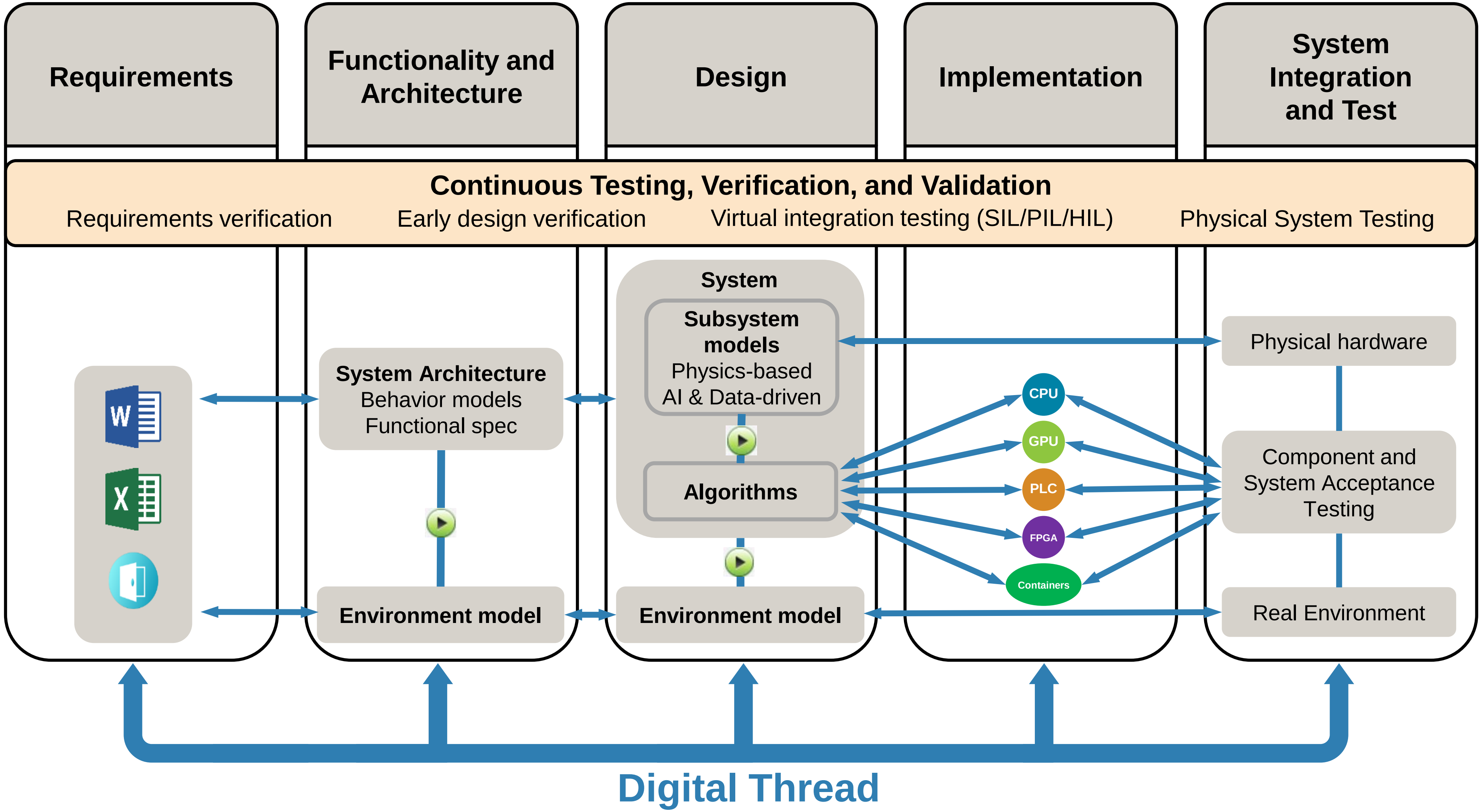
AI with Model-Based Design:

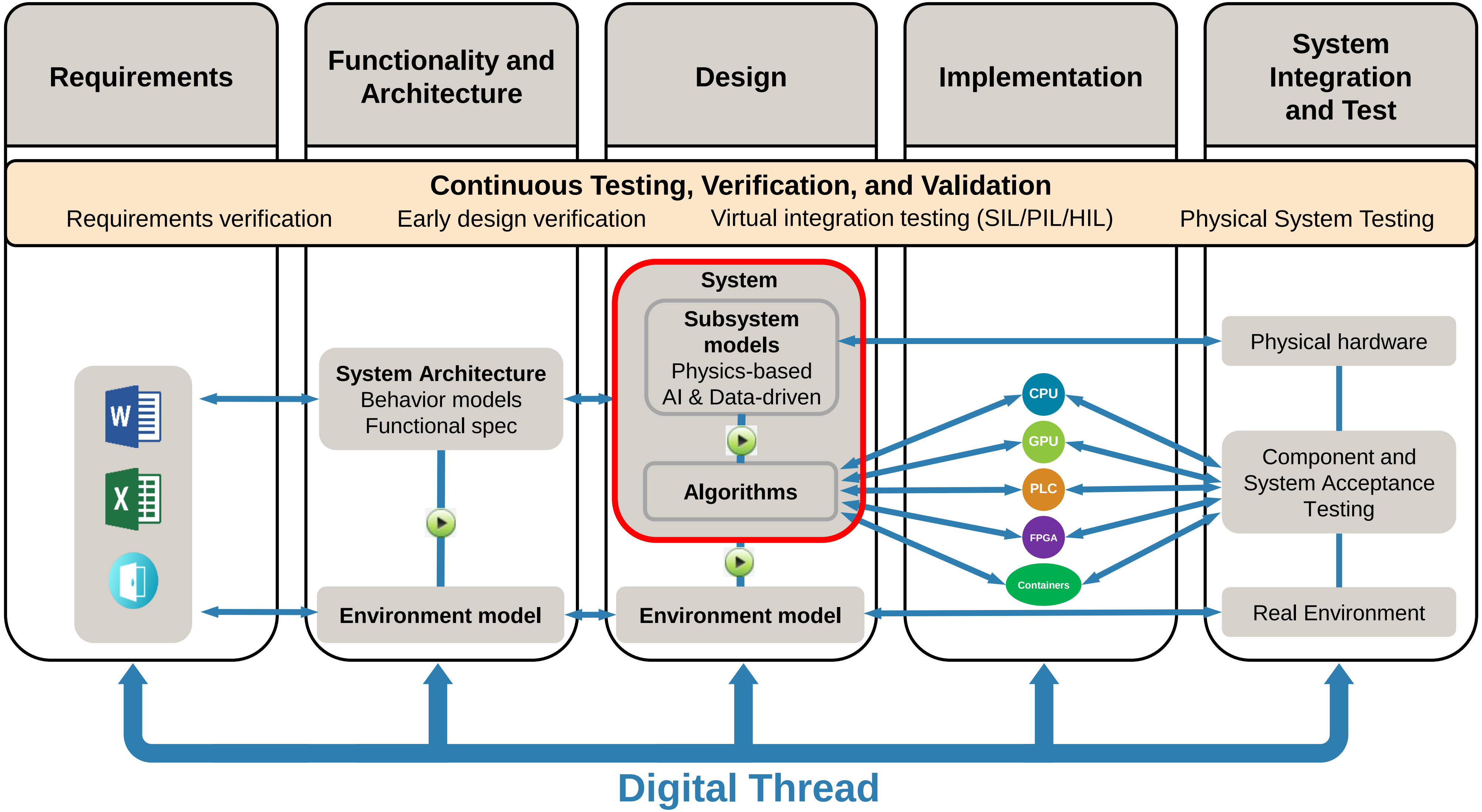
Virtual Sensor Modeling

MathWorks®

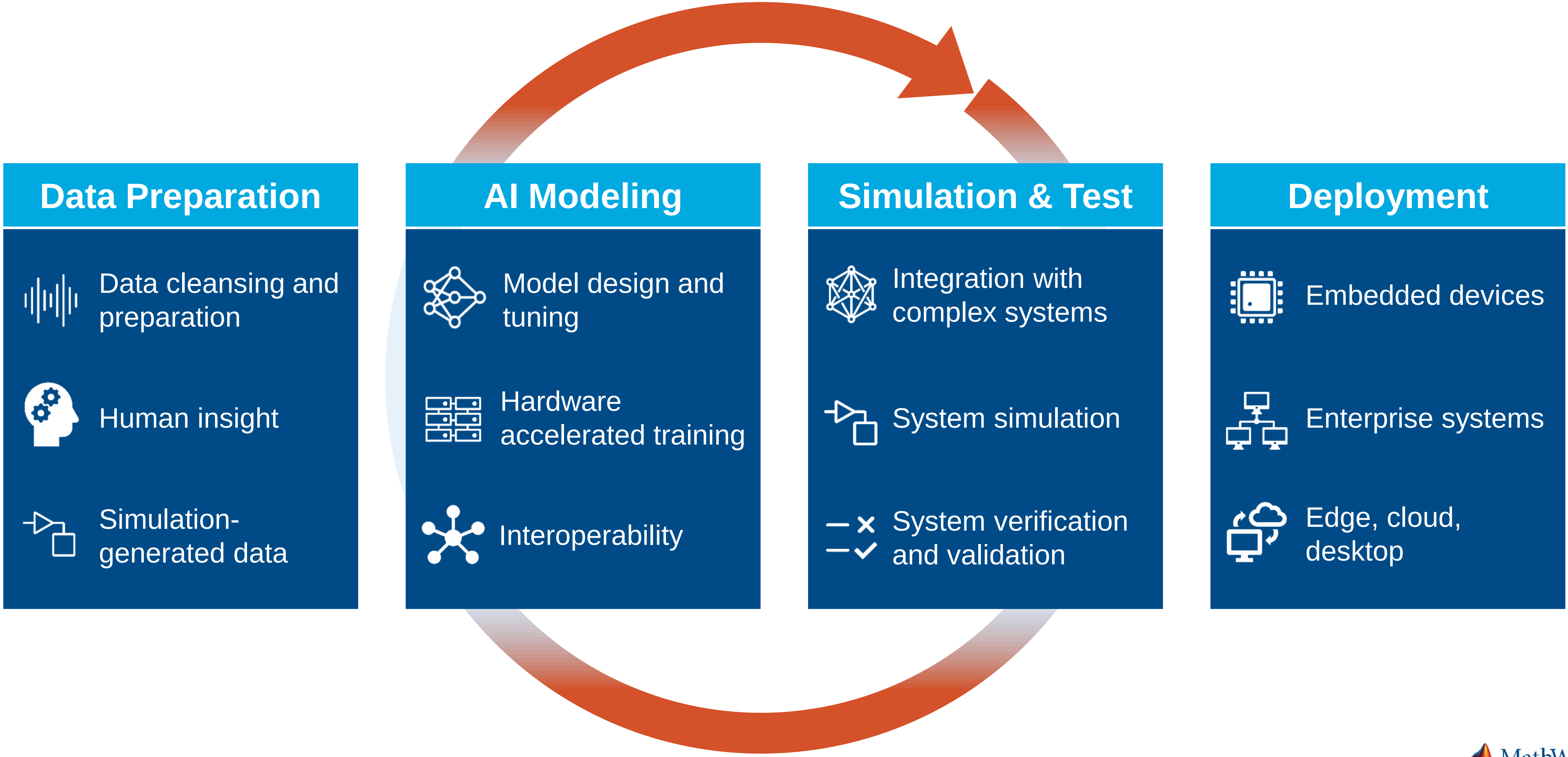
Model-Based Design



Integrating AI into Model-Based Design



AI-driven system design



Data Preparation

- Data cleansing and preparation
- Human insight
- Simulation-generated data

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

Simulation & Test

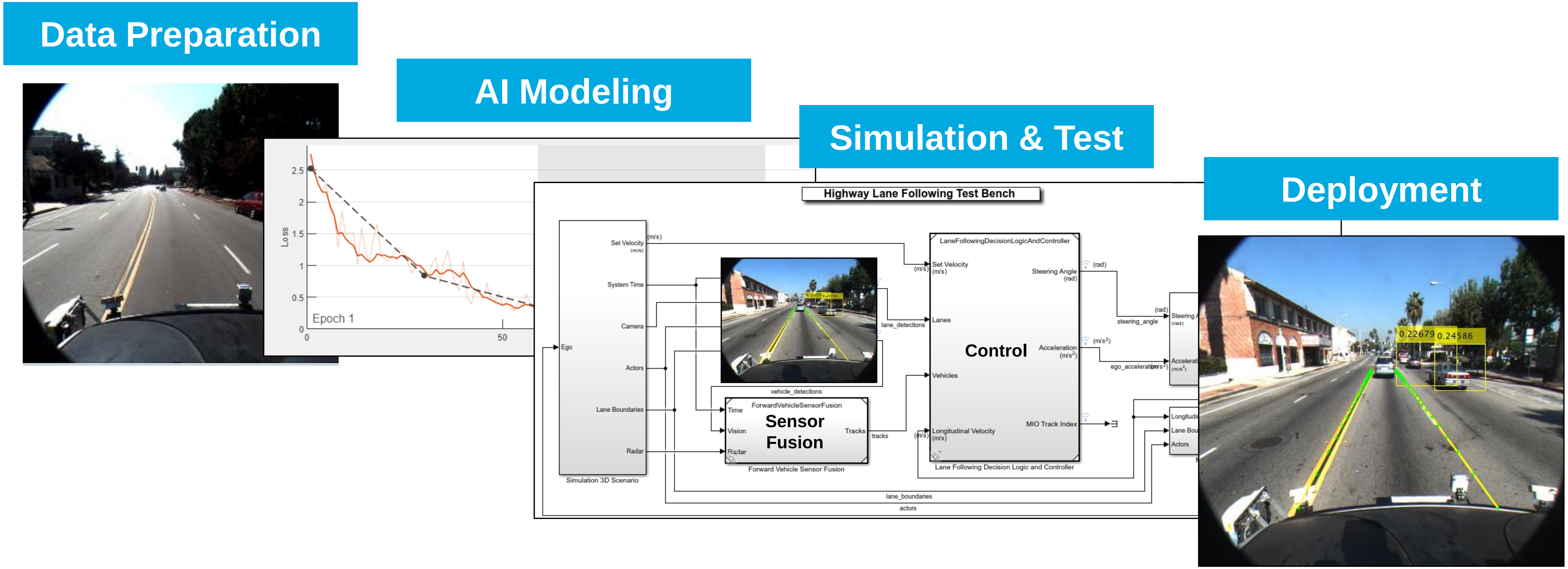
- Integration with complex systems
- System simulation
- System verification and validation

Deployment

- Embedded devices
- Enterprise systems
- Edge, cloud, desktop

AI for images and video

AI for vehicle detection



Data Preparation

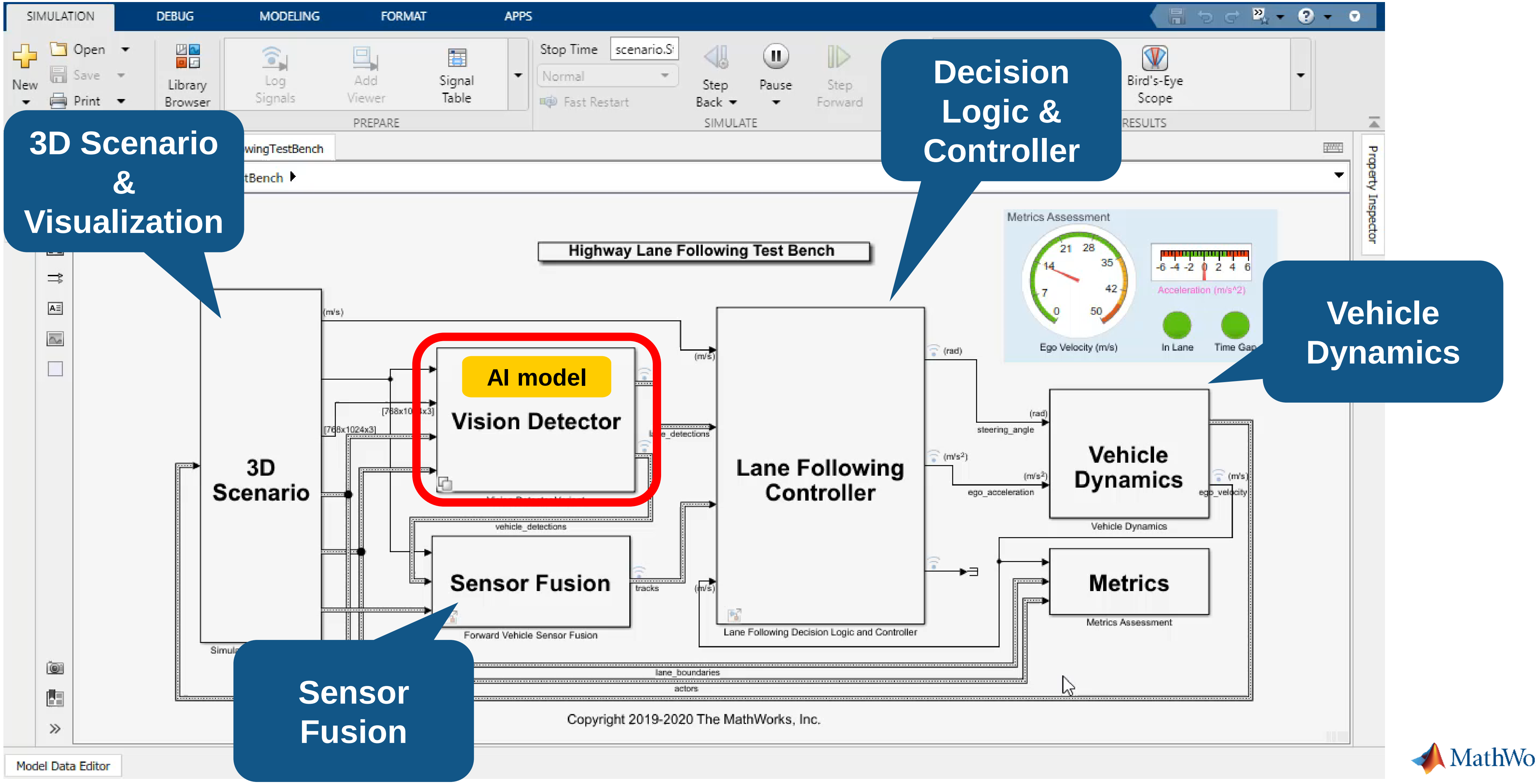
AI Modeling

Simulation & Test

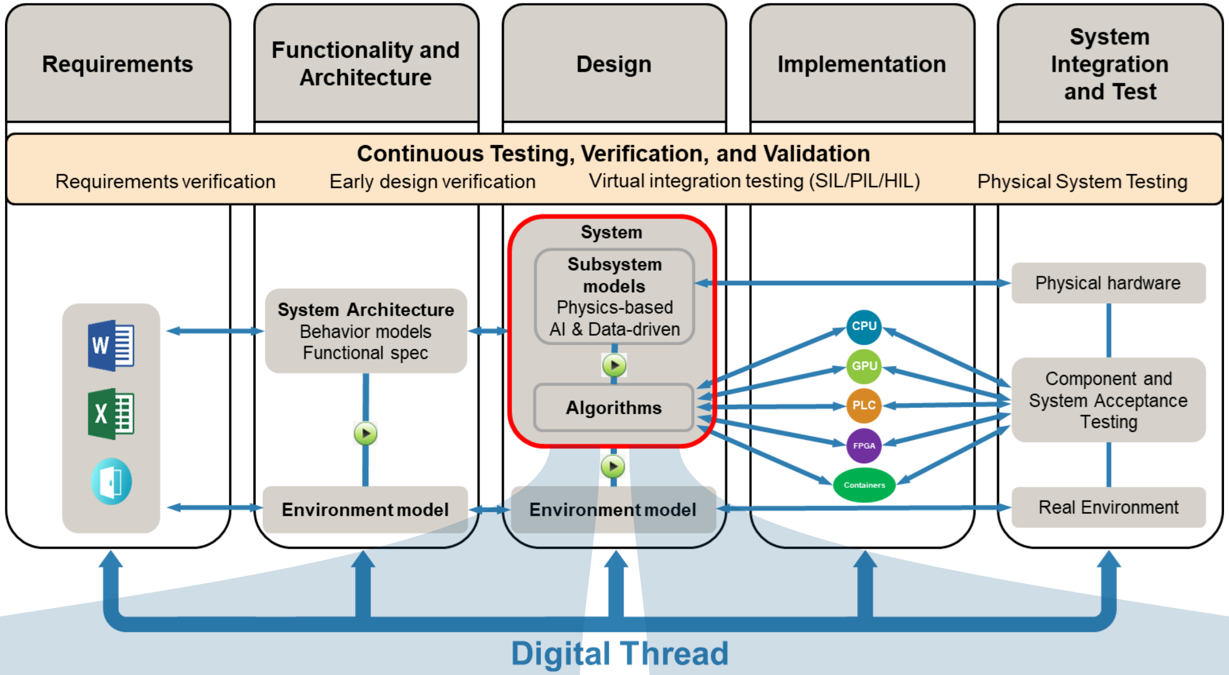
Deployment

AI is often part of a larger system

Simulate and test all your components together in Simulink



Integrate AI models into MBD for system-level simulation and code generation



Requirements

Functionality and Architecture

Design

Implementation

System Integration and Test

Continuous Testing, Verification, and Validation

Requirements verification

Early design verification

Virtual integration testing (SIL/PIL/HIL)

Physical System Testing

System Architecture
Behavior models
Functional spec

Environment model

System

Subsystem models
Physics-based
AI & Data-driven

Algorithms

Environment model

CPU

GPU

PLC

FPGA

Containers

Physical hardware

Component and System Acceptance Testing

Real Environment

Digital Thread

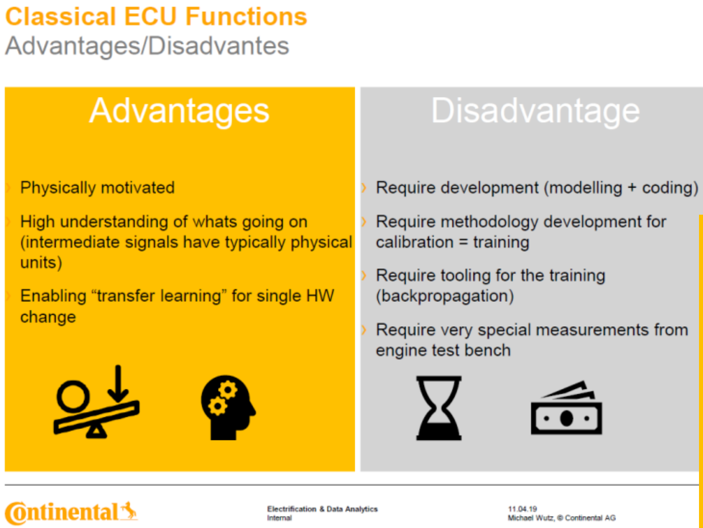
AI for component modeling

- Speeding up desktop and HIL simulations
- Modeling component dynamics from data when first-principles models cannot be obtained

AI for algorithm development

- Virtual sensor modeling
- Sensor fusion
- Object detection

Continental uses data-driven engine temperature models for ECU development

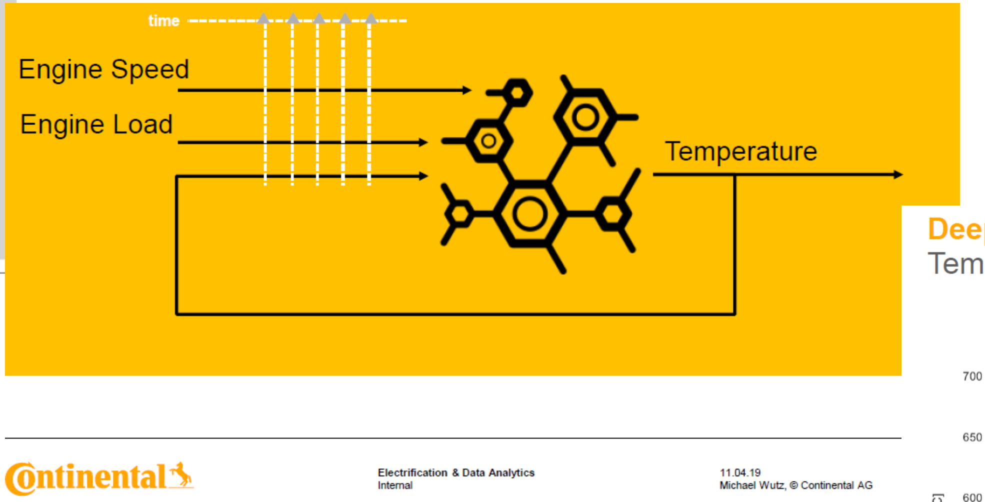


Classical ECU Functions
Advantages/Disadvantes

Advantages	Disadvantage
Physically motivated High understanding of whats going on (intermediate signals have typically physical units) Enabling “transfer learning” for single HW change	Require development (modelling + coding) Require methodology development for calibration = training Require tooling for the training (backpropagation) Require very special measurements from engine test bench

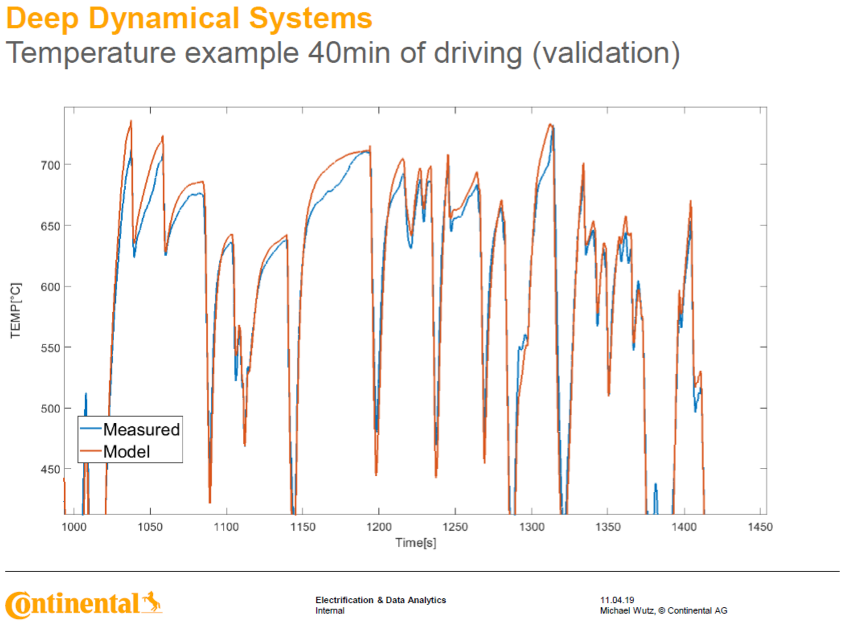
Electrification & Data Analytics
Internal

11.04.19
Michael Wutz, © Continental AG



Electrification & Data Analytics
Internal

11.04.19
Michael Wutz, © Continental AG



Deep Dynamical Systems
Temperature example 40min of driving (validation)

Electrification & Data Analytics
Internal

11.04.19
Michael Wutz, © Continental AG

Link to presentation

Renault Uses Deep Learning Networks to Estimate NO_X Emissions

Challenge

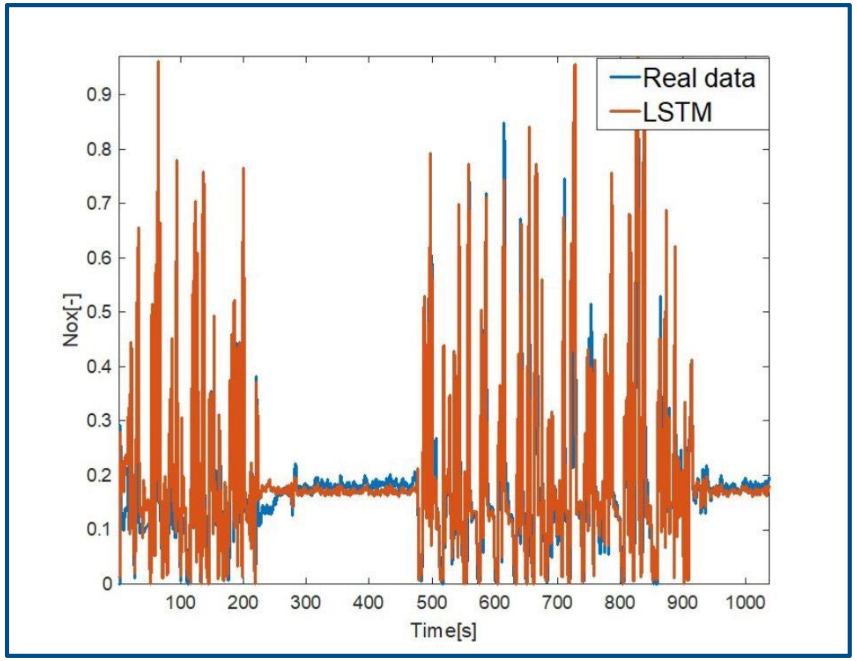
Design, simulate, and improve aftertreatment systems to reduce oxides of nitrogen (NO_X) emissions

Solution

Use MATLAB and Deep Learning Toolbox to model engine-out NO_X emissions using a long short-term memory (LSTM) network

Results

- NO_X emissions predicted with close to 90% accuracy
- LSTM network incorporated into aftertreatment simulation model
- Code generated directly from network for ECU deployment

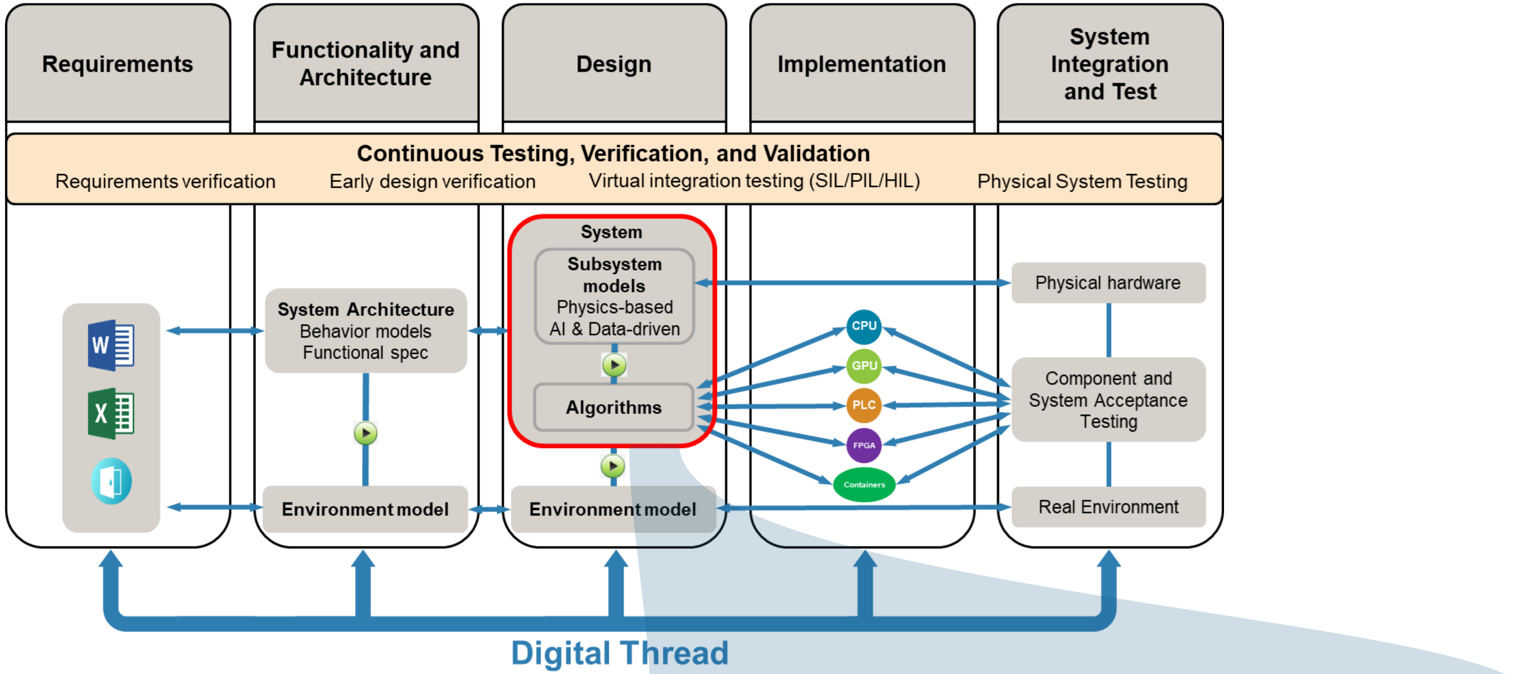


Measured NO_X emissions from an actual engine and modeled NO_X emissions from the LSTM network.

> *“Even though we are not specialists in deep learning, using MATLAB and Deep Learning Toolbox we were able to create and train a network that predicts NO_X emissions with almost 90% accuracy.”*
>
> *- Nicoleta-Alexandra Stroe, Renault*

Link to article

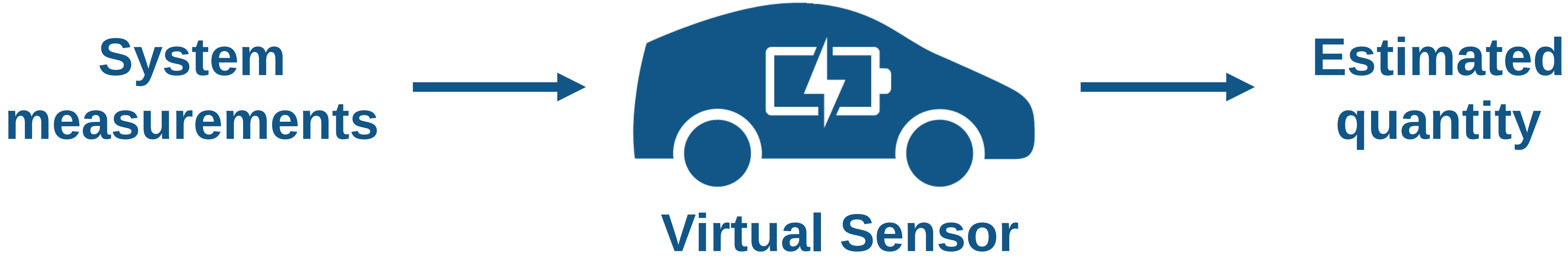
Focus today



AI for algorithm development

- Virtual sensor modeling
- Sensor fusion
- Object detection

A Virtual Sensor mimics a physical sensor using data from other measurements to estimate the quantity of interest

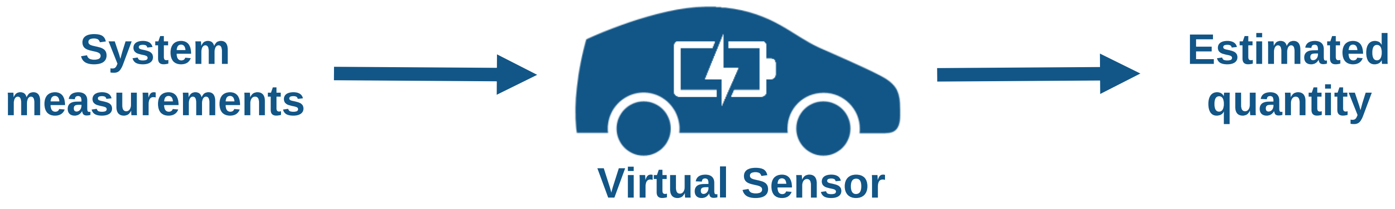


System measurements

Virtual Sensor

Estimated quantity

Why are Virtual Sensors relevant?



A **physical sensor** may be:

- Expensive
- Slow
- Noisy
- Unreliable
- Not feasible
- Unmanufacturable
- Degrading over time
- Requiring redundancy
- etc.

Data-driven vs. first-principles modeling

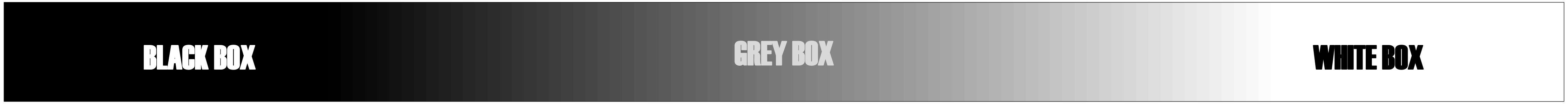
Data-driven models and first-principles models can co-exist

DATA-DRIVEN MODELS

Statistics, optimization, AI

FIRST-PRINCIPLES MODELS

Physics, math, domain knowledge

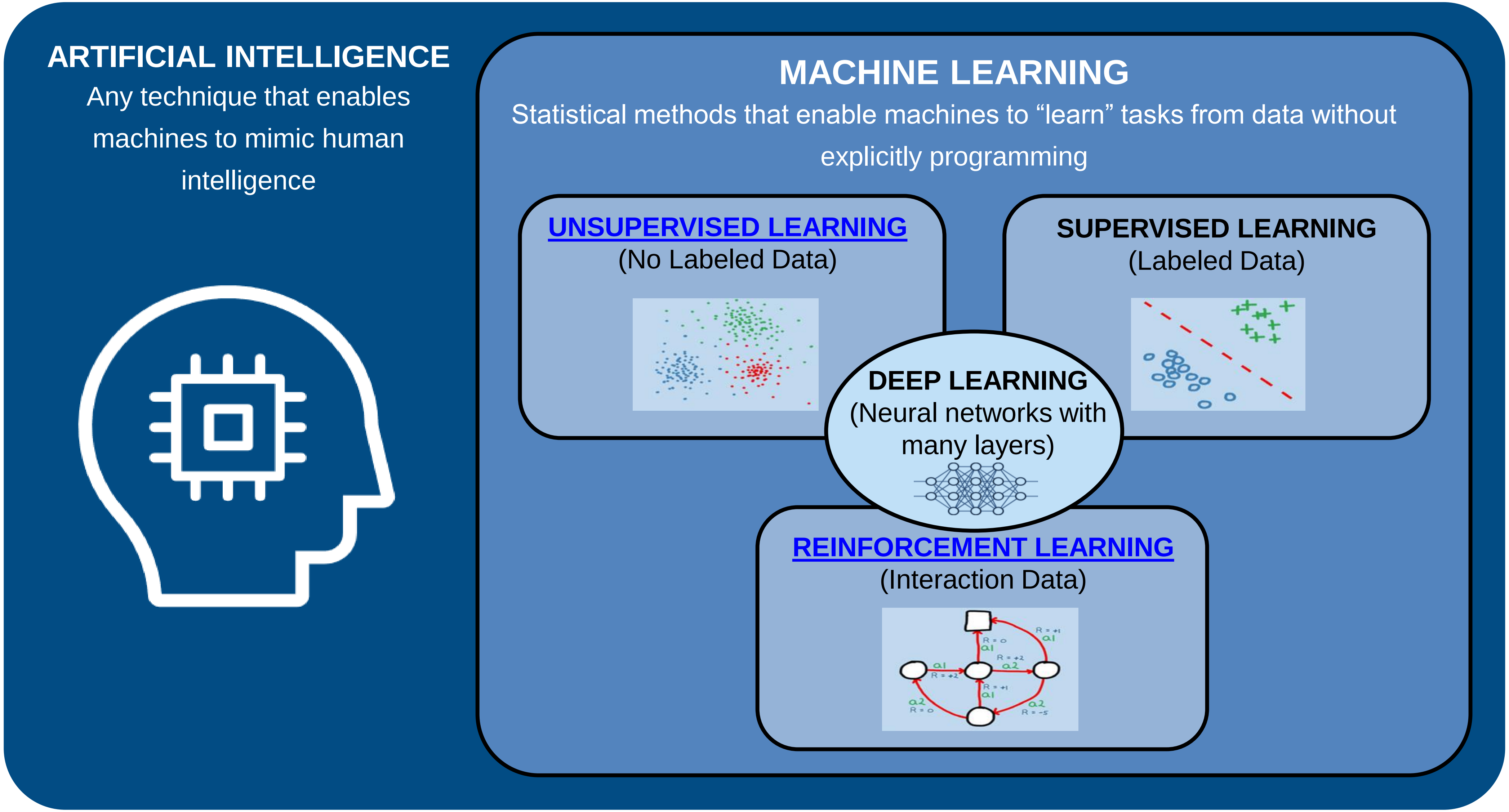


BLACK BOX

GREY BOX

WHITE BOX

The AI megatrend



ARTIFICIAL INTELLIGENCE
Any technique that enables machines to mimic human intelligence

MACHINE LEARNING
Statistical methods that enable machines to “learn” tasks from data without explicitly programming

UNSUPERVISED LEARNING
(No Labeled Data)

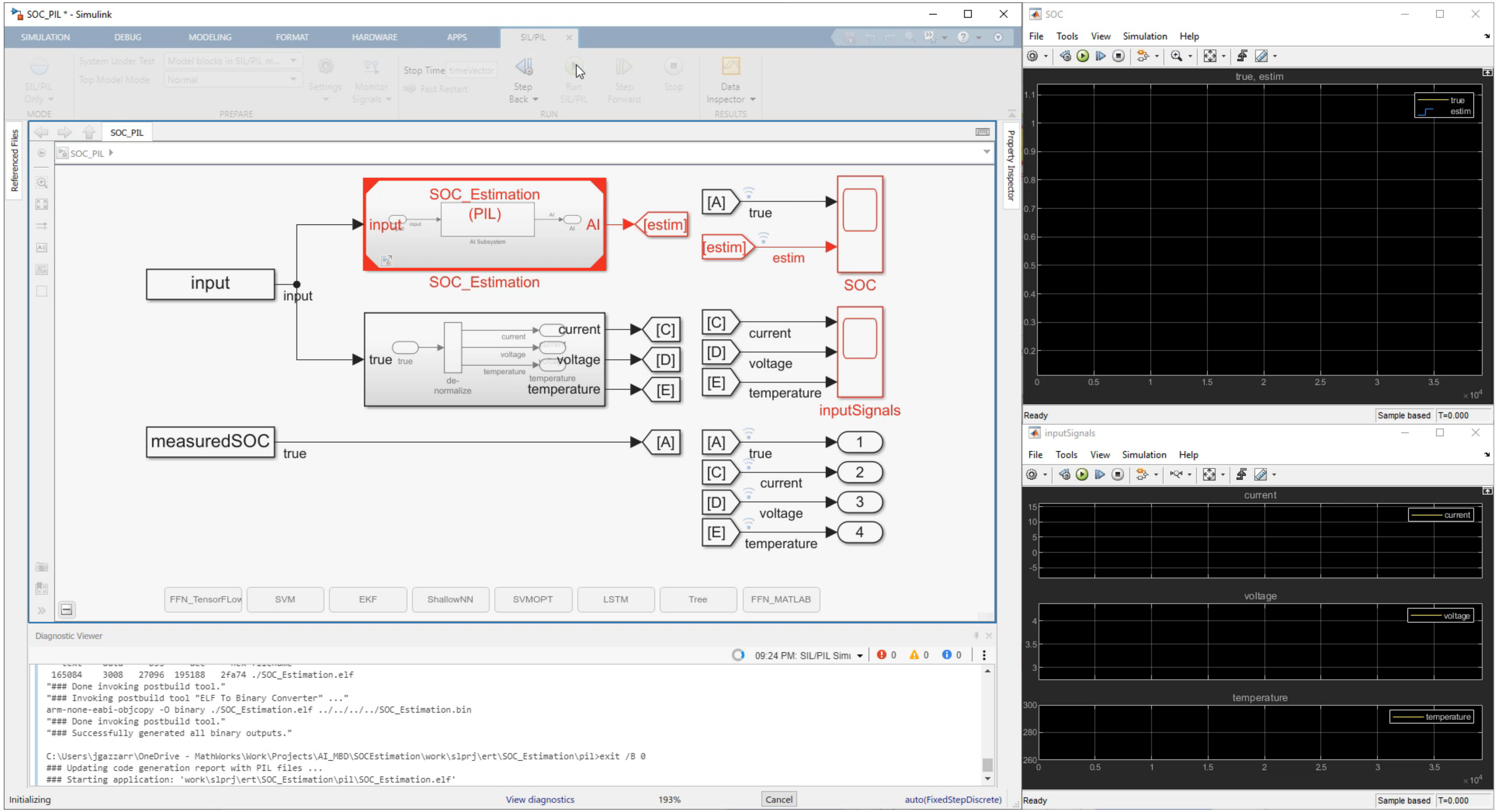
SUPERVISED LEARNING
(Labeled Data)

DEEP LEARNING
(Neural networks with many layers)

REINFORCEMENT LEARNING
(Interaction Data)

Virtual sensor for Battery State of Charge (SOC) estimation

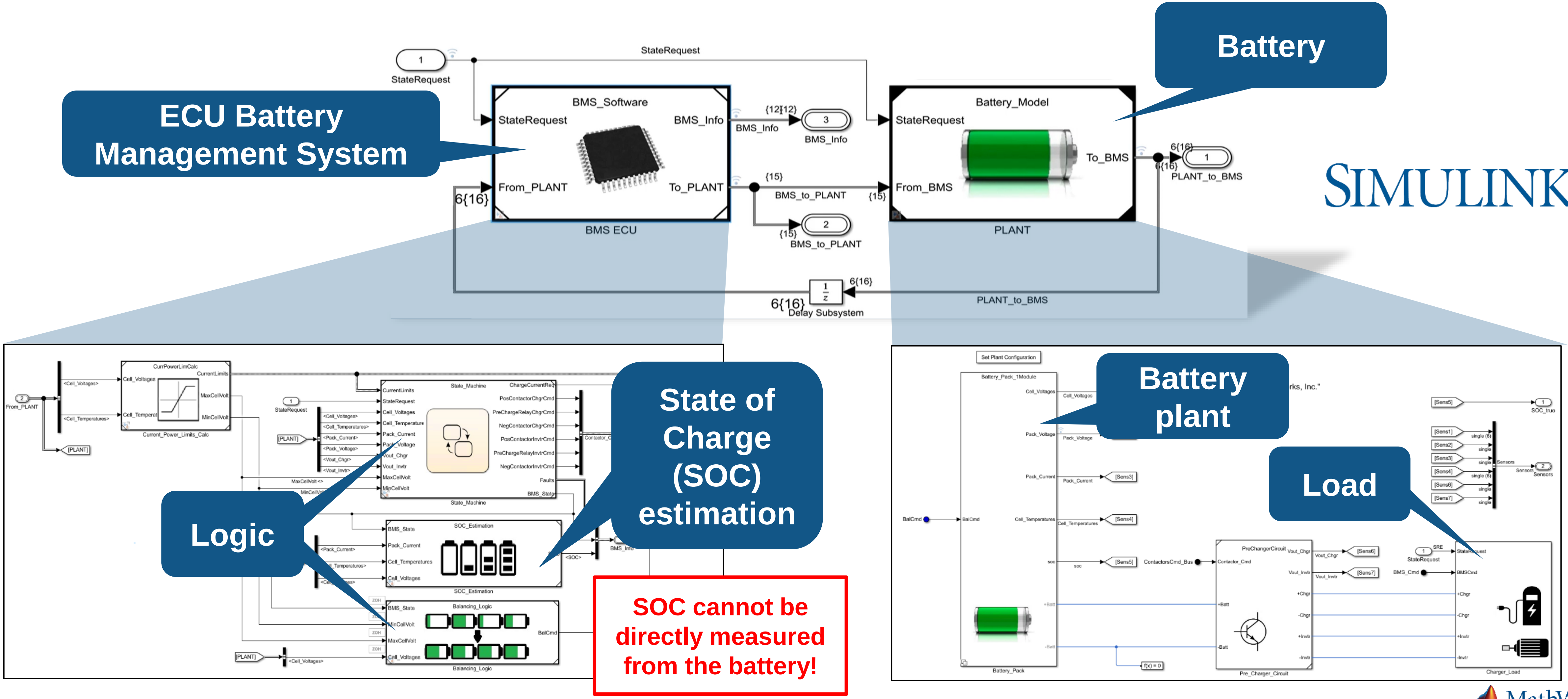
Using AI-based virtual sensors in System-level simulation



```
 165084    3008   27096  195188   2fa74 ./SOC_Estimation.elf
"### Done invoking postbuild tool."
"### Invoking postbuild tool "ELF To Binary Converter" ..."
arm-none-eabi-objcopy -O binary ./SOC_Estimation.elf ../../../../SOC_Estimation.bin
"### Done invoking postbuild tool."
"### Successfully generated all binary outputs."

C:\Users\jgazzarr\OneDrive - MathWorks\Work\Projects\AI_MBD\SOCEstimation\work\slprj\ert\SOC_Estimation\pil>exit /B 0
### Updating code generation report with PIL files ...
### Starting application: 'work\slprj\ert\SOC_Estimation\pil\SOC_Estimation.elf'
```

Having a physical sensor might not be feasible due to cost, manufacturing process, reliability, degradation, etc.



ECU Battery Management System

Battery

SIMULINK®

Logic

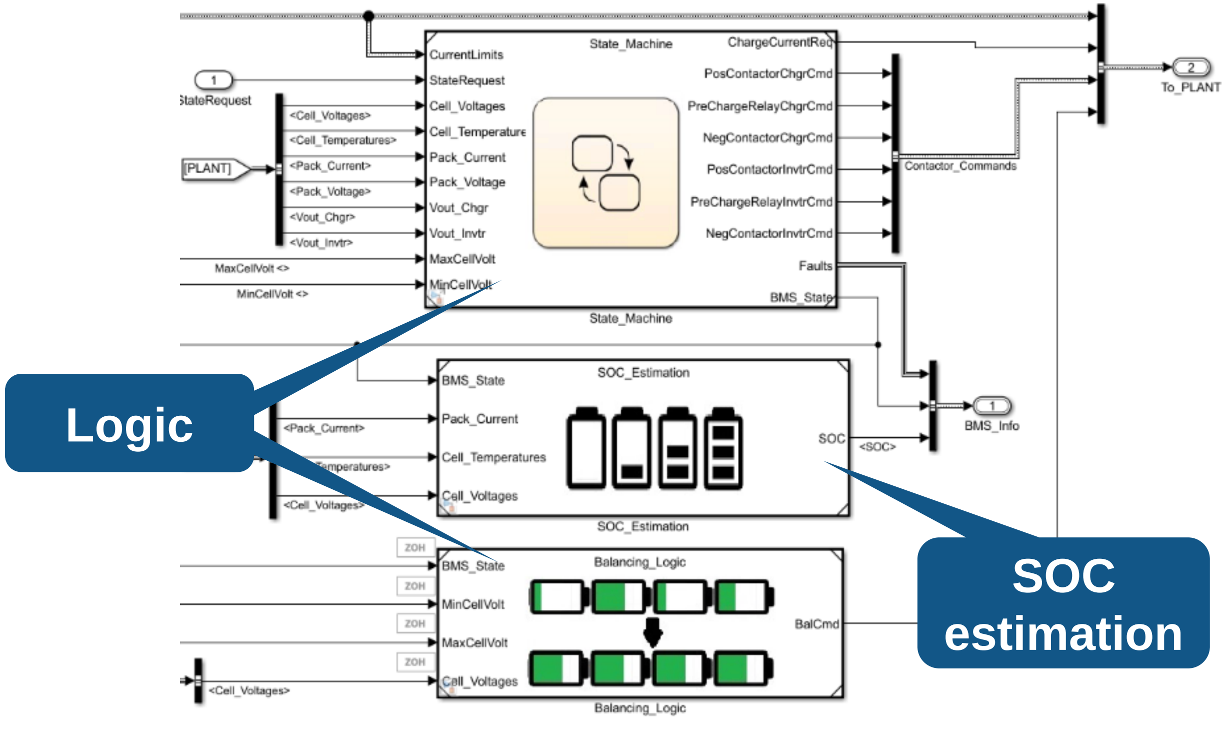
State of Charge (SOC) estimation

SOC cannot be directly measured from the battery!

Battery plant

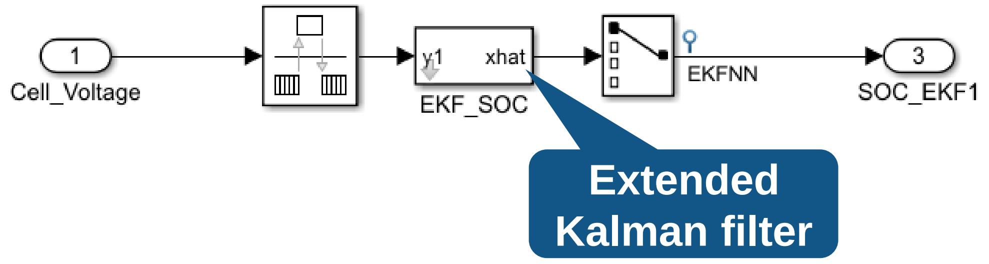
Load

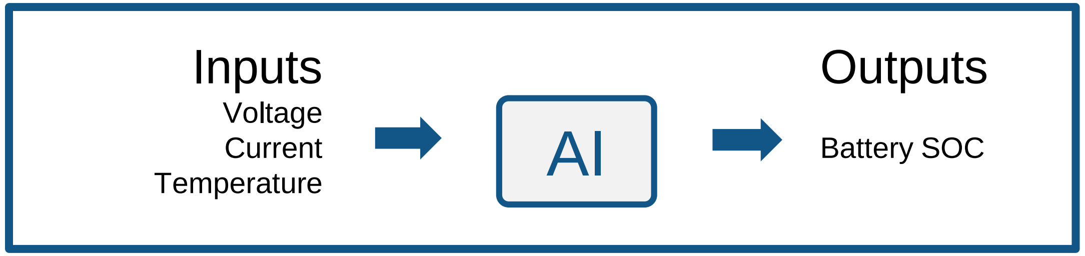
Virtual sensor for Battery State of Charge (SOC) estimation



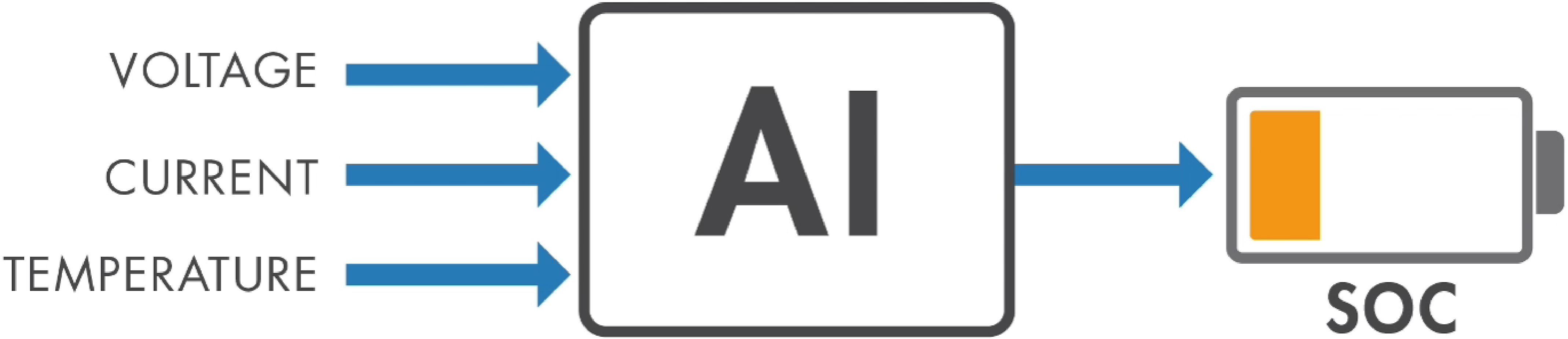
Why AI over Kalman filtering?

- No need of an internal battery model
- Training directly on measured data
- Capture very complex data relationships



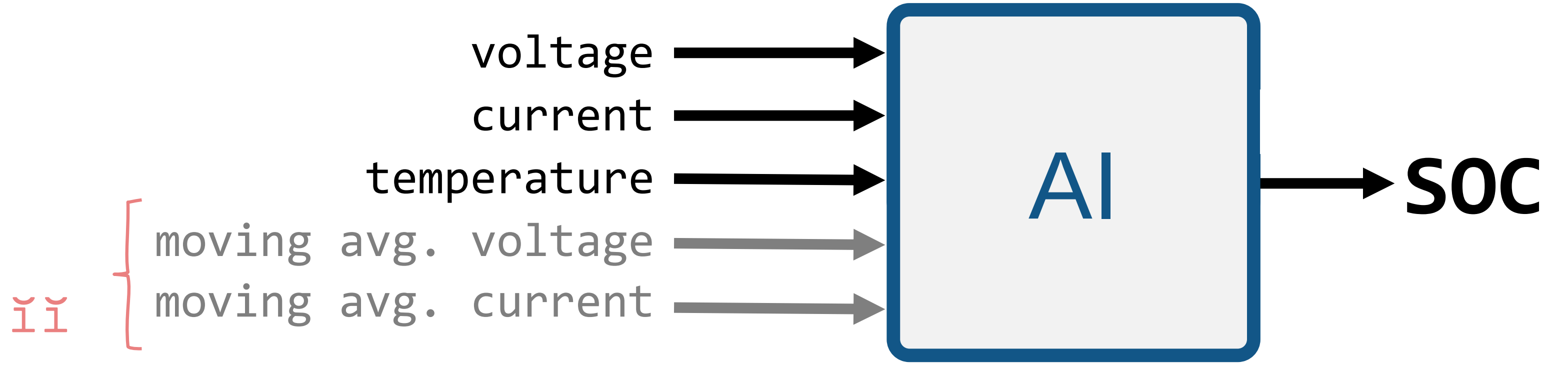


Battery State-of-Charge Estimation Using AI

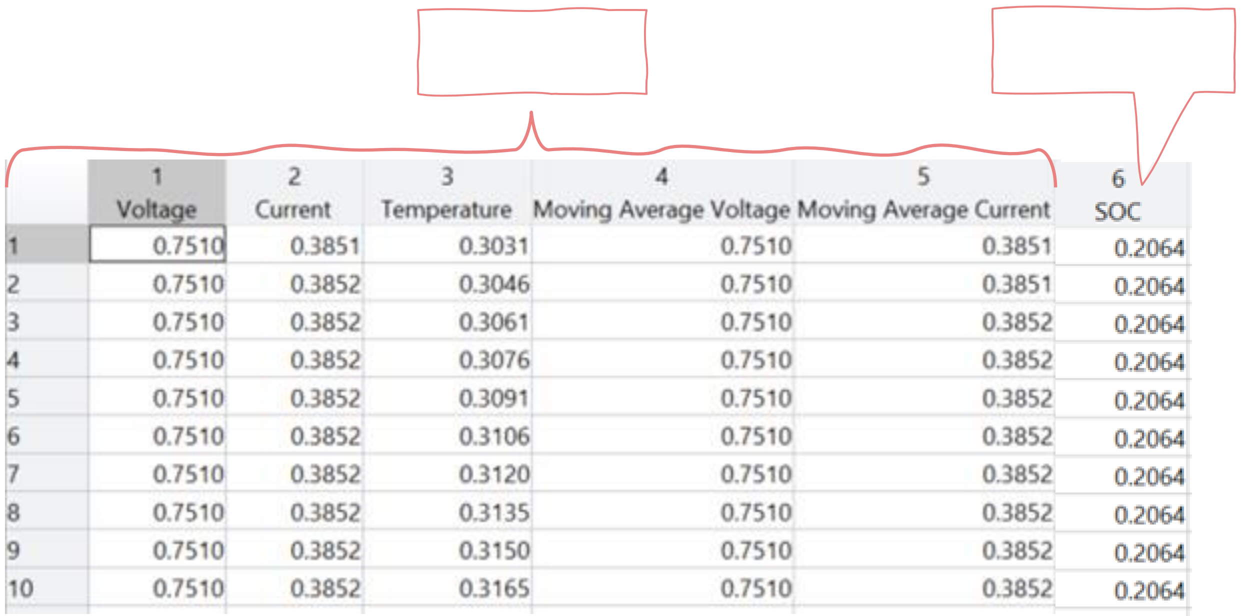


Data Preparation

Data source: McMaster University*



$$SOC(t) = \frac{1}{C}\int_0^t I(p)dp$$



	1 Voltage	2 Current	3 Temperature	4 Moving Average Voltage	5 Moving Average Current	6 SOC
1	0.7510	0.3851	0.3031	0.7510	0.3851	0.2064
2	0.7510	0.3852	0.3046	0.7510	0.3851	0.2064
3	0.7510	0.3852	0.3061	0.7510	0.3852	0.2064
4	0.7510	0.3852	0.3076	0.7510	0.3852	0.2064
5	0.7510	0.3852	0.3091	0.7510	0.3852	0.2064
6	0.7510	0.3852	0.3106	0.7510	0.3852	0.2064
7	0.7510	0.3852	0.3120	0.7510	0.3852	0.2064
8	0.7510	0.3852	0.3135	0.7510	0.3852	0.2064
9	0.7510	0.3852	0.3150	0.7510	0.3852	0.2064
10	0.7510	0.3852	0.3165	0.7510	0.3852	0.2064

* https://data.mendeley.com/datasets/cp3473x7xv/3

AI Modeling

3 options for AI Modeling:

1

Train in MATLAB's **Machine Learning** Framework

2

Import model from **TensorFlow or PyTorch**

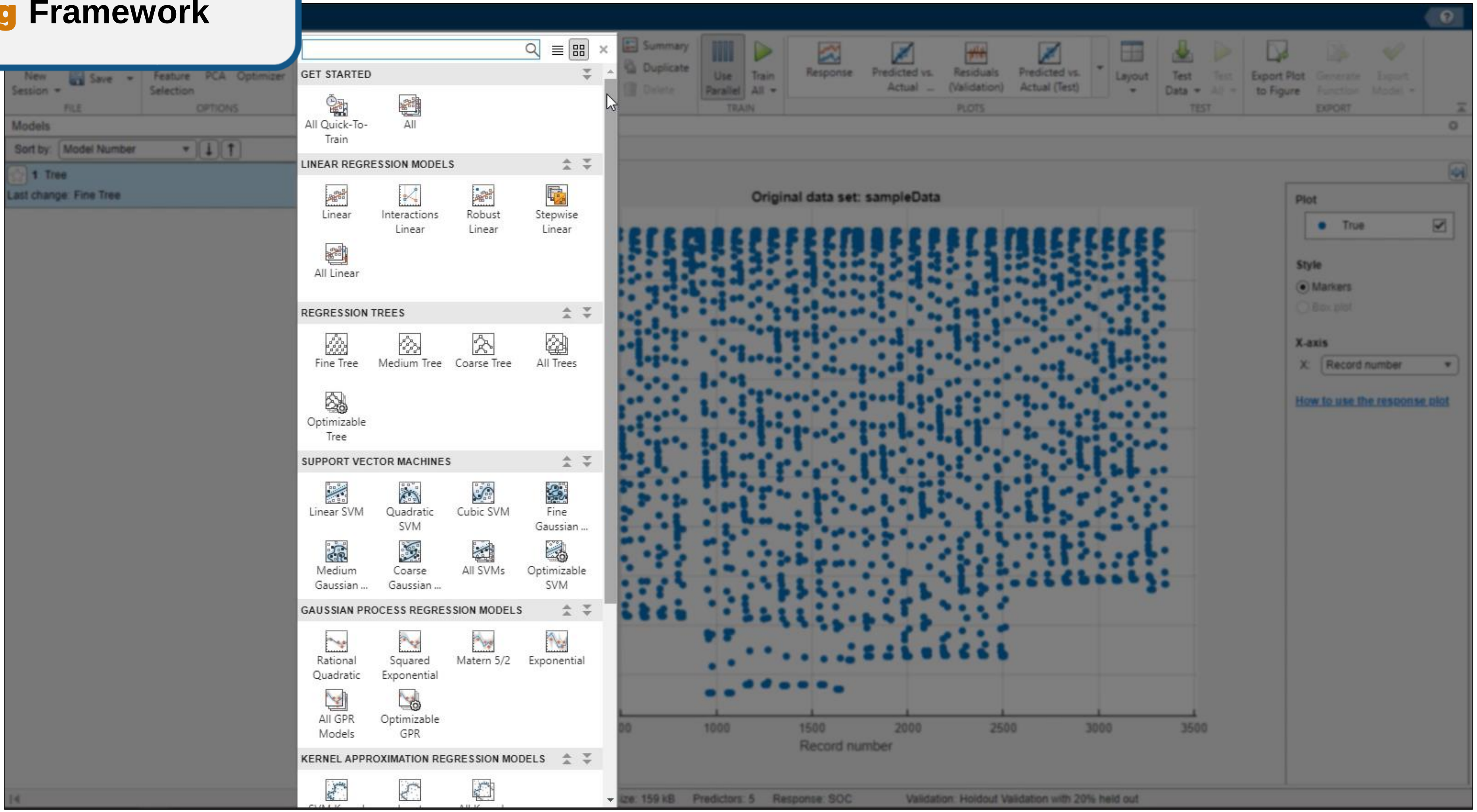
3

Train in MATLAB's **Deep Learning** Framework

AI Modeling

1

Train in MATLAB's **Machine Learning** Framework



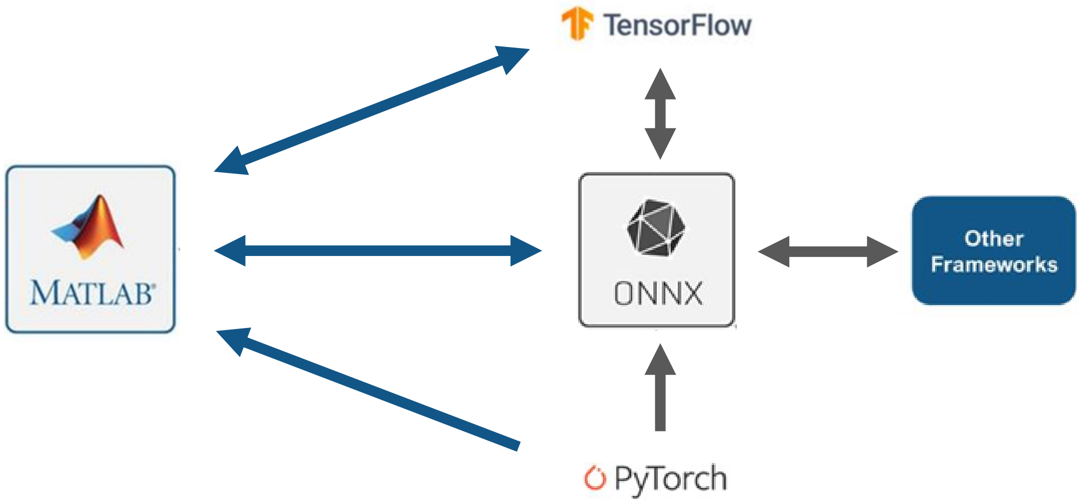
ize: 159 kB Predictors: 5 Response: SOC Validation: Holdout Validation with 20% held out

Data Preparation | AI Modeling | Simulation & Test | Deployment

MATLAB interoperates with other frameworks

Framework interoperability bridges the gap between data science, engineering and production

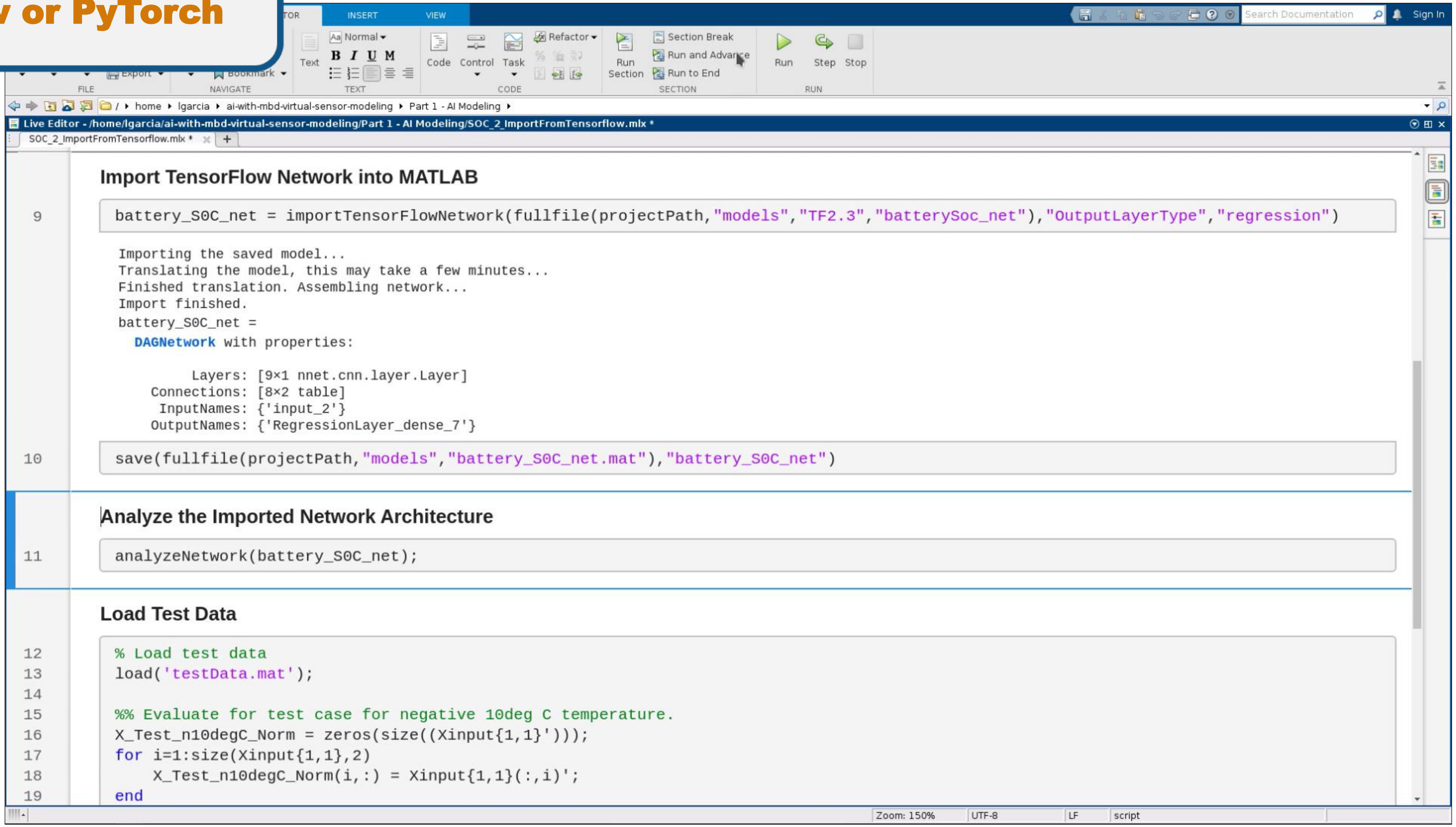
TensorFlow-Keras Import	R2017b
ONNX Converter (Import & Export)	R2018a
TensorFlow Converter (Import)	R2021a
TensorFlow Converter (Export)	R2022b
PyTorch Converter (Import)	R2022b



AI Modeling

2

Import model from
TensorFlow or PyTorch


TOR
INSERT
VIEW
Search Documentation
Sign In
Export
Bookmark
FILE
NAVIGATE
Text
Normal
B I U M
TEXT
Code
Control
Task
Refactor
CODE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
/ home lgarcia ai-with-mbd-virtual-sensor-modeling Part 1 - AI Modeling
Live Editor - /home/lgarcia/ai-with-mbd-virtual-sensor-modeling/Part 1 - AI Modeling/SOC_2_ImportFromTensorflow.mlx *
SOC_2_ImportFromTensorflow.mlx *
Zoom: 150%
UTF-8
LF
script

Import TensorFlow Network into MATLAB

```
battery_SOC_net = importTensorFlowNetwork(fullfile(projectPath,"models","TF2.3","batterySoc_net"),"OutputLayerType","regression")
```

```
Importing the saved model...
Translating the model, this may take a few minutes...
Finished translation. Assembling network...
Import finished.
battery_SOC_net =
  DAGNetwork with properties:

         Layers: [9×1 nnet.cnn.layer.Layer]
    Connections: [8×2 table]
     InputNames: {'input_2'}
    OutputNames: {'RegressionLayer_dense_7'}
```

```
save(fullfile(projectPath,"models","battery_SOC_net.mat"),"battery_SOC_net")
```

Analyze the Imported Network Architecture

```
analyzeNetwork(battery_SOC_net);
```

Load Test Data

```
% Load test data
load('testData.mat');

%% Evaluate for test case for negative 10deg C temperature.
X_Test_n10degC_Norm = zeros(size((Xinput{1,1}')));
for i=1:size(Xinput{1,1},2)
    X_Test_n10degC_Norm(i,:) = Xinput{1,1}(:,i)';
end
```

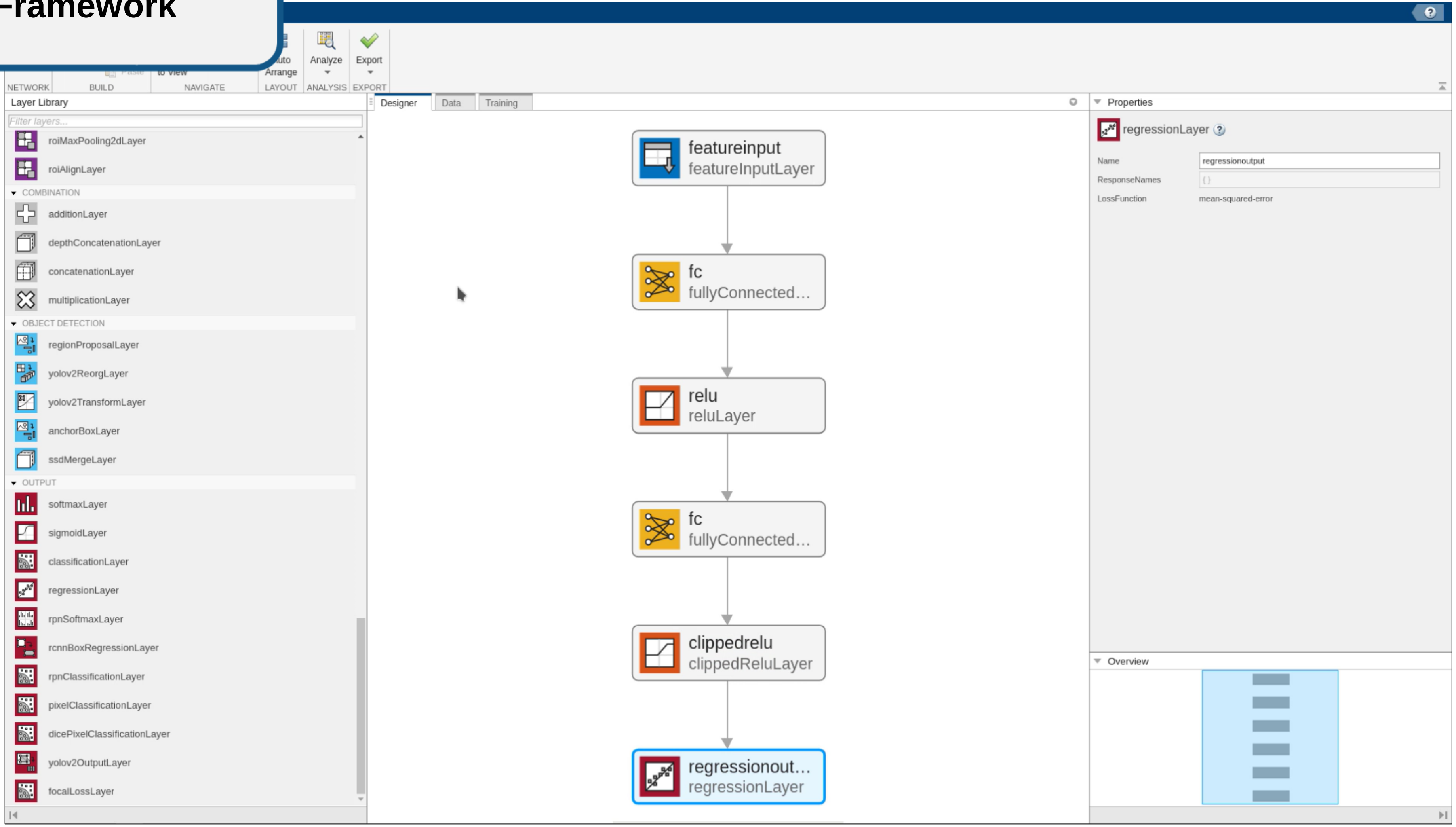
Data Preparation | **AI Modeling** | Simulation & Test | Deployment

MathWorks

AI Modeling

3

Train in MATLAB's Deep Learning Framework



Name: regressionoutput

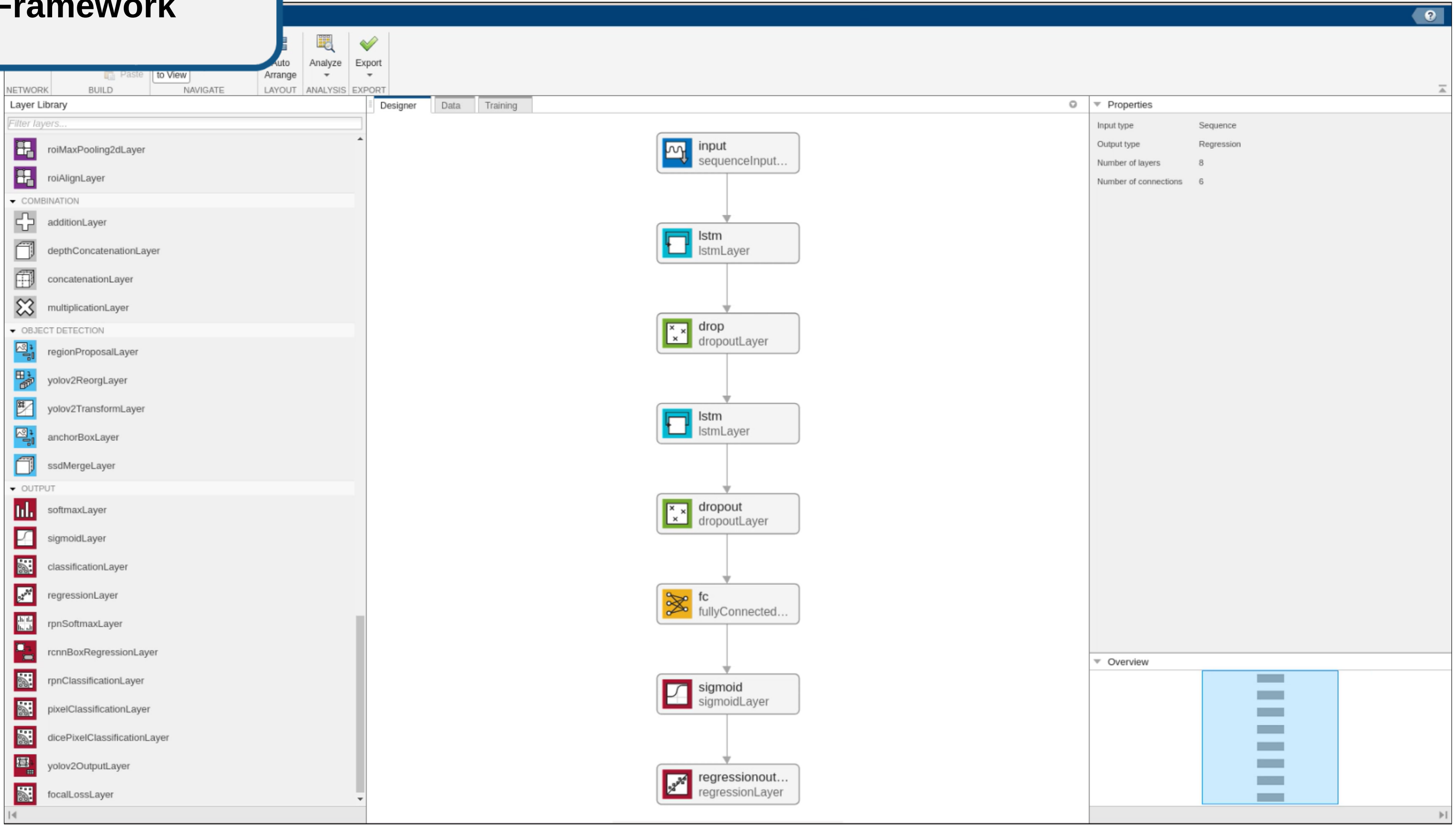
ResponseNames: {}

LossFunction: mean-squared-error

AI Modeling

3

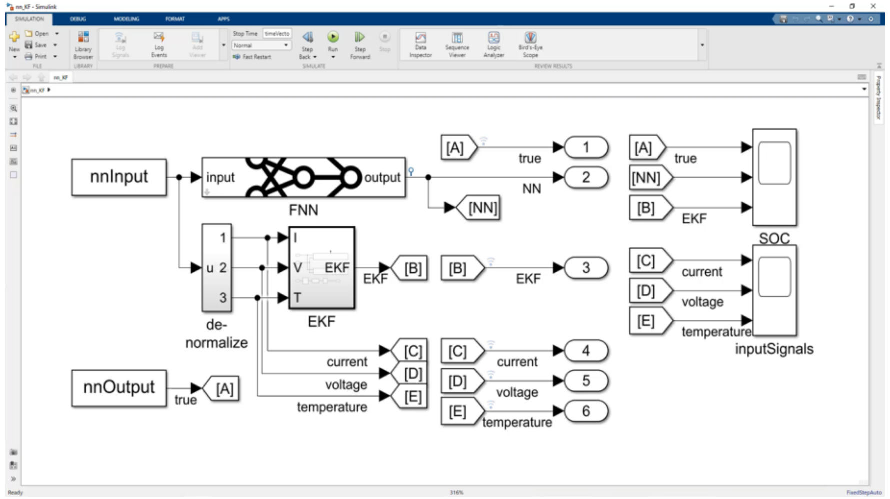
Train in MATLAB's **Deep Learning** Framework



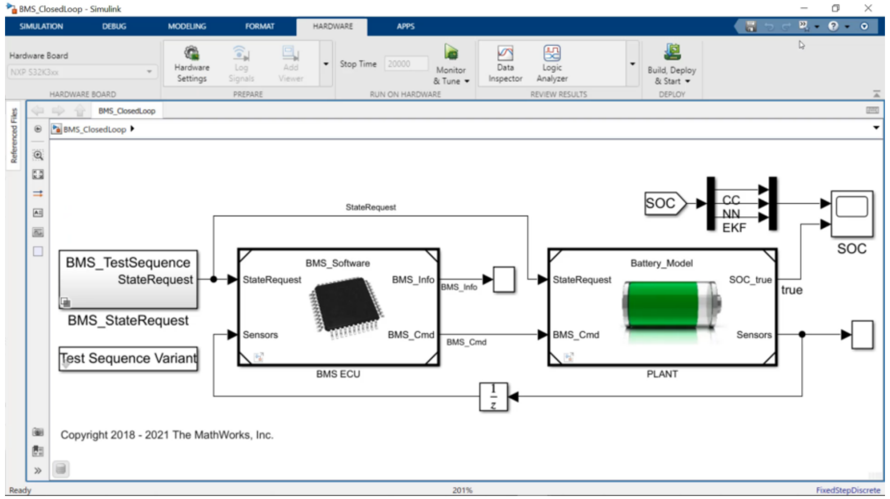
Input type	Sequence
Output type	Regression
Number of layers	8
Number of connections	6

Integrate your AI model for system-level simulation and test

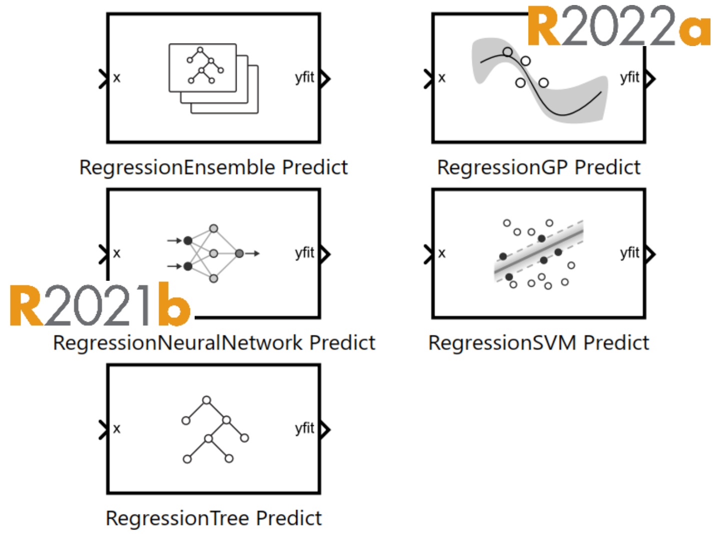
Integration of trained AI model into Simulink

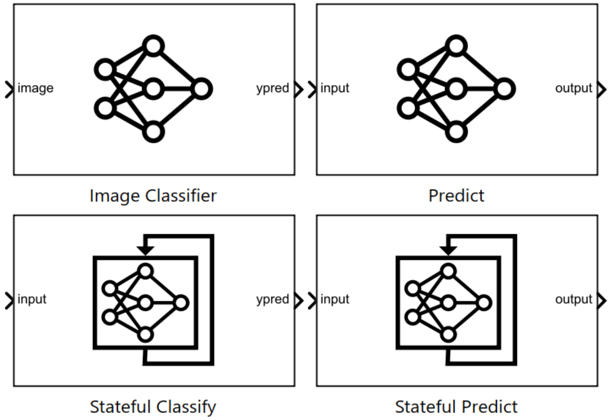


System-level simulation

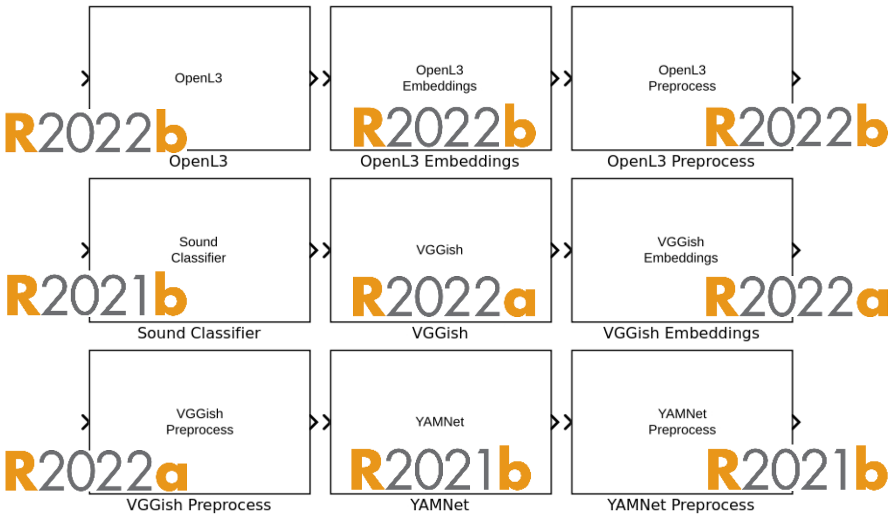


AI libraries in Simulink are expanding to include more AI blocks for more applications

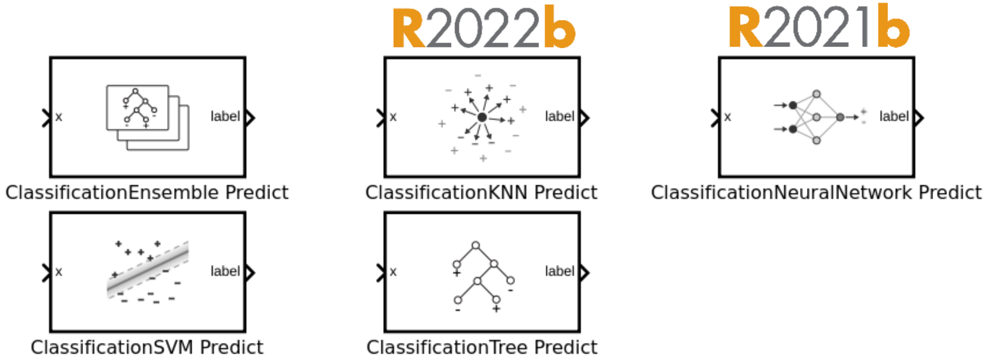


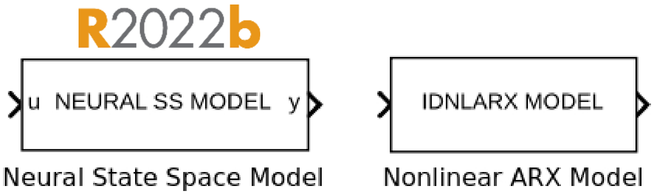


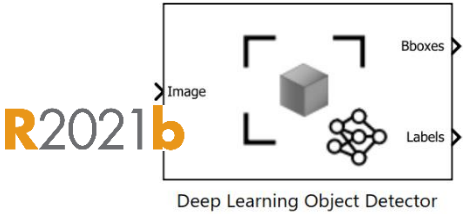
Deep Learning Toolbox



Audio Toolbox


R2022b
R2021b
x
label
ClassificationEnsemble Predict
ClassificationKNN Predict
ClassificationNeuralNetwork Predict
ClassificationSVM Predict
ClassificationTree Predict





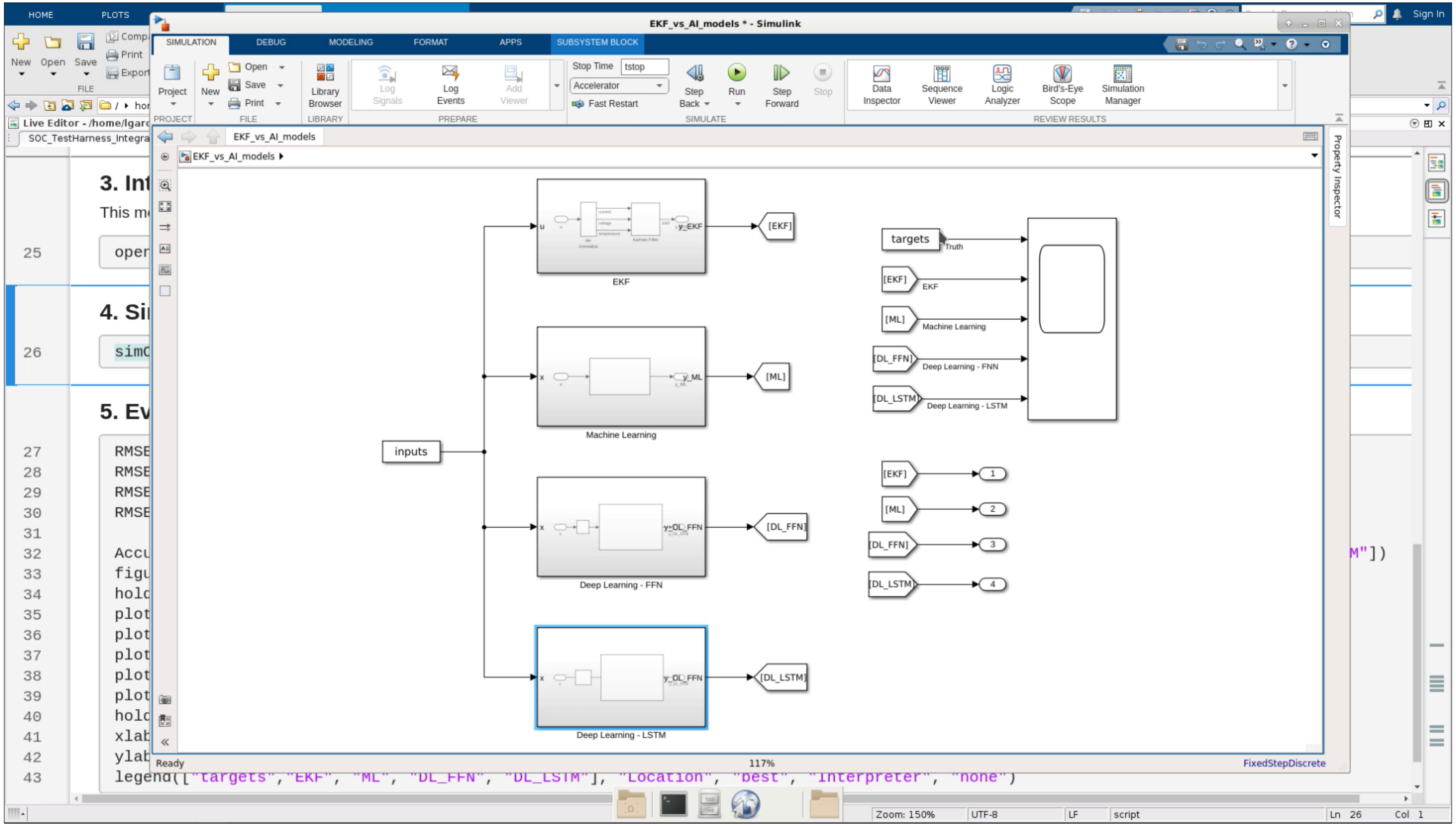
Computer Vision Toolbox

System Identification Toolbox

Statistics and Machine Learning Toolbox

Data Preparation | AI Modeling | **Simulation & Test** | Deployment

Integration of trained AI models into Simulink



3. Int

This m

```
oper
```

4. Si

```
simO
```

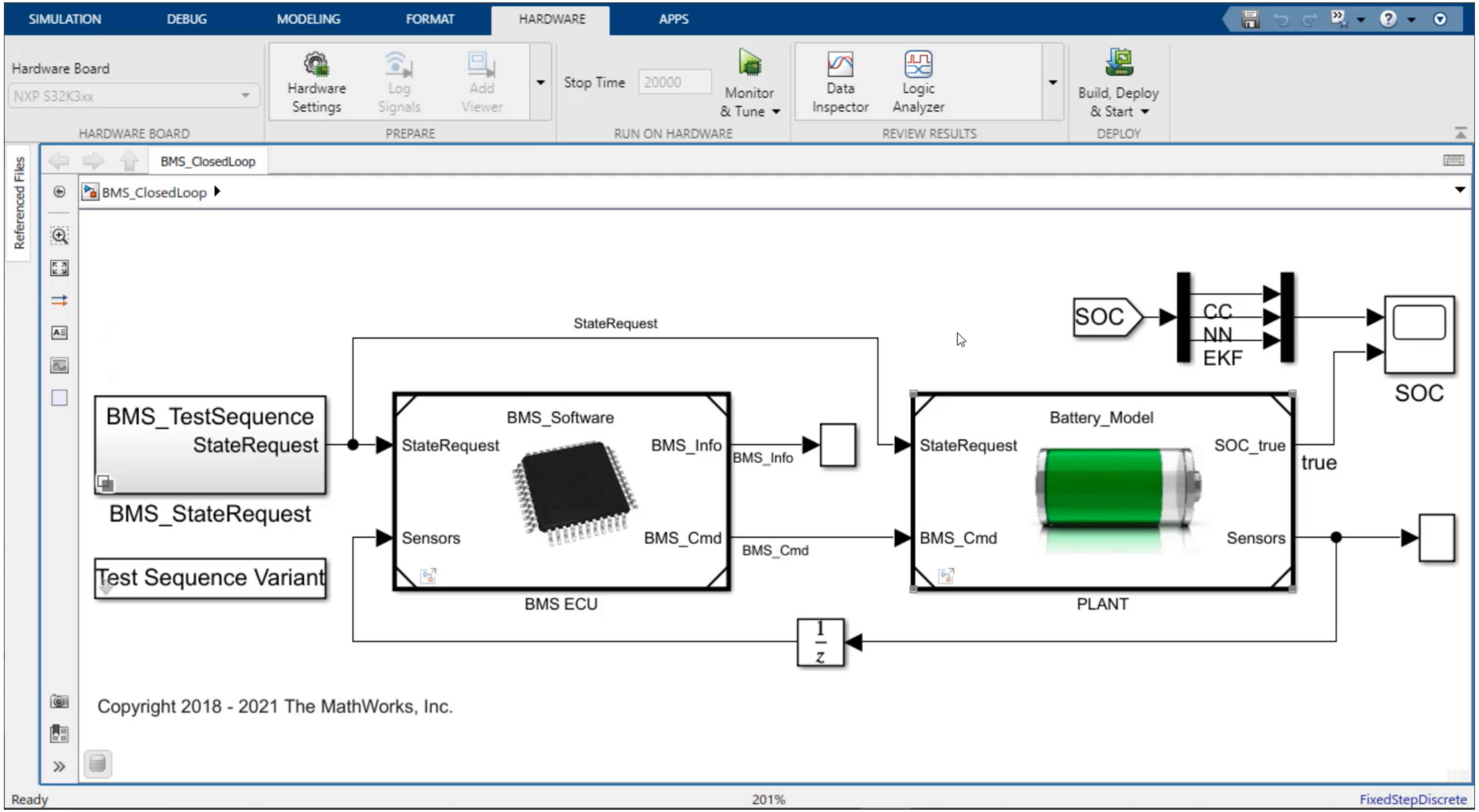
5. Ev

```
RMSE
RMSE
RMSE
RMSE

Accu
figu
hold
plot
plot
plot
plot
plot
hold
xlab
ylab
legend(["targets","EKF","ML","DL_FFN","DL_LSTM"], "Location", "best", "Interpreter", "none")
```

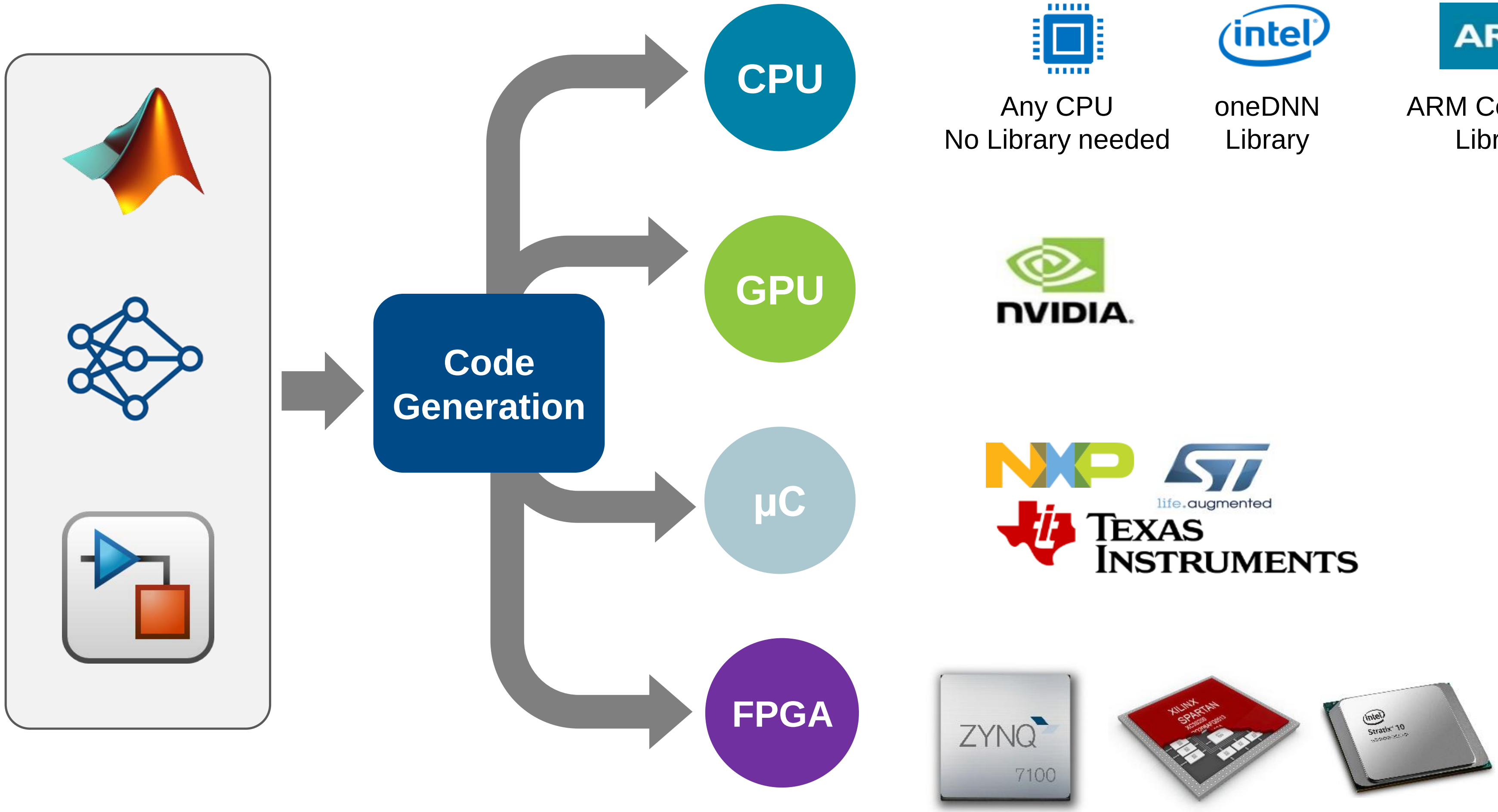
Data Preparation | AI Modeling | **Simulation & Test** | Deployment

Closed-Loop System-Level Simulation



Data Preparation | AI Modeling | **Simulation & Test** | Deployment

Deploy to target with zero coding errors



Code Generation

CPU

Any CPU
No Library needed

oneDNN
Library

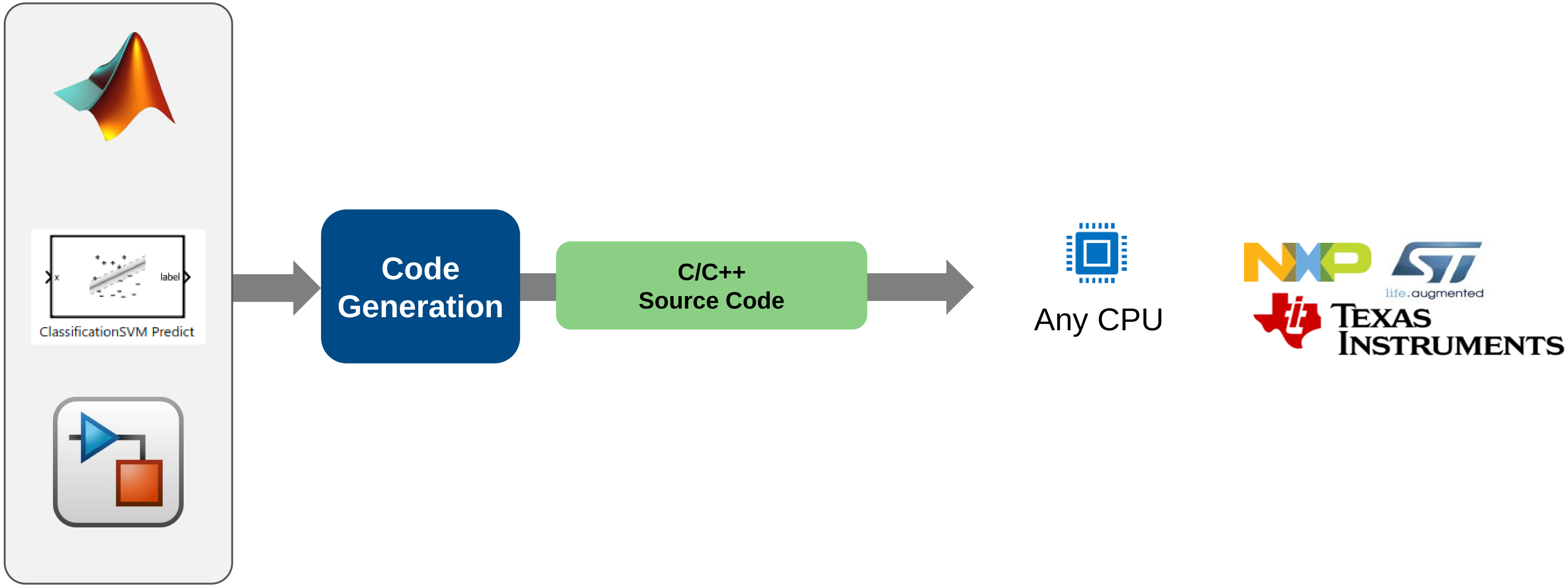
ARM Compute
Library

GPU

µC

FPGA

Use Embedded Coder to Generate Code for Machine Learning

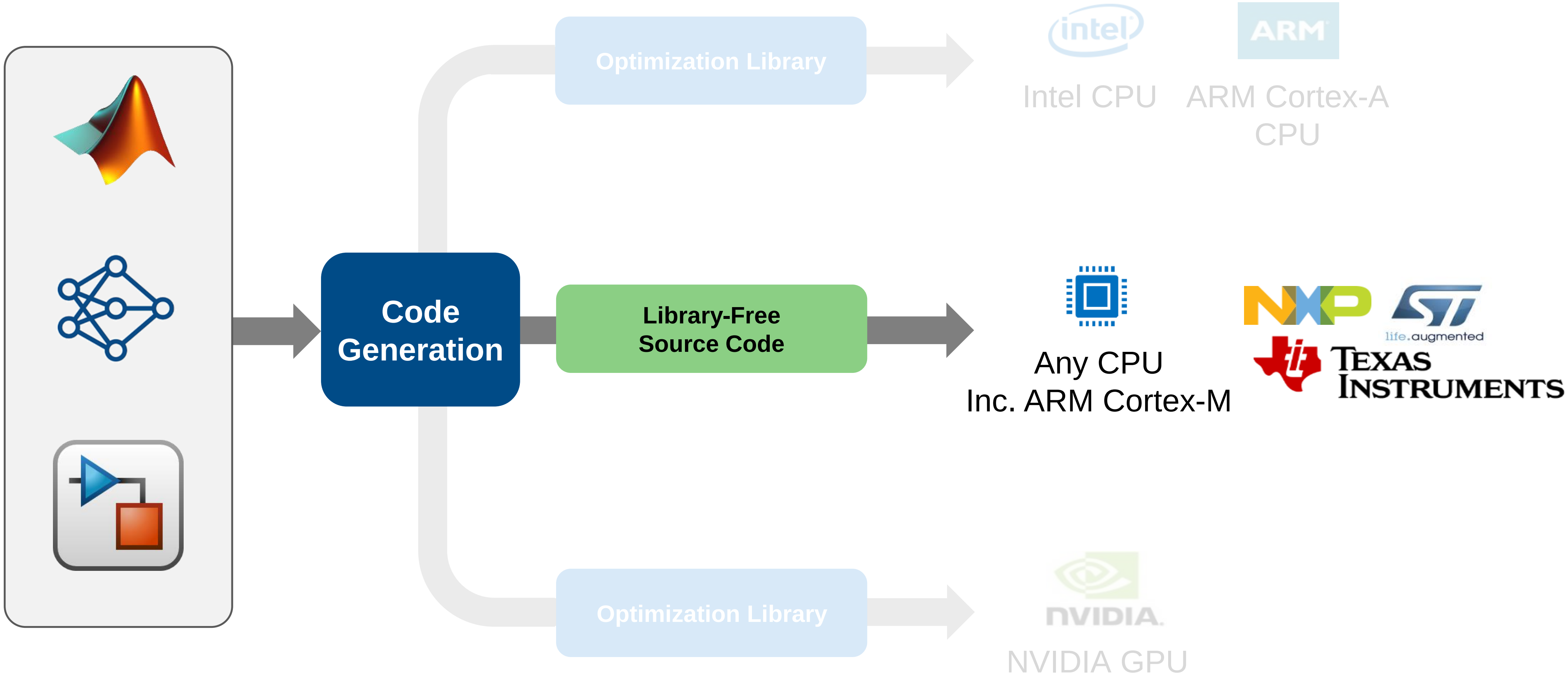


Code Generation

C/C++
Source Code

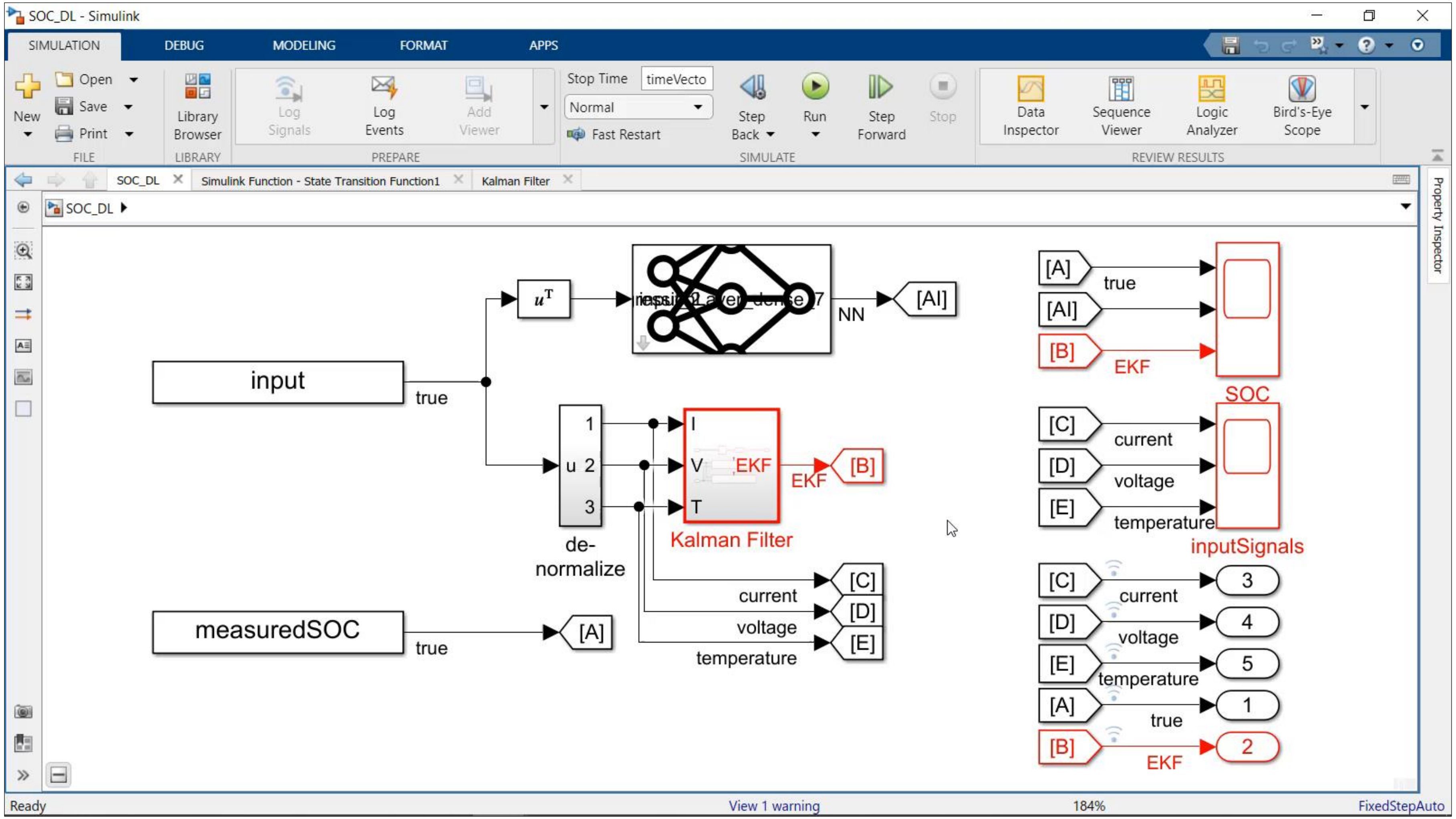
Any CPU

Generate Library-Free C/C++ Code for Deep Learning Networks



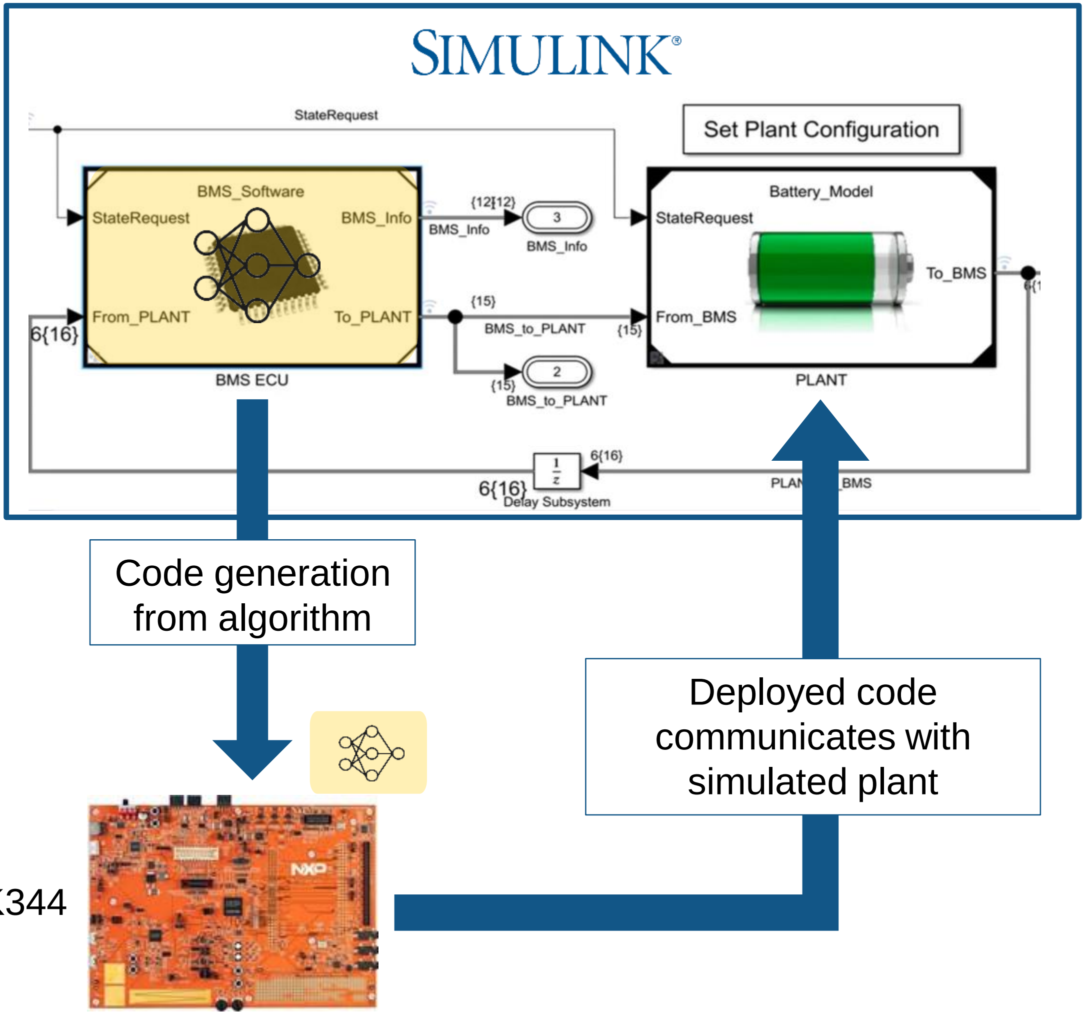
Data Preparation | AI Modeling | Simulation & Test | Deployment

Generate Library-Free C/C++ Code for Deep Learning Networks



Data Preparation | AI Modeling | Simulation & Test | **Deployment**

Processor-in-the-Loop Testing on ARM Cortex-M7 Processor



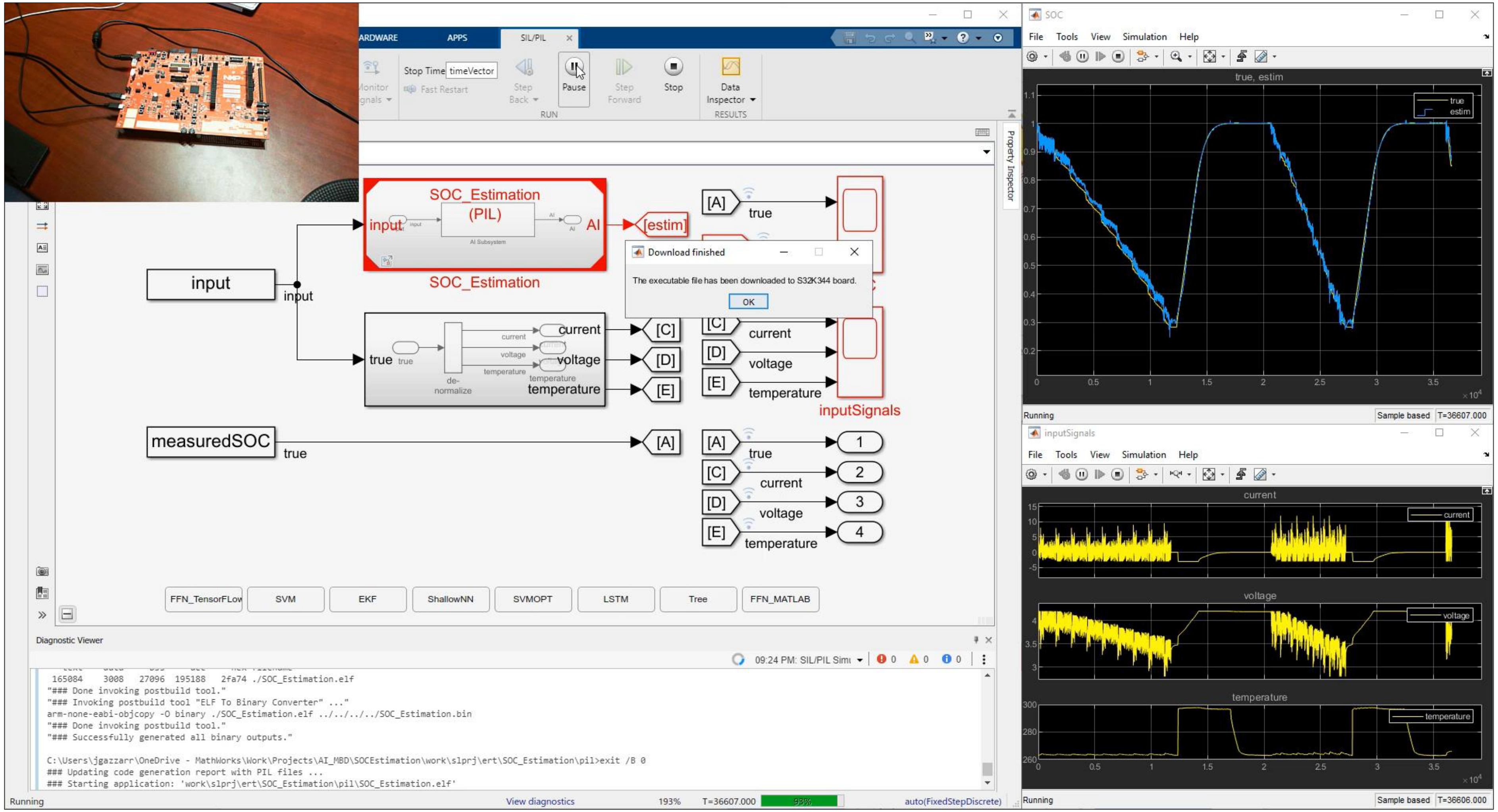
Code generation from algorithm

Deployed code communicates with simulated plant

NXP S32K344 board

Arm® Cortex®-M

Processor-in-the-Loop Testing on ARM Cortex-M7 Processor



Download finished

The executable file has been downloaded to S32K344 board.

OK

```
 165084    3008   27096  195188   2fa74 ./SOC_Estimation.elf
"### Done invoking postbuild tool."
"### Invoking postbuild tool "ELF To Binary Converter" ..."
arm-none-eabi-objcopy -O binary ./SOC_Estimation.elf ../../../../SOC_Estimation.bin
"### Done invoking postbuild tool."
"### Successfully generated all binary outputs."

C:\Users\jgazzarr\OneDrive - MathWorks\Work\Projects\AI_MBD\SOCEstimation\work\slprj\ert\SOC_Estimation\pil>exit /B 0
### Updating code generation report with PIL files ...
### Starting application: 'work\slprj\ert\SOC_Estimation\pil\SOC_Estimation.elf'
```

Data Preparation | AI Modeling | Simulation & Test | **Deployment**

Manage AI tradeoffs for your system

	EKF Extended Kalman Filter	**Tree** Fine Regression Tree	**FFN** 1-hidden layer Feedforward Network	**LSTM** Stacked Long Short-Term Memory Network
Preprocessing effort	🔵	🟡	🟡	🔵
Training Speed	N/A	🔵	🟡	🔴
Interpretability	🔵	🔵	🔴	🔴
Inference Speed	🟡	🔵	🔵🔵	🔴
Model Size	🔵	🟡	🔵	🔴
Accuracy (RSME)	🔵	🟡	🔵	🔵🔵

Results are specific to this Battery SOC Estimation example

Much Better 🔵🔵 Better 🔵 Okay 🟡 Worse 🔴

Model compression can bridge the gap between AI modelling and embedded deployment

Data Preparation

- Data cleansing and preparation
- Human insight
- Simulation-generated data

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

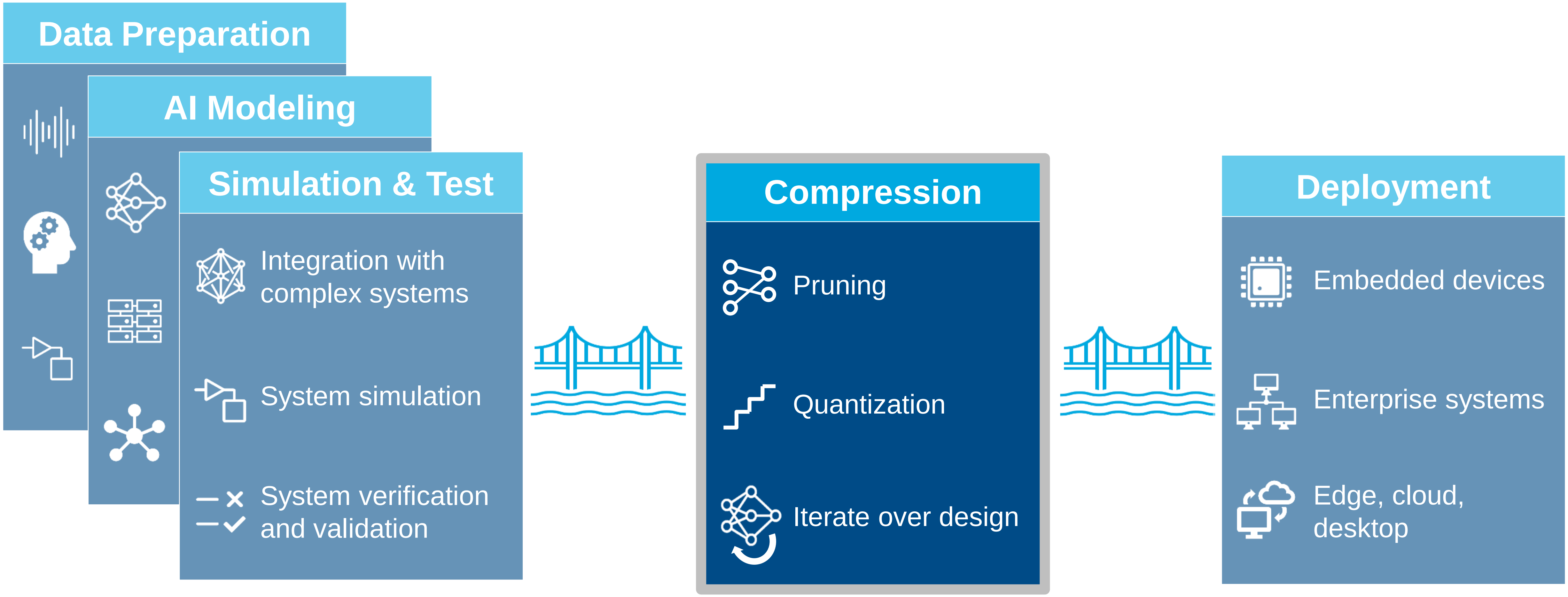
Simulation & Test

- Integration with complex systems
- System simulation
- System verification and validation

Deployment

- Embedded devices
- Enterprise systems
- Edge, cloud, desktop

Model compression can bridge the gap between AI modelling and embedded deployment



Data Preparation

AI Modeling

Simulation & Test

- Integration with complex systems
- System simulation
- System verification and validation

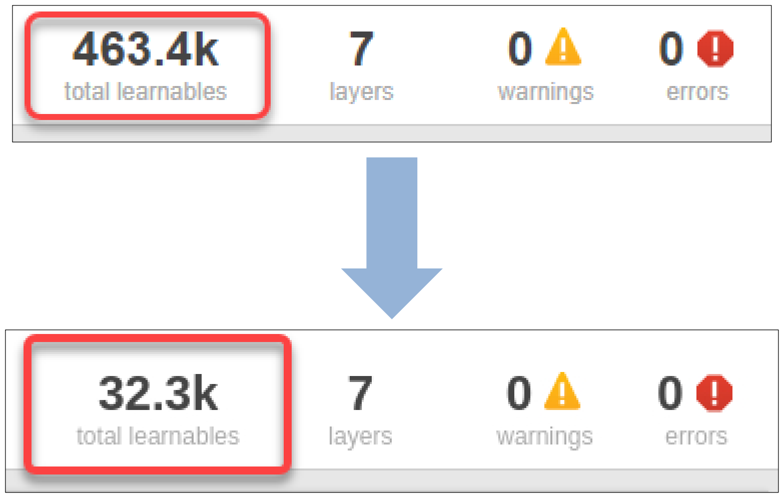
Compression

- Pruning
- Quantization
- Iterate over design

Deployment

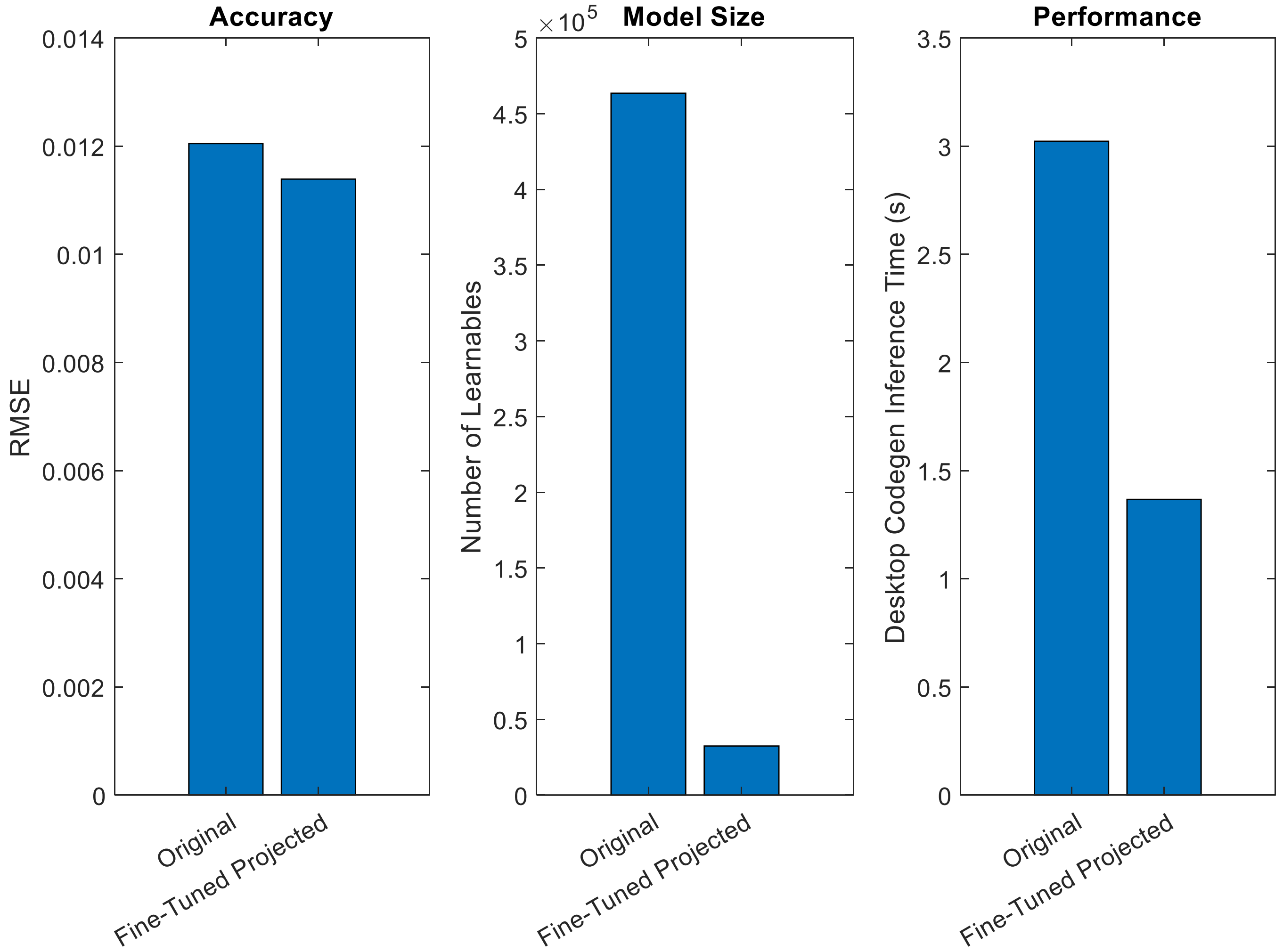
- Embedded devices
- Enterprise systems
- Edge, cloud, desktop

Model Compression Using Projection



463.4k total learnables | 7 layers | 0 ⚠ warnings | 0 ❗ errors

32.3k total learnables | 7 layers | 0 ⚠ warnings | 0 ❗ errors



Manage AI tradeoffs for your system

	EKF Extended Kalman Filter	**Tree** Fine Regression Tree	**FFN** 1-hidden layer Feedforward Network	**LSTM** Stacked Long Short-Term Memory Network	**LSTM*** * Compressed Stacked Long Short-Term Memory Network
Preprocessing effort	🔵	🟡	🟡	🔵	🔵
Training Speed	N/A	🔵	🟡	🔴	🔴
Interpretability	🔵	🔵	🔴	🔴	🔴
Inference Speed	🟡	🔵	🔵🔵	🔴	🟡
Model Size	🔵	🟡	🔵	🔴	🔵
Accuracy (RSME)	🔵	🟡	🔵	🔵🔵	🔵🔵

Results are specific to this Battery SOC Estimation example

Much Better 🔵🔵 Better 🔵 Okay 🟡 Worse 🔴

Key takeaways

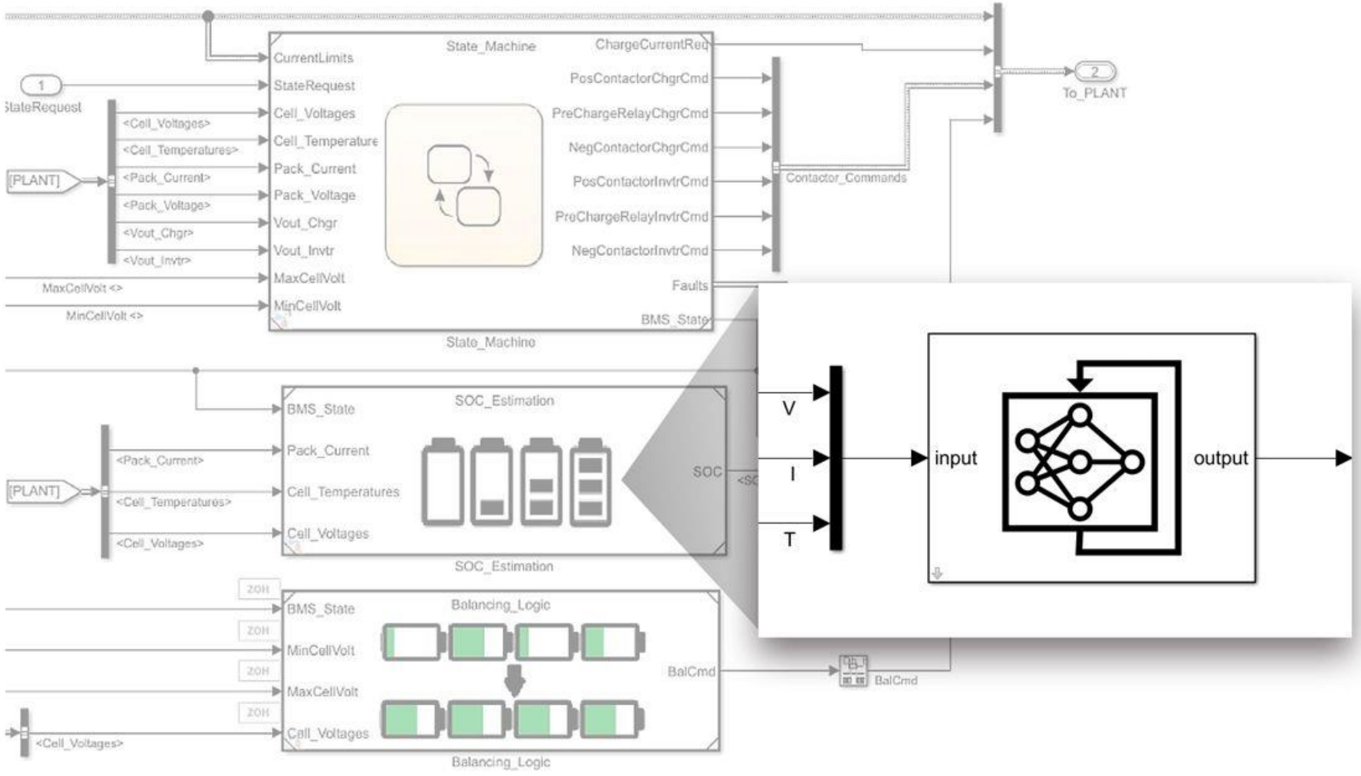
Data Preparation	AI Modeling	Simulation & Test	Compression	Deployment
Toolboxes for **domain-specific** pre/post-processing	**Low-code** workflow for AI Modeling through Apps Import models from **TensorFlow**, **PyTorch** or other DL Frameworks	**Simulink blocks for AI** models make integration easy	**Model compression techniques** to reduce model size and speed up inference	**Code generation** for embedded targets (incl. library free source code for Deep Learning)

Select and implement the optimal AI technique

Model Accuracy
Deployment Efficiency

In summary, build a virtual sensor using AI and integrate into Simulink for system-level simulation and code generation

Questions?


State_Machine
CurrentLimits
StateRequest
Cell_Voltages
Cell_Temperature
Pack_Current
Pack_Voltage
Vout_Chgr
Vout_Invtr
MaxCellVolt
MinCellVolt
ChargeCurrentReq
PosContactorChgrCmd
PreChargeRelayChgrCmd
NegContactorChgrCmd
PosContactorInvtrCmd
PreChargeRelayInvtrCmd
NegContactorInvtrCmd
Faults
BMS_State
1
StateRequest
[PLANT]
<Cell_Voltages>
<Cell_Temperatures>
<Pack_Current>
<Pack_Voltage>
<Vout_Chgr>
<Vout_Invtr>
MaxCellVolt <>
MinCellVolt <>
Contactor_Commands
2
To_PLANT
SOC_Estimation
BMS_State
Pack_Current
Cell_Temperatures
Cell_Voltages
SOC
Balancing_Logic
ZOH
MinCellVolt
MaxCellVolt
BalCmd
<Cell_Voltages>
V
I
T
input
output