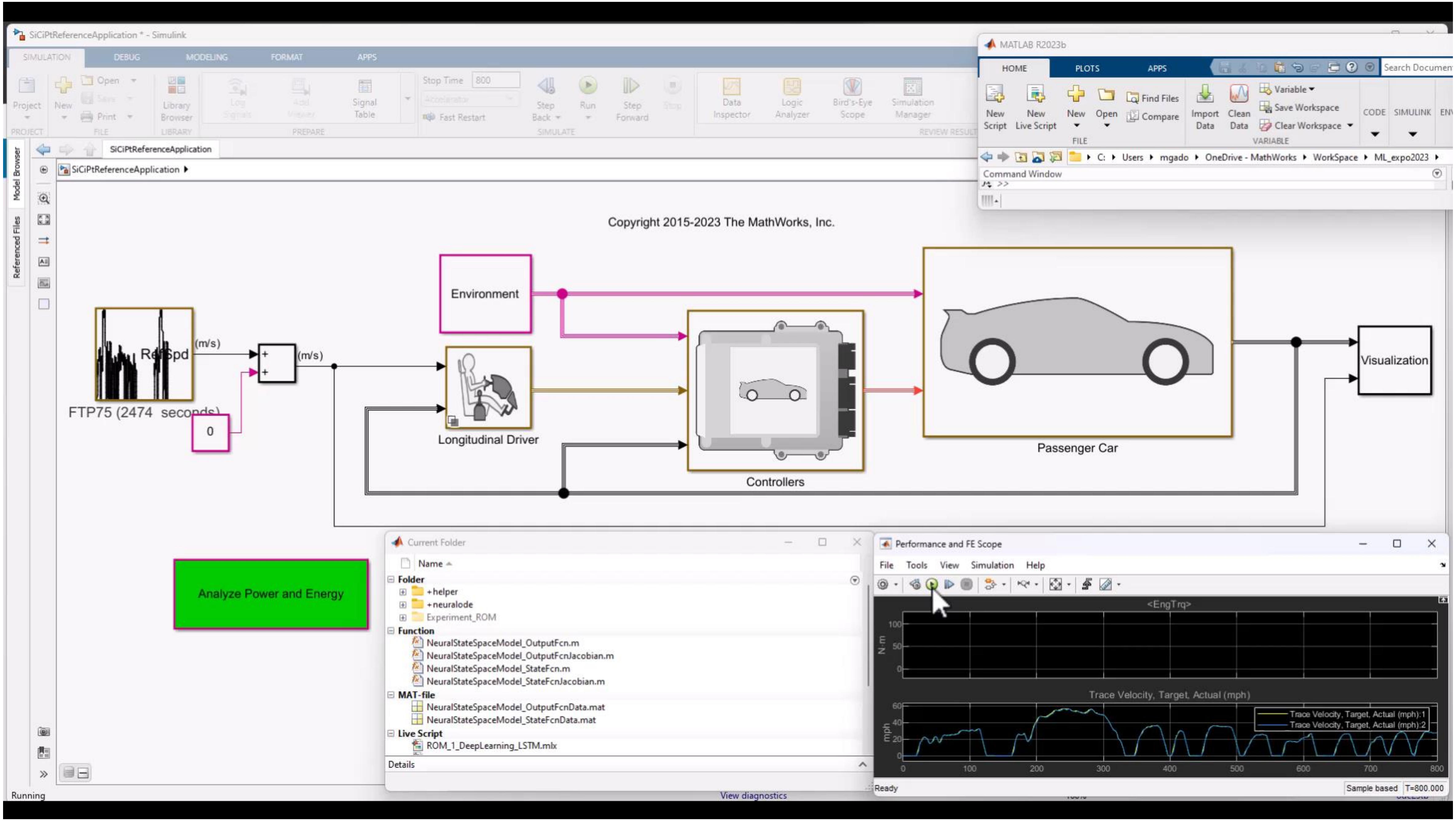
MATLAB EXPO

FRANCE

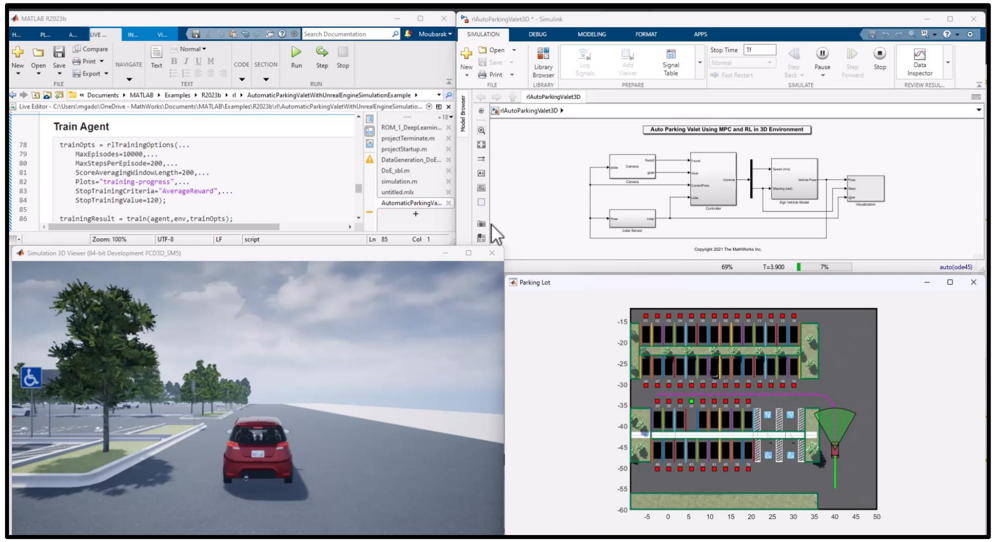
L'IA au service de la simulation des systèmes

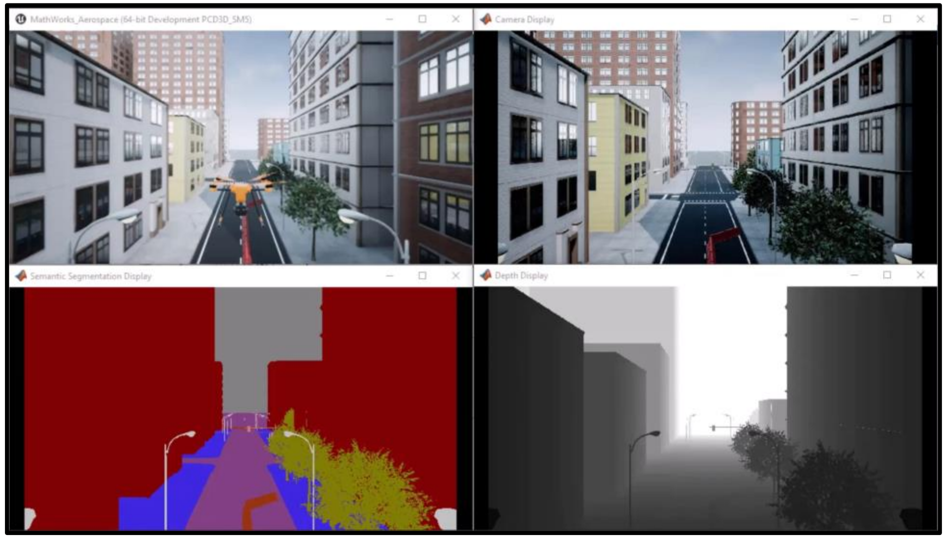
Moubarak Gado, MathWorks

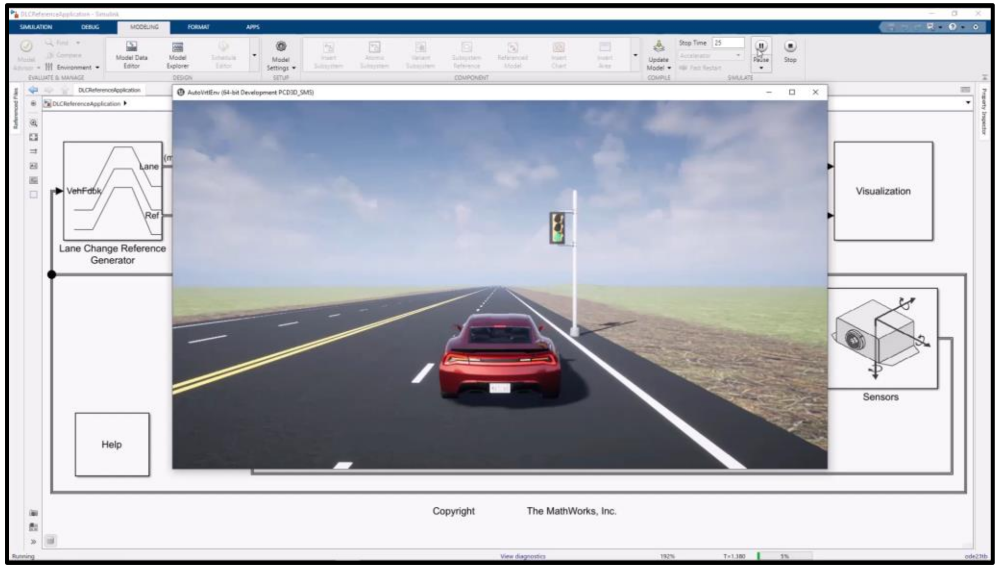
MathWorks®


SiCiPtReferenceApplication * - Simulink
SIMULATION DEBUG MODELING FORMAT APPS
Copyright 2015-2023 The MathWorks, Inc.
Environment
RefSpd
(m/s)
(m/s)
FTP75 (2474 seconds)
0
Longitudinal Driver
Controllers
Passenger Car
Visualization
Analyze Power and Energy
MATLAB R2023b
HOME PLOTS APPS
Command Window
Current Folder
Name
Folder
+helper
+neuralode
Experiment_ROM
Function
NeuralStateSpaceModel_OutputFcn.m
NeuralStateSpaceModel_OutputFcnJacobian.m
NeuralStateSpaceModel_StateFcn.m
NeuralStateSpaceModel_StateFcnJacobian.m
MAT-file
NeuralStateSpaceModel_OutputFcnData.mat
NeuralStateSpaceModel_StateFcnData.mat
Live Script
ROM_1_DeepLearning_LSTM.mlx
Details
Performance and FE Scope
File Tools View Simulation Help
<EngTrq>
Trace Velocity, Target, Actual (mph)
Running
Ready

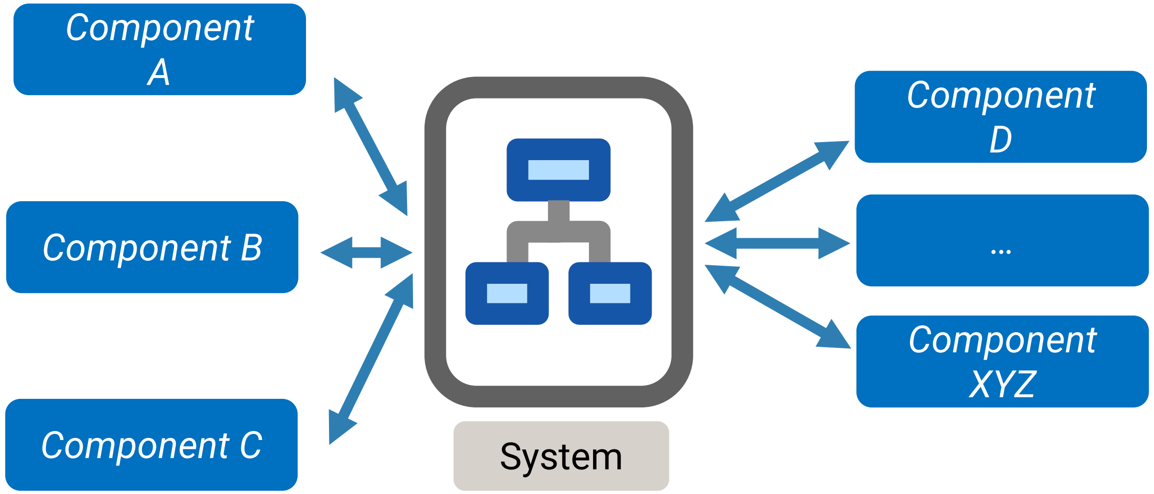
System-level Simulation



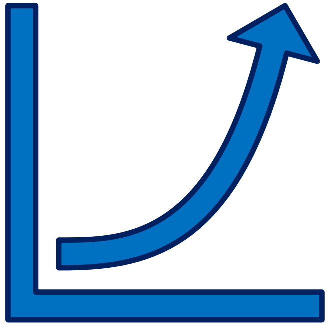




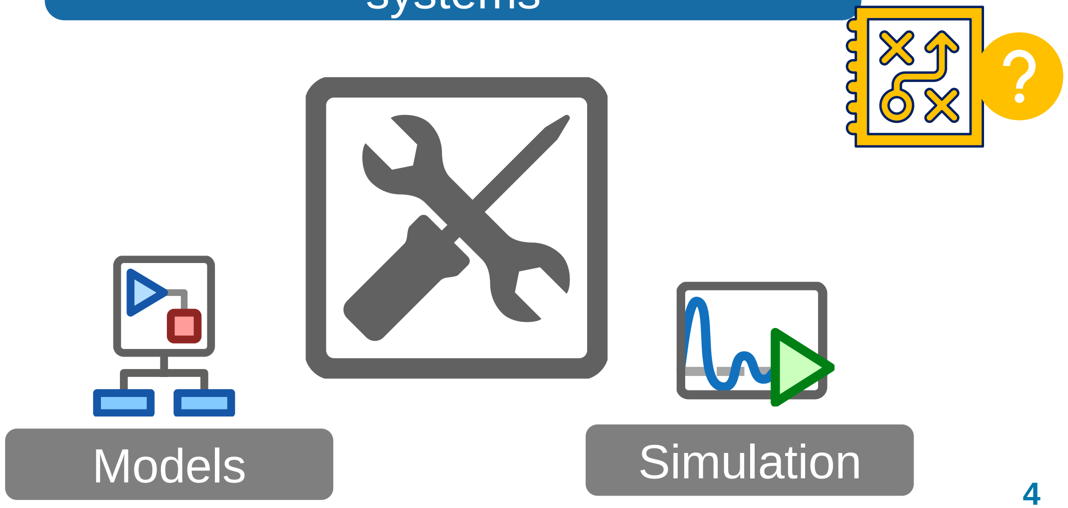
Systems complexity is increasing



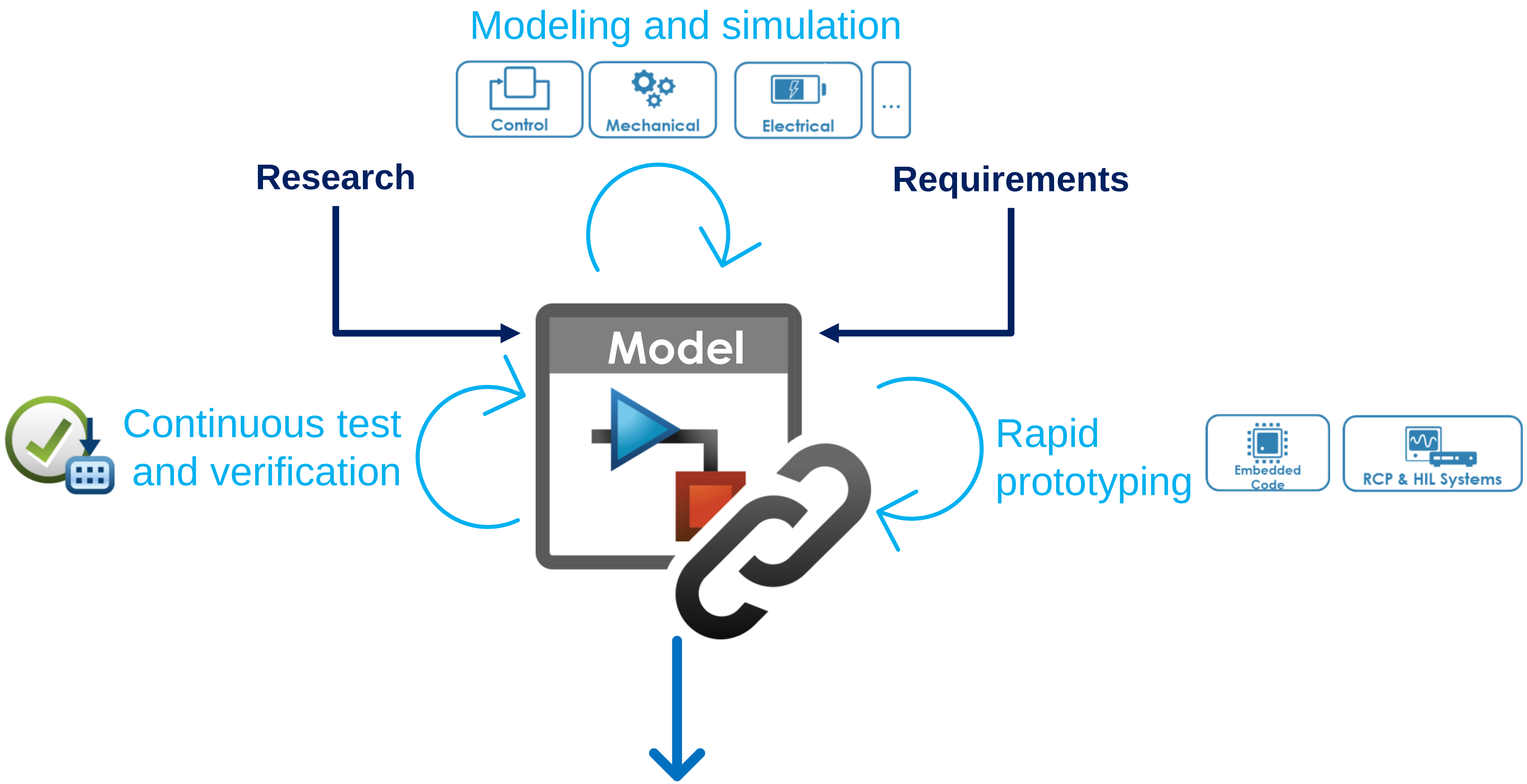
Increasing system complexity



Common tools used for designing systems



Building complex systems with Model-Based Design (MBD)



Generation of outputs
Production of code
Reports
Certification artifacts

AI as a new tool to address modeling and simulation challenges

Improve algorithm accuracy:
train AI model using high quality data

Managing complexity:
replace algorithms that would be too difficult to design otherwise

Save time:
replace models that would be too long to simulate

User stories

Automotive

Mercedes-Benz:
Deep Neural Networks virtual sensors on ECU

Aerospace

Lockheed Martin:
Deep Learning based fleet performance optimization

Robotics & Smart manufacturing

ASTRI:
AI driven digital twin for robotic welding system

Automotive

Vitesco:
Reinforcement Learning based controller for powertrain control

Food and beverage

Coca-Cola:
Virtual Sensor with Machine Learning to improve beverage diagnostics

Medical

Dutch Epilepsy Clinics Foundation:
Diagnosis of epileptic seizures using Machine Learning

Automotive

Renault:
Estimating Nox emission with Deep Learning

Autonomous Vehicle

Monarch tractor:
AI for camera and sensor data analysis in smart electric tractor

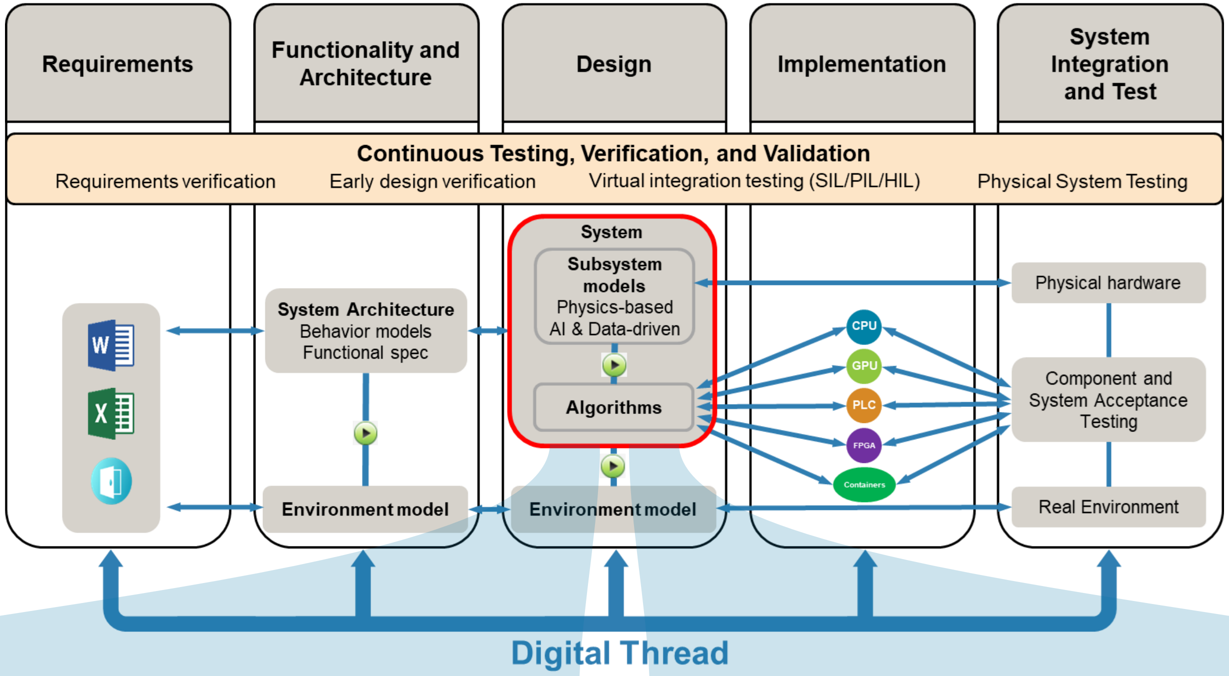
Energy

Plug Power:
AI based predictive model for fuel cell

AI, simulation and MBD: MATLAB and Simulink for system design workflow

- Easy to use interfaces and apps
- Domain specific examples
- Use AI in your area of expertise without being AI specialist
- Common and collaborative workflow
- Integrate AI models developed in 3rd party frameworks (TensorFlow, PyTorch, …)

Where can you integrate AI into Model-Based Design?



2 categories in general

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

Observed (major) trends for AI in simulation

Data synthesis

Use AI for realistic data generation

Component modeling

Use AI for physical environment modeling, Reduced order models

Algorithm modeling

Virtual Sensors, Predictive maintenance, ADAS, Signal Processing, Natural Language Processing, Electrification

Control systems, planning, decision making

Advanced control algorithms, end to end modeling

Observed (major) trends for AI in simulation

Data synthesis

Measurements are difficult to obtain. I want to do a what if analysis.

Use AI (generative AI models like GANs, diffusion models, …) to generate realistic data. Use AI based digital twin to generate data for what if analysis

Algorithm modeling

I want to design a condition monitoring algorithm and deploy it on hardware. I have enough labeled data

Use AI to train a classification algorithm and use automatic code generation tools to deploy it on embedded hardware.

Control systems, planning, decision making

I want to replace several subsystems with a single black box and design a control algorithm, but traditional methods don't work

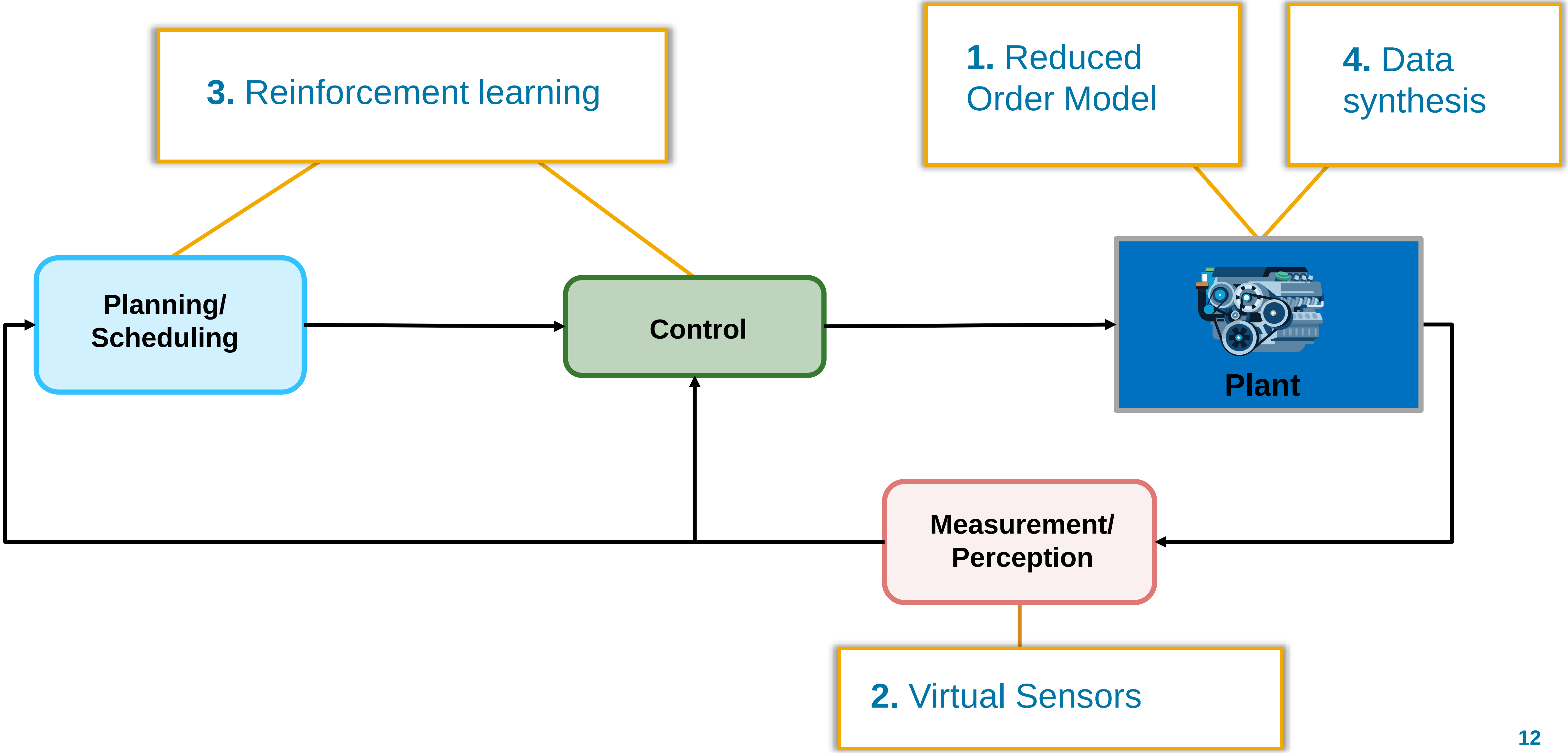
Use Reinforcement Learning (where simulation and AI combined) to build an end-to-end solution that is self-tuned through a training process

Component modeling

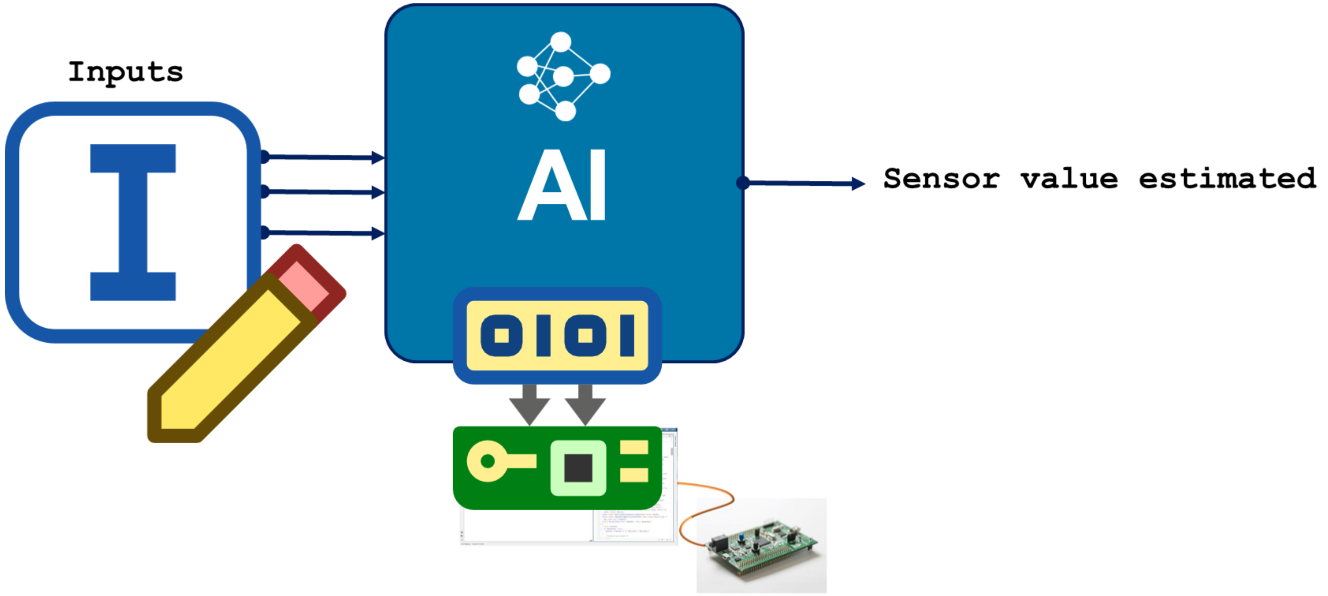
I have high quality data (inputs and responses) for a physical component. Can I represent the component without using high fidelity tools?

Use AI to train a reduced order model. You can use leverage AI-based ROM early in design process and high-fidelity model later

Observed (major) trends for AI in simulation



Application example: Virtual sensors



What
A software component that mimics the behavior of a physical sensor by leveraging information available from other measurements and estimate the quantity of interest.

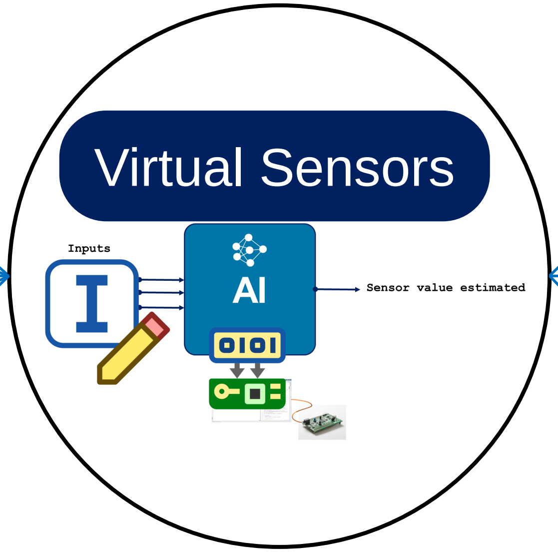
When
Physical sensors are impractical, expensive, slow, noisy, unreliable, not feasible, etc.

How
Kalman Filters, Grey-Box Models
Lookup tables
Time series modeling
AI (**Machine Learning** and **Deep Learning**)

Application example: Virtual sensors

Industries

- Automotive
- Medical devices
- Aerospace
- Industrial Automation and Machinery
- Energy production



Applications

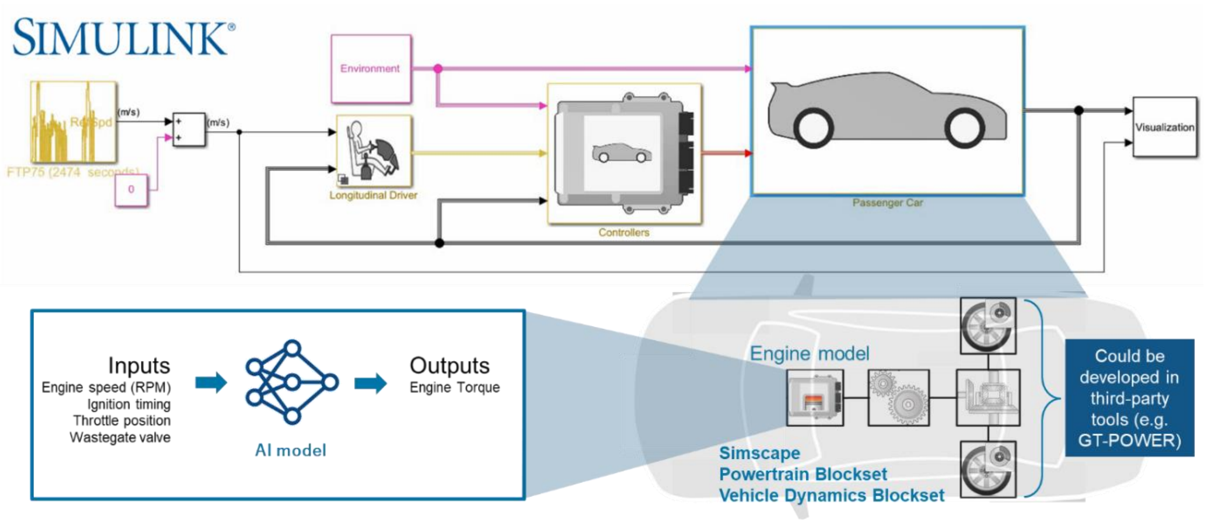
- Control systems
- Electrification
- Fault detection and Predictive maintenance

User Story

Poclain Hydraulics:
Soft Sensors to measure Motor Temperature in real time using Deep Learning

Case study

I need to provide other teams with good trade-off models for faster simulation reuse



AI for component modeling

*Replacing a first-principles engine model with an AI-based **Reduced Order Model***

Application example: Reduced Order Modeling

- Data-driven and adaptive methods: feature extraction, selection
- Reduced computational time and memory, real-time model updating
- Accelerated design process: faster parametric studies and optimization
- More time for exploration and iteration: edge cases, alternative evaluation, faster high-fidelity simulations
- Integration of 2D and 3D models from other tools into system level simulation, enhanced controller design
- Perform hardware-in-the-loop testing without complete system hardware

From first principles models to reduced order models

What is a model?

A simplified abstraction of a system, concept, phenomenon

Physics based model

A useful (not perfect) representation using governing laws of nature that embed concepts of time, space and causality.

Explainable and clear physical meaning,
Can be parameterized

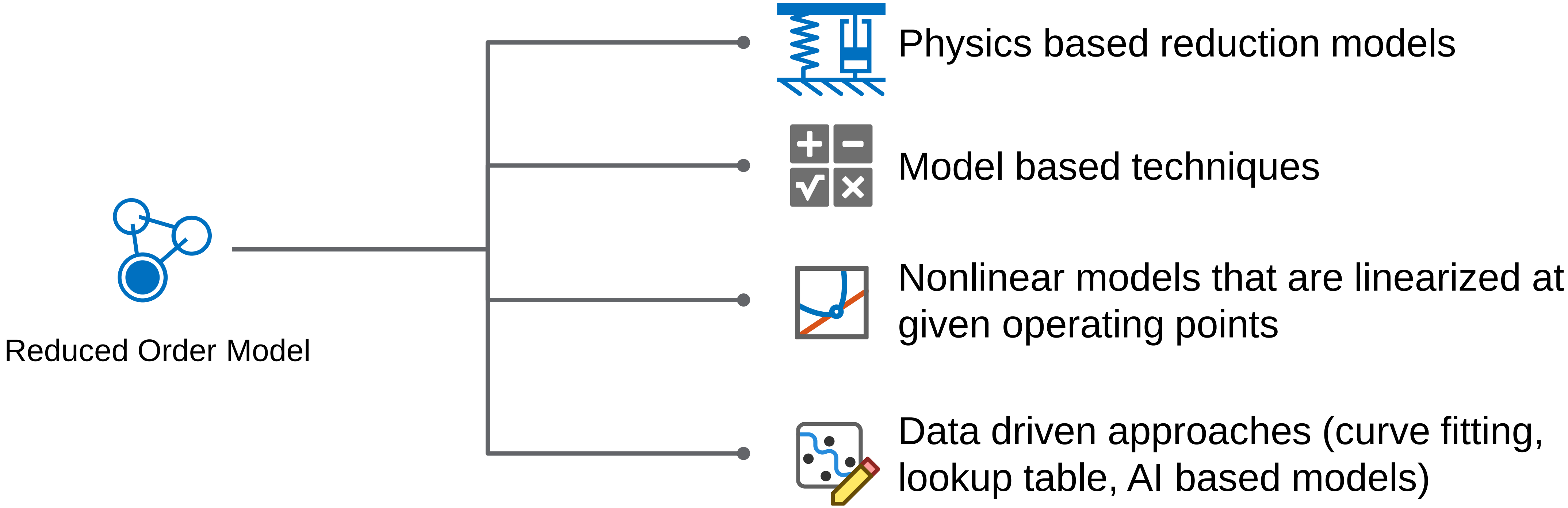
→

Reduced Order Model

Techniques that aim to simplify the original high-fidelity model in a lower-dimensional approximation and extracting most relevant features

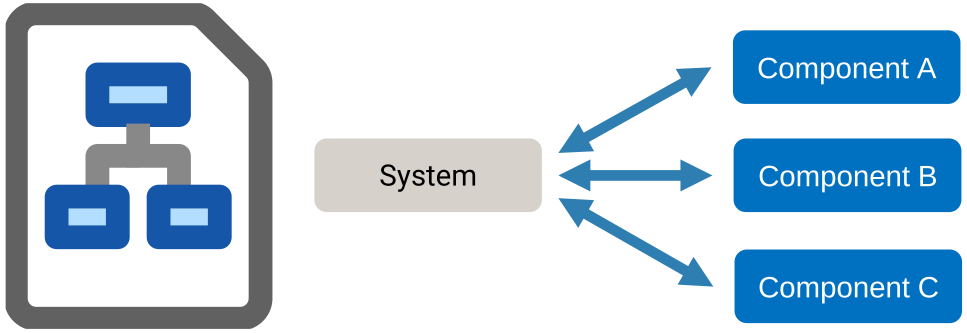
Can run faster

From first principles models to reduced order models



Data-driven vs. first-principles modeling

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

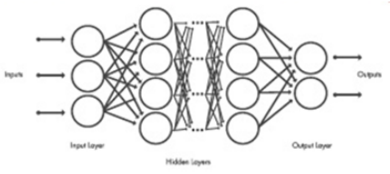


DATA-DRIVEN MODELS
Statistics, optimization, AI

FIRST-PRINCIPLES MODELS
Physics, math, domain knowledge

BLACK BOX	GREY BOX	WHITE BOX

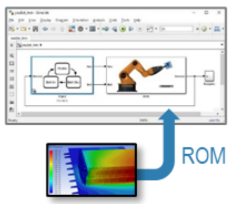
AI-BASED
- Machine learning deep learning
- Reinforcement learning



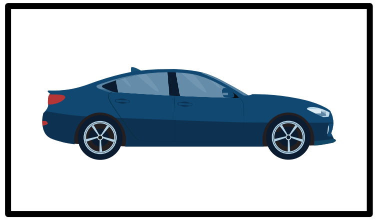
PARAMETER ESTIMATION / HYBRID MODELS
- Kalman estimator
- System identification
- Regression

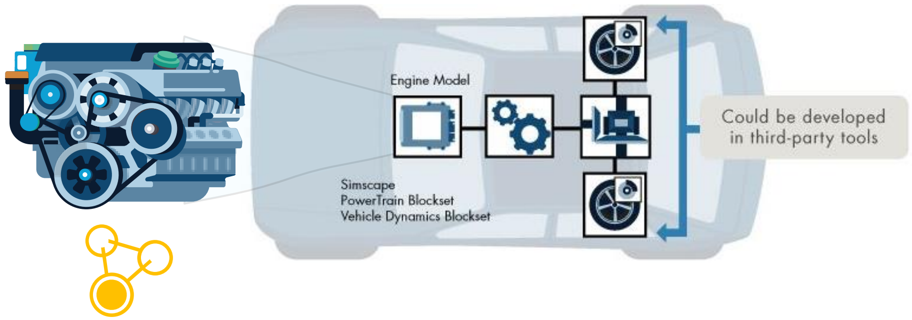
$$\hat{x}_k = \underbrace{\hat{x}_k^-}_{\text{Predict}} + \underbrace{K_k(y_k - C\hat{x}_k^-)}_{\text{Update}}$$

PHYSICS-BASED
- Systems/components (electrical, mechanical, algorithms, etc.)
- Can integrate models from other tools such as FEM

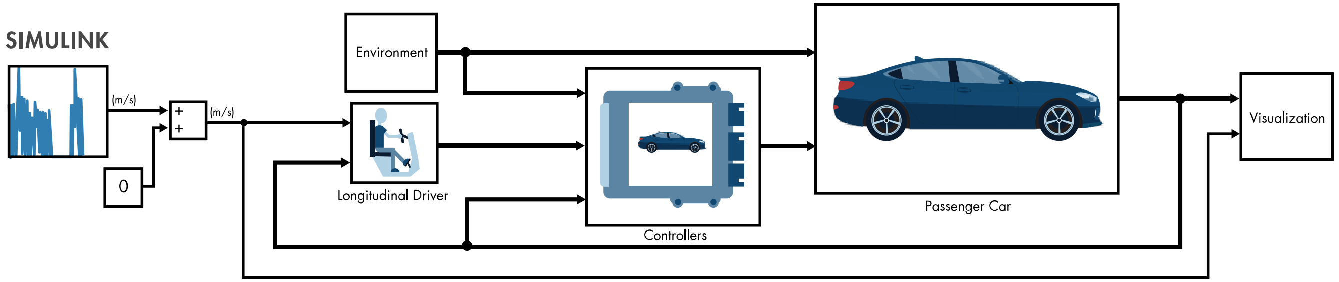


Case study: ROM of engine model

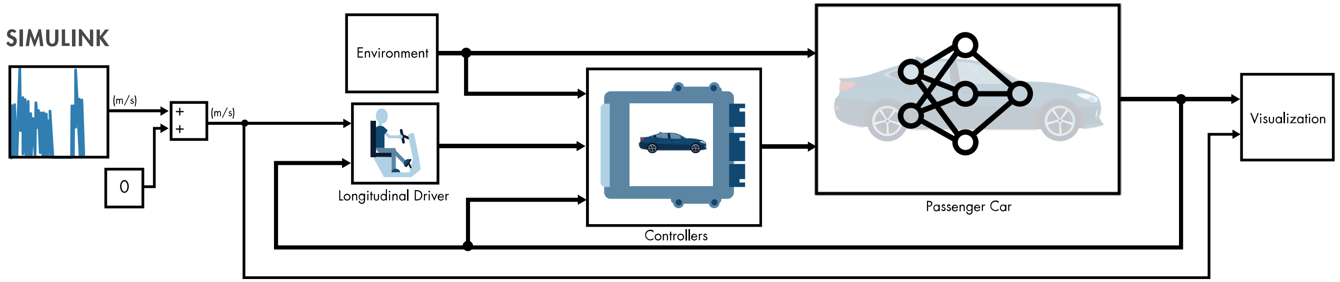




HIGH FIDELITY MODEL



REDUCED ORDER MODEL



Challenges with AI and Simulation for designing complex systems

AI model integration

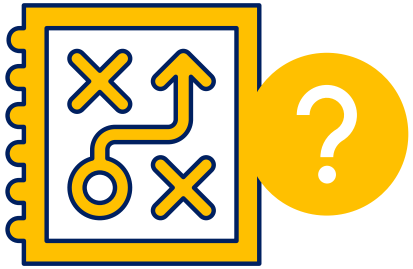
Some teams are using TensorFlow and PyTorch, other are using MATLAB and Simulink. How can the teams work together?

Choosing best AI technique

How to choose the right AI techniques and algorithms?

Data

Data preparation is time consuming



Challenges

Moving from Prototype to production is time-consuming

How can I deploy easily on embedded device easily and get to production faster ?

Errors and uncertainties

Can I quantify uncertainties Quantitying errors and uncertainties?

Managing trade-off

How to balance trade-off between complexity and fidelity of the reduced model ?

Model validation and verification

How to validate and verify the AI model and its predictions

MATLAB/Simulink for AI and complex system design

- Choosing best AI technique
- Managing trade-off
- Data
- AI model integration
- Model validation and verification
- Errors and uncertainties
- Development to production made easy

Over 500+ examples using AI for domain-specific applications
Fast and easy experimentation: train and quickly compare different AI models
Choose the best AI technique not only for design, but also for deployment efficiency on intended system

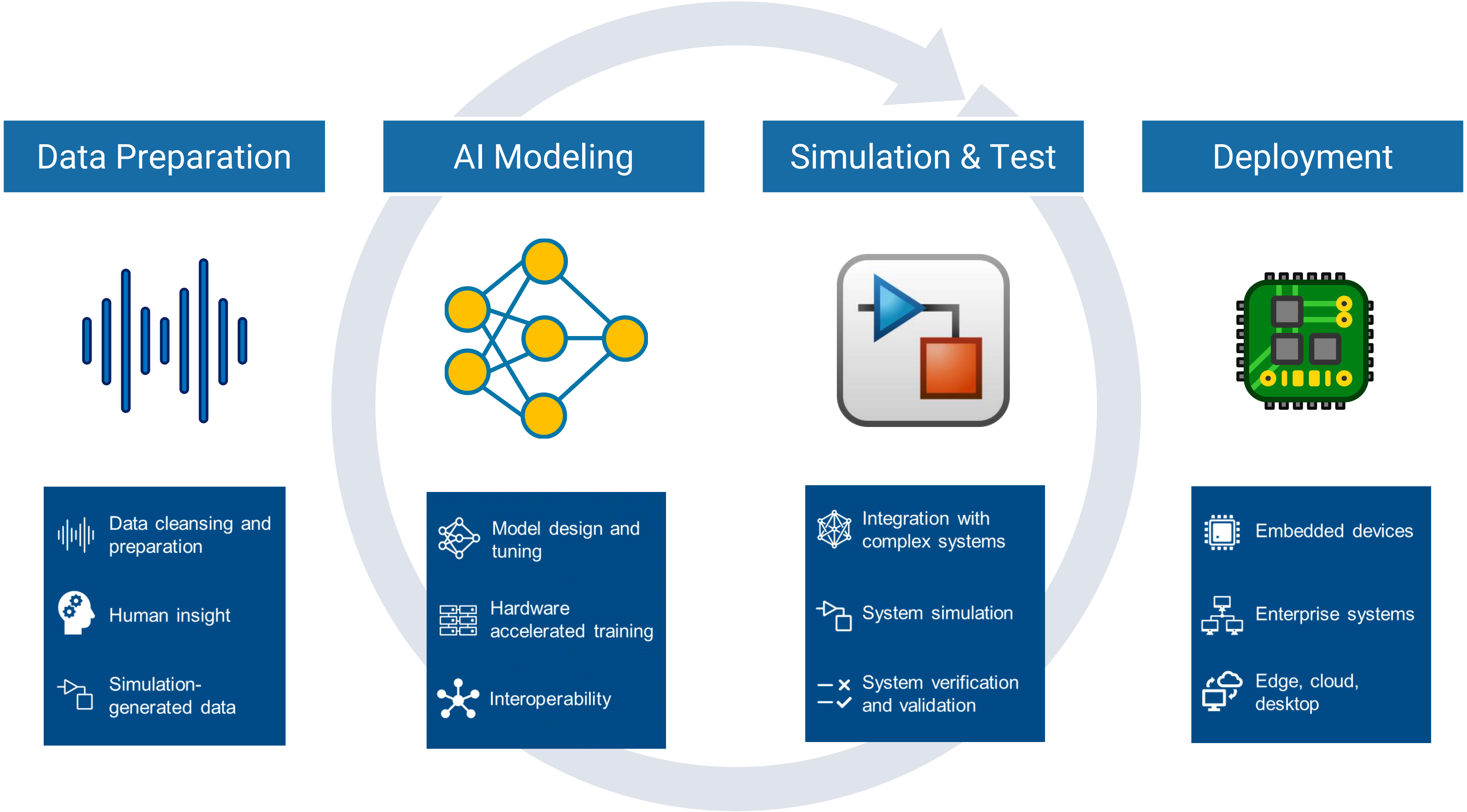
Specific tools to save time in every stage of design process

With Simulink, you can integrate easily your AI model (MATLAB, TensorFlow, PyTorch) into the overall simulation environment

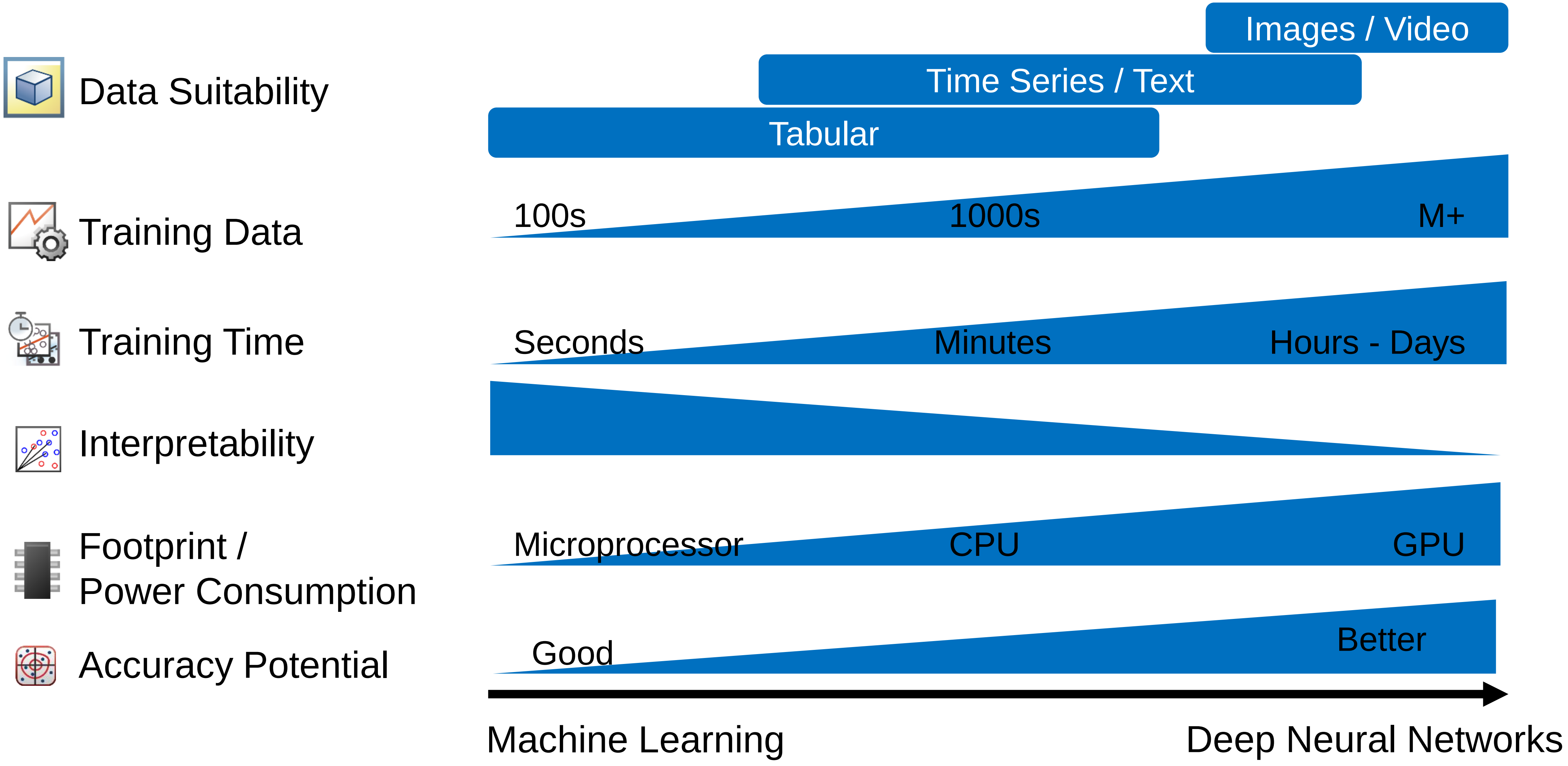
Systematically test your model by simulating different test scenario before deploying to production
MATLAB has a growing list of Verification, validation and explanaible AI functionality

Automatically generate source for embedded AI (CPUs, GPUs or FPGAs)

AI-driven system design workflow



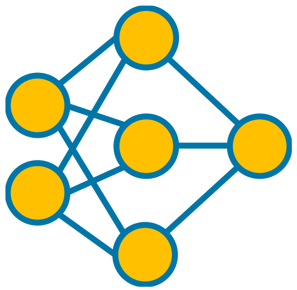
AI workflow – What technique to Consider?



AI-driven system design

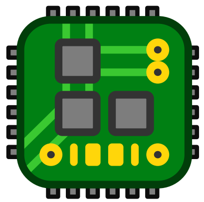
Data Preparation

AI Modeling



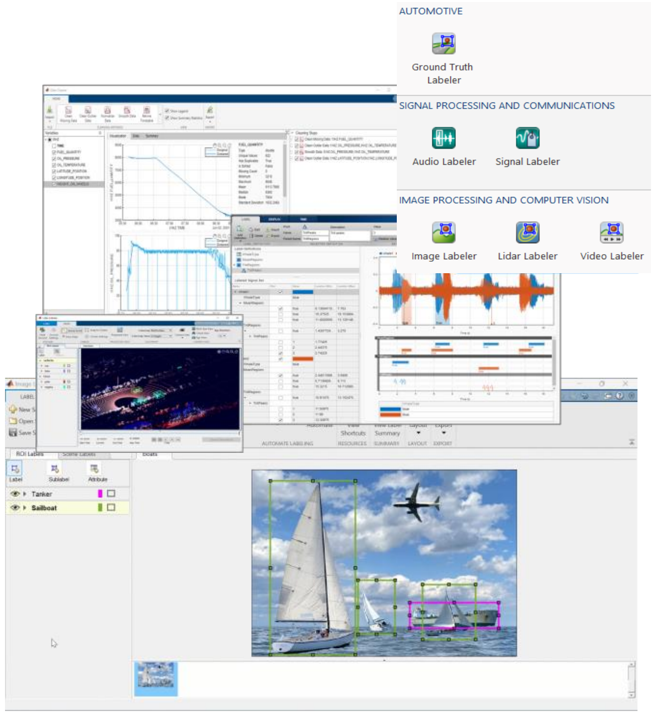
Simulation & Test

Deployment

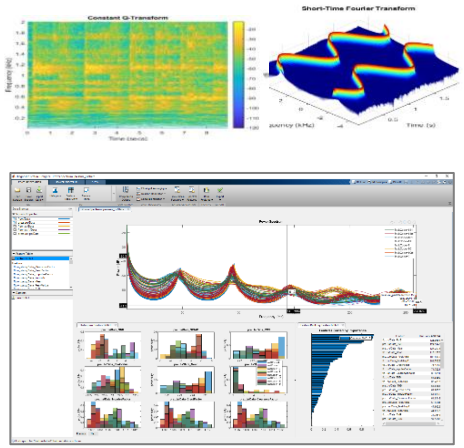


MATLAB is a Data Manipulation Environment

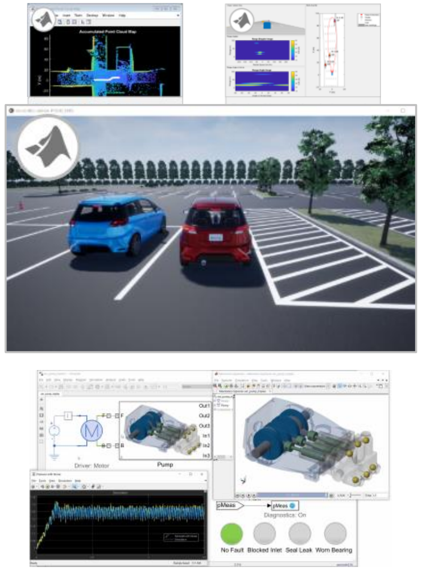
Spend less time preprocessing and labeling your data



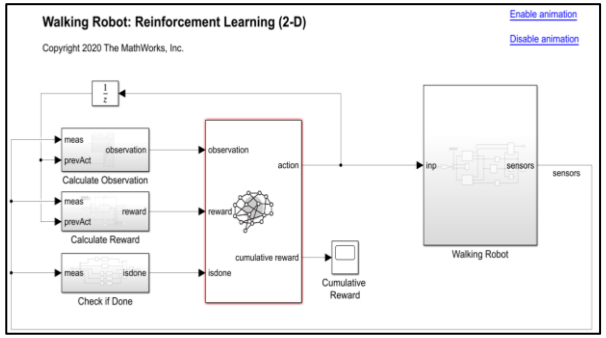
Extract useful features from raw data



Data Simulation & Validation Use Simulink and Simscape to generate realistic data or build Digital Twin



Use MATLAB and Simulink to create environment models for training agents (Reinforcement Learning)



Feature extraction

Time series

Signal processing techniques
Wavelet
Time, frequency, time/frequency transformation

Images

Deep Learning is now the state of the art
Specialized feature extraction techniques
(HOG, SURF, LBP, …)

Domain specific feature extraction techniques

Predictive Maintenance Toolbox
`DiagnosticFeatureDesigner` App

Audio Toolbox
`audioFeatureExtractor`

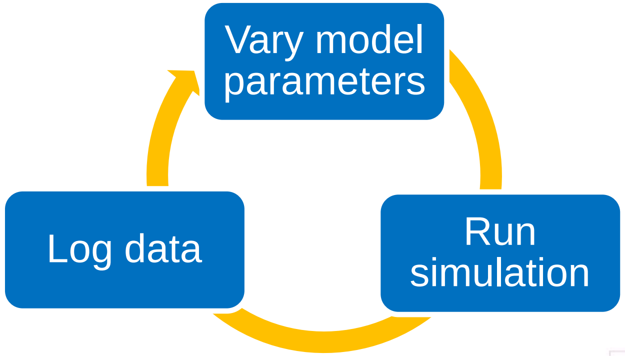
Text Analytics Toolbox

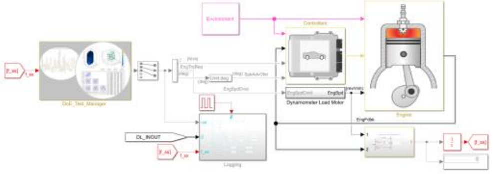
Example: Reduced order modeling

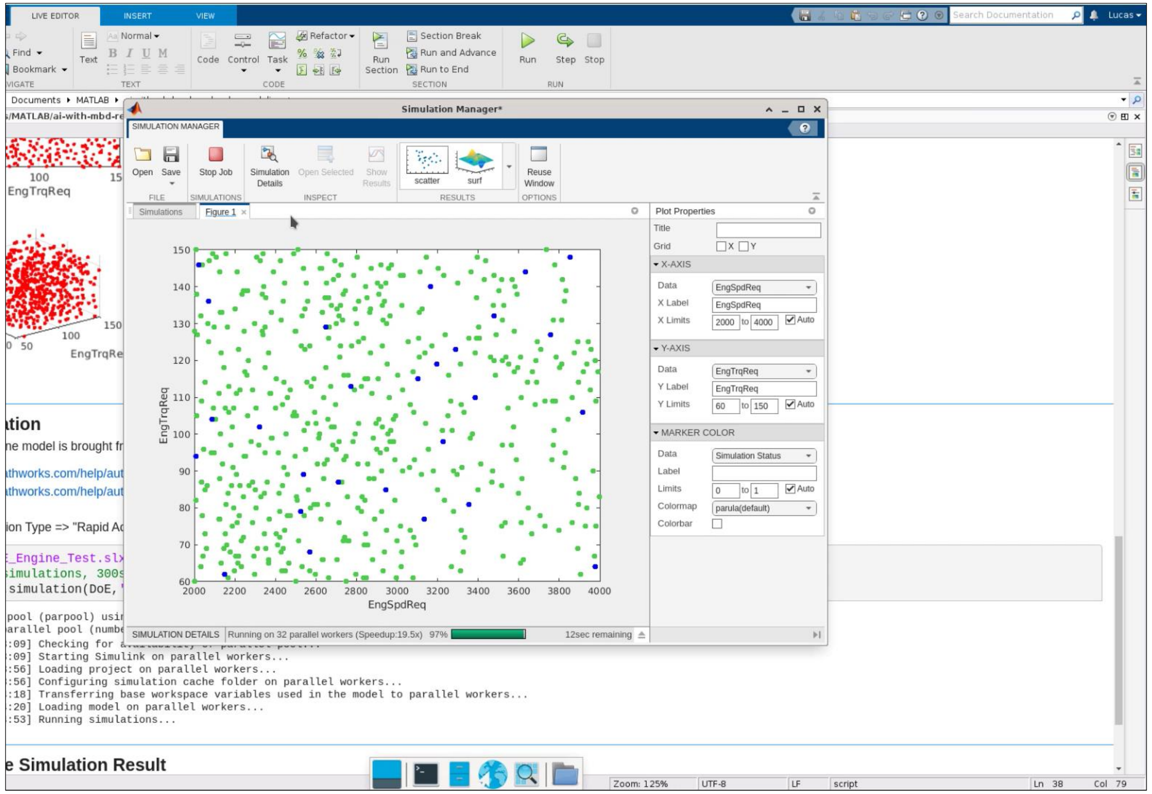
Design of Experiments & synthetic Data Generation

DoE = 512×3 table

	EngTrqReq	EngSpdR...	SpkAdvOfst
1	60	2000	-30
2	128	2500	15
3	94	2750	8
4	111	2875	-19
5	77	2625	-11
6	144	2125	4
7	85	2563	-21
8	119	3313	-28
9	68	2938	21



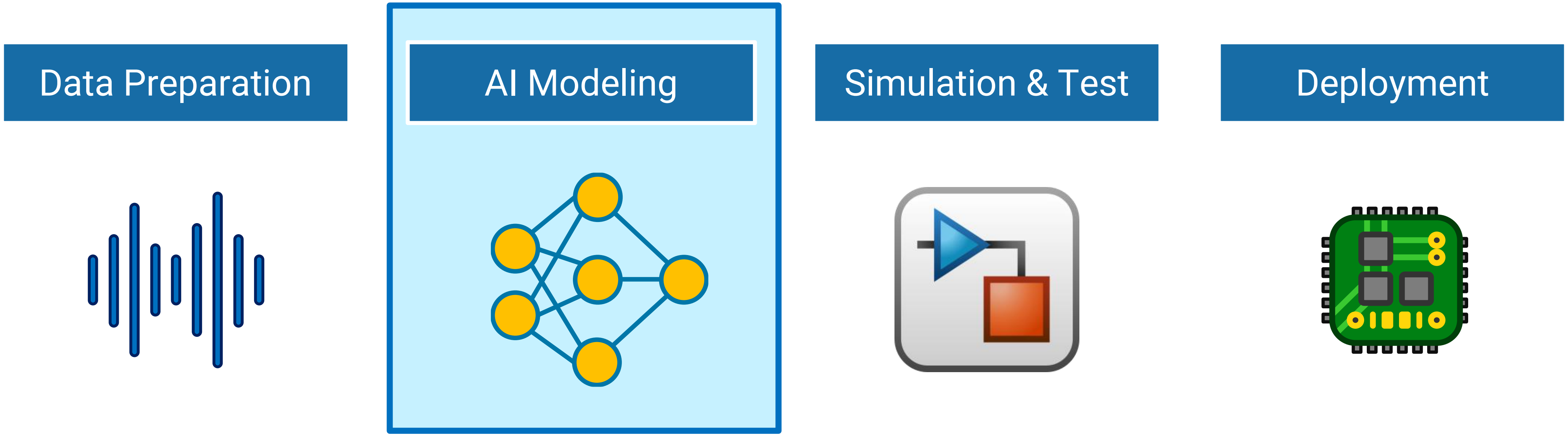




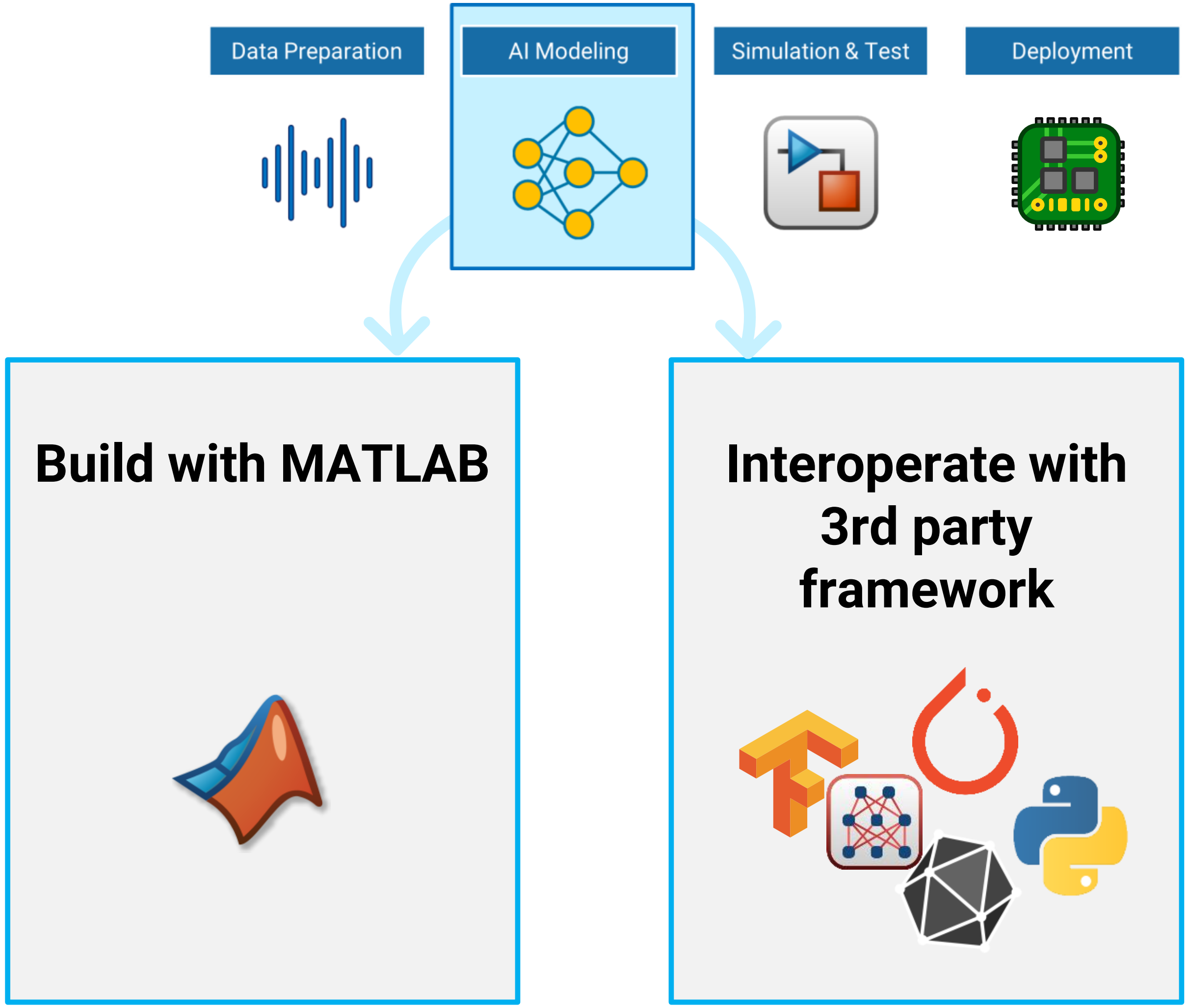
Input features
- Engine speed (RPM)
- Ignition timing
- Throttle position
- Wastegate valve

Response
- Engine Torque

AI-driven system design



AI modeling
Multiple approaches



Start with a complete set of algorithms, pre-built models and domain specific examples

Algorithms

Machine Learning
Deep Learning
Reinforcement Learning
Predictive Maintenance
Bayesian Optimization
… and more

Flexible modeling approach

Low code or ***programmatic***

choose approach that best suits your needs

Pre-builts models

Audio

Lidar

Natural Language Processing

Computer Vision

Access model from MATLAB Deep Learning Model Hub on Github

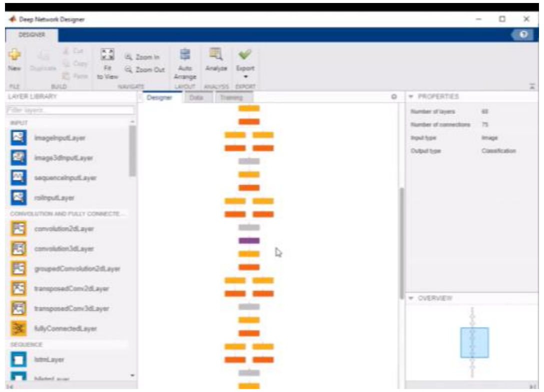
Application specific reference examples

Increase productivity using Apps for design and analysis

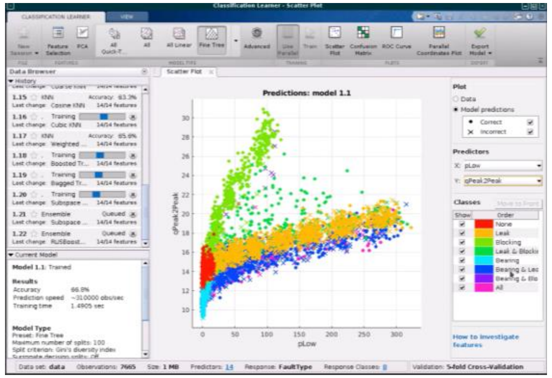
Design your AI model

↓

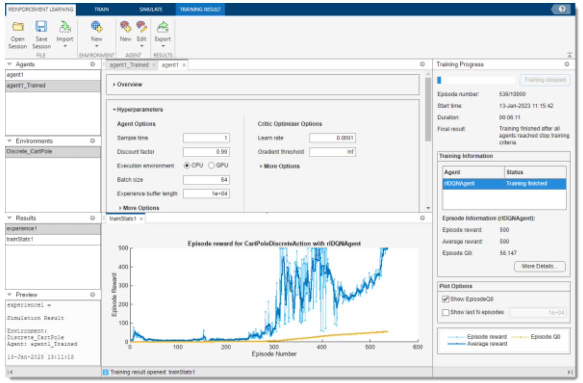
Run multiple **experiments**, compare results and optimize your AI model



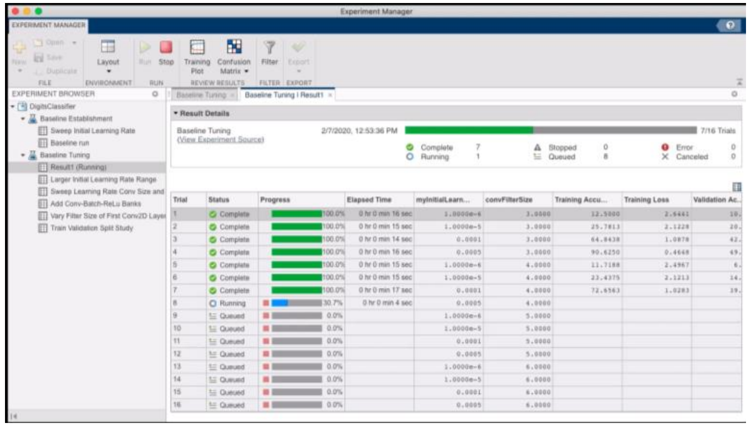
Deep Network Designer app to build, visualize, and edit deep learning networks



Machine Learning Apps to train machine Learning Models



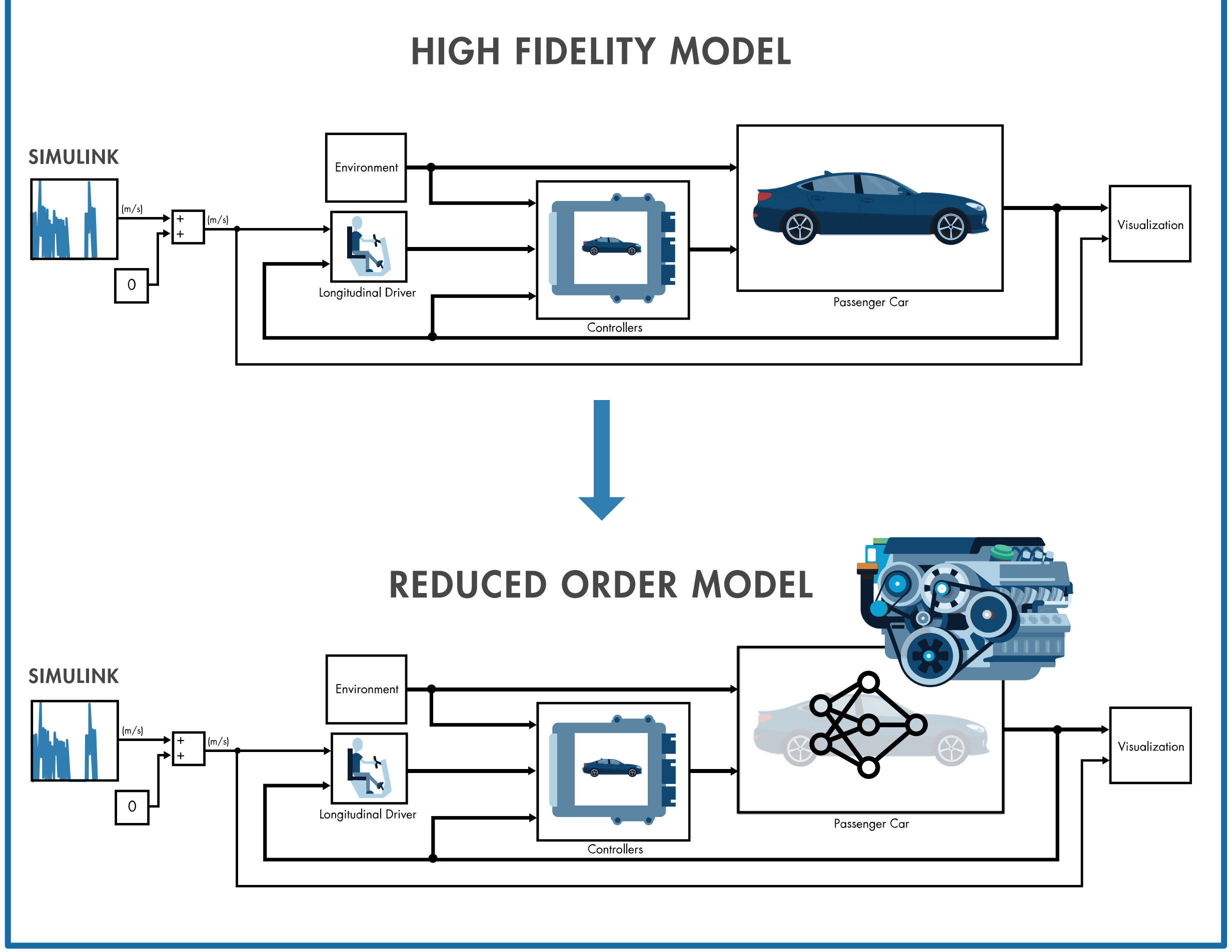
Reinforcement Learning Designer app to design, train, and simulate agents for existing environments

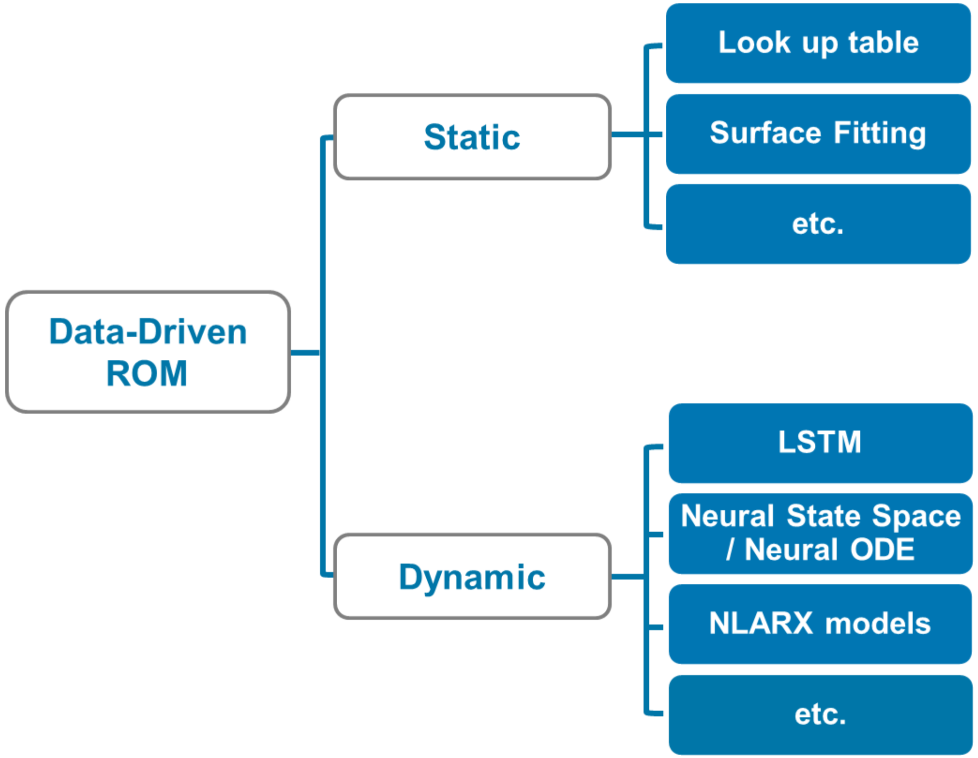


Experiment Manager app to manage multiple deep learning experiments, analyze and compare results and code

Example: engine model AI based ROM using LSTM

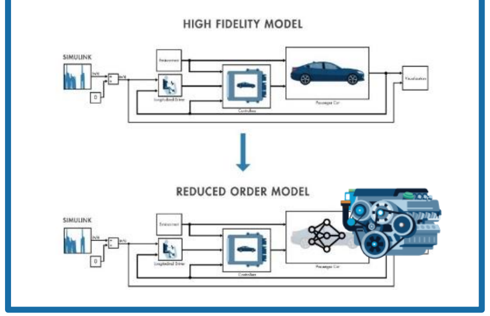
Data-driven Reduced Order Model



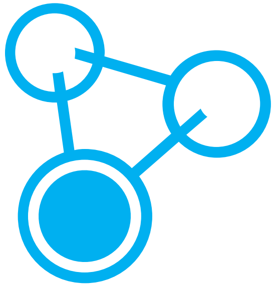


Example: engine model AI based ROM using LSTM

Data-driven Reduced Order Model

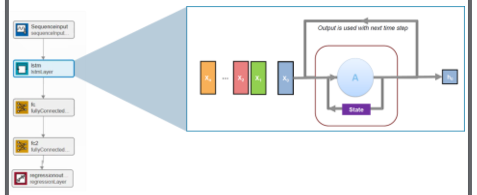


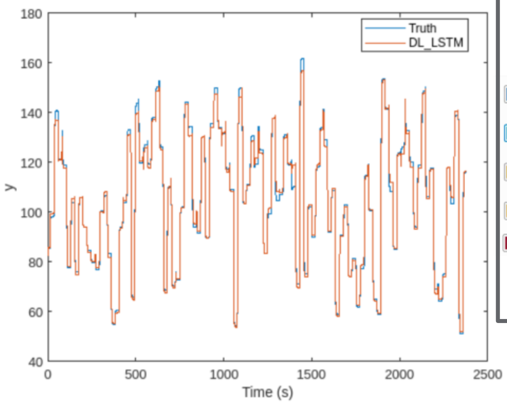
Input features
Engine speed (RPM)
Ignition timing
Throttle position
Wastegate valve



Response
Engine Torque

LSTM





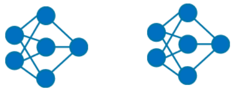
accuracyResults = *1×2 table*

		RMSE	R^2
1	Test	1.8252	0.9953

Neural State Space

nonlinear state function (f) and output function (g)

$$\begin{cases} \dot{x} = f(x,u) \\ y = g(x,u) \end{cases}$$

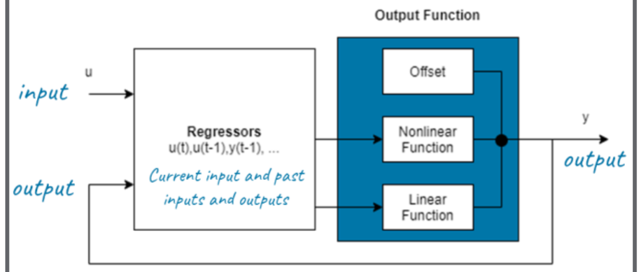


State network (*f*) Output network (*g*)

accuracyResults = *1×2 table*

		RMSE	R^2
1	Test	6.2167	0.9443

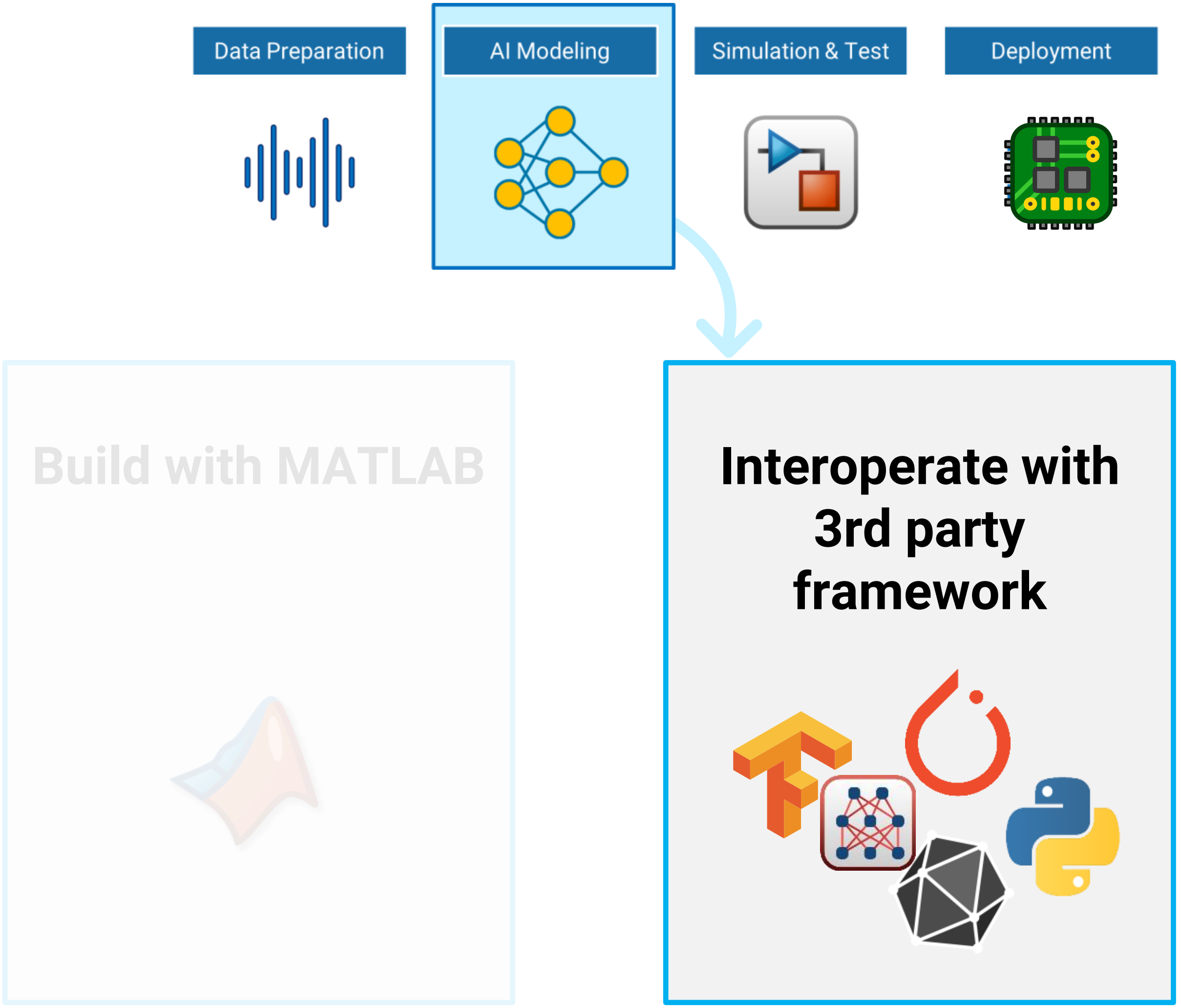
Nonlinear ARX (NLARX)



accuracyResults = *1×2 table*

		RMSE	R^2
1	Test	6.6068	0.9356

AI modelling
Multiple approaches



MATLAB interoperates with other frameworks

Interoperate with 3rd party framework

Convert a Deep Learning TensorFlow / PyTorch / ONNX model

Coexecute a TensorFlow or PyTorch or any Python model from MATLAB

Example: Import trained network from TensorFlow

- models
  - lstm_vehicle_engine_rom_tf
    - variables
    - keras_metadata.pb
    - saved_model.pb

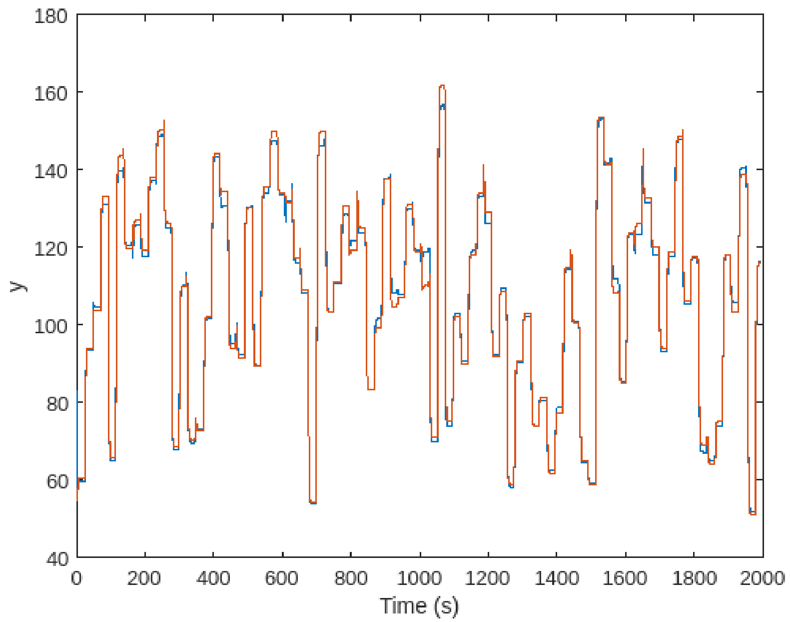
Import TensorFlow Network into MATLAB

```
TFModel = "lstm_vehicle_engine_rom_tf";
net = importTensorFlowNetwork(fullfile(projectPath,"models",TFModel),...
    "TargetNetwork", "dlnetwork")
```

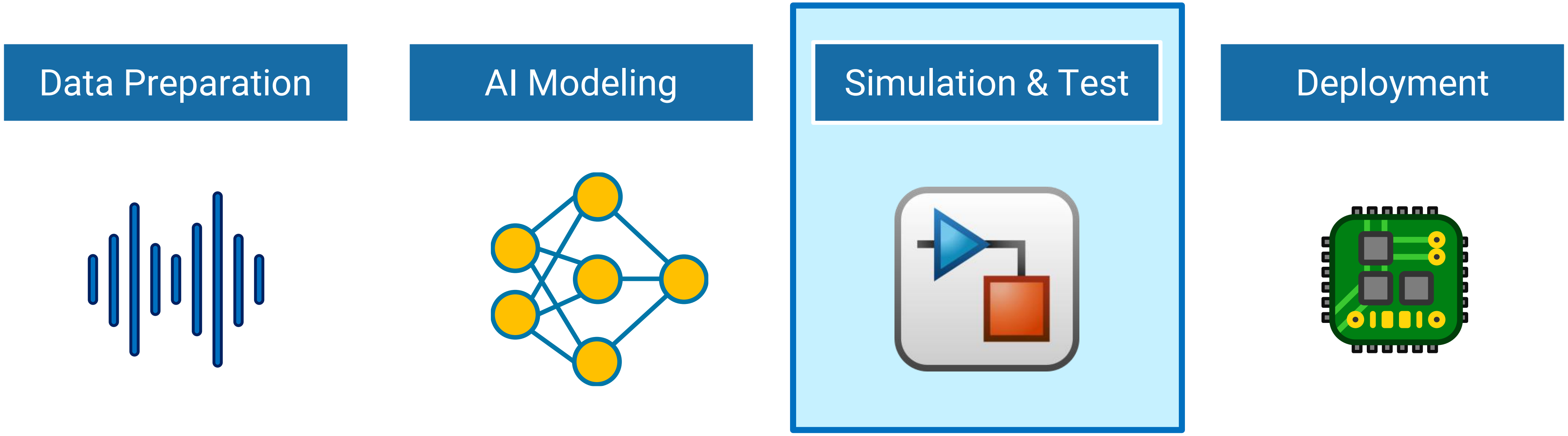
```
Importing the saved model...
Translating the model, this may take a few minutes...
Finished translation. Assembling network...
Import finished.
net =
  dlnetwork with properties:

         Layers: [6×1 nnet.cnn.layer.Layer]
    Connections: [5×2 table]
     Learnables: [7×3 table]
          State: [2×3 table]
     InputNames: {'input_2'}
    OutputNames: {'dense_2_'}
    Initialized: 1

  View summary with summary.
```

```
YPred = predict(net, X);
```

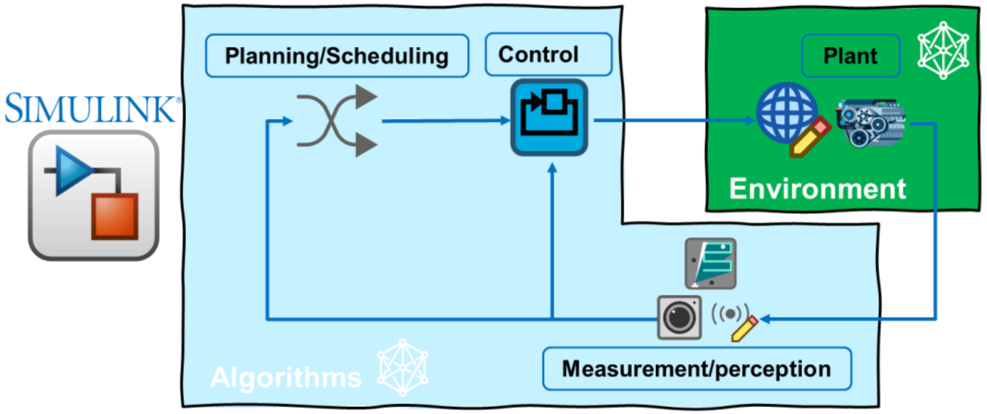
```
Ts = 0.1;
t = Ts*(0:size(X,2)-1)';
plot(t,YPred); hold on, plot(t,Y); hold off
xlabel("Time (s)")
ylabel("y")
```

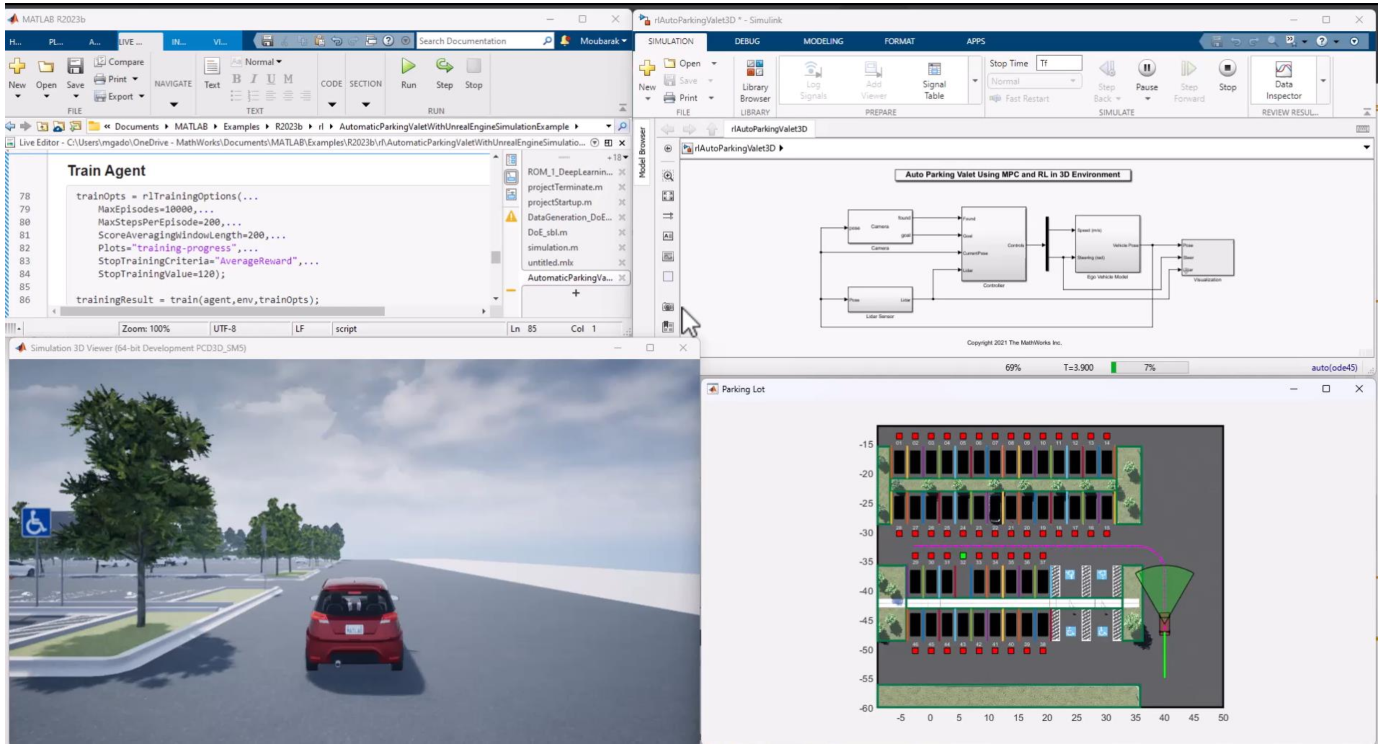


AI-driven system design



AI is part of a larger system

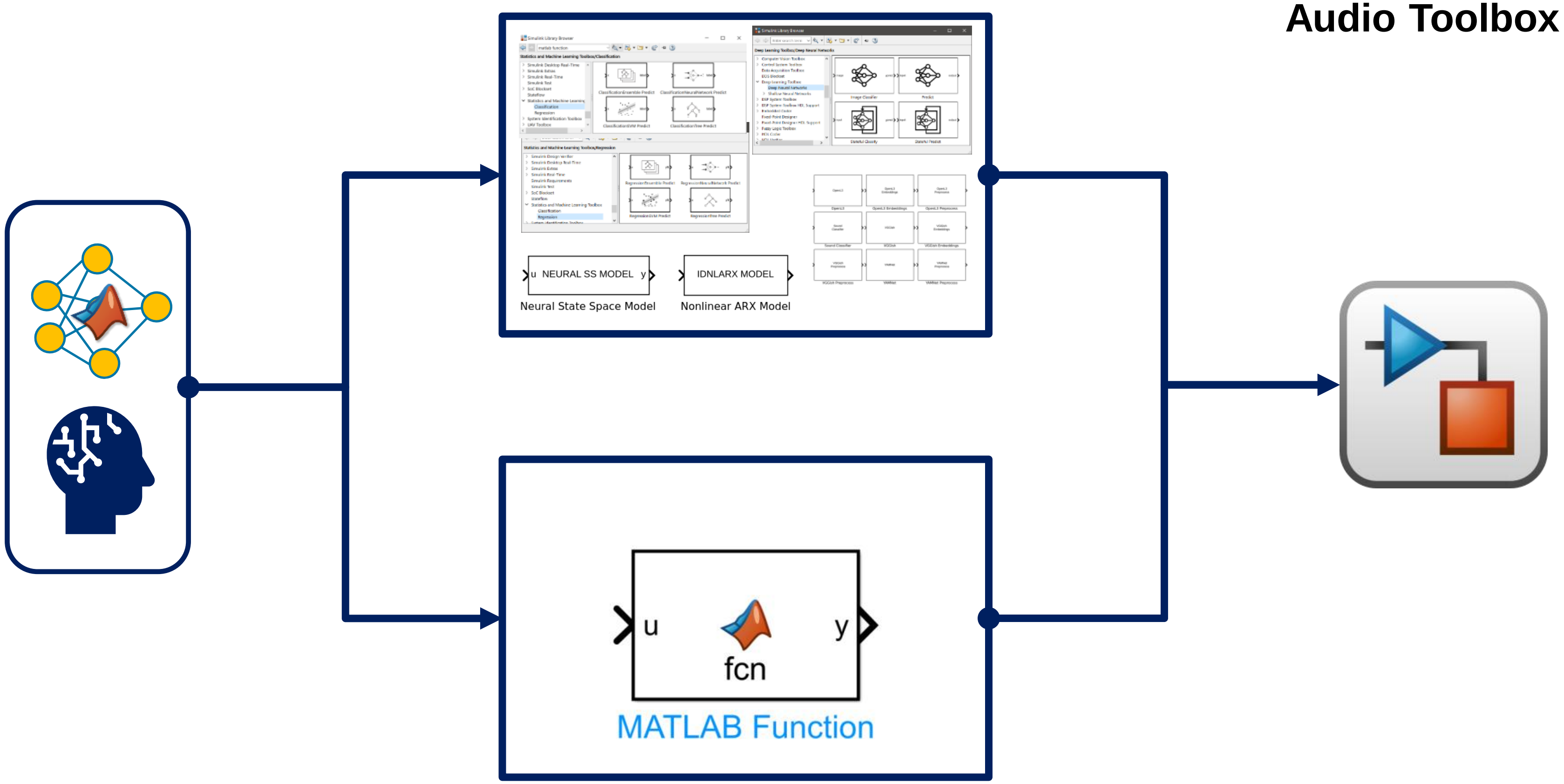




Integrate your AI model into Simulink

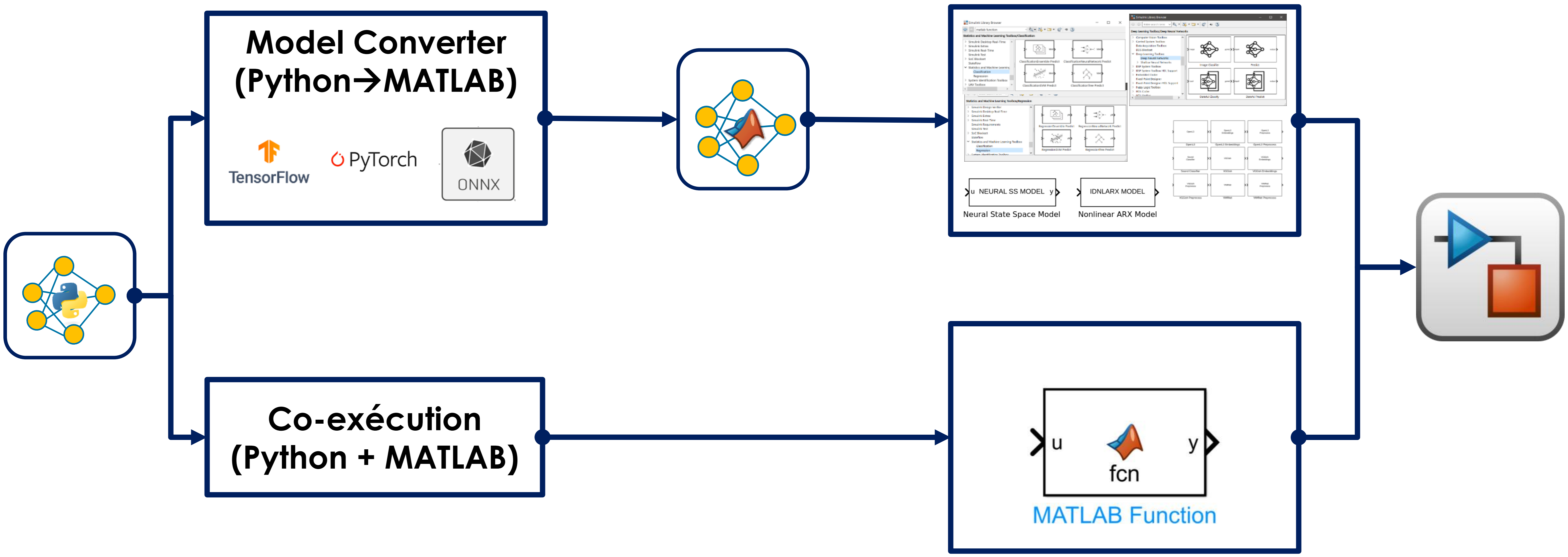
Deep Learning Toolbox
Statistics and Machine Learning Toolbox
System identification Toolbox
Computer Vision Toolbox
Audio Toolbox

Use **AI libraries blocks** (recommended workflow)



Use **MATLAB Function Blocks**
(when no equivalent built-in block)

What if I have Python AI models ?

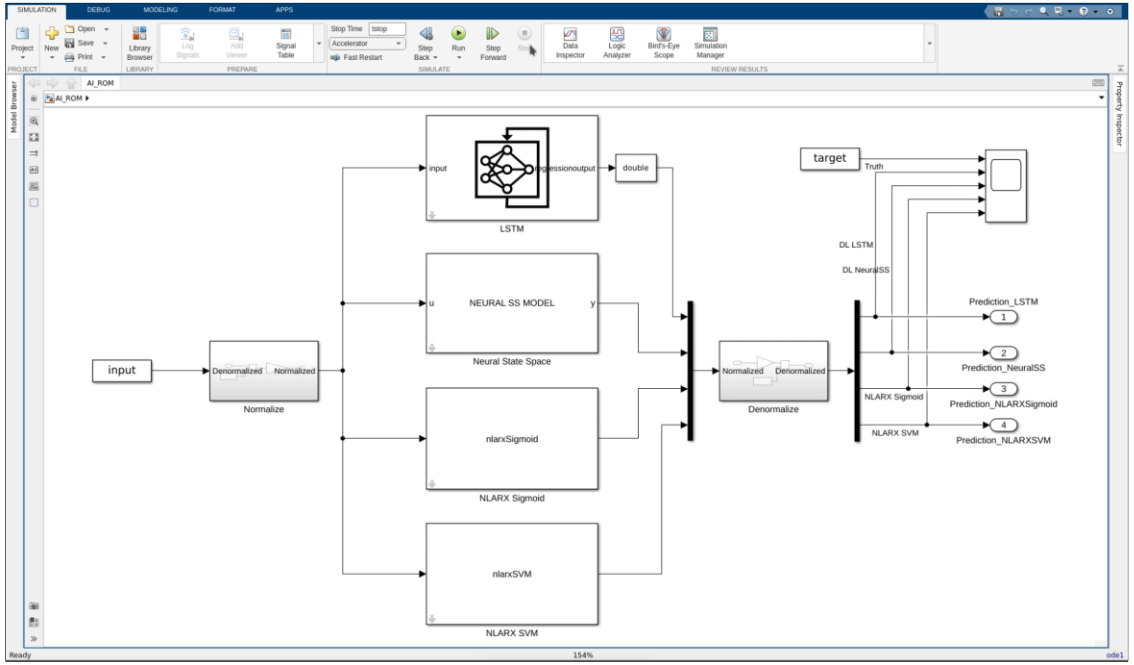


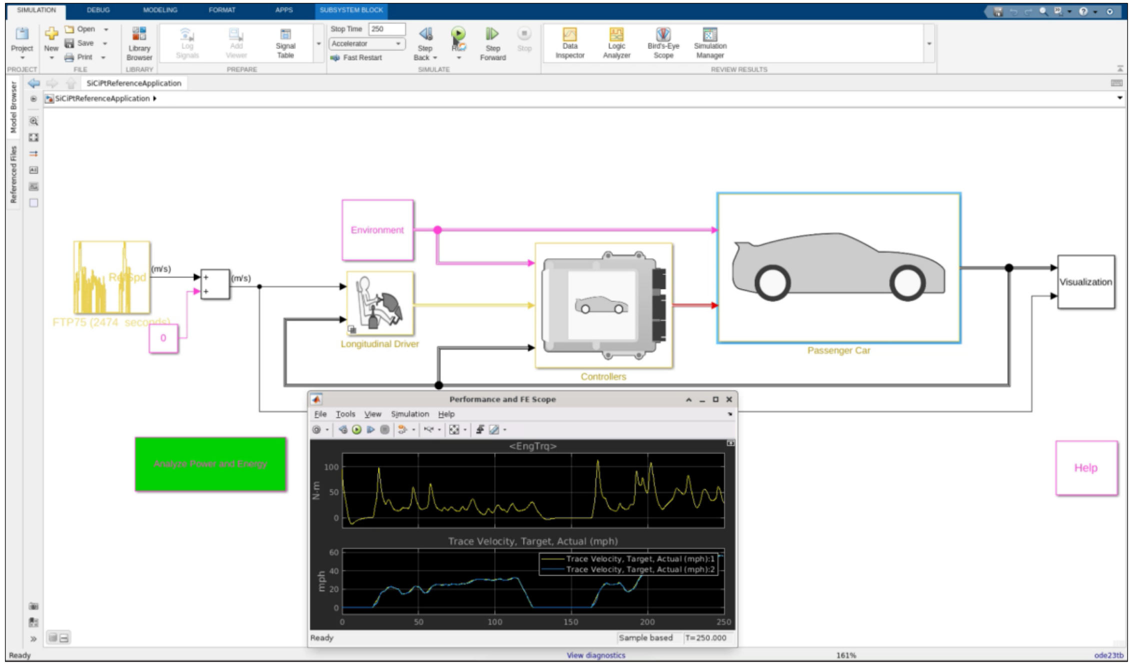
Whether you use MATLAB or not, Simulink is an enabler of your AI model

- Use result of simulation to inform model selection and use variants to compare design options
- Test scenarios that would be difficult, expensive, or dangerous to run on hardware or in a physical environment
- Experiment with multiple AI models of an algorithm and rapidly compare tradeoffs in accuracy, model size and on-device performance.
- Uncover system integration issues earlier

Example: AI-based engine reduced-order-model

Integration of trained AI model into Simulink → **System-level simulation**



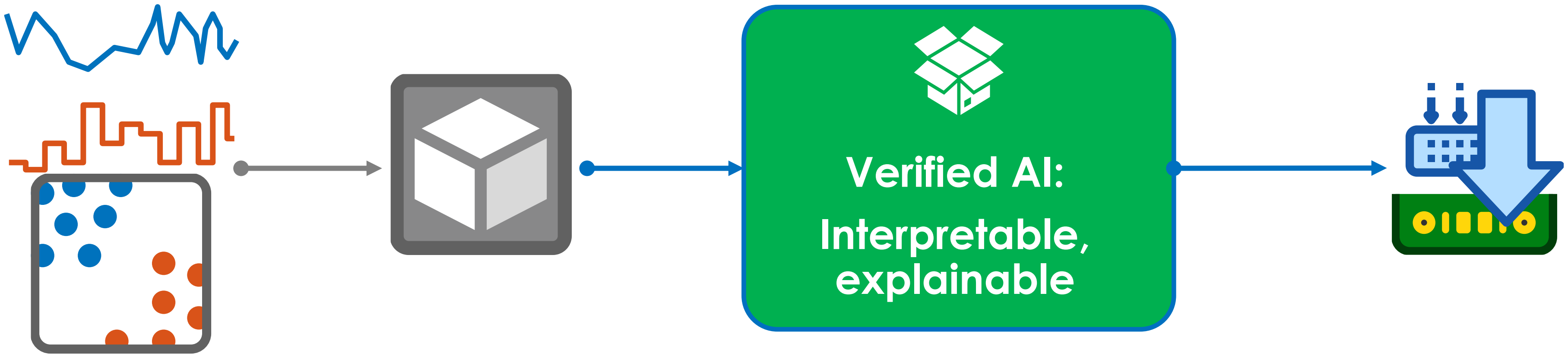


Integrate AI models into Simulink for system-level simulation and test

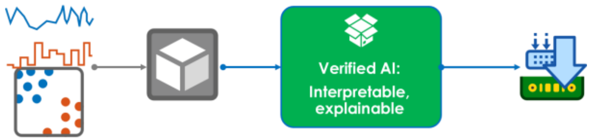
Integration of trained AI models into Simulink

Path	Time Plot (Dark Band = Self Time)	Total Time (s)	Self Time (s)	Number of Calls
AI_ROM		49.440	45.732	142760
LSTM		2.643	0.000	0
NLARX Sigmoid		0.284	0.000	0
Neural State Space		0.195	0.000	0
Scope		0.188	0.188	23795
From Workspace2		0.161	0.161	23794
Demux		0.128	0.128	95184
From Workspace1		0.054	0.054	23794
Prediction_LSTM		0.040	0.040	23794
Prediction_NeuralSS		0.006	0.006	23794
Prediction_NLARXSigmoid		0.005	0.005	23794
Prediction_NLARXSVM		0.004	0.004	23794
NLARX SVM		0.001	0.000	0
Normalize		0.000	0.000	0
Cast To Double		0.000	0.000	3
Denormalize		0.000	0.000	0

Understanding and Verifying your AI models



Understanding and Verifying your AI models

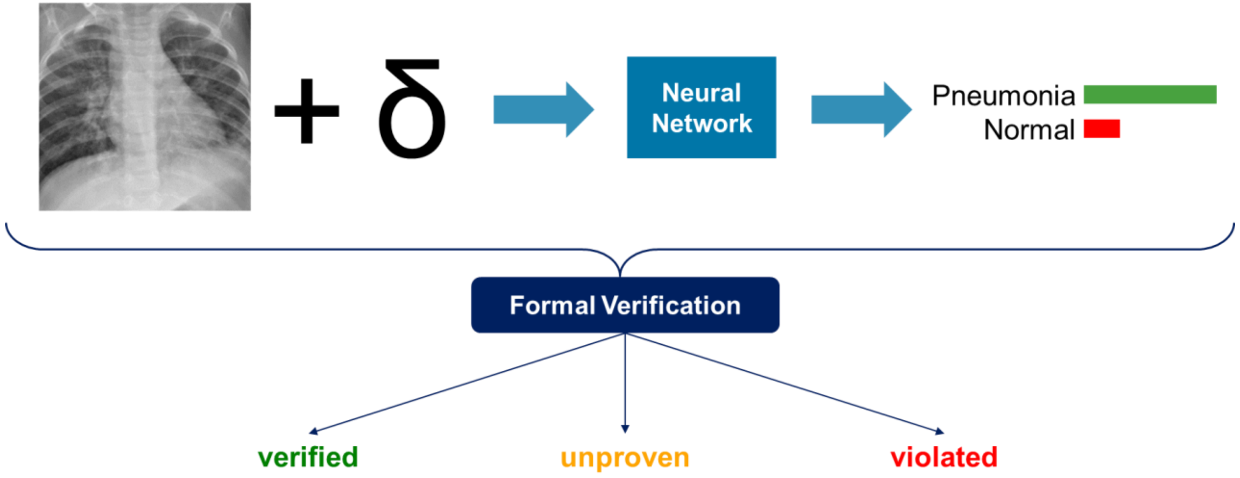


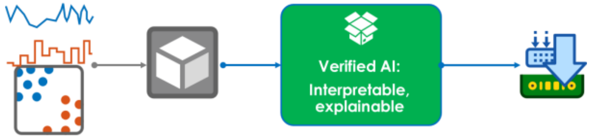
Interpretability methods

Global methods
Local methods

	Pre-model	Intrinsic	Post-hoc
Model-specific		Regression Tree Models Naive Bayes Clustering	Predictor Importance (trees & ensemble) Occlusion Sensitivity CAM, Grad-CAM, Activations
Model-agnostic	PCA tSNE MRMR Correlation		Partial Dependence Plot Individual Conditional Exp LIME Shapley values KNN Fuzzy Logic

Understanding and Verifying your AI models





Neural Network Verification

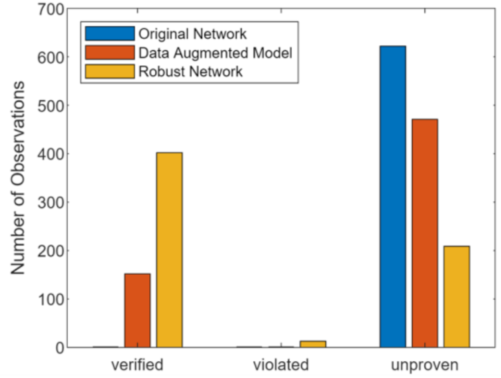
R2022b

Deep Learning Toolbox Verification Library

by MathWorks Deep Learning Toolbox Team STAFF

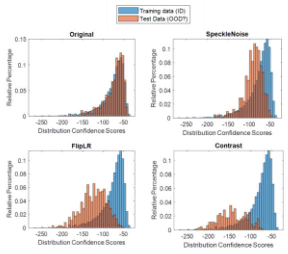
Verify and test robustness of deep learning networks

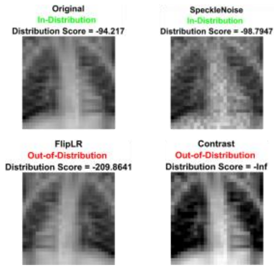
https://www.mathworks.com/help/deeplearning/verification.html

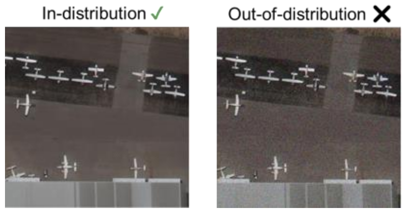



```
perturbation = 0.01;
XLower = XTest - perturbation;
XUpper = XTest + perturbation;


result = verifyNetworkRobustness(dlnet,...
    XLower,XUpper,TTest);
```

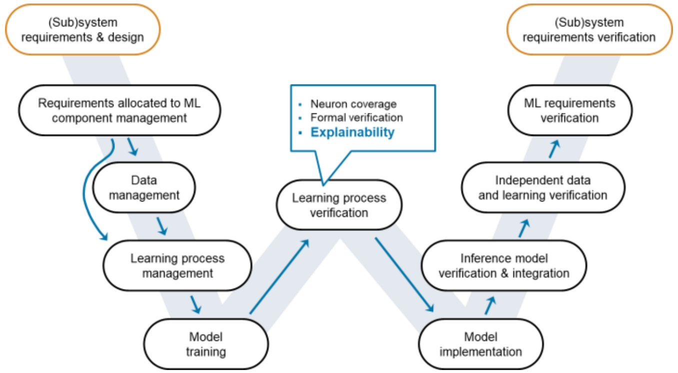
```
summary(result)
      verified      402
      violated       13
      unproven      209
```



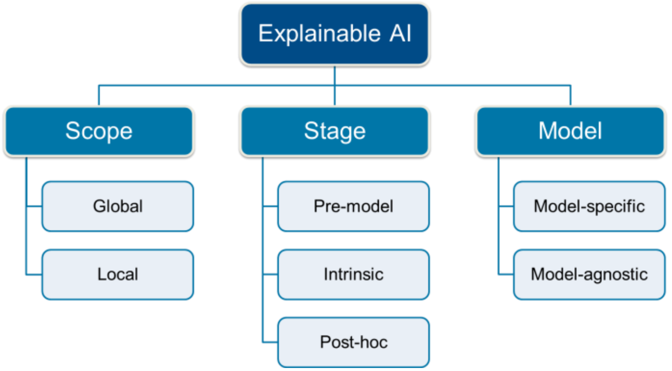




Why MATLAB for Explainable AI?



- Explainable AI plays an important role in Verification and Validation of AI-enabled systems

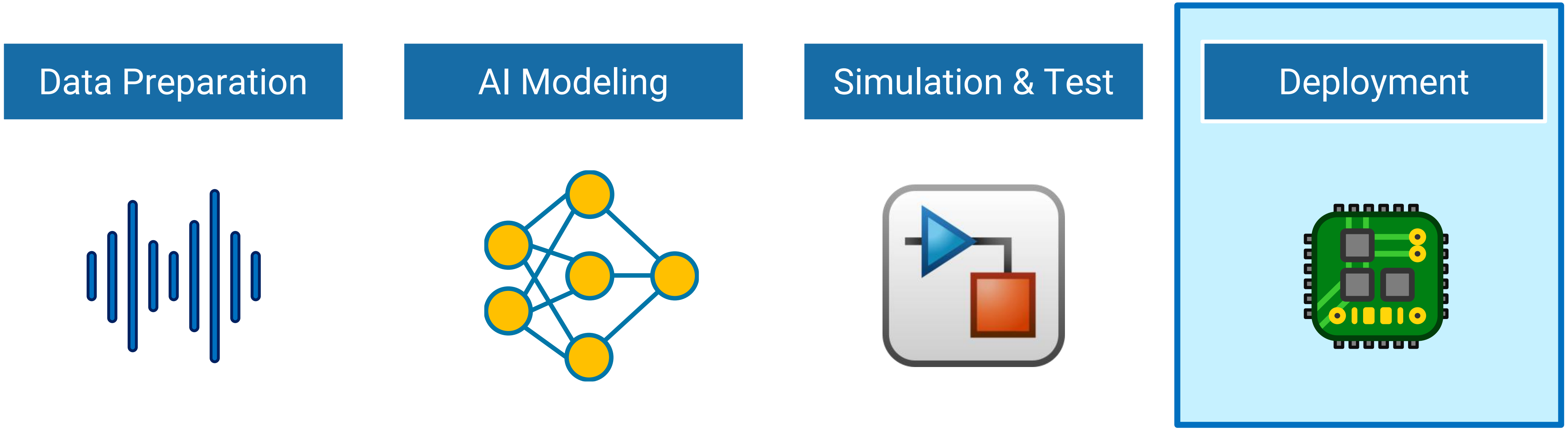


- MATLAB has a growing list of Explainable AI functionality
  - There is no *one-size-fits-all* method

- MathWorks is actively engaging with research groups and certification bodies

EUROCAE WG-114 / SAE G-34 Standardization Working Group “Artificial Intelligence in Aviation”

AI-driven system design



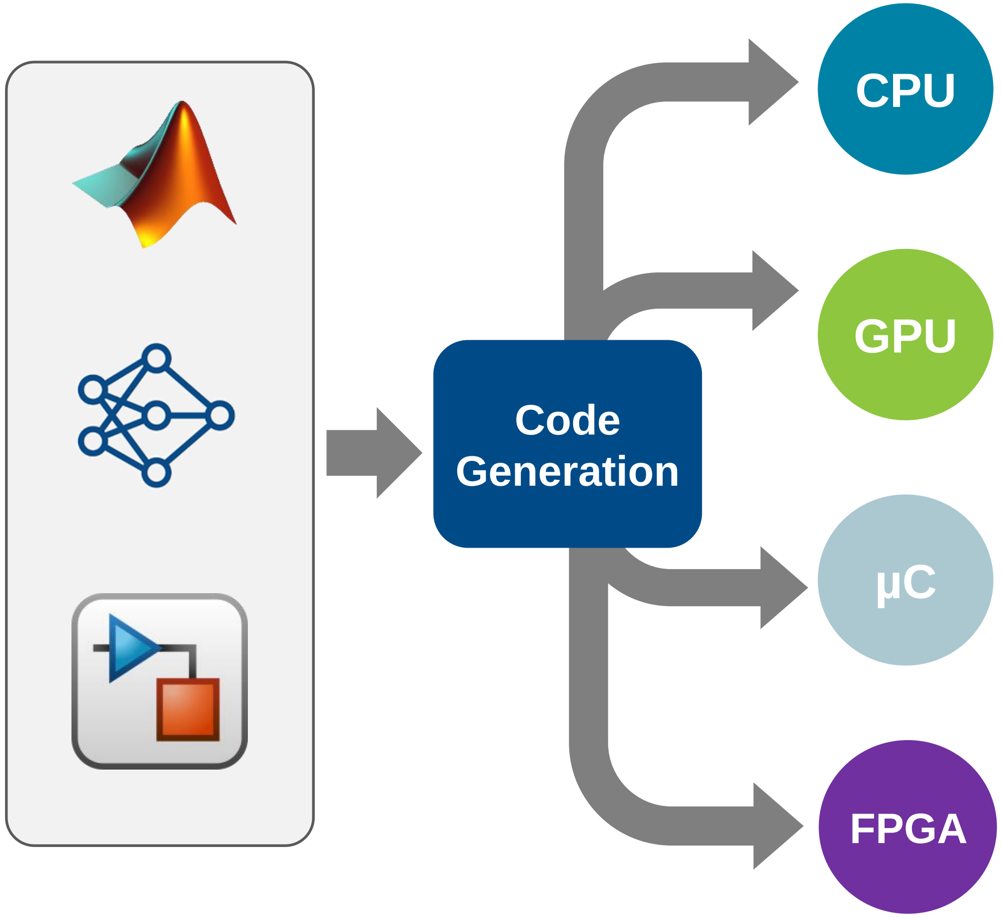
From development to production

Save time and reduce errors

Simplify process, eliminate compatibility issues, deploy on different platforms

End-to-end workflow for designing, testing, and deploying

Deploy to many targets with zero coding errors



Any CPU No Library Needed

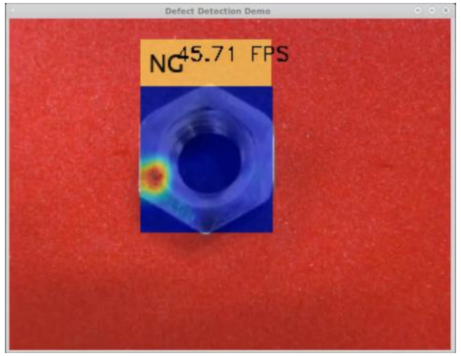
OneDNN Library

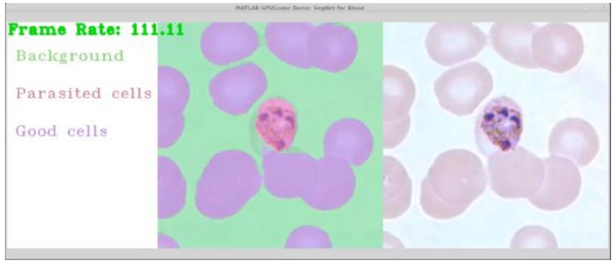
ARM Compute Library

NVIDIA TensorRT and cuDNN Libraries

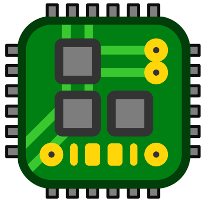
R2022a

TensorFlow Lite Support for TensorFlow Lite





Code generation workflows for embedded target



Through MATLAB

Using apps

GPU Coder　　HDL Coder　　MATLAB Coder

Using command line

codegen

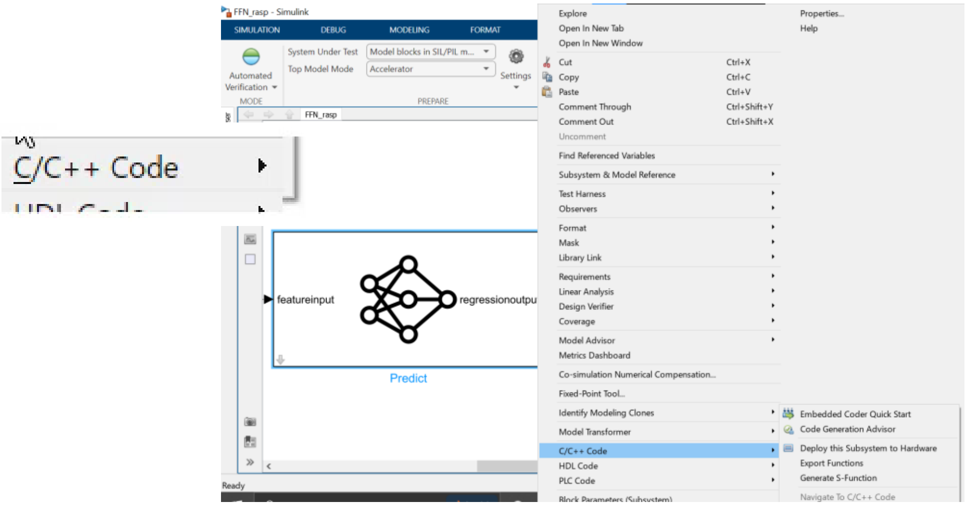
Example
Generate generic C++ code through command line for LSTM trained model

```
% Set Up a Code Generation Configuration Object for a Static Library
cfg = coder.config('lib');
cfg.GenCodeOnly = true;
cfg.TargetLang = 'C++';

% Set Up a Configuration Object for Deep Learning Code Generation
dlcfg = coder.DeepLearningConfig('none');

% Attach the Deep Learning Configuration Object to the Code Generation Configuration Object
cfg.DeepLearningConfig = dlcfg;

% Generate Source C++ Code by Using codegen
codegen -config cfg netPredictLSTM.m -args {ones(3, 500, 'single')} -d netPredictLSTM_Generic -report
```

```
codegen options files function -args {func_inputs1} ... -args {func_inputsN}

codegen project
```

Through Simulink (more suitable for MBD workflow)

Embedded Coder　　Simulink Coder　　HDL Coder　　PLC Coder

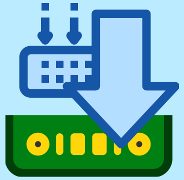


Getting closer to real hardware prototype

Development

Deployment use cases

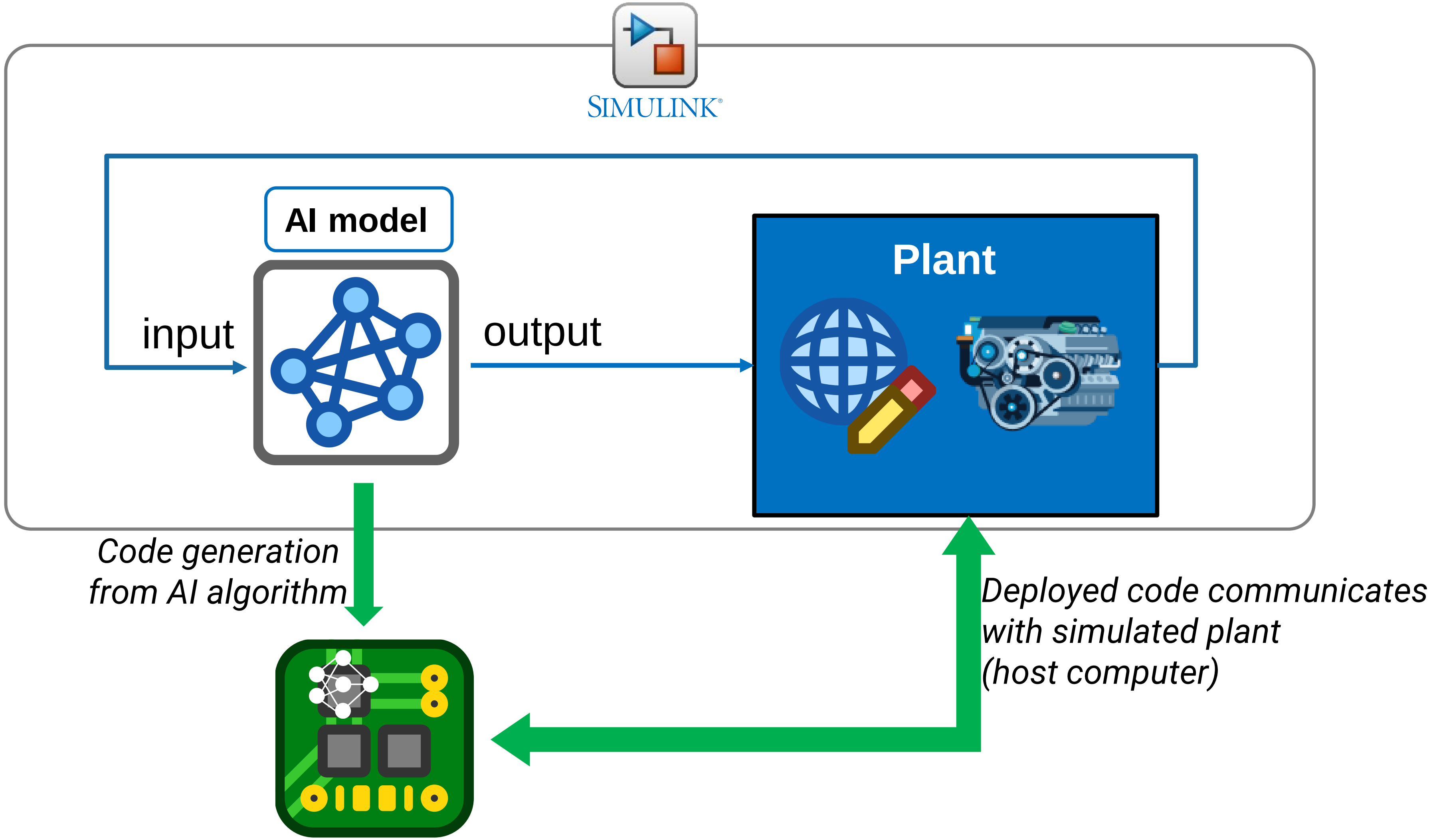
Test



Production

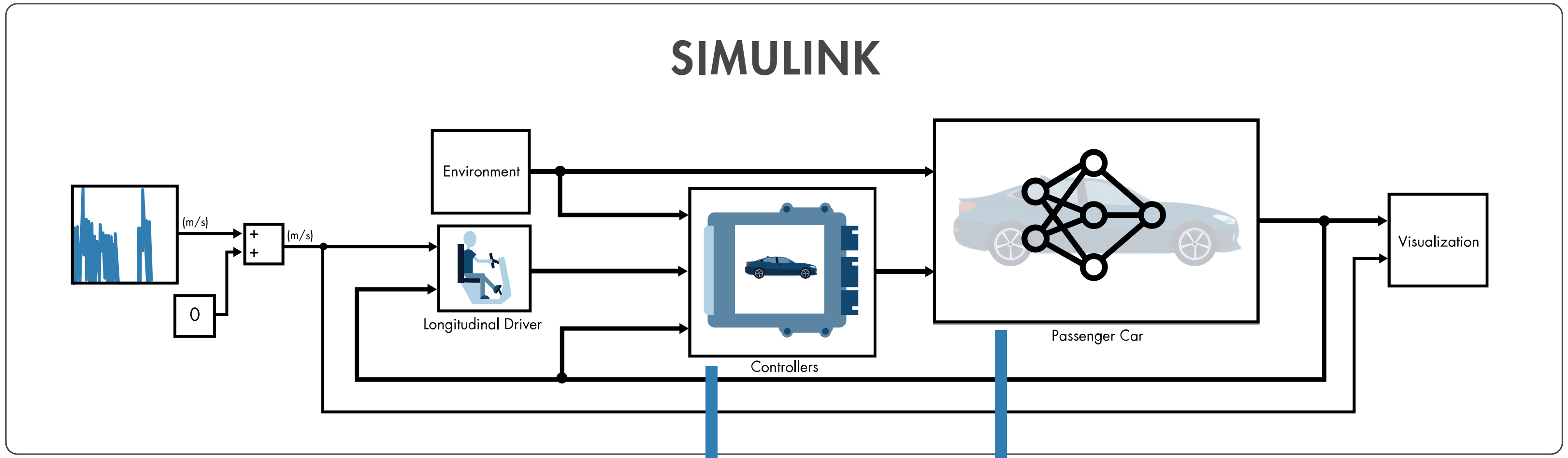
Get closer to real hardware

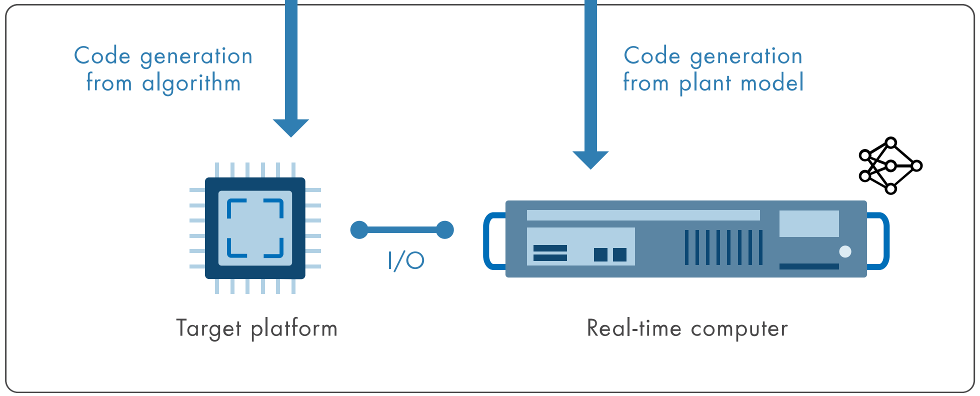
System-level test: Processor-in-the-loop simulation

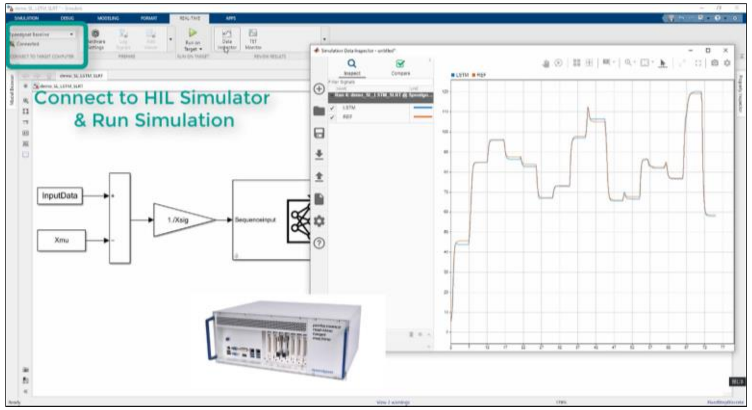


Deploy and validate your embedded AI algorithm on real production processor

System-level test: Hardware-in-the-loop simulation



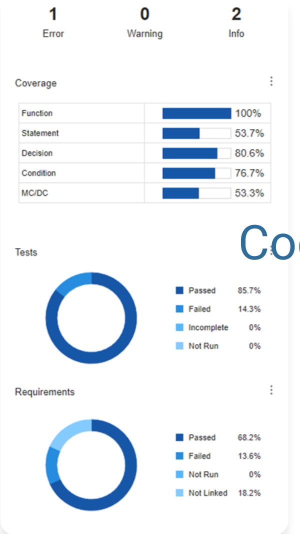




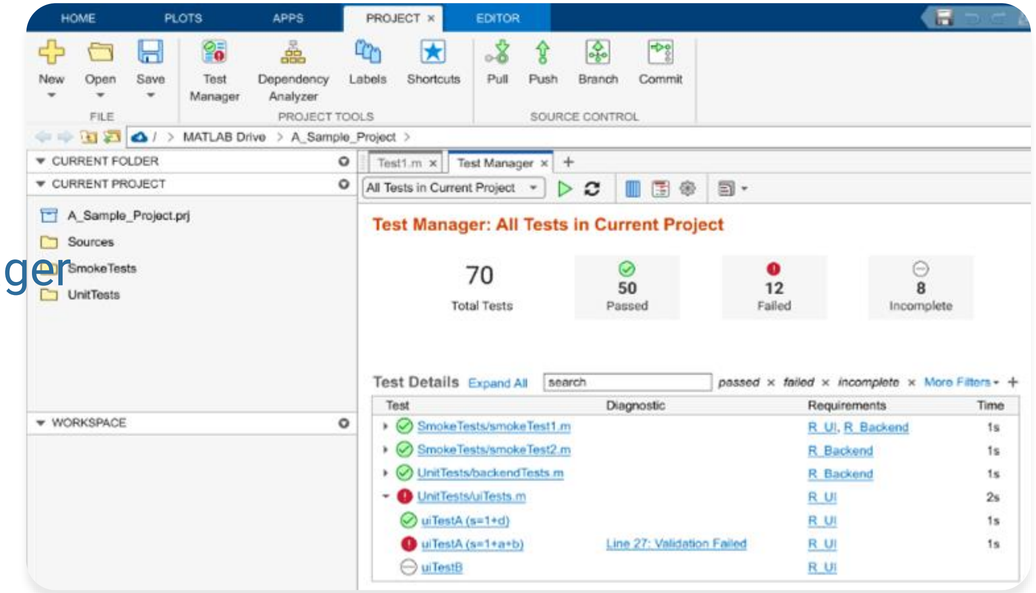
Engine AI-based ROM example

Increasing software quality with MATLAB Test

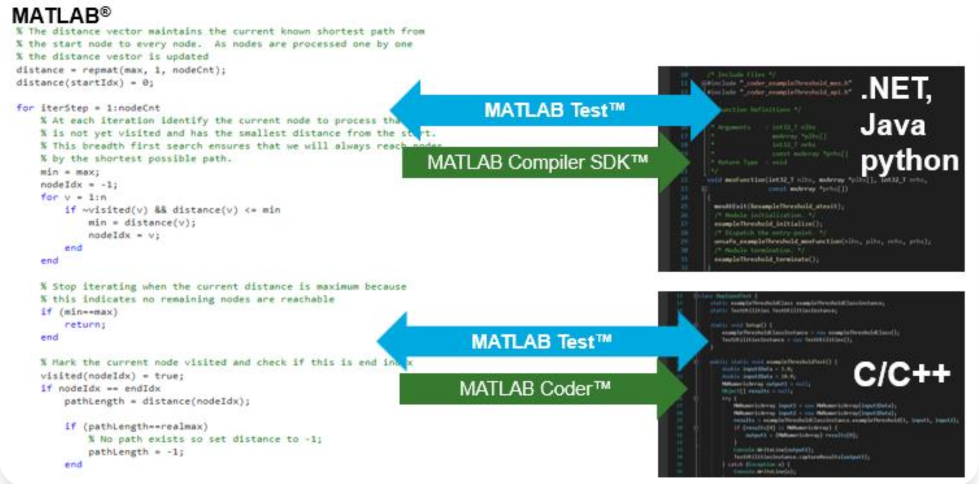
Code quality dashboard



Test Manager



Equivalence testing



Advanced coverage

Code Coverage Report

The code coverage report provides a detailed analysis of the source code covered by the tests.

Overall Coverage Summary

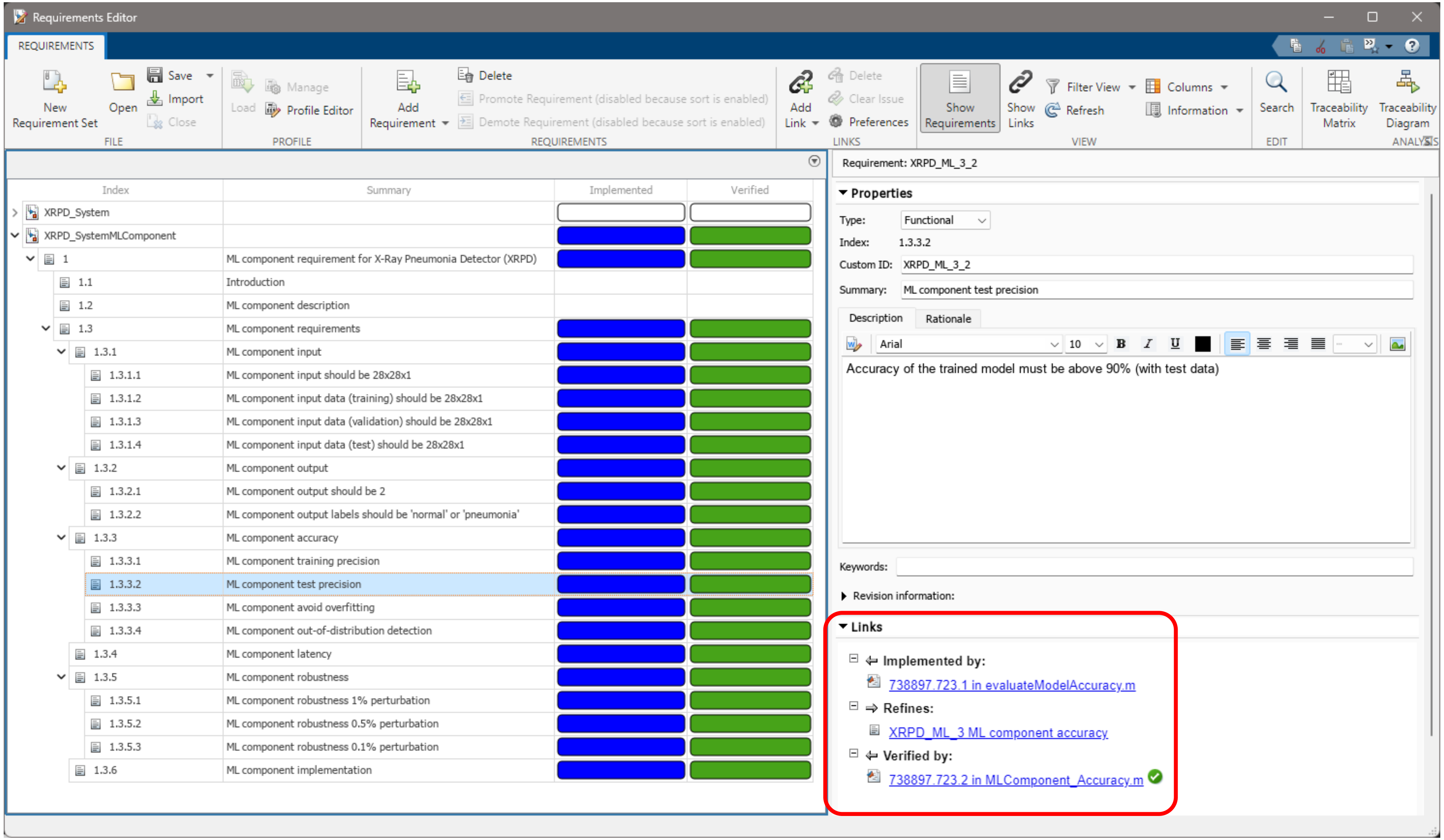
Summary of the code coverage metrics for all source files.

Total Files

5

Coverage Metric	Executable	Missed	Code Coverage
Function	3	0	100%
Statement	82	36	56.09%
Decision	36	5	86.11%
Condition	30	6	80%
MC/DC	15	6	60%

Link to Requirements Verification

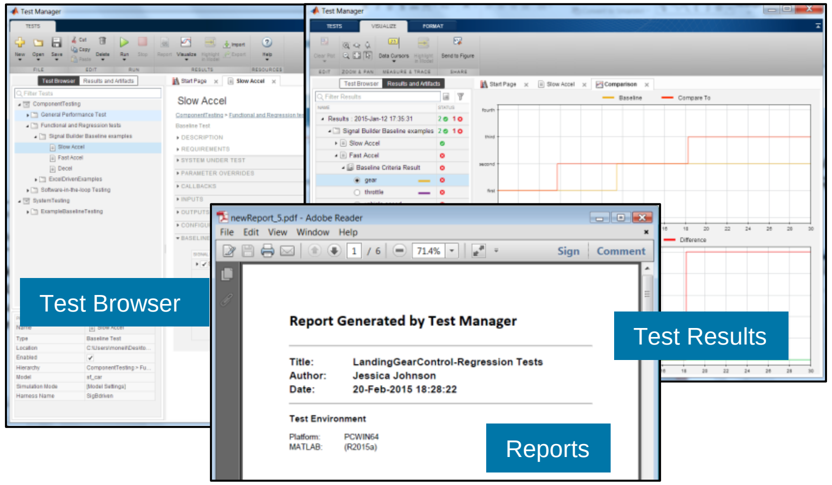


Simulink Test

Develop, manage, and execute simulation-based tests

Test Manager

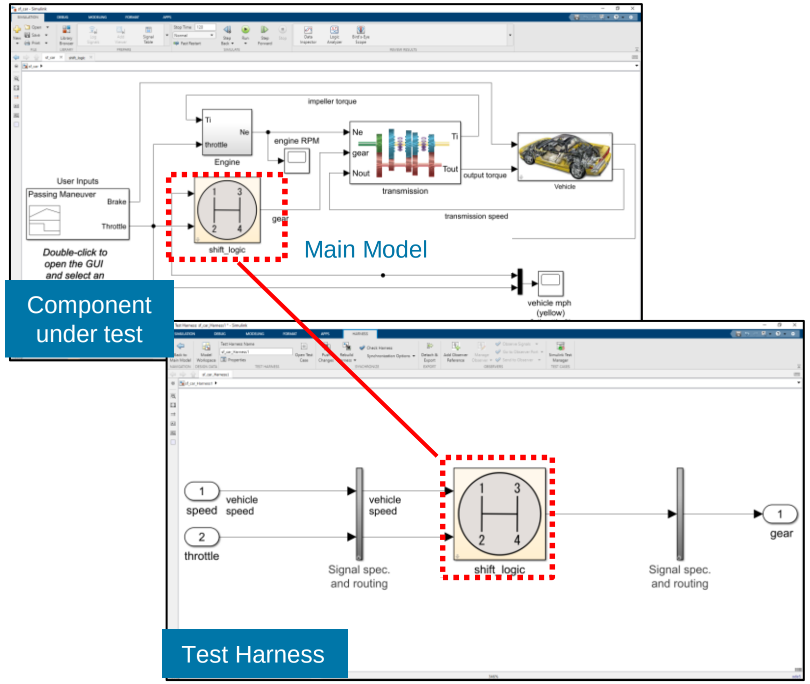
- Author, manage, organize tests
- Execute simulation, equivalence and baseline tests
- Review, export, report



Examples

Test Harnesses

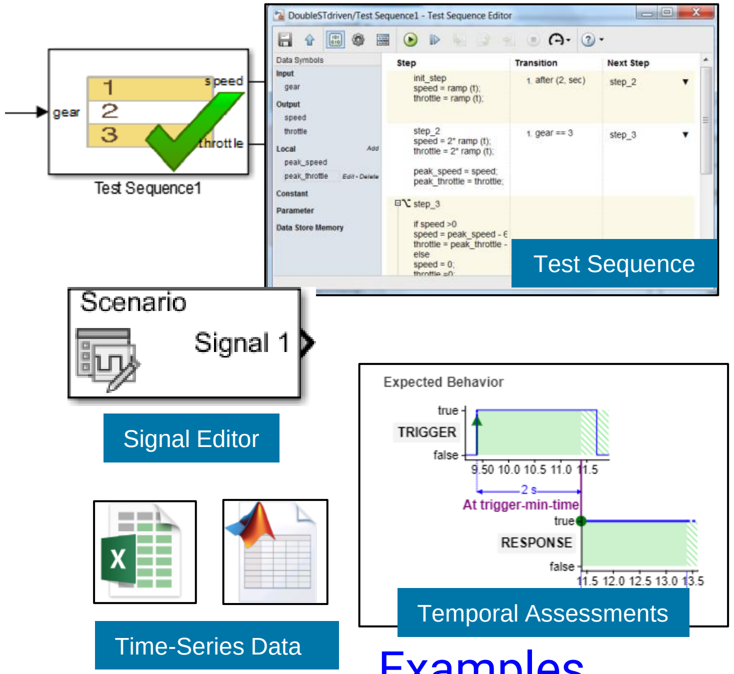
- Isolate Component Under Test
- Synchronized, simulation test environment



Examples

Test Authoring

- Specify test inputs, expected outputs, and tolerances
- Construct complex test sequences and assessments

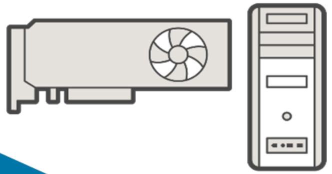


Examples

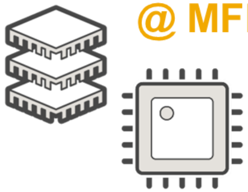
How to optimized performance in hardware constrained environment?

Memory: 32GB
@ 10TFLOPs

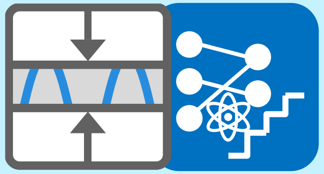
Memory: 8GB
@ GFLOPs



Memory: 1kB~1MB
@ MFLOPs



Compression



Pruning and Model Compression techniques to reduce model size and speed up inference

How to optimized performance in hardware constrained environment?

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

Pruning and Model Compression techniques to reduce model size and speed up inference

Quantization

- Conversion from floating point to fixed point

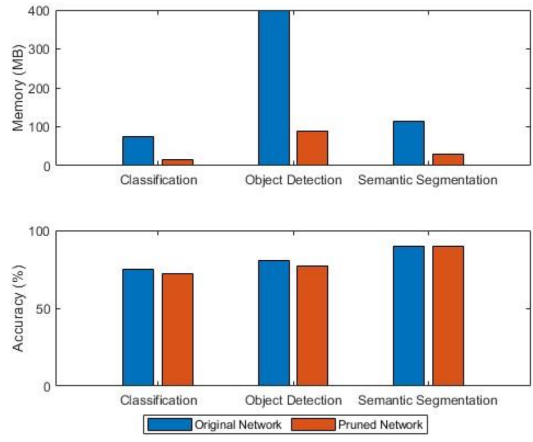
Pruning

- Removing unimportant parts of the network

Projection

- Project learnable parameters into a lower dimensional space

Deep Network Quantization App	R2020a
Taylor Pruning	R2022a
Projection	R2022b

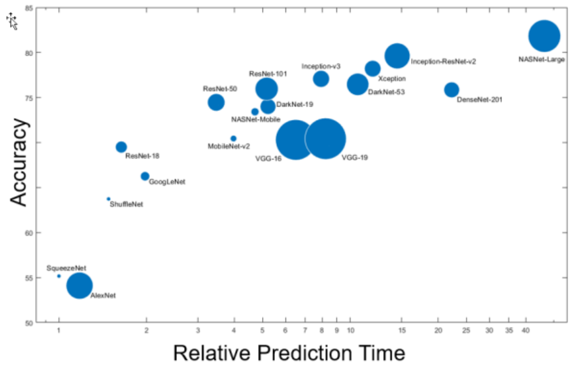


AI model compression workflow

Determine Hardware Constraints → **1** Select Model → **2** Simplify Model → **3** Quantize Model Parameters → Deploy & Integrate

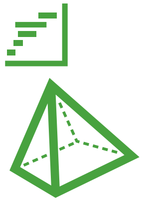
Select Model

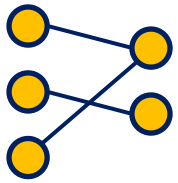
Size aware model selection



Simplify Model

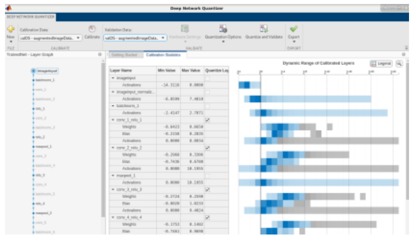
Projection and **Pruning**



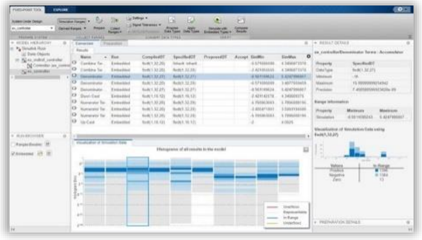


Quantize Model

Deep Learning Toolbox Model Quantization Library



Fixed-Point Designer



Conclusion

- Many promising application in the intersection between AI and Simulation
- Combining AI and simulation for designing complex system is all about tradeoffs
- MATLAB and Simulink
  - Run simulation of AI model at the system level and collect metric
  - Refine model and implement the optimal AI technique
  - Balance AI accuracy and deployment efficiency
  - One toolchain for seamless interaction between AI and simulation
  - Select and implement the optimal AI technique balan

Key takeaways

Data Preparation

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

Toolboxes for **domain-specific** pre/post-processing

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

Low-code workflow for AI Modeling through Apps

Import models from **TensorFlow**, **PyTorch** or other DL Frameworks

Simulation & Test

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

Simulink blocks for AI models make integration easy

Compression

- Size aware model selection
- Compression techniques (pruning, projection)
- Quantization

Model compression techniques to reduce model size and speed up inference

Deployment

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

Code generation for embedded targets

Thank you!

Q&A