DENSO TEN

2019/5/28

MATLAB EXPO 2019

車載制御ECUへのAI適用における実装開発プロセスの構築

横山　夏軌
株式会社デンソーテン
AE事業本部 先行システム開発部 技術開発室

■アジェンダ

DENSO TEN

- 会社紹介
- 背景と課題
- MATLAB®活用によるAI実装検討
- MATLAB®応用によるAI実装開発プロセスの構築
  - ネットワークモデルのSimulink®モデル変換
  - ECU実装時の問題点検証
  - AI専用ライブラリの開発
  - Simulink®モデルからレイヤー情報への逆変換
- まとめ

DENSO TEN

会社紹介

■会社概要

DENSO TEN

社　名	株式会社デンソーテン （DENSO TEN Limited）
本　社	兵庫県神戸市兵庫区御所通1-2-28
代表者	岩田　悟志（代表取締役社長）
資本金	53億円 （デンソー51%／トヨタ35%／富士通14%）
設　立	1972（昭和47）年10月25日
従業員	単独 2,938名　 連結 10,699名（2018年3月31日現在）

主な事業

コネクティッド事業
安全運転管理テレマティクスサービス(通信型ドライブレコーダー)
緊急通報システム、AIを活用したタクシー需要予測サービス　など

CI事業
ディスプレイオーディオ、カーナビゲーション、CDチューナー、音響システム　など

AE事業
ハイブリッド制御ECU、エンジン制御ECU
エアバッグ制御ECU　など

■デンソーテンの歩み

DENSO TEN

1920年
川西機械製作所

1949年
神戸工業株式会社

富士通

1968年
富士通と合併

1972年
富士通テン株式会社

DENSO TEN

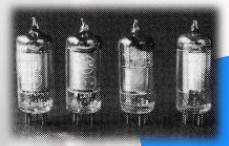
2017年株式会社デンソーテン

1944年
紫電改

1956年
タクシー用FM無線機

2003年
デジタル無線タクシー配車システム

2010年
マルチアングルビジョン



1948年
真空管

1955年
初代クラウン用
オートラジオ

1997年
ナビ内蔵*AV*一体器
AVN
「*A*udio」「*V*isual」「*N*avigation」

つながるサービス
Future Link

2014年
タクシー配車システム

クラウド型

2015年
ドライブレコーダー

Wi-Fiで
自動地図更新ナビ

2016年
ドライブレコーダー
内蔵ナビ

REC 録ナビ

1984年
エンジン制御*ECU*

2005年
ハイブリッド*ECU*

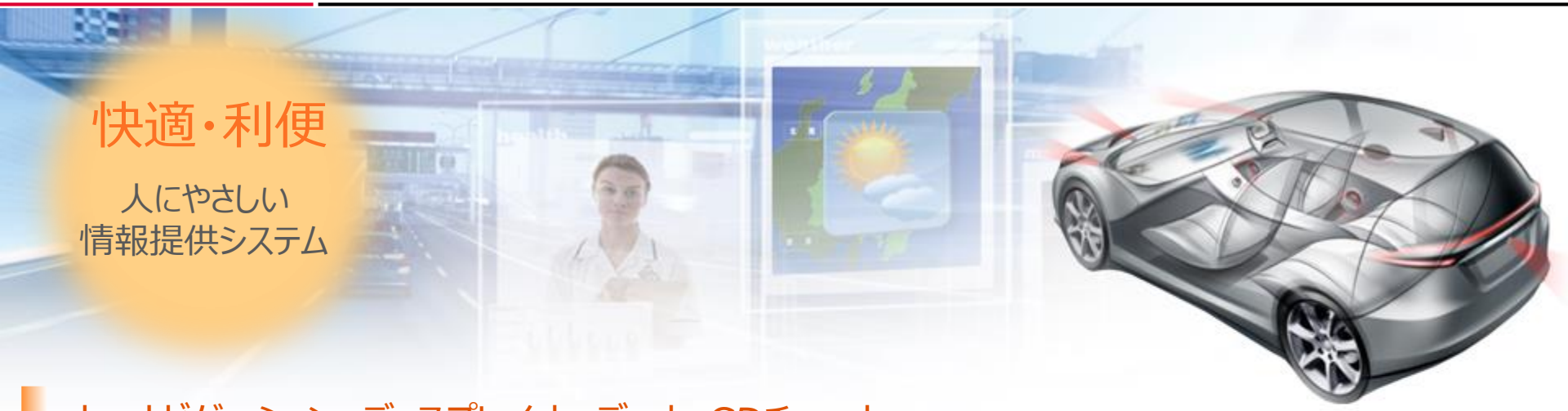
■事業ビジョン

DENSO TEN

人とクルマ、社会とクルマをつなぎ、
自由で快適なモビリティ社会の実現に貢献します。

■主な製品 –快適・利便–

DENSO TEN



快適・利便

人にやさしい
情報提供システム

カーナビゲーション、ディスプレイオーディオ、CDチューナー





音響システム・アンプ


AI活用タクシー乗車需要予測サービス

クラウド型タクシー配車システム

■主な製品 –安心・安全–

DENSO TEN

安心・安全

社会のネットに
クルマを組込む
基盤システム

緊急通報システム

エアバッグ制御*ECU*

盗難防止装置（*VSS*）

通信型ドライブレコーダー（商用車向け）

■主な製品 –環境–

DENSO TEN

環境

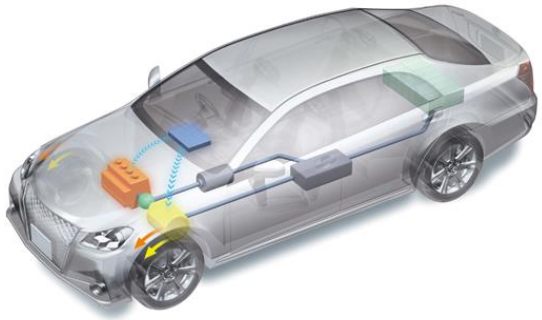
地球を守る
パワートレーン制御システム

ハイブリッドECU

電動パワー ステアリングECU

エンジン制御ECU

バッテリー マネジメントシステム



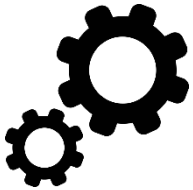
DENSO TEN

背景と課題

■各業界で普及するAIと成果

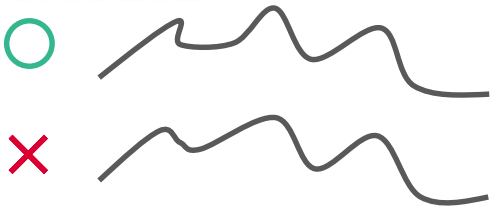
DENSO TEN

…　機械　電子　美容　医療　…



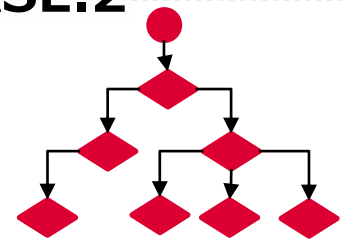
外観検査　故障診断　スキンケア　病理診断

CASE.1



良/不良の違いが熟練者にしか分からない。

CASE.2



無数の例外条件が存在する。

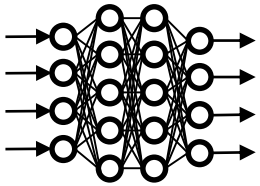
CASE.3

現象の定式化が困難。

精度・速度を

AI適用により向上

INPUT



OUTPUT

専門家同等以上の精度で高速に実現

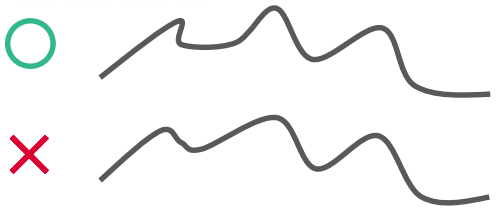
■ ECUにAIを搭載するモチベーション

DENSO TEN

自動車制御

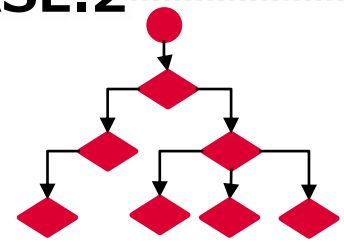
自動車制御でも、
悩み事は同じ

CASE.1



良/不良の違いが熟練者にしか分からない。

CASE.2



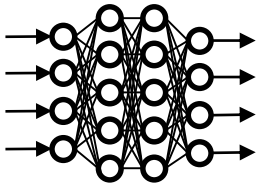
無数の例外条件が存在する。

CASE.3

現象の定式化が困難。

と言うことは…

AI適用により解決？



開発効率化や
性能向上を期待

■AIのECU実装に対する課題

DENSO TEN

現状

「**着想・発案**」が目的
高スペック、高自由度

・アルゴ高速化・省メモリ化
・マイコン機能活用
・プログラミング言語変換

python Chainer ギャップ C

実験 → データ加工 → AI設計・学習（サーバ（GPU搭載）） → 結果解析 → ✕ → 実装

机上検討から先が
繋がらない…

「**実現**」が目的
低スペック・低コスト

AIをECUに実装し、
製品化に繋げる**開発プロセス**の構築が必要

あるべき姿

実験 → データ加工 → AI設計・学習（サーバ（GPU搭載）） → シミュレーション → 実装 → 製品化

結果解析

DENSO TEN

MATLAB®活用によるAI実装検討

■なぜ実装に繋がらないのか？

DENSO TEN

Pythonを使う理由は？

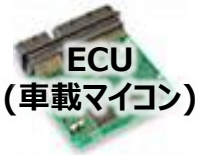
無償で使える
豊富なAI関連ライブラリ
充実したAI関連の実施例

Python→C言語の実用的な変換ツールがない。

これが**課題**？

設計者が手動でC言語に変換

マイコンに対応したコンパイラはC言語が基本



C言語でないとECUに載らない

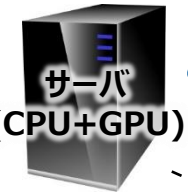
■ 課題の本質は何か？

DENSO TEN

MATLAB & SIMULINK

Simulink®による
モデリング＆シミュレーション

↳ ここが車載制御
開発プロセスの**肝**



python

Embedded Coder®により
コード生成

単純にPython® → C
が出来ても**本質的な**
課題解決はしない。

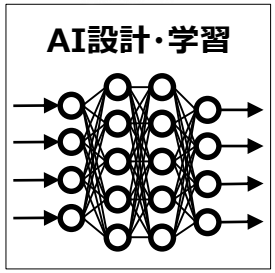
C言語コードの実装

↳ AI制御開発でも**従来同様にMBDをやりたい**

■MATLAB®活用により解決？

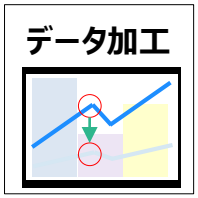
DENSO TEN

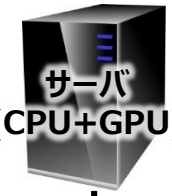
AI設計もできる
……らしい



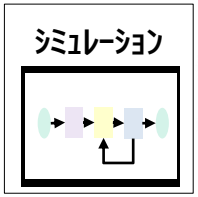
＋

豊富なライブラリに
よる**データ処理**





MATLAB® & SIMULINK®



GUIベースの高度な
シミュレーション機能

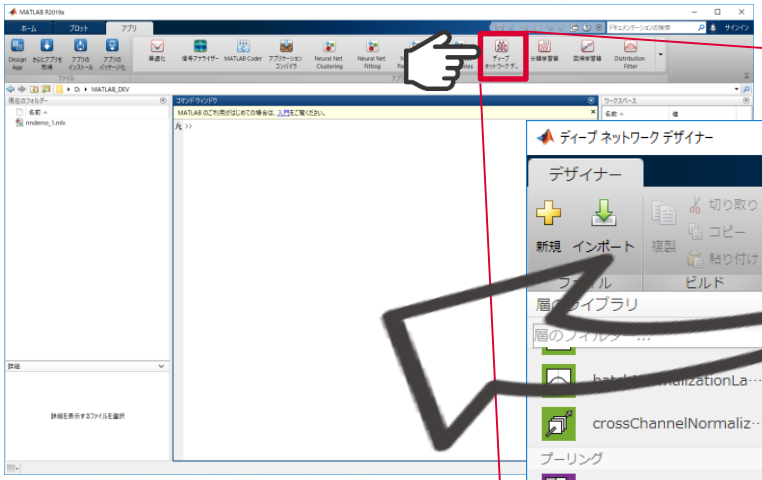
強力な**可視化**機能

車載制御開発で実績のある
コード生成機能

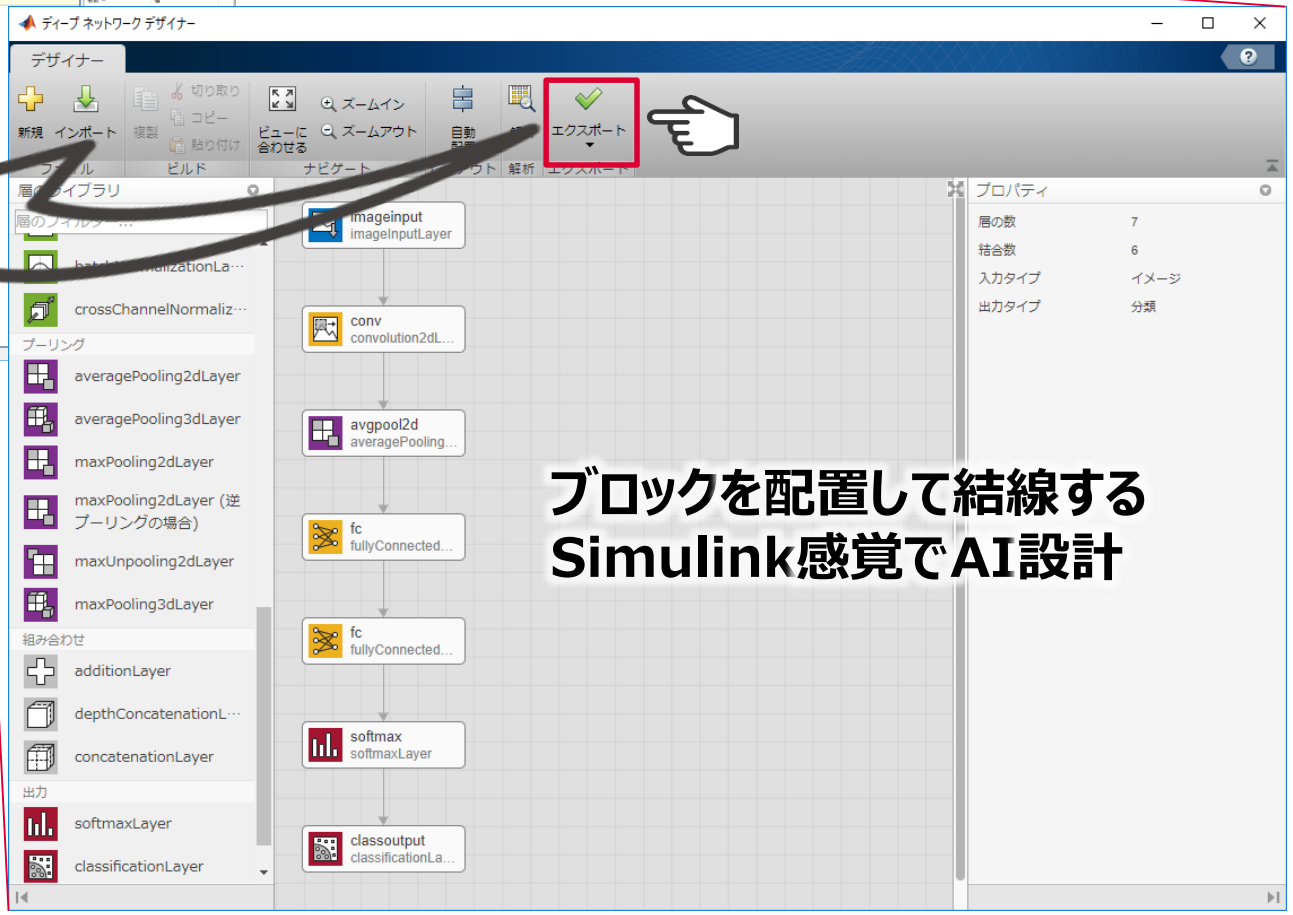
MATLAB®活用により
AI設計から実装まで
シームレスにつながる
……はず

■Deep Learning Toolbox™によるAI設計 DENSO TEN

Deep Learning Toolbox™の**ディープネットワークデザイナー**
を使って、GUIベースでAI(ネットワークモデル)設計



設計したAIを
ワークスペースに
エクスポート

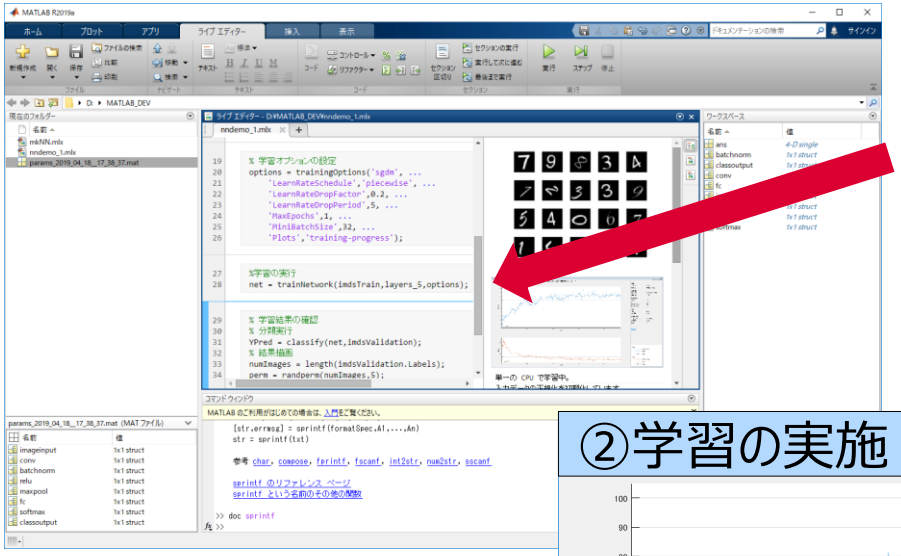


ブロックを配置して結線する
Simulink感覚でAI設計

■作成したAI(ネットワークモデル)の学習

DENSO TEN

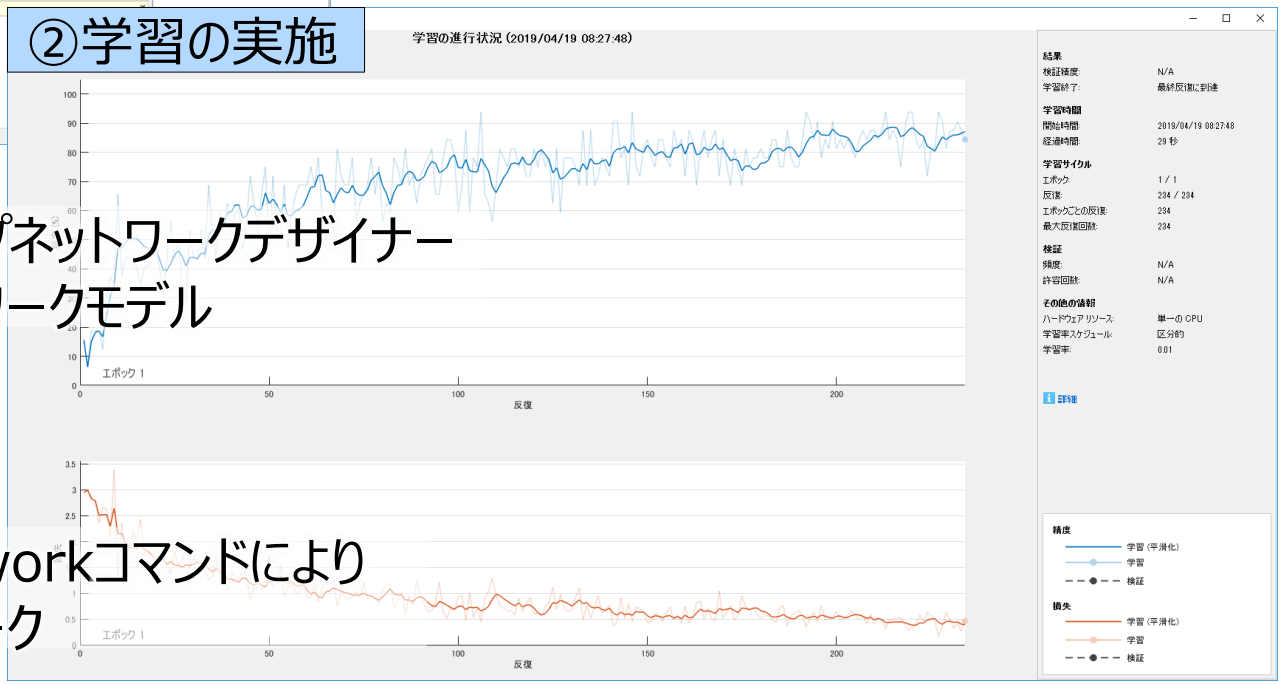
作成したネットワークモデルを学習

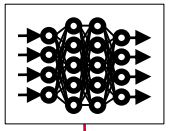


①MATLABからコマンド実行

```
% 学習オプションの設定
options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.2, ...
    'LearnRateDropPeriod',5, ...
    'MaxEpochs',1, ...
    'MiniBatchSize',32, ...
    'Plots','training-progress');
% 学習実行
net = trainNetwork(imdsTrain,layers,options);
```

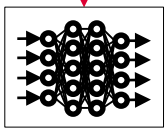
②学習の実施





layers:ディープネットワークデザイナー
で作成したネットワークモデル

学習

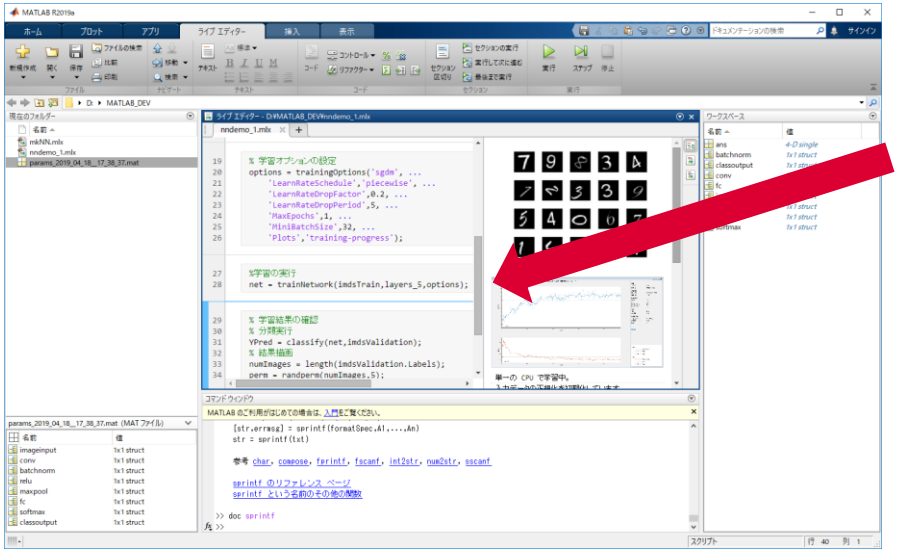


net:trainNetworkコマンドにより
学習したネットワーク

■学習済みネットワークモデルのコード生成

DENSO TEN

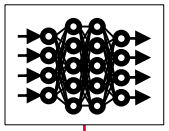
作成したネットワークモデルを学習



①MATLABからコマンド実行

```
% コード生成の実行
cfg = coder.config('lib');
cfg.GenCodeOnly = true;
cfg.TargetLang = 'C++';
dlcfg = coder.DeepLearningConfig('arm-compute');
dlcfg.ArmComputeVersion = '18.05';
dlcfg.ArmArchitecture = 'armv8';
cfg.DeepLearningConfig = dlcfg;
codegen -config cfg test_classify ...
        -args ones(28,28,'uint8') -report
```

net:trainNetworkコマンドにより学習したネットワーク



コード生成

②学習済みモデルからコード生成

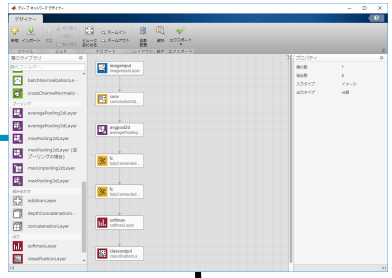
C++ソースコード(ARM用)

③ターゲットに実装

Raspberry Pi

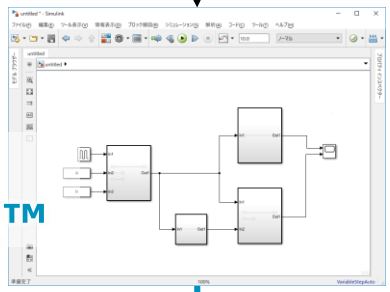
■学習済みネットワークのシミュレーション検討

DENSO TEN



ネットワークモデル

ここを何とかしたい



Smulink®モデル

出来そうなことは分かった
→Deep Learning Toolbox™

すでに出来ている
→Embedded Coder®

Cソースコード

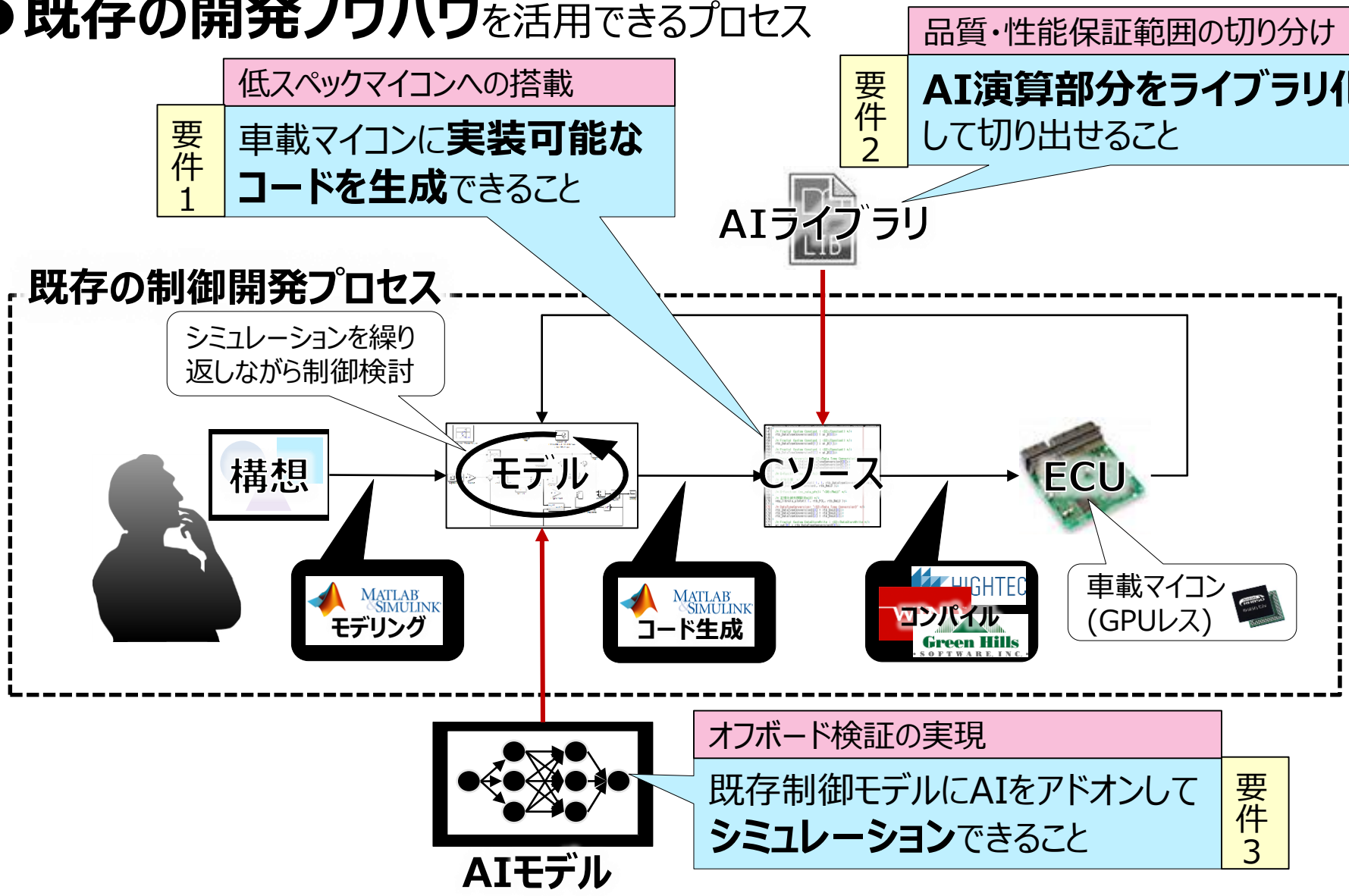
DENSO TEN

MATLAB®応用によるAI実装開発プロセスの構築

■開発コンセプト

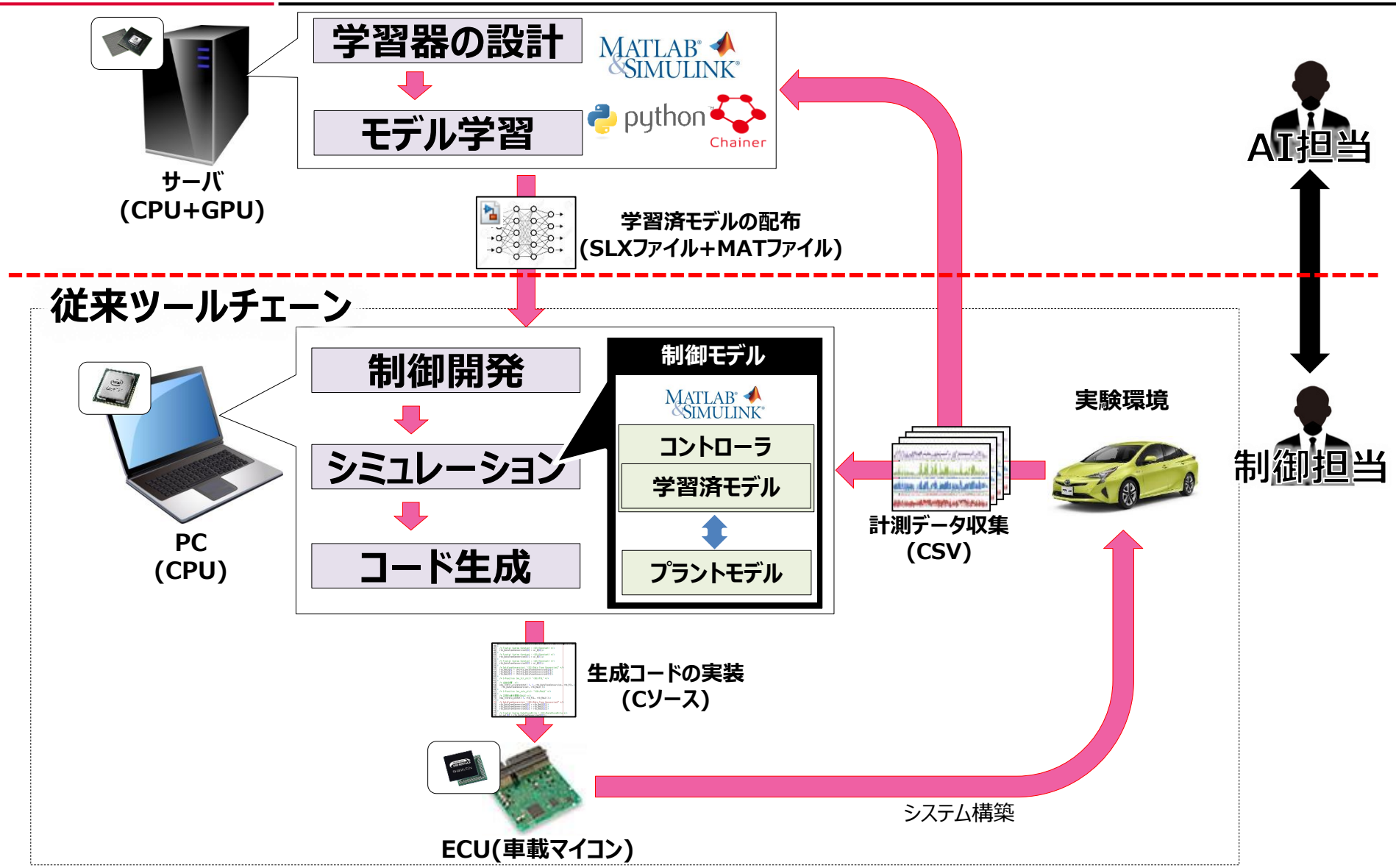
DENSO TEN

●既存の開発ノウハウを活用できるプロセス



■MATLAB®によるツールチェーン構築

DENSO TEN



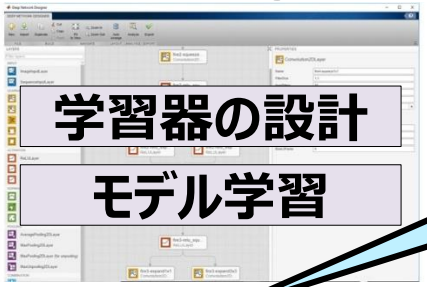
■開発技術のポイント

DENSO TEN

サーバ
(CPU+GPU)

python
仮学習済モデル

Deep Learning Toolbox™

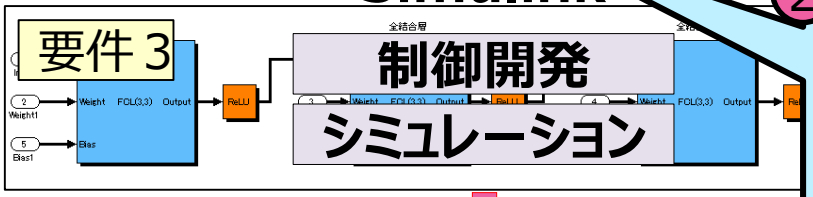


① S-function活用し、ECU実装向けのブロックセットによる**Simulinkモデル変換**

④ Simulink®から**Layer情報へ逆変換**

PC
(CPU)

Simulink



② Simulink®変換時に**ECU実装時の問題点検証**
例）ライブラリ非対応の処理が含まれる。
処理負荷が大きすぎる。

Embedded Coder®
(C言語)

要件１

AI実装用独自ライブラリ
(C言語、アセンブラ)

③ マイコンに応じた高速化手法を適用した**AI専用ライブラリ**

要件１	実装可能なコードを生成
要件２	AI演算部分をライブラリ化
要件３	AIモデルをシミュレーション

DENSO TEN

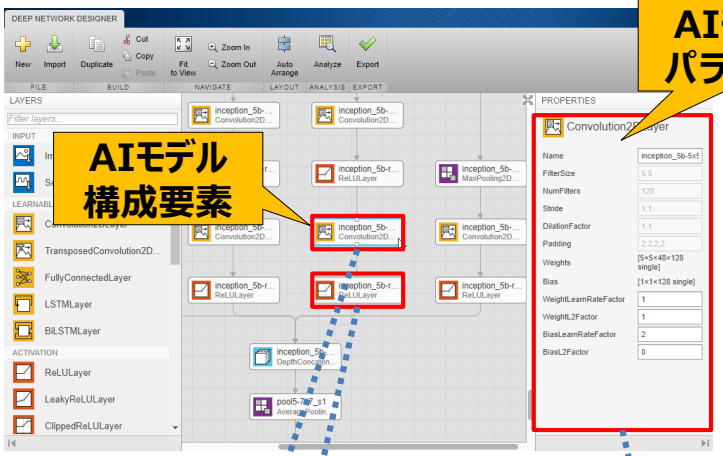
MATLAB®応用によるAI実装開発プロセスの構築

ネットワークモデルのSimulink®モデル変換

■AIのSimulink®モデル変換イメージ

DENSO TEN

Deep Learning Toolbox™



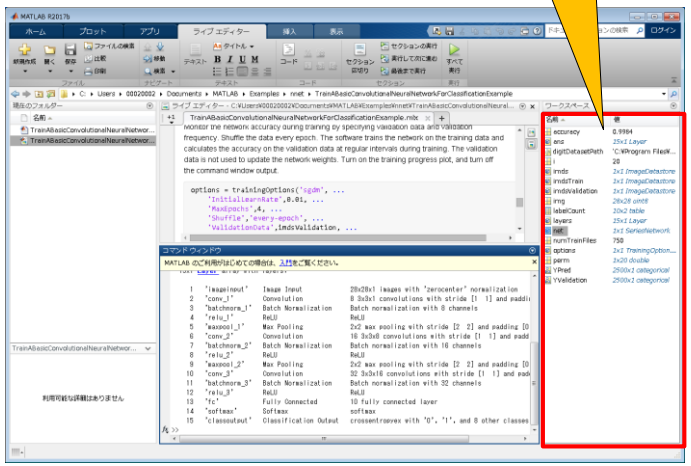
AIモデル
パラメータ

AIモデル
構成要素

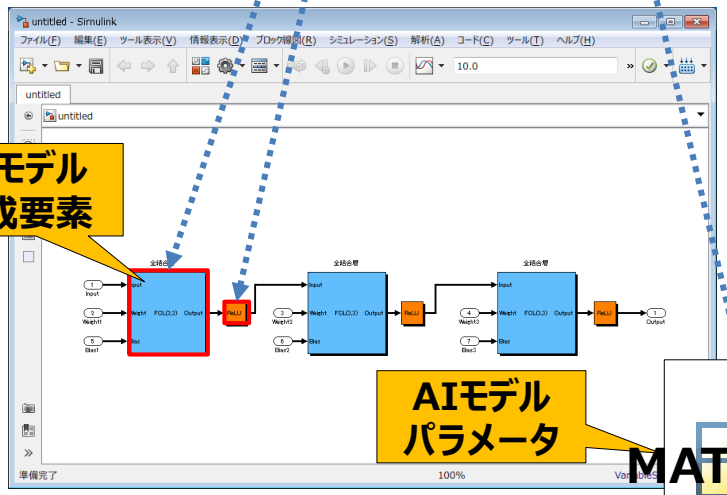
MATLAB ®データ形式で保存
(MATLAB API活用)

MATLAB®

AIモデル
構成要素
＋
パラメータ



Simulink®



AIモデル
構成要素

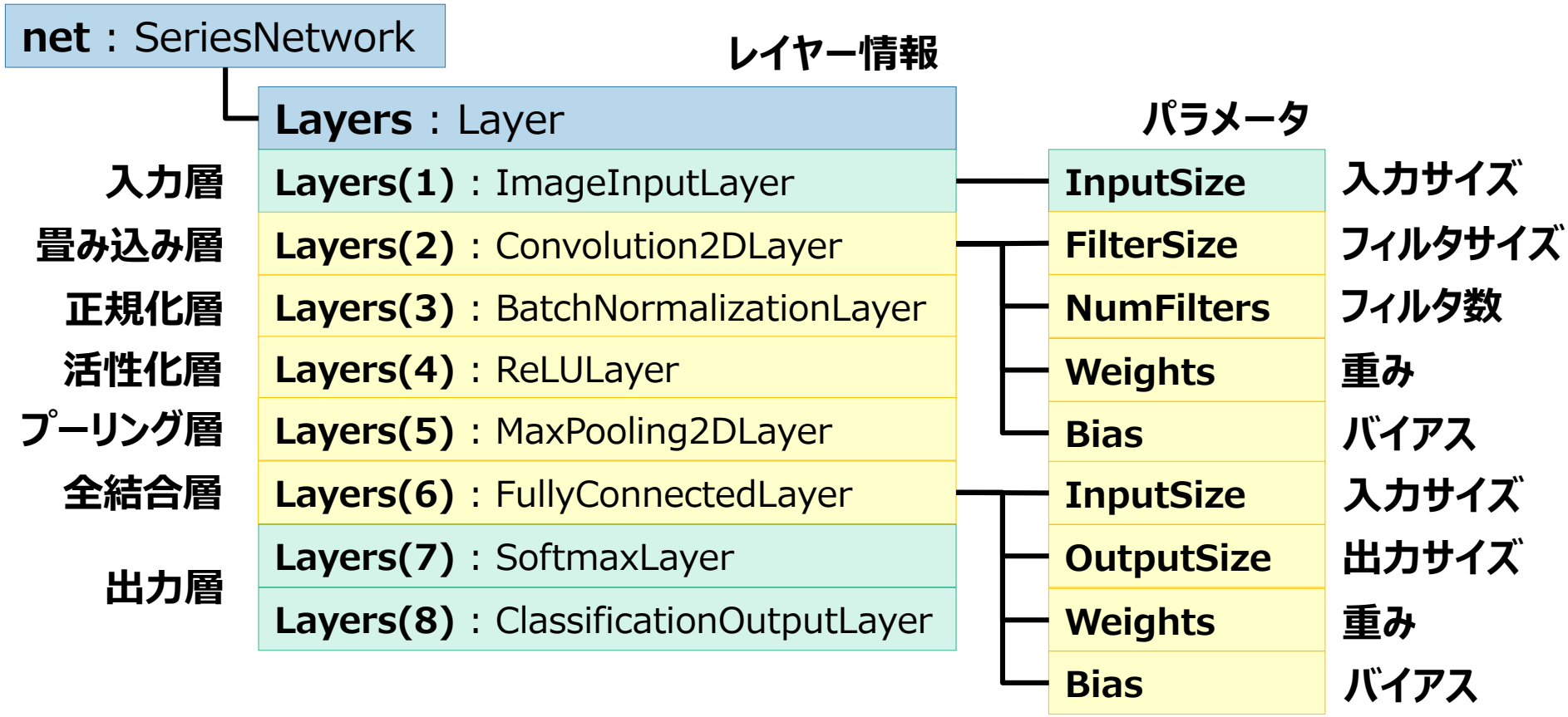
AIモデル
パラメータ

MAT-file

MATLAB®データから
Simulink®モデル生成
(Simulink® API活用)

DENSO TEN

学習済みモデルのデータ構造

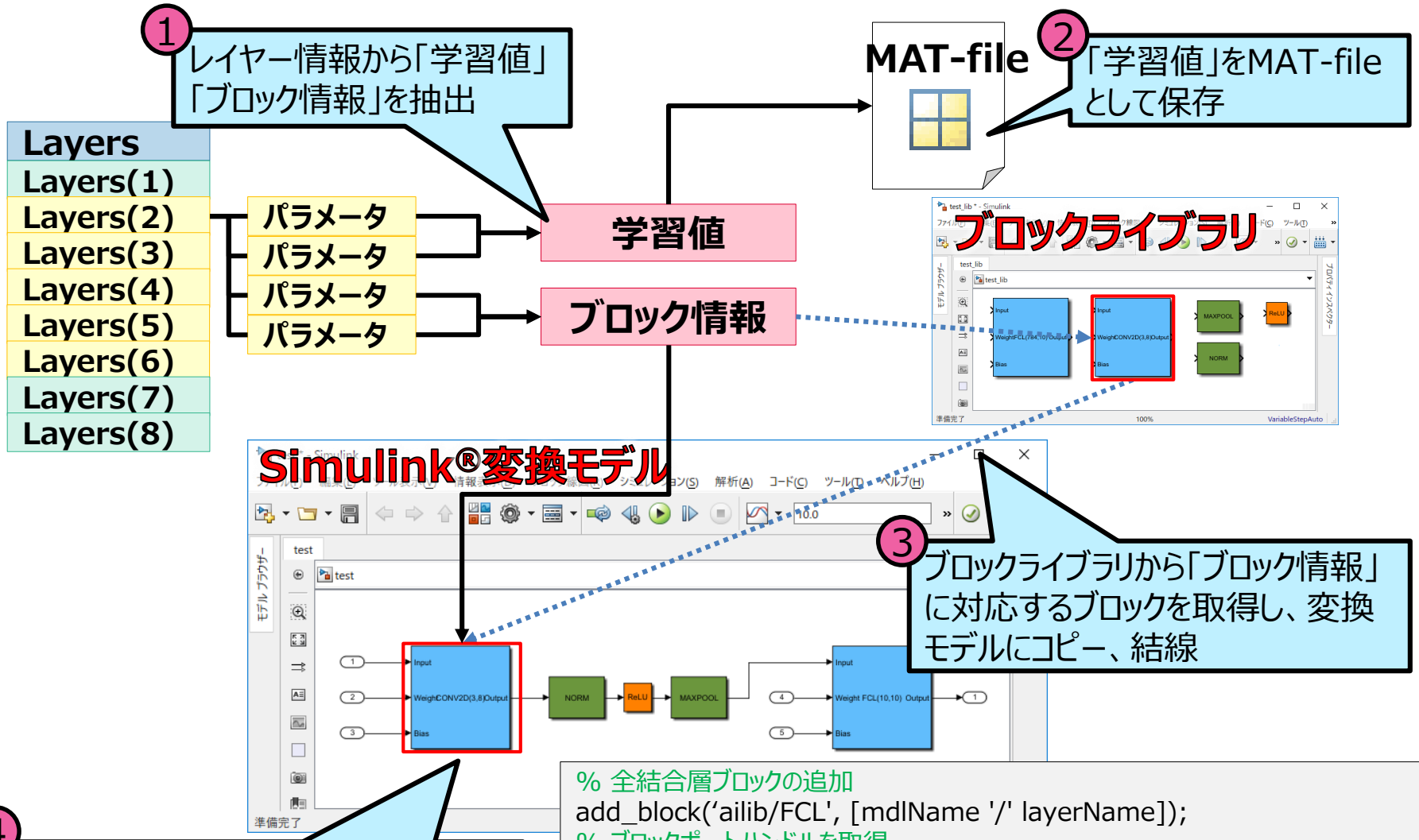


MATLAB APIによりデータの取得・加工が可能

```
% 畳み込み層からフィルタ数の取り出し
numfilter = net.Layers(2).NumFilters;
```

■MATLAB®/Simulink®APIによるAIモデル変換[2/2]

DENSO TEN

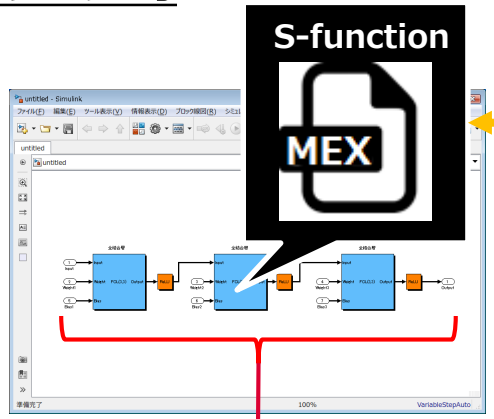



```
% 全結合層ブロックの追加
add_block('ailib/FCL', [mdlName '/' layerName]);
% ブロックポートハンドルを取得
curBlkPortHdl = get_param([mdlName '/' layerName],'PortHandles');
% ブロックを結線
add_line(name, preBlkPortHdl.Outport(1), curBlkPortHdl.Inport(1));
```

■シミュレーション / コード生成の両立

DENSO TEN

シミュレーション時



リンク

AI専用ライブラリ
(C言語)

Cソース

畳み込み層や全結合層など
の複雑な演算を隠蔽

コード生成時

リンク

Embedded Coder ®

Target Language Compiler

プログラム言語を
生成するコンパイラ

STF

コード生成ルール

生成コード

実装

DENSO TEN

MATLAB®応用によるAI実装開発プロセスの構築

ECU実装時の問題点検証

■利用可能なレイヤーの制限

DENSO TEN

様々な深層学習のレイヤー

入力
- imageInputLayer
- image3dInputLayer
- sequenceInputLayer

畳み込みと全結合
- convolution2dLayer
- convolution3dLayer
- groupedConvolution2dLayer
- transposedConv2dLayer
- transposedConv3dLayer
- fullyConnectedLayer

シーケンス
- lstmLayer
- bilstmLayer
- sequenceFoldingLayer
- sequenceUnfoldingLayer
- flattenLayer

アクティベーション
- reluLayer
- leakyReluLayer
- clippedReluLayer
- tanhLayer
- eluLayer

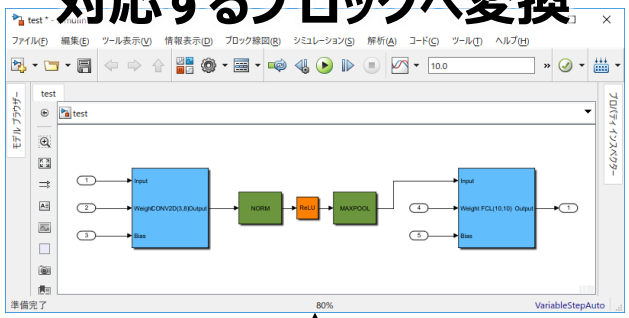
正規化とユーティリティ
- dropoutLayer
- batchNormalizationLayer
- crossChannelNormalizationLayer

プーリング
- averagePooling2dLayer
- averagePooling3dLayer
- maxPooling2dLayer
- maxPooling2dLayer (逆プーリングの場合)
- maxUnpooling2dLayer
- maxPooling3dLayer

組み合わせ
- additionLayer
- depthConcatenationLayer
- concatenationLayer

出力
- softmaxLayer
- classificationLayer
- regressionLayer

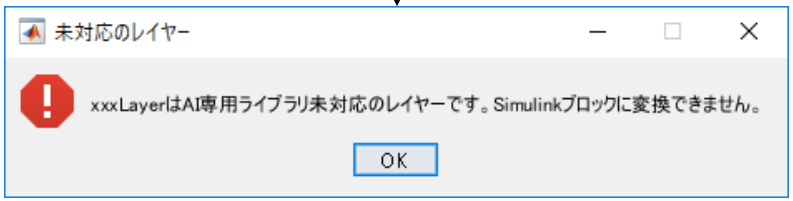
対応するブロックへ変換



ライブラリ対応

利用可能レイヤーの識別

ライブラリ非対応



ブロック変換を制限

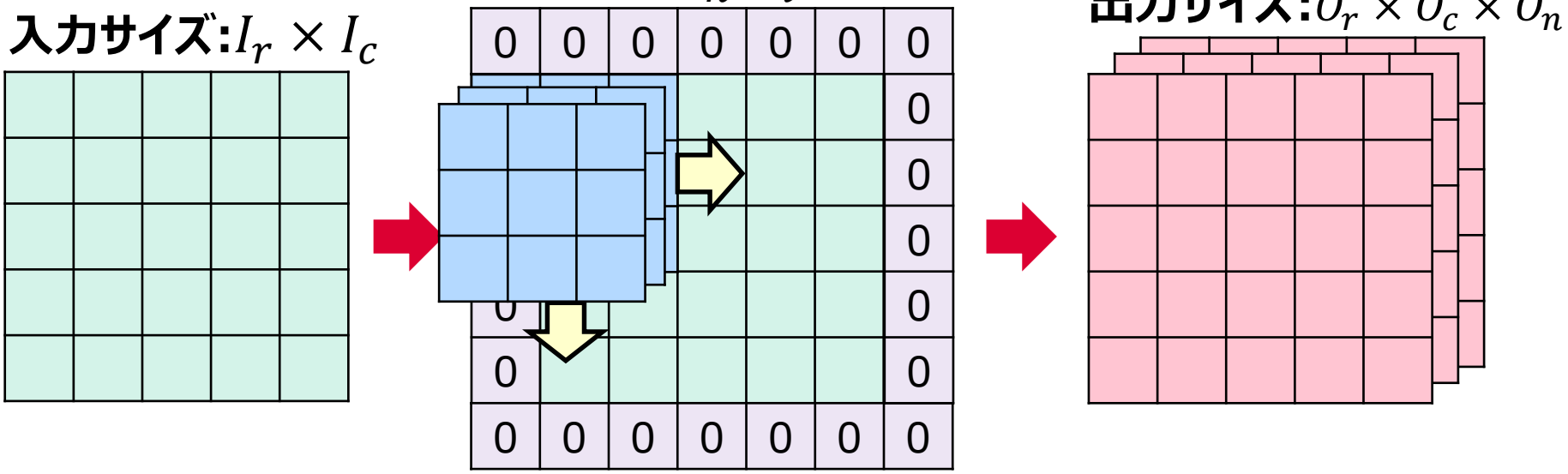
■演算規模の制限

DENSO TEN

畳み込み層の計算量

フィルタサイズ:$F_r \times F_c$
フィルタ数:F_n
パディングサイズ:$[P_t, P_b, P_l, P_r]$
ストライド:$[S_h, S_v]$

入力サイズ:$I_r \times I_c$



出力サイズ:$O_r \times O_c \times O_n$

演算規模Nを定式化し、許容上限をターゲットに応じて制限

$$N = (F_r \times F_c) \times \frac{I_r + P_l + P_r}{S_h} \times \frac{I_c + P_t + P_b}{S_v} \times F_n$$

DENSO TEN

MATLAB®応用によるAI実装開発プロセスの構築

AI専用ライブラリの開発

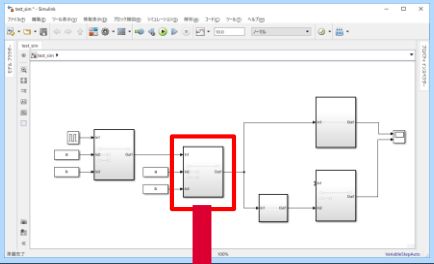
■AI専用ライブラリによる制御/AIの分離

DENSO TEN

開発担当別に取り扱う問題を抽象化

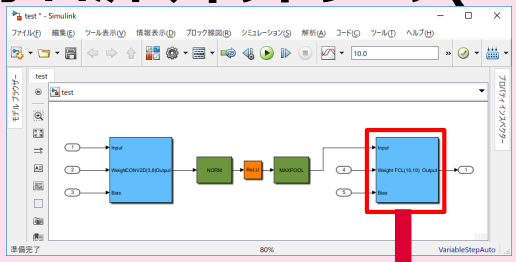
制御担当

制御モデル



AI担当

学習済みネットワーク(AI)



ライブラリ担当

AI専用ライブラリ

AIへの関わり方	関心ごと
■インテリジェントな機能ブロックとしてAIを扱う。 ■AIの入出力I/Fに対して機能要求を行う。	■AIを含む制御モデルが要求通りの振る舞いをすること。
■要求された入出力に対して適当なネットワークモデルを設計・学習する。	■AIの入力に対して高速・高精度に出力を返すこと。
■各レイヤーの演算をマイコン機能を駆使してプログラミングする。	■各レイヤーの入出力に対して高速かつ正確に出力を返すこと。

■階層別の並列開発による効率化

DENSO TEN

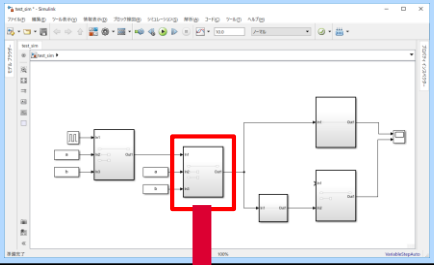
取り扱う開発課題に応じた階層を改良

制御担当

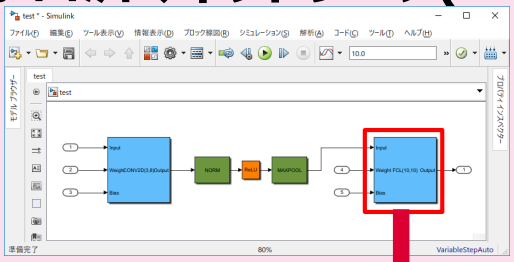
AI担当

ライブラリ担当

制御モデル



学習済みネットワーク(AI)

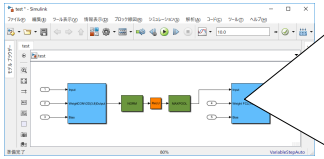


AI専用ライブラリ

開発課題(例)	改良対象		
AI出力に応じて制御モードを切り替えたい	○		
特定の入力に対する誤判定を改善したい		○	
ターゲットマイコンを変更したい			○

■S-FunctionによるAI専用ライブラリ埋め込み

DENSO TEN

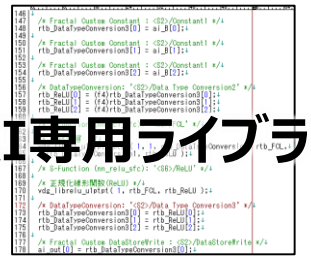


S-function

```
…
/* S-Functionからの出力設定 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    …
    AI専用ライブラリ関数を実行して結果を返す。
    …
}
…
```

CALL

RETURN



AI専用ライブラリ

CALL

RETURN

STF(TLC)

```
…
%%ブロックに対するコード出力
%function Outputs(block, system) Output
    …
    %openfile buff
    AI専用ライブラリ関数を実行するコードを作成。
    %closefile buff
    …
%endfunction
…
```

GEN

生成コード

DENSO TEN

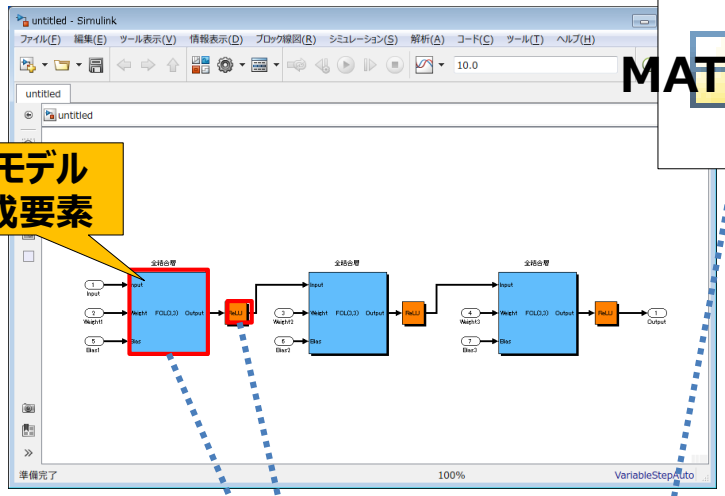
MATLAB®応用によるAI実装開発プロセスの構築

Simulink®モデルからレイヤー情報への逆変換

■AIのSimulink®モデル逆変換イメージ

DENSO TEN

Simulink®



AIモデル
構成要素

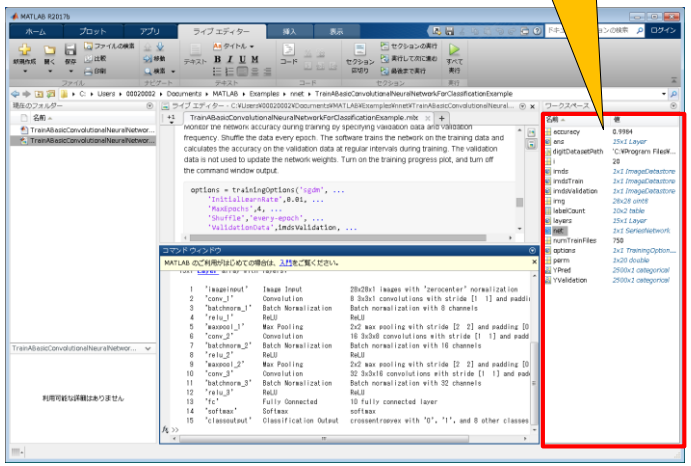
AIモデル
パラメータ

MAT-file

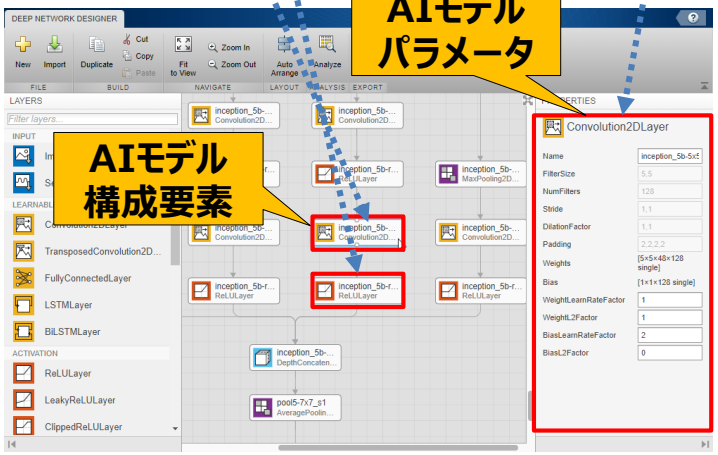
MATLAB®データ形式で保存
(Simulink® API活用)

MATLAB®

AIモデル
構成要素
＋
パラメータ



Deep Learning Toolbox™



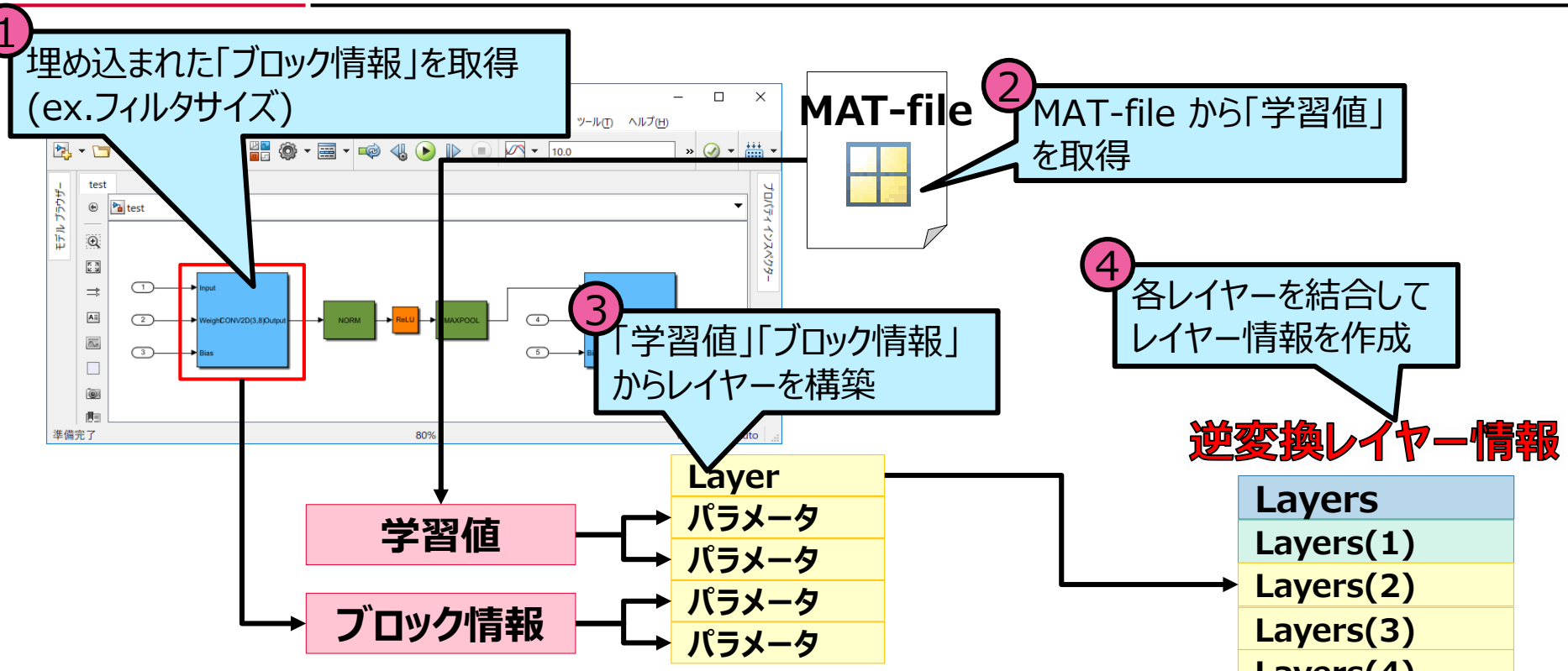
AIモデル
パラメータ

AIモデル
構成要素

MATLAB®データからレイヤー
情報生成
(MATLAB® API活用)

■MATLAB®/Simulink®APIによるAIモデル逆変換

DENSO TEN




```
% レイヤー情報の構築
layers = [
    imageInputLayer(imgin.size, “Name”, “imgin”)
    convolution2dLayer(conv.size, conv.num, "Name", "conv", ...
                       "Padding", "same", "Bias", conv.Bias,"Weights", conv.Weights)
    ～略～
    fullyConnectedLayer(fc.size, "Name", "fc", "Bias", fc.Bias, "Weights", fc.Weights)
    softmaxLayer("Name", "softmax")
    classificationLayer("Name", "classoutput","Classes", classoutput.Classes) ];
```

DENSO TEN

まとめ

■開発結果

DENSO TEN

要件ごとの達成状況

No.	内容	達成状況	
要件１	実装可能なコードを生成	AIのSimulink®変換により達成	○
要件２	AI演算部分をライブラリ化	ライブラリをS-function埋め込みで達成	○
要件３	AIモデルをシミュレーション	AIのSimulink®変換により達成	○

開発技術のポイントごとの対応結果

No.	内容	対応
ポイント１	ネットワークモデルのSimulink®モデル変換	SeriesNetworkクラスを解析し、MATLAB®/Simulink®APIにより変換
ポイント２	ECU実装時の問題点検証	Simulink®変換時にレイヤーの種別及び計算量の制限を実施
ポイント３	AI専用ライブラリの開発	レイヤー単位のC言語ライブラリ開発
ポイント４	Simulink®モデルからレイヤー情報への逆変換	MATLAB®/Simulink®APIによりSeriesNetworkクラスを作成

■成果

DENSO TEN

■AI設計から実装までの
一貫性のあるAI開発プロセスを構築

■将来の**製品化を見据えた仕組み作り**に目途付け

- 使用したMATLAB®/Simulink®製品ファミリ
  MATLAB®
  Simulink®
  Simulink Coder®
  Embedded Coder®
  Deep Learning Toolbox™


DENSO TEN