransfer.mat', ...
            'netTransfer');
    end

    out = mynet.predict(in);
```

*コード生成に利用できるメソッドはpredict, activationsのみ(R2018a時点)

GPU Coderの使い方：専用Appを利用してコード生成



GPU Coder - fog_rectification.prj

Select Define Check Generate Finish

1

GPU Coder

Entry-Point Functions:

fog_rectification

+ Add Entry-Point Function

GPU Coder - fog_rectification.prj

Define Input Types

2

To convert MATLAB to GPU, you must define the type of each input for every entry point function. [Learn more](#)

To **automatically define input types**, call fog_rectification or enter a script that calls fog_rectification in the MATLAB prompt below:

>> run_fog_rectification

Autodefine Input Types

GPU Coder - fog_rectification.prj

Generate Code GENERATE VERIFY CODE

3

Build type: Source Code

Output file name: Source Code

Language: MEX

Hardware Board: Dynamic Library (.dll)

Device: Executable (.exe)

Toolchain: Automatically locate an installed toolchain

More Settings Generate

Back Next

Code Generation Report

MATLAB code Call stack C code

Static Code Metrics Report

Target Source Files

- fog_rectification.cu
- fog_rectification.h
- fog_rectification_initialize.cu
- fog_rectification_initialize.h
- fog_rectification_terminate.cu
- fog_rectification_terminate.h
- fog_rectification_types.h
- rtGetInf.cu
- rtGetInf.h
- rtGetNaN.cu
- rtGetNaN.h
- rt_nonfinite.cu
- rt_nonfinite.h
- rtwtypes.h

GPU Static Metrics Report

GPU Kernels

- fog_rectification_kernel1
- fog_rectification_kernel2
- fog_rectification_kernel3
- fog_rectification_kernel4
- fog_rectification_kernel5
- fog_rectification_kernel6
- fog_rectification_kernel7
- fog_rectification_kernel8
- fog_rectification_kernel9
- fog_rectification_kernel10
- fog_rectification_kernel11
- fog_rectification_kernel12
- fog_rectification_kernel13
- fog_rectification_kernel14
- fog_rectification_kernel15

File: fog_rectification.cu

```

1 /*
2  * Prerelease License - for engineering feedback and testing purposes
3  * only. Not for sale.
4  * File: fog_rectification.cu
5  *
6  * MATLAB Coder version      : 3.4
7  * C/C++ source code generated on : 05-Sep-2017 09:14:16
8  */
9
10 /* Include Files */
11 #include "rt_nonfinite.h"
12 #include "fog_rectification.h"
13
14 /* Variable Definitions */
15 __constant__ real_T const_b[9];
16
17 /* Function Declarations */
18 static_global__ void fog_rectification_kernel1(const uint8_T *input
19 *b_input);

```

4

Summary All Messages (6) Build Log

C source code generated on: 05-Sep-2017 09:14:28

Coding target: Static Library

Number of errors: 0

Number of warnings: 0

Number of notices: 6

Tell Us What You Think

We value your feedback. Please take a few minutes to answer this short questionnaire regarding the Code Generation Report.

>>Provide Feedback

組み込み機器への実装ワークフロー

```

% Example MATLAB algorithm
function [out] = my_algorithm(in)
% ...
endfunction
    
```

MATLAB アルゴリズム
(functional reference)

GPU Coder

Build type

MATLABから
生成しCUDA
コードを直接Call

手書きしたmain関数
(C++)から生成した
CUDAコードをCall

手書きしたmain関数
(C++)から生成した
CUDAコードをCall

.mex

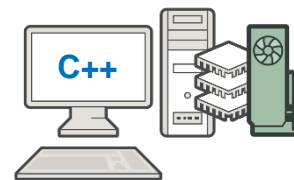
.lib

Cross-compiled
.lib

Desktop
GPU

Desktop
GPU

Embedded GPU



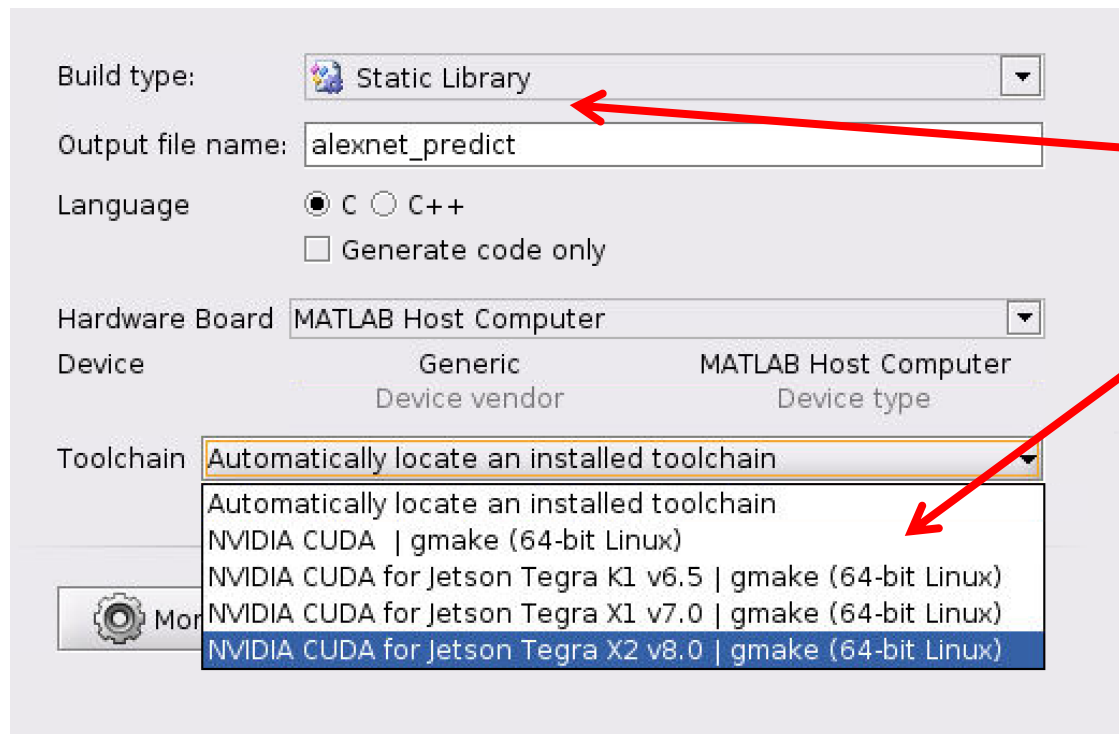
1 機能確認

2 単体検証

3 統合検証

4 リアルタイム検証

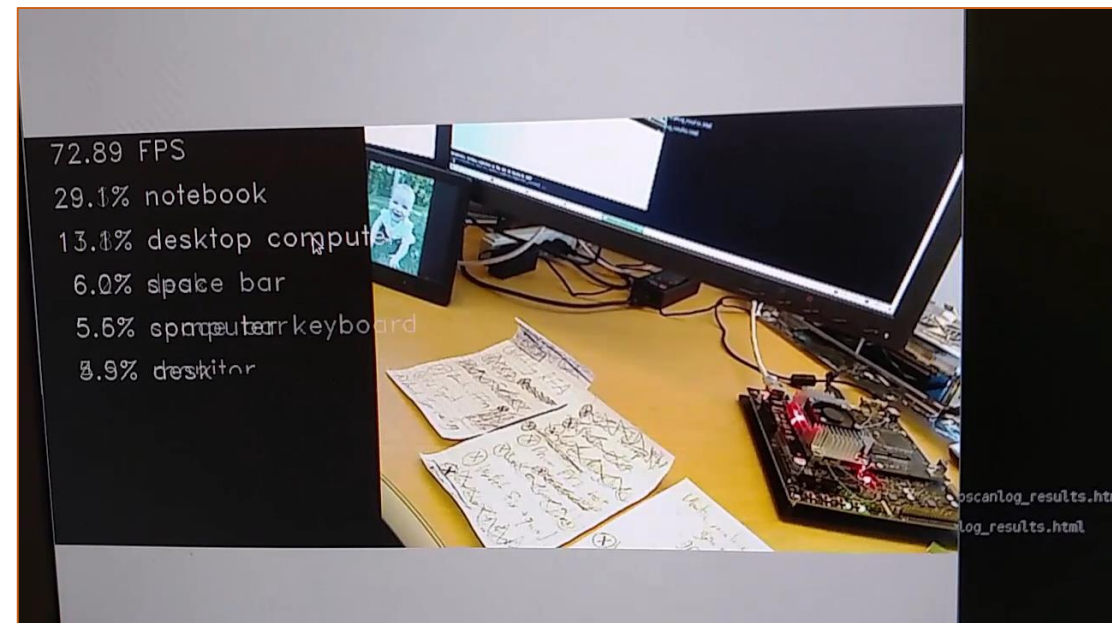
組み込みGPUへの実装：ターゲットハードウェア用にクロスコンパイル



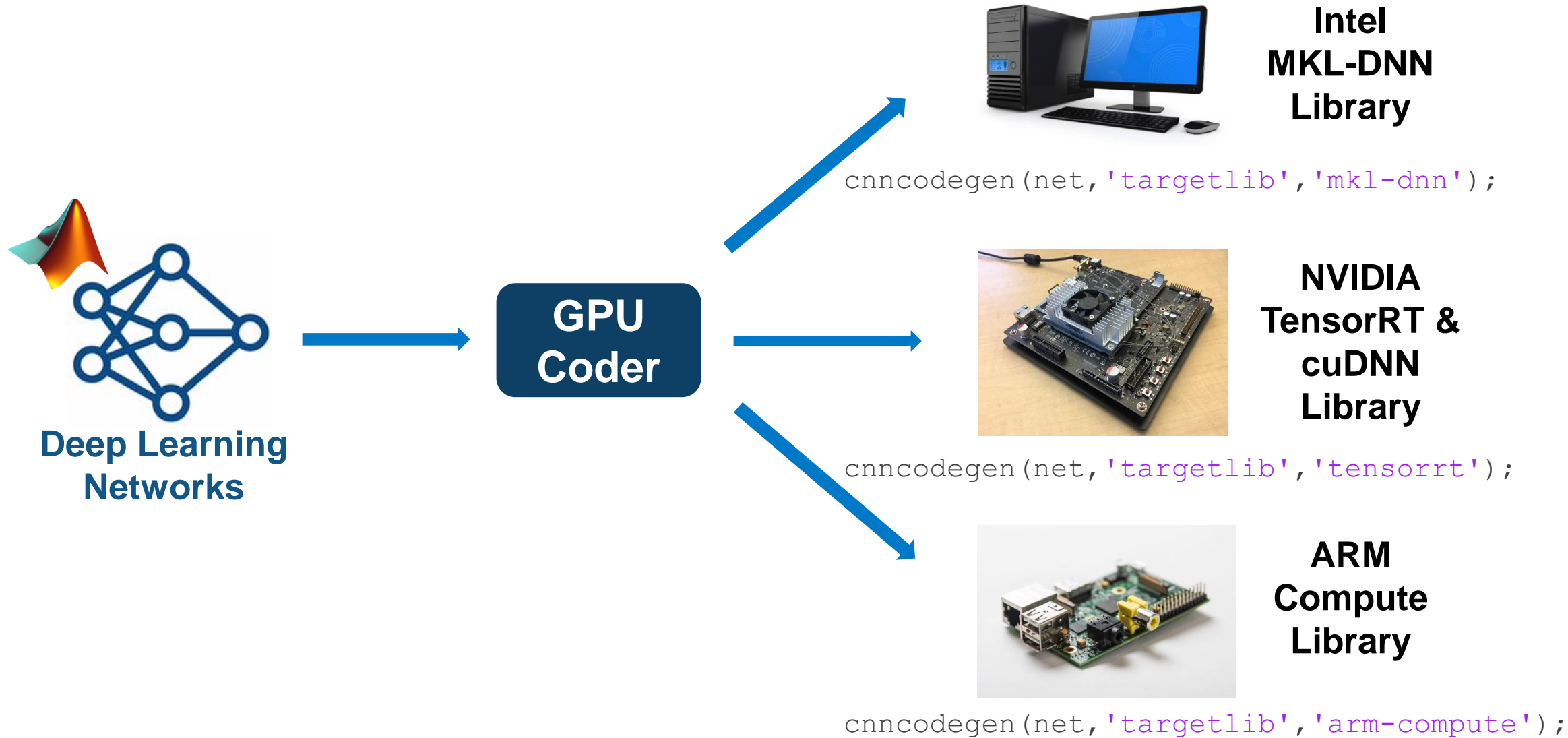
変更が必要な箇所は2つ

1. ビルドタイプを'lib'に変更

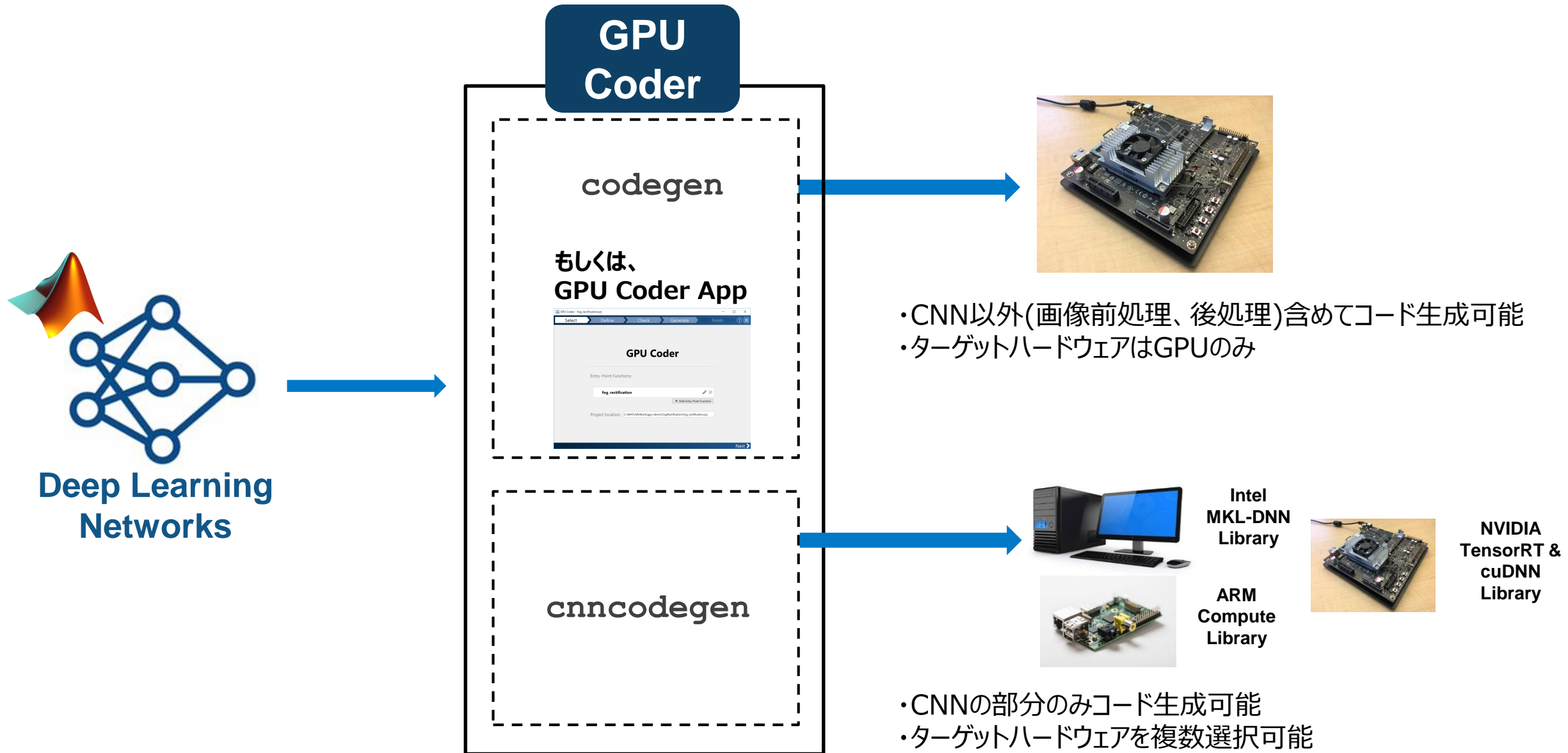
2. 適切なツールチェーンを選択



CPUをターゲットとしたコード生成も可能

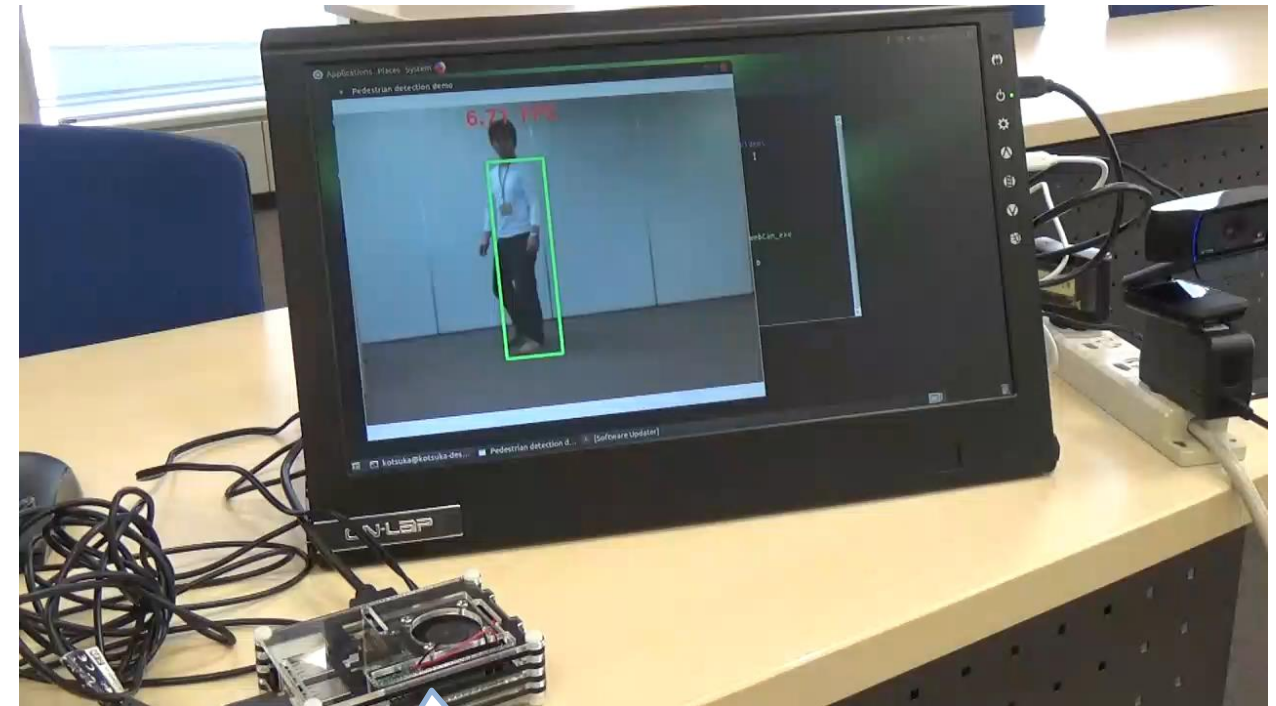


ターゲットハードウェアとコード生成方法の対応について



歩行者検出用ネットワークのRaspberry Pi3実装例

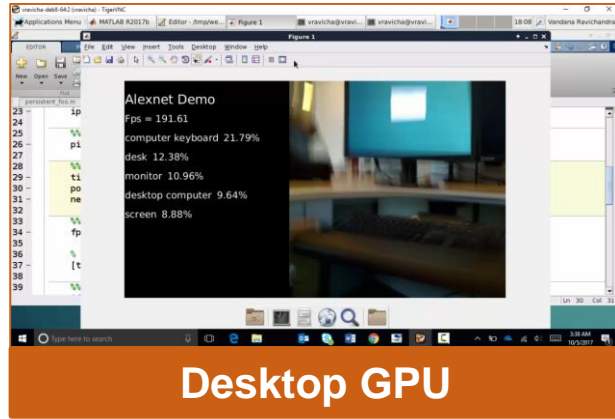
| | UUNet(歩行者検出用ネットワーク) |
|---------------------------|--|
| Technique | 固定サイズの検索窓を利用、歩行者のサイズは固定。 CNNは画像分類の目的でのみ利用 |
| レイヤ数 | 12 |
| ウェイトサイズ | 3 MB |
| Jetson TX2での フレームレート | ~66 FPS |
| RaspberryPi3での フレームレート | ~7 FPS |
| 分類精度 | 85 % |



RaspberryPi3

One MATLAB, 5 hardware platforms

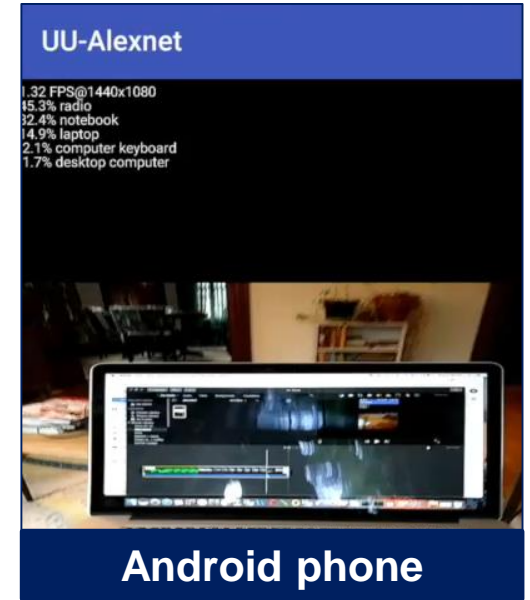
R2017b



Desktop GPU



Embedded GPU



UU-Alexnet

R2018a



Desktop CPU

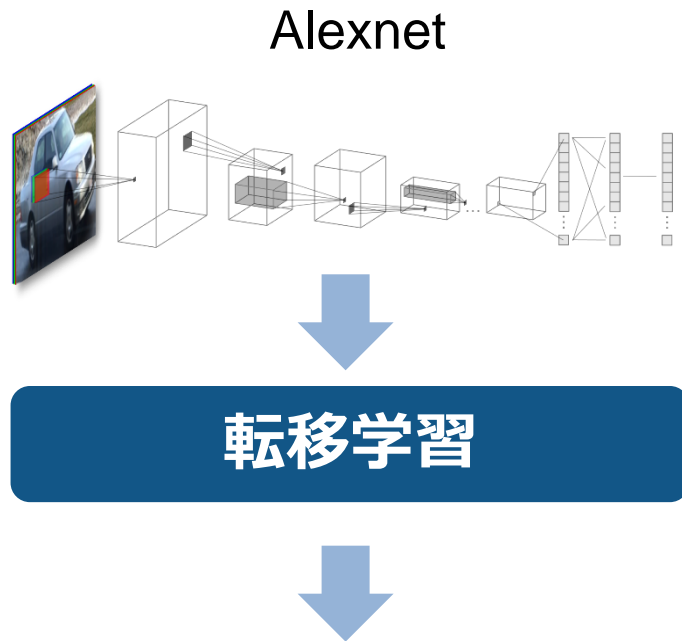


Raspberry Pi board

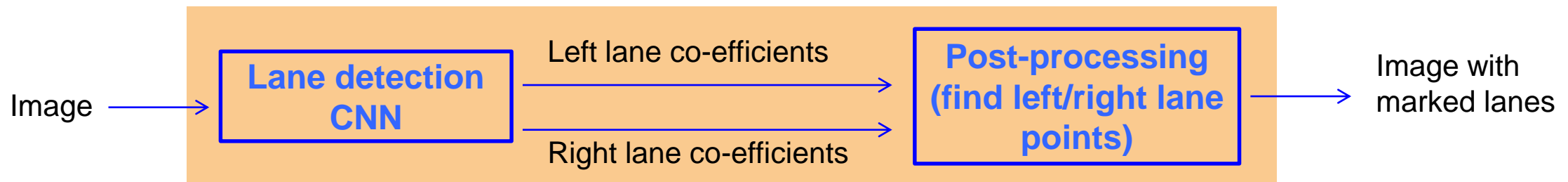
Agenda

- Introduction
- MATLAB上でのディープラーニング開発フロー
- GPU Coder™による効率的なGPU/CPU実装
- **すぐに試せる例題集**
- まとめ

GPU Coderですぐに試せるディープラーニングサンプル

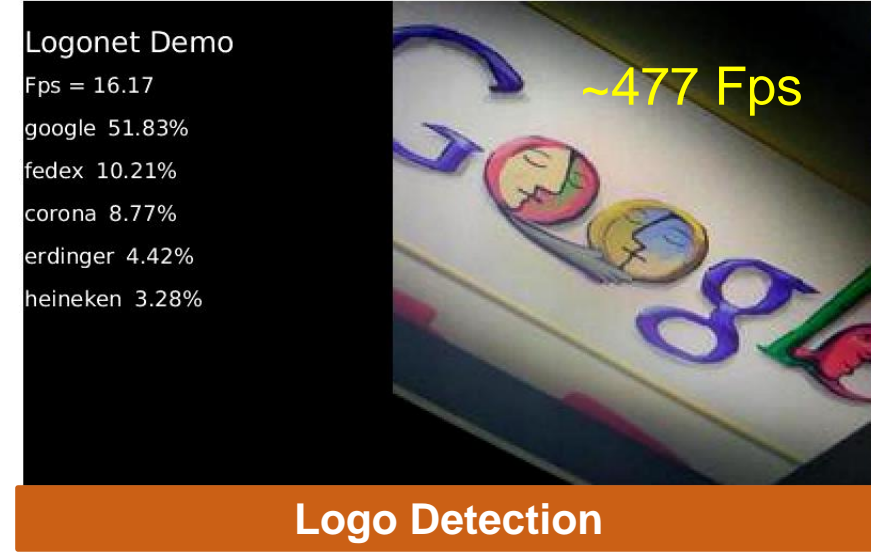


CNNの出力がレーンを表す放物線の係数となる($y = ax^2 + bx + c$)

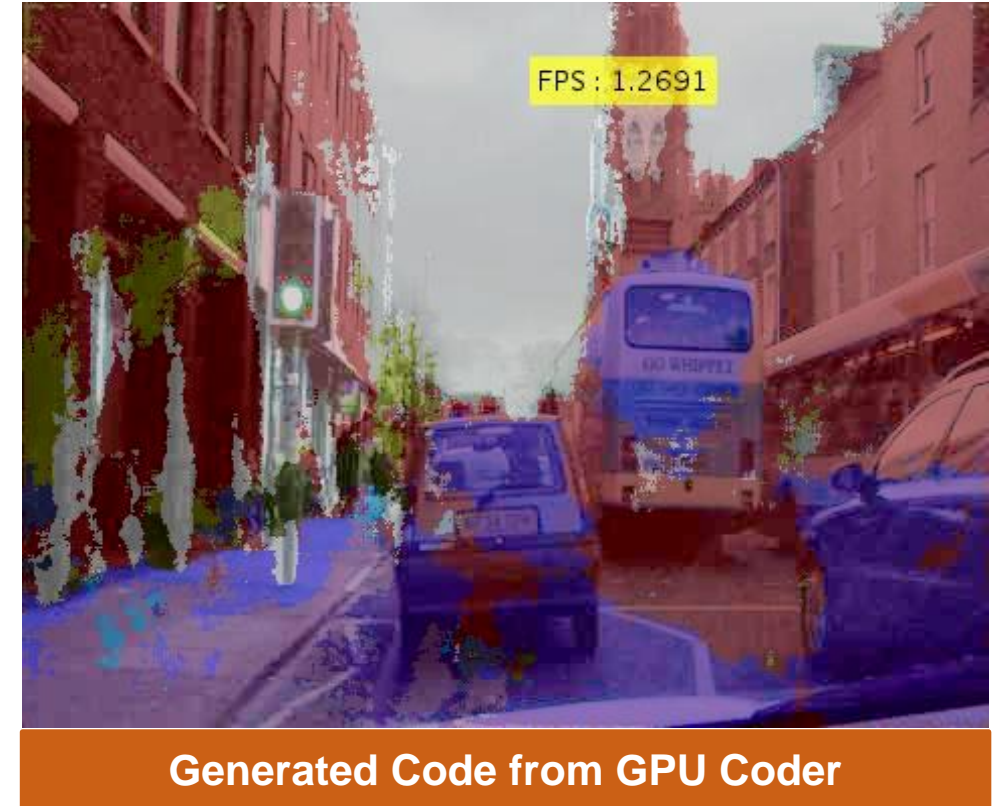
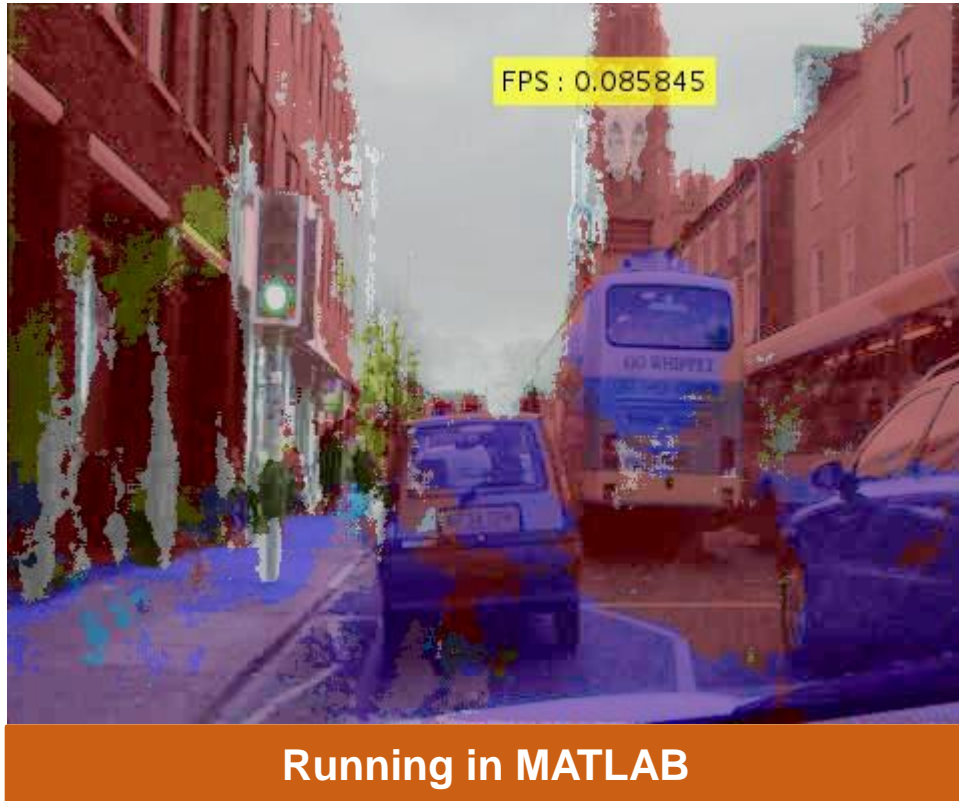


GPU coderを利用し、画像の前処理、後処理を含むアルゴリズム全体をコード生成

GPU Coderですぐに試せるディープラーニングサンプル



GPU Coderですぐに試せるディープラーニングサンプル



GPU Coderで~20倍程度高速化

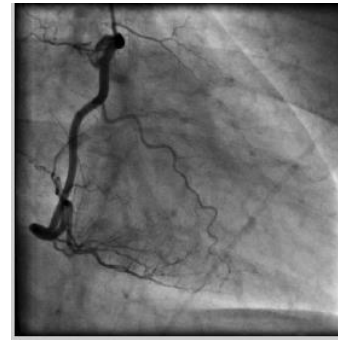
GPU Coderですぐに試せるコンピュータビジョン系サンプル



霧(ノイズ)除去



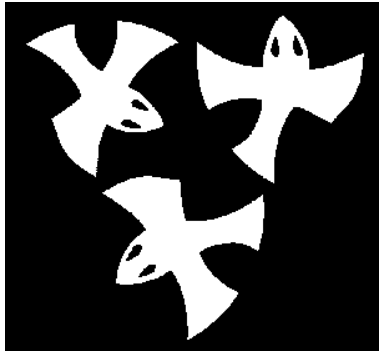
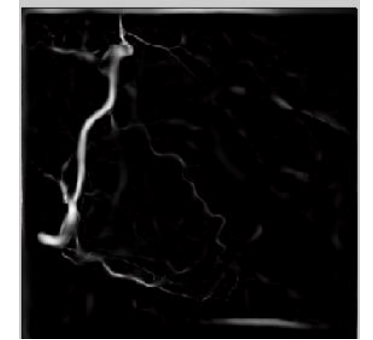
5x speedup



線強調フィルタ



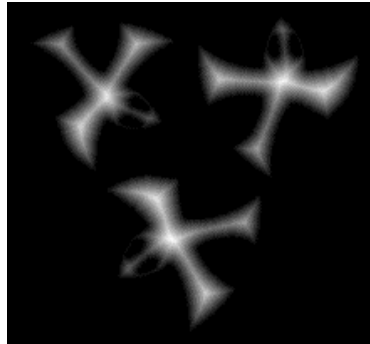
3x speedup



距離変換



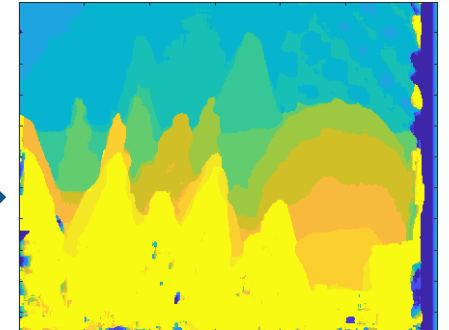
8x speedup



ディスパリティ算出



50x speedup



レイトレーシング



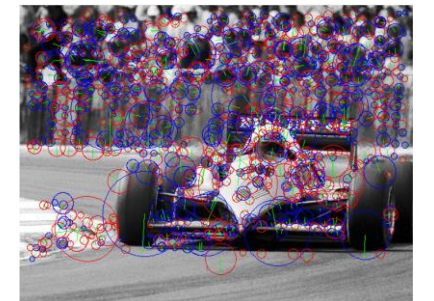
18x speedup



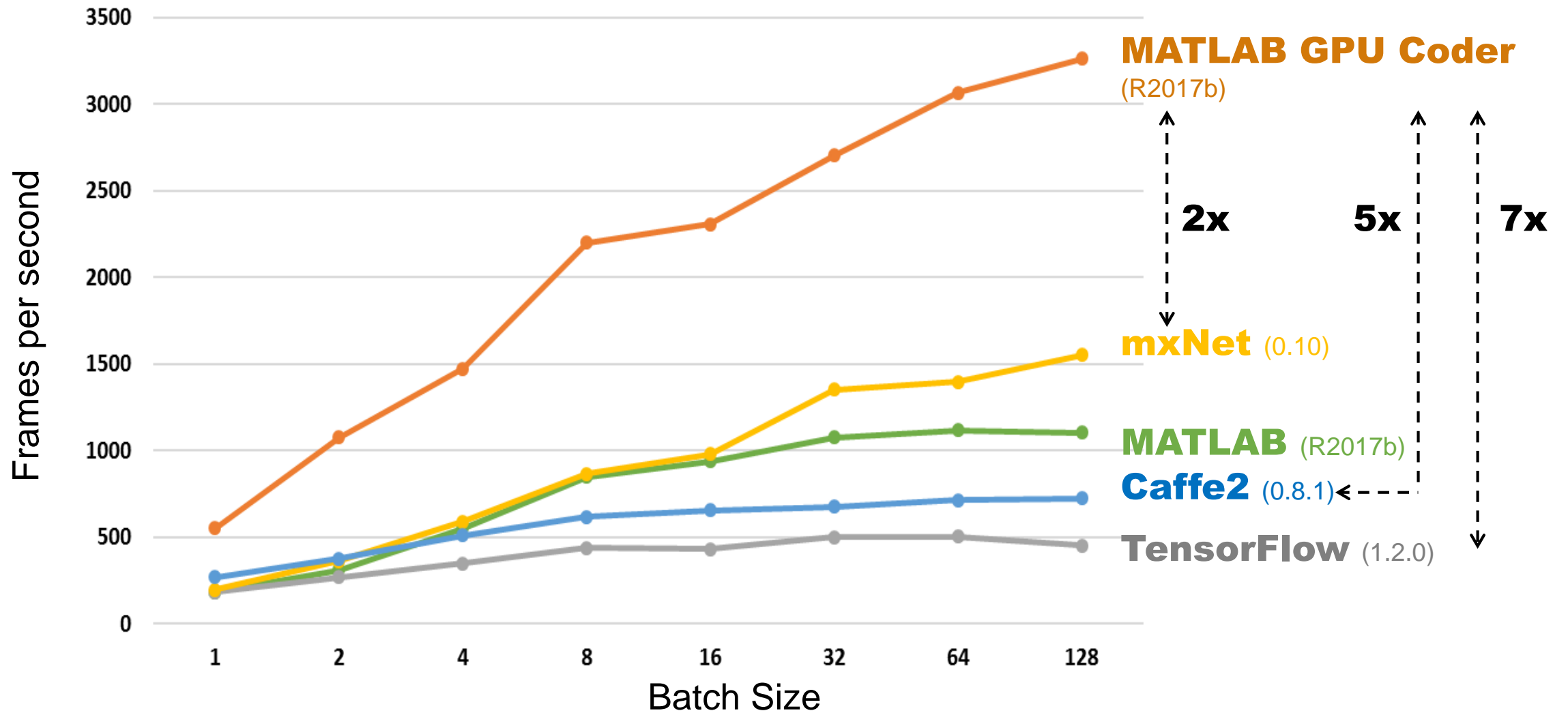
特徴点抽出



700x speedup



Alexnet Inference Frame-Rateパフォーマンス(NVIDIA Titan Xpを利用)



| | |
|-------|---|
| CPU | Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz |
| GPU | Pascal Titan Xp |
| cuDNN | v5 |

Agenda

- Introduction
- MATLAB上でのディープラーニング開発フロー
- GPU Coder™による効率的なGPU/CPU実装
- すぐに試せる例題集
- まとめ

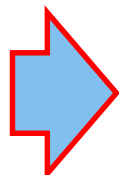
まとめ：ディープラーニングの組み込み機器実装ソリューション ～GPU・CPU実装編～

①統合開発環境MATLAB

- ✓ コンピュータービジョン・ディープラーニングのアルゴリズム開発環境として強力なMATLAB
- ✓ アルゴリズム開発からGPUまで、同一環境上で実現可能

②GPU Coder

- ✓ CUDAの文法を知らなくても自動コード生成でGPUを利用可能
- ✓ エンジニアのスキルに依存しない、再現性の高いコード生成
- ✓ GPU以外のデバイス, ARM系プロセッサにも実装可能
- ✓ すぐに始められるサンプル集

 MATLABを使って、コンピュータービジョン・ディープラーニングのアルゴリズムの開発から実装までを効率的に実現!





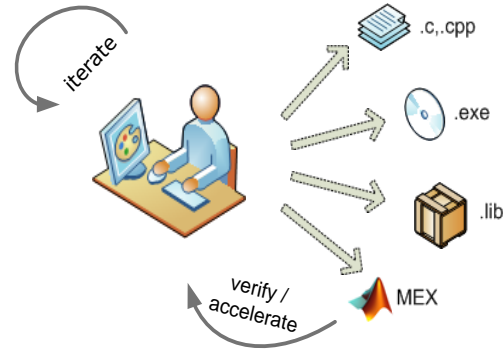
© 2018 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

GPU Coder関連製品

GPU Coderに
必須となります

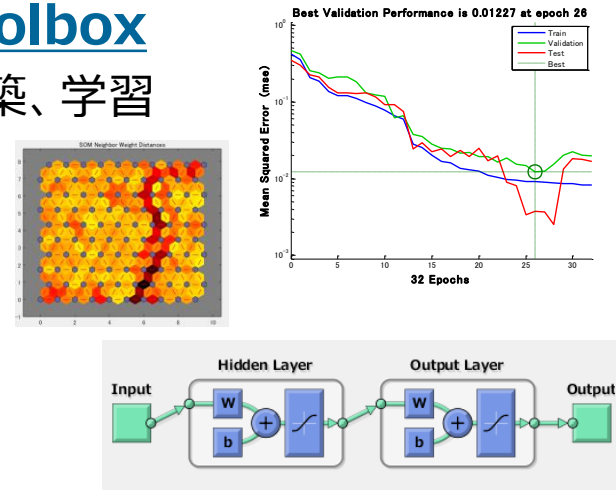
MATLAB Coder™

- MATLABプログラムからC/C++コードを生成
- MATLAB上で、アルゴリズム開発から実装までフローを統合



Neural Network Toolbox

- ニューラルネットワークの構築、学習
- データフィッティング
- クラスタリング
- パターン認識
- 深層学習
- GPUによる計算の高速化



Parallel Computing Toolbox

- MATLAB & Simulink と連携した並列処理
- 対話的な並列計算実行
- GPGPU による高速演算
- ジョブおよびタスクの制御



Embedded Coder®

- MATLABプログラム/Simulinkモデルから組み込み用C/C++コードを自動生成

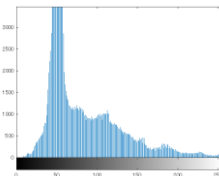
```

#include "structuredflow.h"
#include "Chart.h"
void Chart(void);
void Chart(void)
{
    start();
    do {
        if(input1) {
            one();
        } else if(input2) {
            two();
        } else {
            three();
        }
        loopy();
    } while(looptest());
}
    
```

GPU Coder関連製品：画像処理・コンピュータビジョン

Image Processing Toolbox™

- コーナー、円検出
- 幾何学的変換
- 各種画像フィルタ処理
- レジストレーション（位置合せ）
- セグメンテーション（領域分割）
- 画像の領域の定量評価



Computer Vision System Toolbox™

- カメラキャリブレーション
- 特徴点・特徴量抽出
- 機械学習による物体認識
- 動画ストリーミング処理
- トラッキング
- ステレオビジョン・3D表示

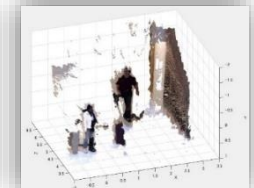
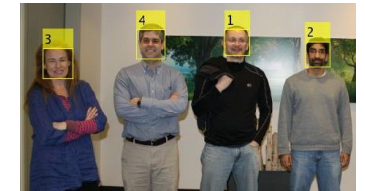
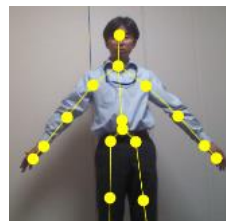


Image Acquisition Toolbox™

- デバイスから画像、動画直接取り込み
 - フレームグラバボード
 - DCAM, Camera Link®
 - GigE Vision®, Webカメラ
 - Microsoft® Kinect® for Windows®



Statistics and Machine Learning Toolbox™

- 機械学習
- 多変量統計
- 確率分布
- 回帰と分散分析
- 実験計画
- 統計的工程管理

