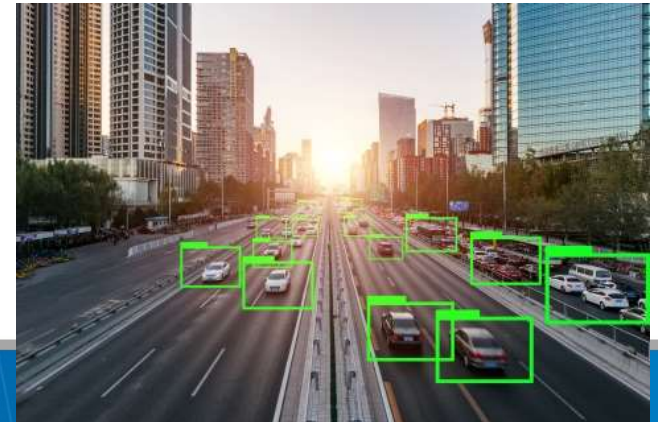


AI for Simulink Users

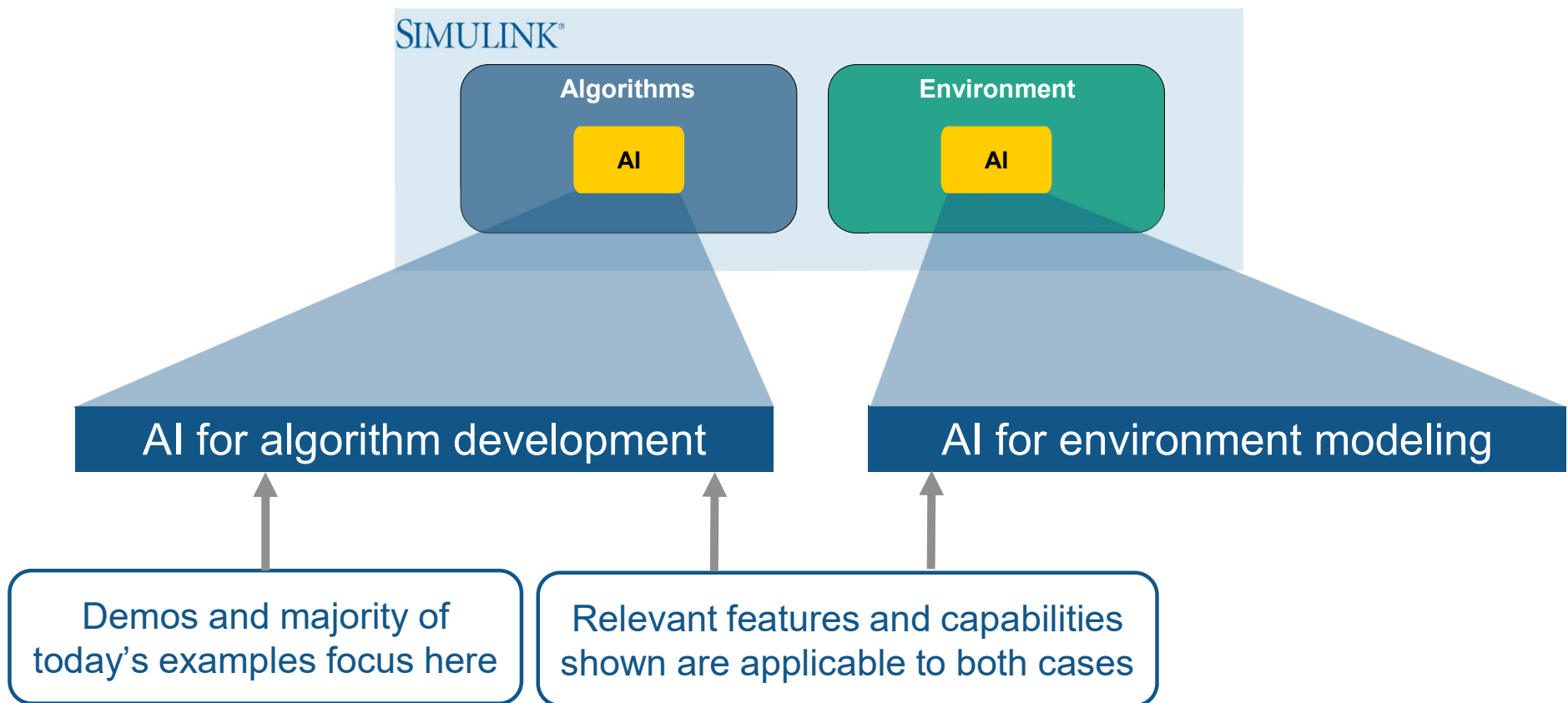


Bill Chou, Product Manager – Coder Products

Bernhard Suhm, Product Manager – Machine Learning

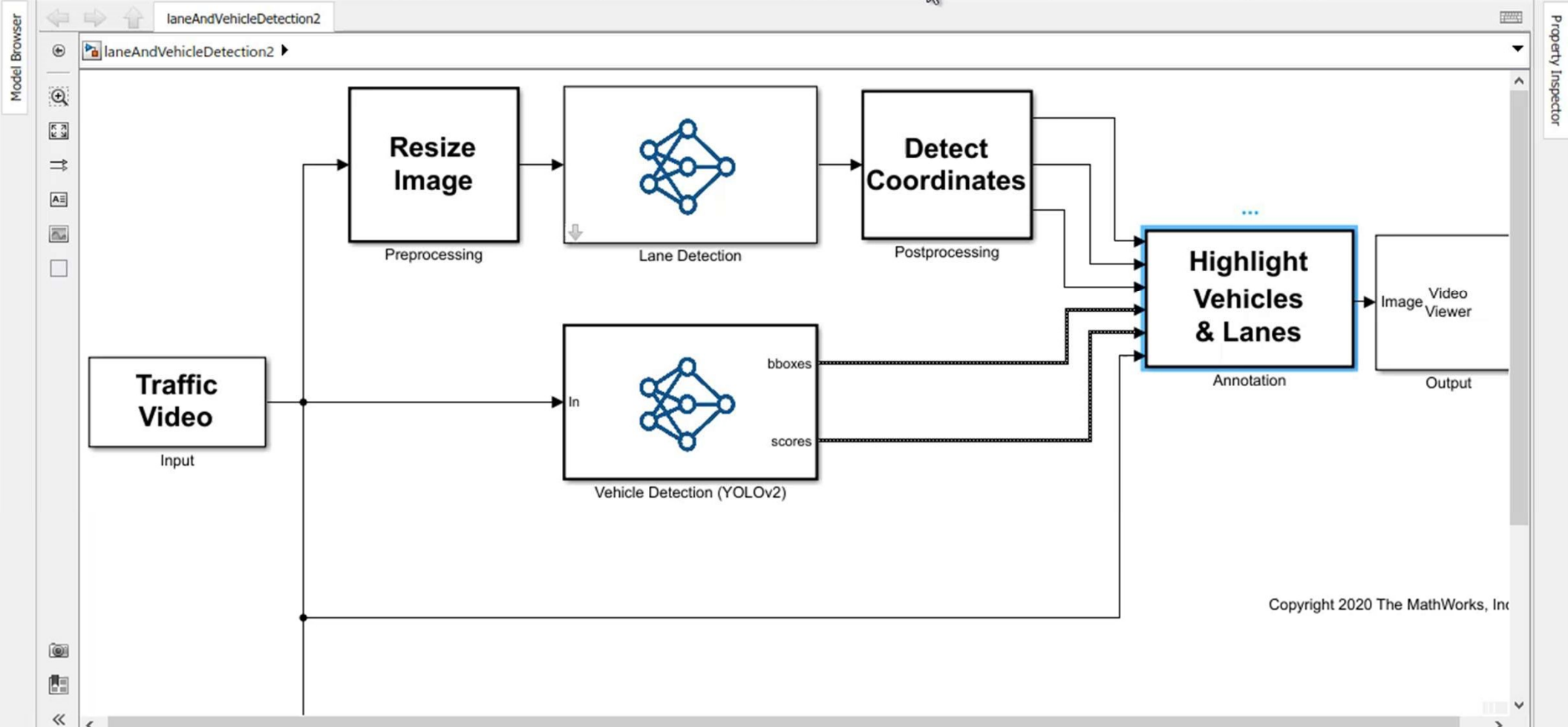
Emmanouil Tzorakoleftherakis, Product Manager – Controls Systems

Today's Objective: How to Build AI Functionality into your Systems



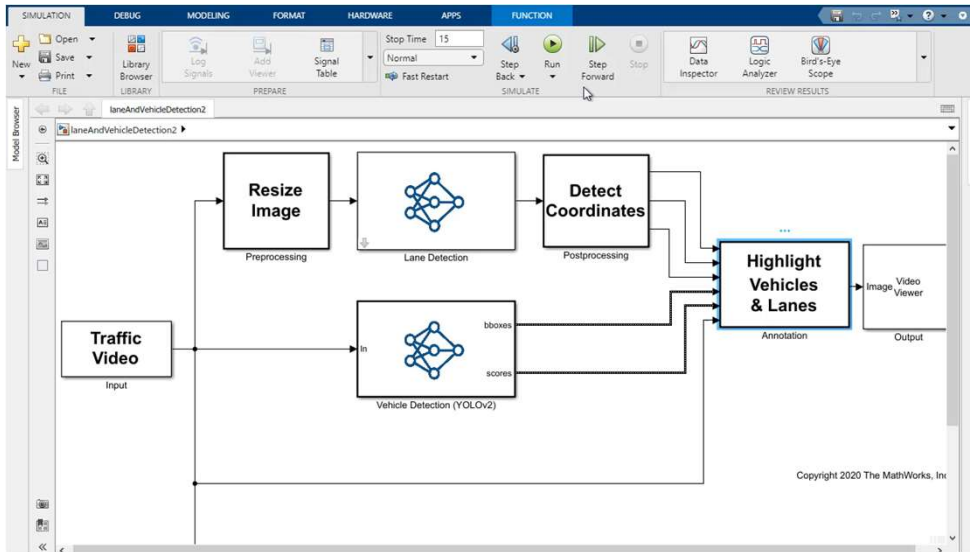
SIMULATION DEBUG MODELING FORMAT HARDWARE APPS **FUNCTION**

+ New Open Save Print FILE
 Library Browser Log Signals Add Viewer Signal Table PREPARE
 Stop Time: 15 Normal Fast Restart Step Back Run Step Forward Stop SIMULATE
 Data Inspector Logic Analyzer Bird's-Eye Scope REVIEW RESULTS



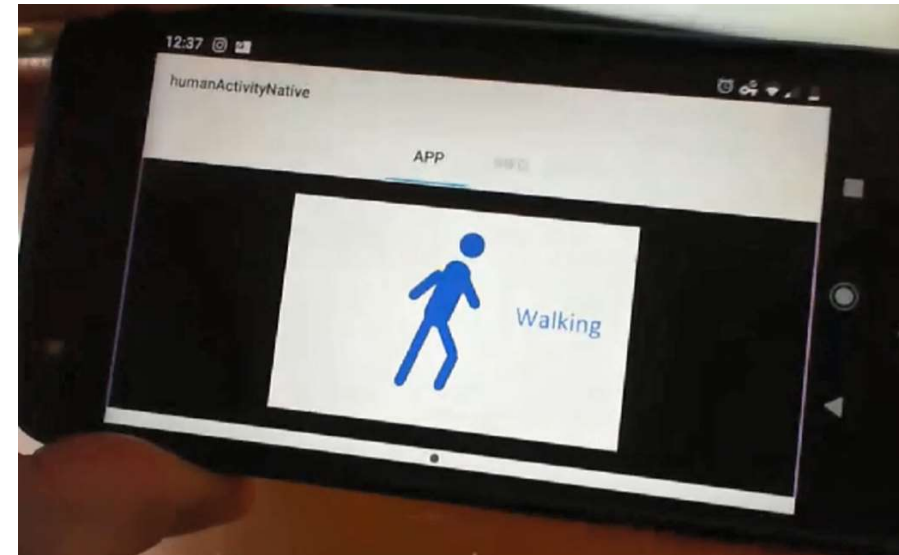
Today's Objective: How to Build AI Functionality into your Systems

Deep Learning in Simulink



Demo: Lane and vehicle detection

Traditional Machine Learning in Simulink



Demo: Human activity recognition

Why should I integrate my AI components into Simulink?

Learning Algorithms Are Driving the AI Megatrend

Statistics and Machine Learning Toolbox



Oversteering Detection
BMW



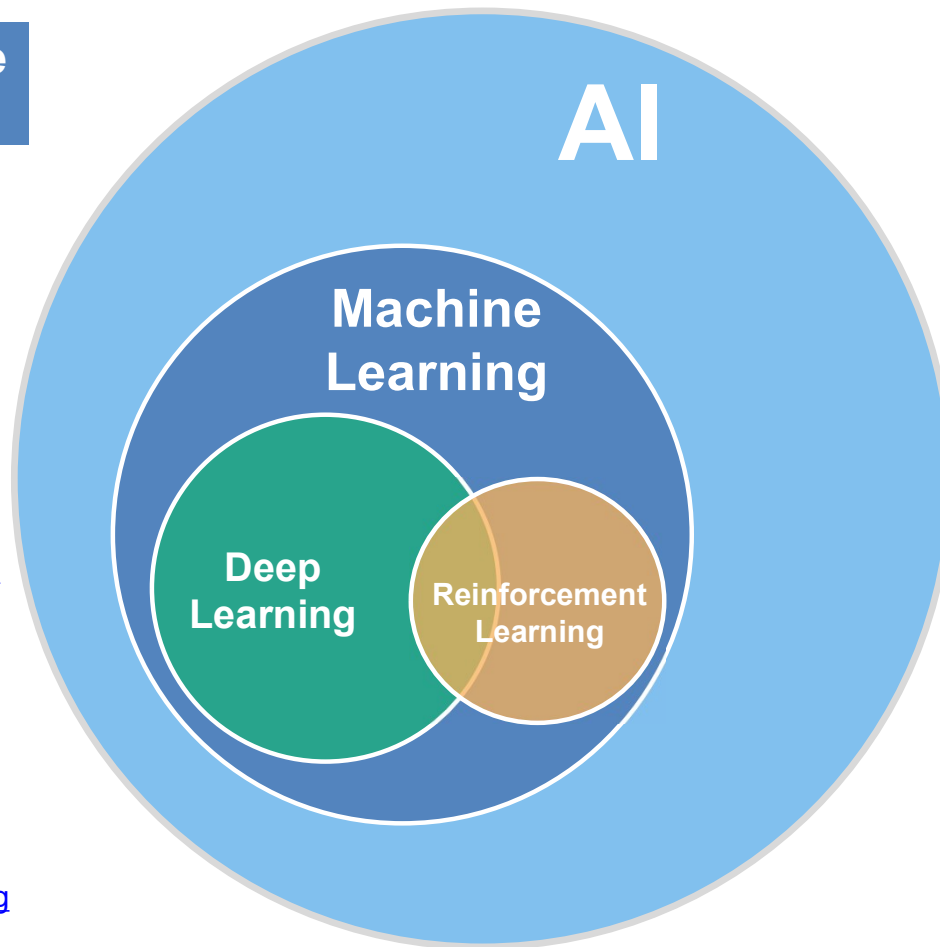
Predictive Maintenance
Baker Hughes



Digital Twin
Atlas Copco



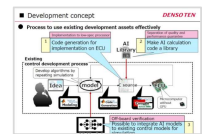
Automatic Ground-Truth Labeling
Caterpillar



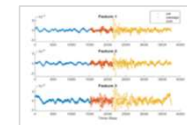
Deep Learning Toolbox



Defect Detection
Airbus

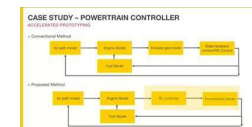


ECU Control
Denso



Seismic Event Detection
Shell

Reinforcement Learning Toolbox



Powertrain control
Vitesco Technologies

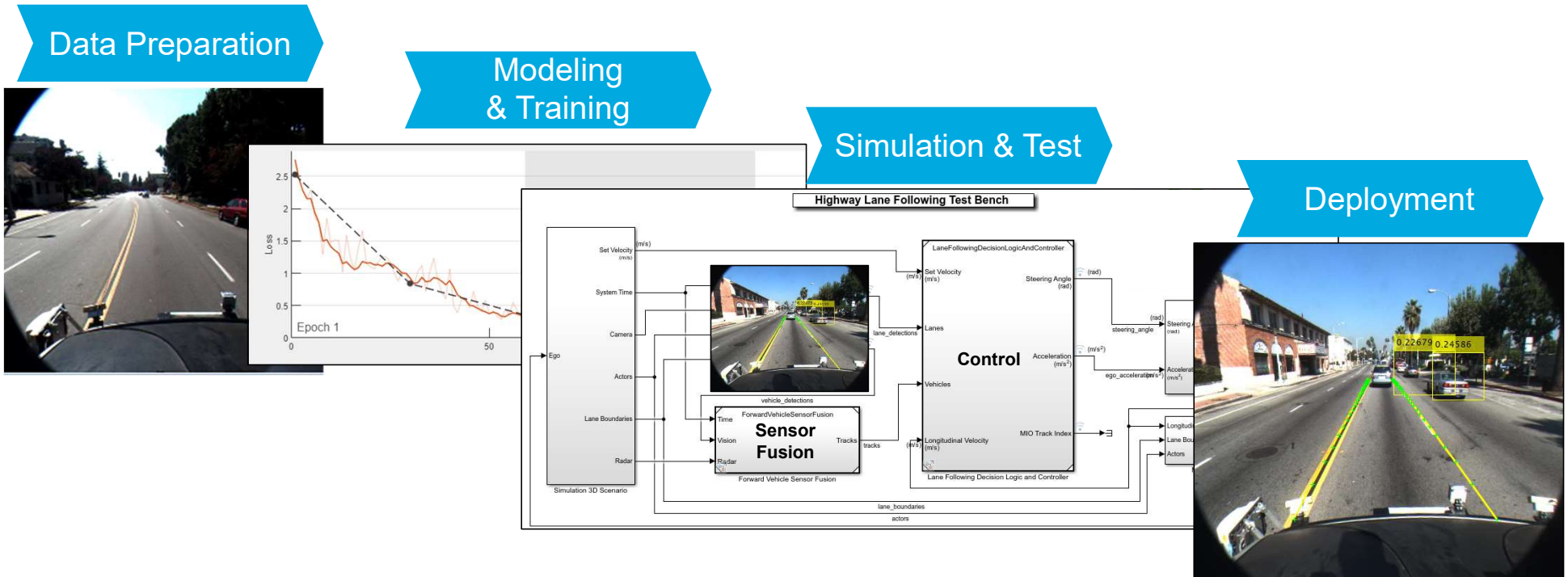
AI Examples: Machine Learning for Fault Identification



Why Machine Learning over traditional quantitative/qualitative methods?

- Higher accuracy
- Process may be challenging or impossible to model

AI Examples: Deep Learning for Vehicle Detection



Why Deep Learning over traditional Computer Vision?

- No feature engineering
- Higher accuracy

Additional AI Examples

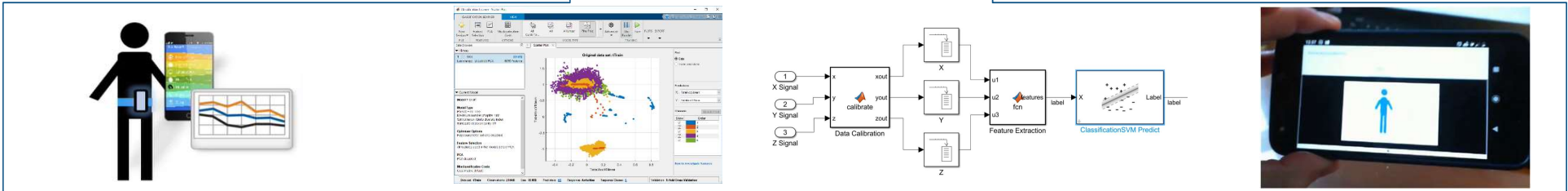
Data Preparation

Modeling & Training

Simulation & Test

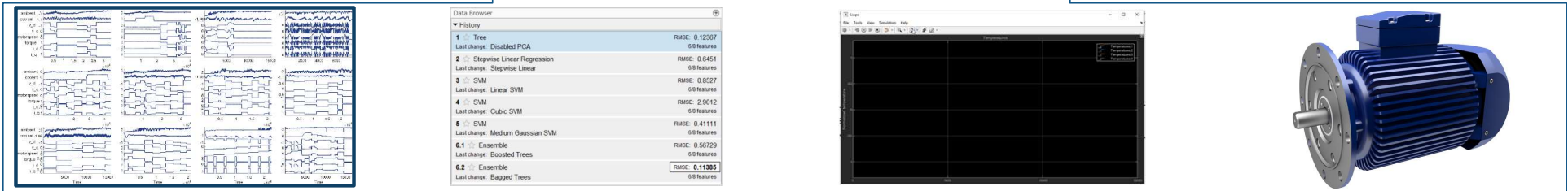
Deployment

Human Activity Recognition



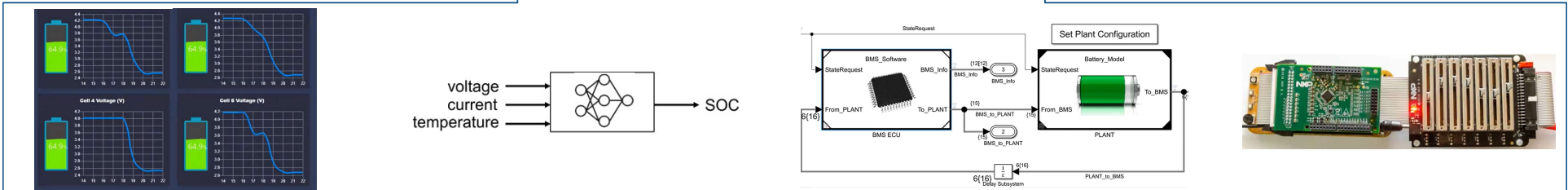
The diagram illustrates the Human Activity Recognition process. It starts with an icon of a person with a smartphone and a tablet displaying a line graph. This leads to a screenshot of a MATLAB workspace showing a scatter plot of 'Original data set (train)'. The data is then processed through a 'Data Calibration' block, which outputs X, Y, and Z signals. These signals are fed into a 'Feature Extraction' block, which outputs 'u1', 'u2', and 'u3'. Finally, these features are used by a 'ClassificationSVM Predict' block to output a 'Label'. To the right, a photograph shows a smartphone screen displaying a blue human silhouette, representing the final activity recognition result.

Electric Motor Temperature Estimation



The diagram illustrates the Electric Motor Temperature Estimation process. On the left, a grid of plots shows various sensor signals over time. These signals are processed through a 'Data Browser' window, which lists several models and their RMSE values: 1. Tree (RMSE: 0.12367), 2. Stepwise Linear Regression (RMSE: 0.0451), 3. SVM (RMSE: 0.0527), 4. SVM (RMSE: 2.9012), 5. SVM (RMSE: 0.41111), 6.1 Ensemble (RMSE: 0.50729), and 6.2 Ensemble (RMSE: 0.11385). The best model is selected for training. The resulting model is then used in a simulation environment (MATLAB/Simulink) to estimate the temperature of an electric motor, which is shown as a blue motor on the right.

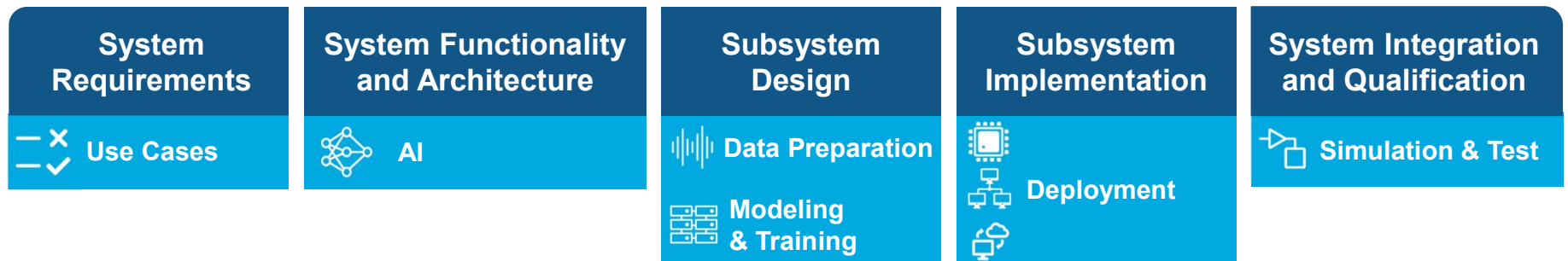
Battery State of Charge Estimation



The diagram illustrates the Battery State of Charge (SOC) Estimation process. On the left, four plots show 'Cell 4 Voltage (V)' and 'Cell 6 Voltage (V)' over time, with a green battery icon indicating a 64.9% SOC. The inputs 'voltage', 'current', and 'temperature' are fed into a neural network model, which outputs the SOC. This model is then implemented in a Simulink block diagram. The 'BMS Software' block receives 'StateRequest' and 'BMS_Info' from the 'BMS ECU' and outputs 'BMS_Info' to the 'Battery Model' block. The 'Battery Model' block receives 'StateRequest' and 'From_BMS' from the 'PLANT' and outputs 'To_BMS' to the 'BMS ECU'. The 'BMS ECU' also receives 'From_PLANT' and 'BMS_Info' from the 'BMS ECU' and outputs 'To_PLANT' to the 'PLANT'. The 'PLANT' block receives 'From_PLANT' and 'To_PLANT' and outputs 'From_BMS' to the 'BMS ECU'. The 'BMS ECU' block is connected to a 'Battery Subsystem' block, which is connected to the 'PLANT' block. On the right, a photograph shows a green battery management system (BMS) board connected to a battery pack.

Systems Complexity Is Increasing

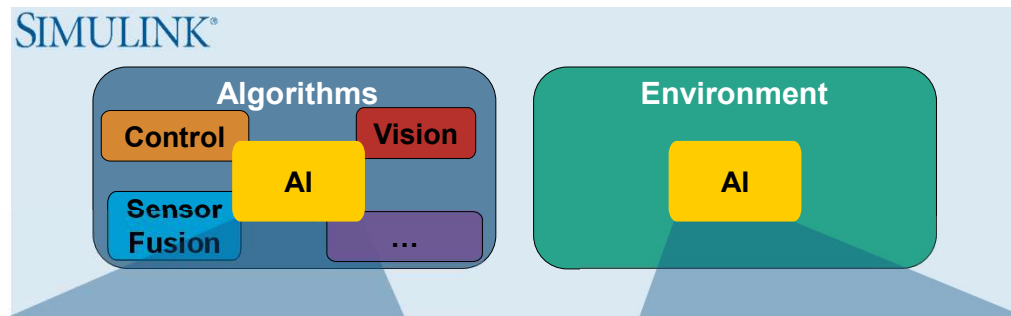
Model-Based Design and AI can help build complex systems



AI-driven system design workflow



Integrating AI Models into Simulink



AI for algorithm development

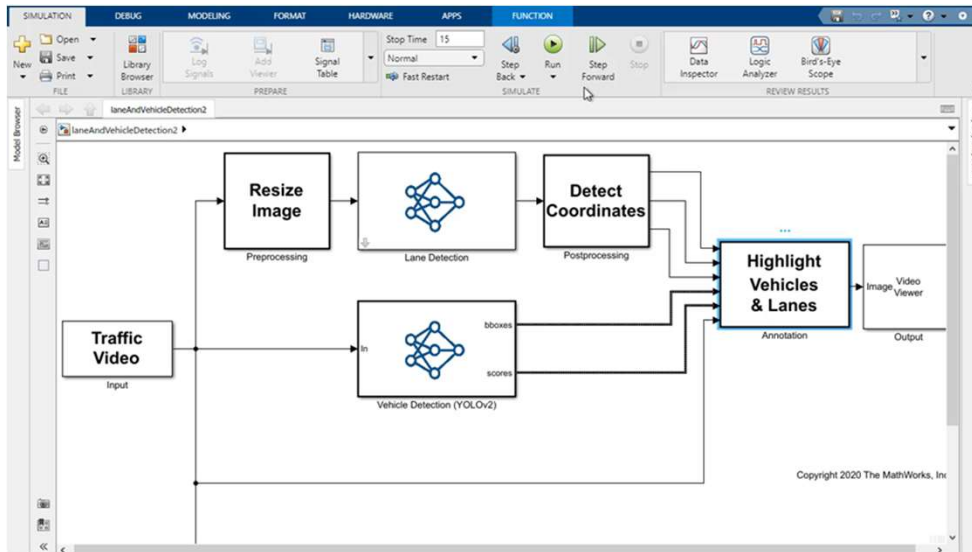
- Simulate for system-level testing
- Verify system requirements
- Deploy overall design to CPU, GPU, ECU, FPGA or a mix of targets

AI for environment modeling

- Speed up high-fidelity model
- Use data-driven model where mathematical modeling is challenging
- Enable HIL tests for above
- Share component with non-experts in a particular modeling domain or tool

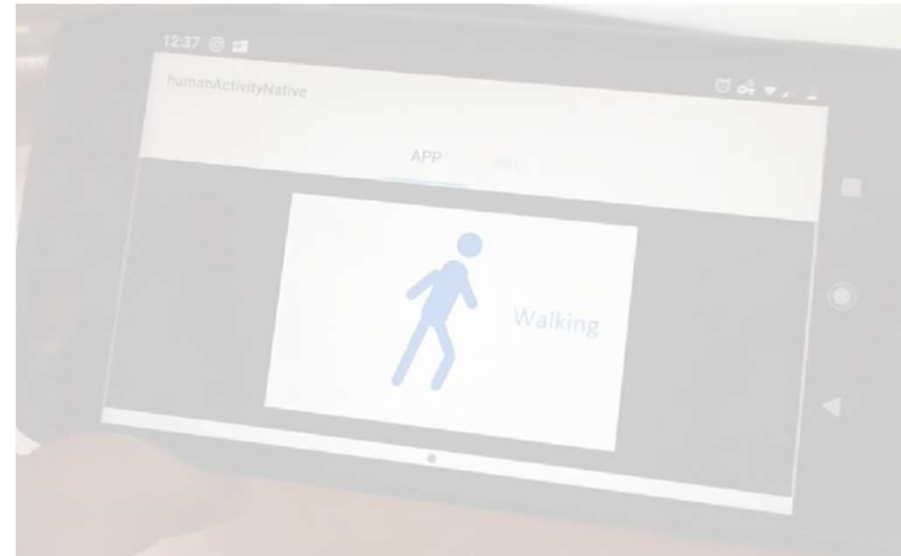
Today's Objective: How to Build AI Functionality into your Systems

Deep Learning in Simulink



Demo: Lane and vehicle detection

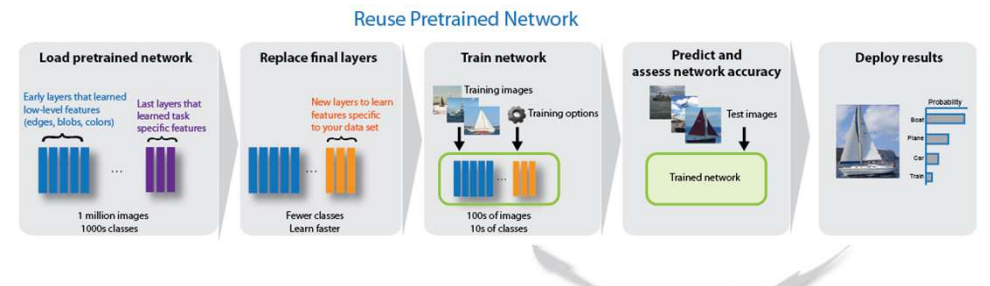
Traditional Machine Learning in Simulink



Demo: Human activity recognition

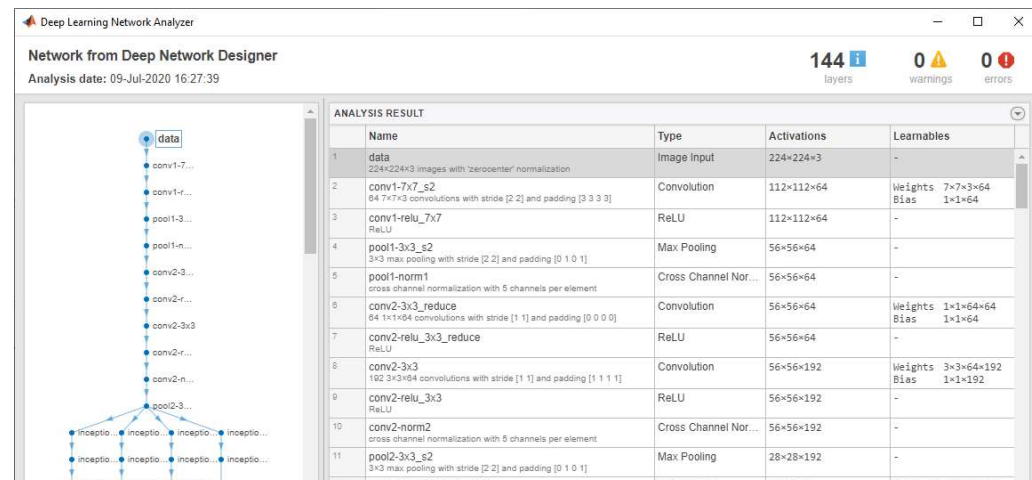
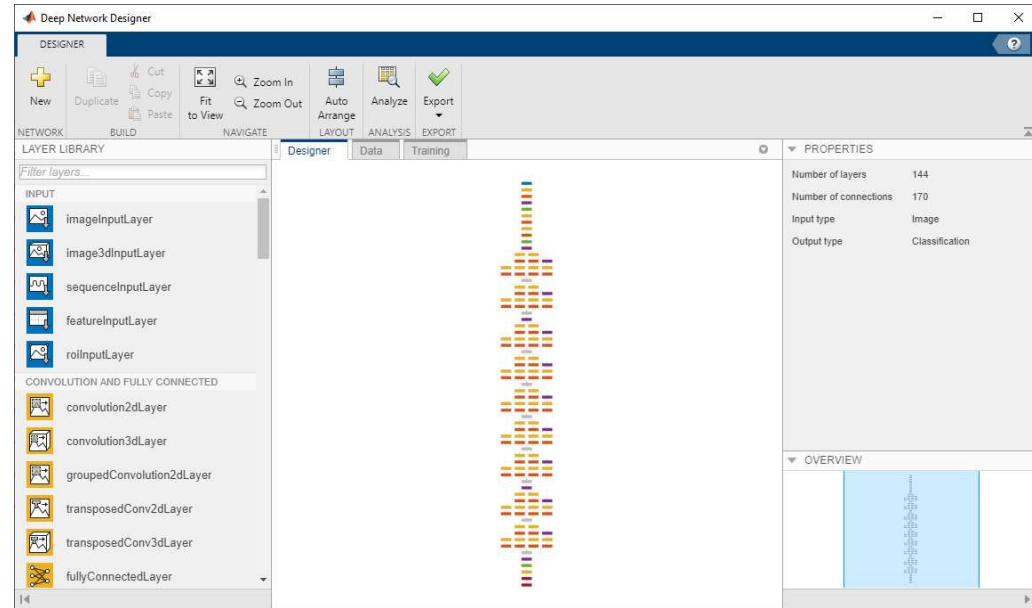
Deep Learning Toolbox

- Leverage pre-defined networks & pretrained networks
- Visually create networks to enable faster design
- Find optimal network using experiments
- Explain and visualize how a network works
- Interoperate with other frameworks



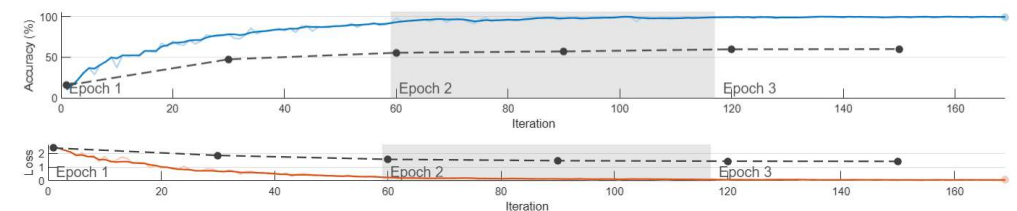
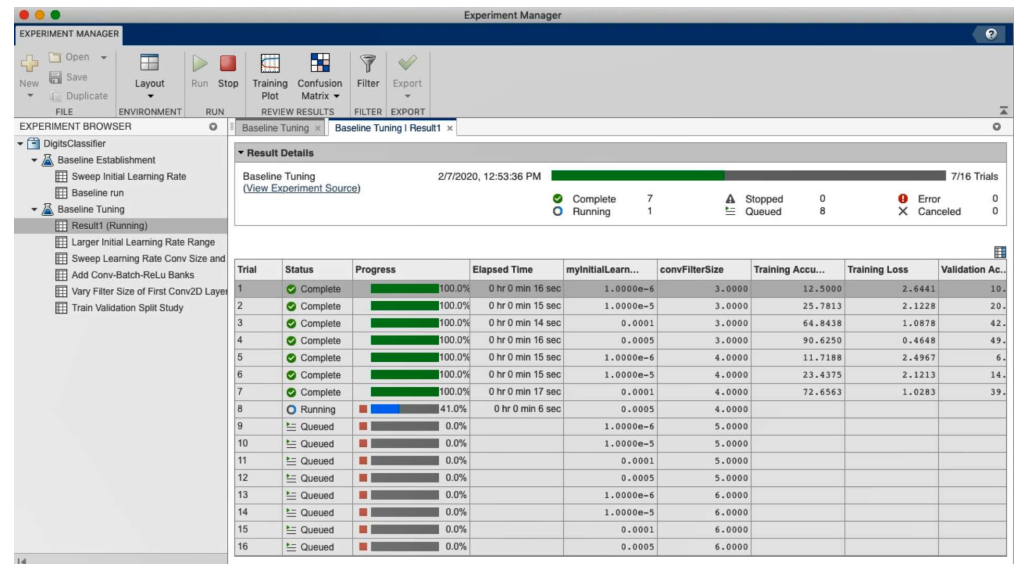
Deep Learning Toolbox

- Leverage pre-defined networks & pretrained networks
- Visually create networks to enable faster design
- Find optimal network using experiments
- Explain and visualize how a network works
- Interoperate with other frameworks



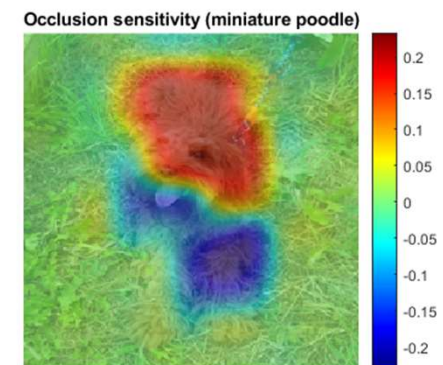
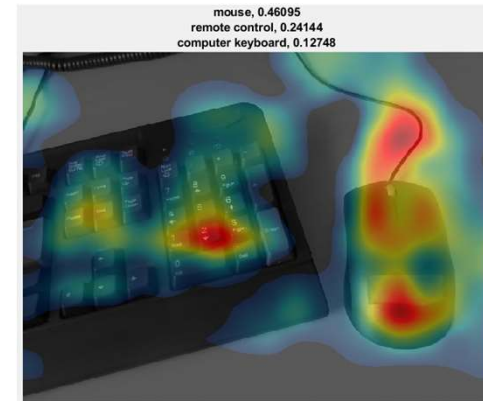
Deep Learning Toolbox

- Leverage pre-defined networks & pretrained networks
- Visually create networks to enable faster design
- Find optimal network using experiments
- Explain and visualize how a network works
- Interoperate with other frameworks



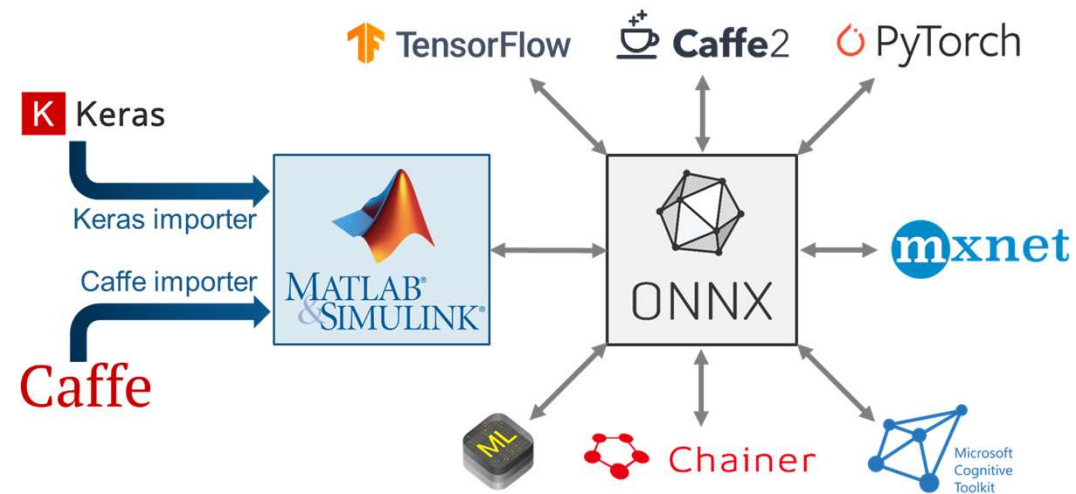
Deep Learning Toolbox

- Leverage pre-defined networks & pretrained networks
- Visually create networks to enable faster design
- Find optimal network using experiments
- Explain and visualize how a network works
- Interoperate with other frameworks

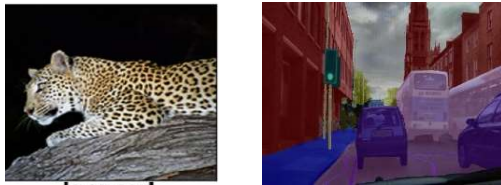


Deep Learning Toolbox

- Leverage pre-defined networks & pretrained networks
- Visually create networks to enable faster design
- Find optimal network using experiments
- Explain and visualize how a network works
- **Interoperate with other frameworks**

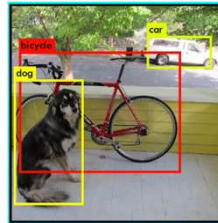


Deep Learning Networks



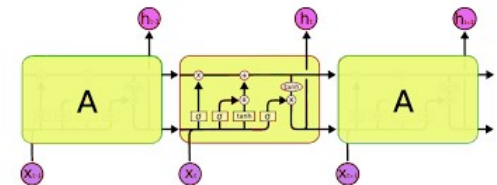
**Image Classification,
Semantic Segmentation**

- ResNet
- Inception v3
- MobileNet v2
- GoogLeNet
- VGG
- SegNet
- DeepLab v3+
- ...



Object Detectors

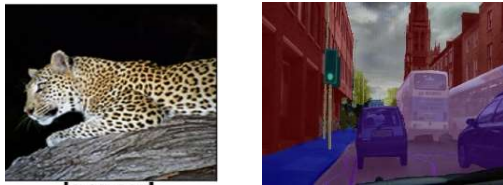
- YOLO v2
- SSD



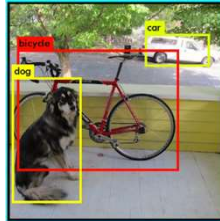
**Sequence Networks for
audio, text, and signal data**

- LSTM
- BiLSTM

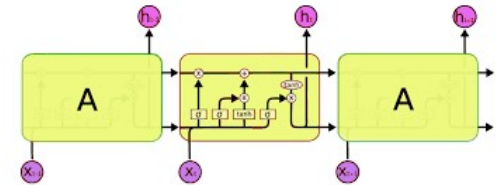
Deep Learning Networks



**Image Classification,
Semantic Segmentation**



Object Detectors



**Sequence Networks for
audio, text, and signal data**

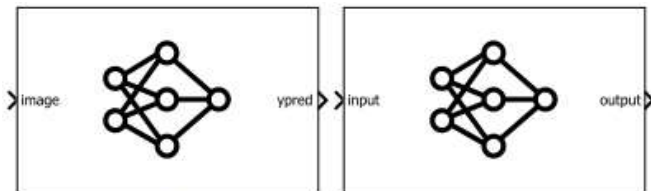


Image Classifier

Predict

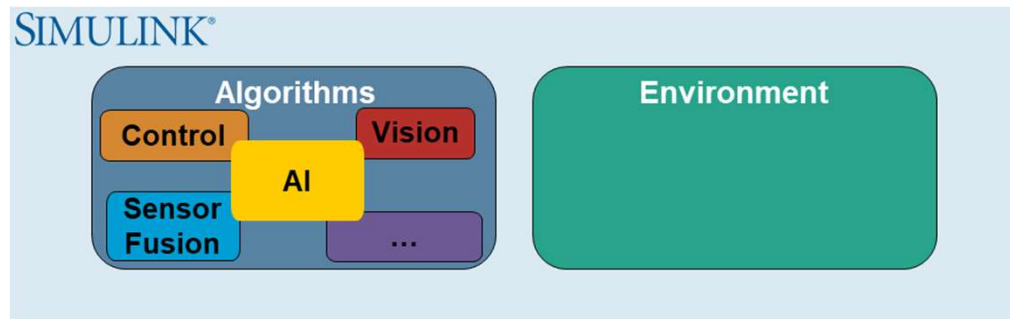


MATLAB Function

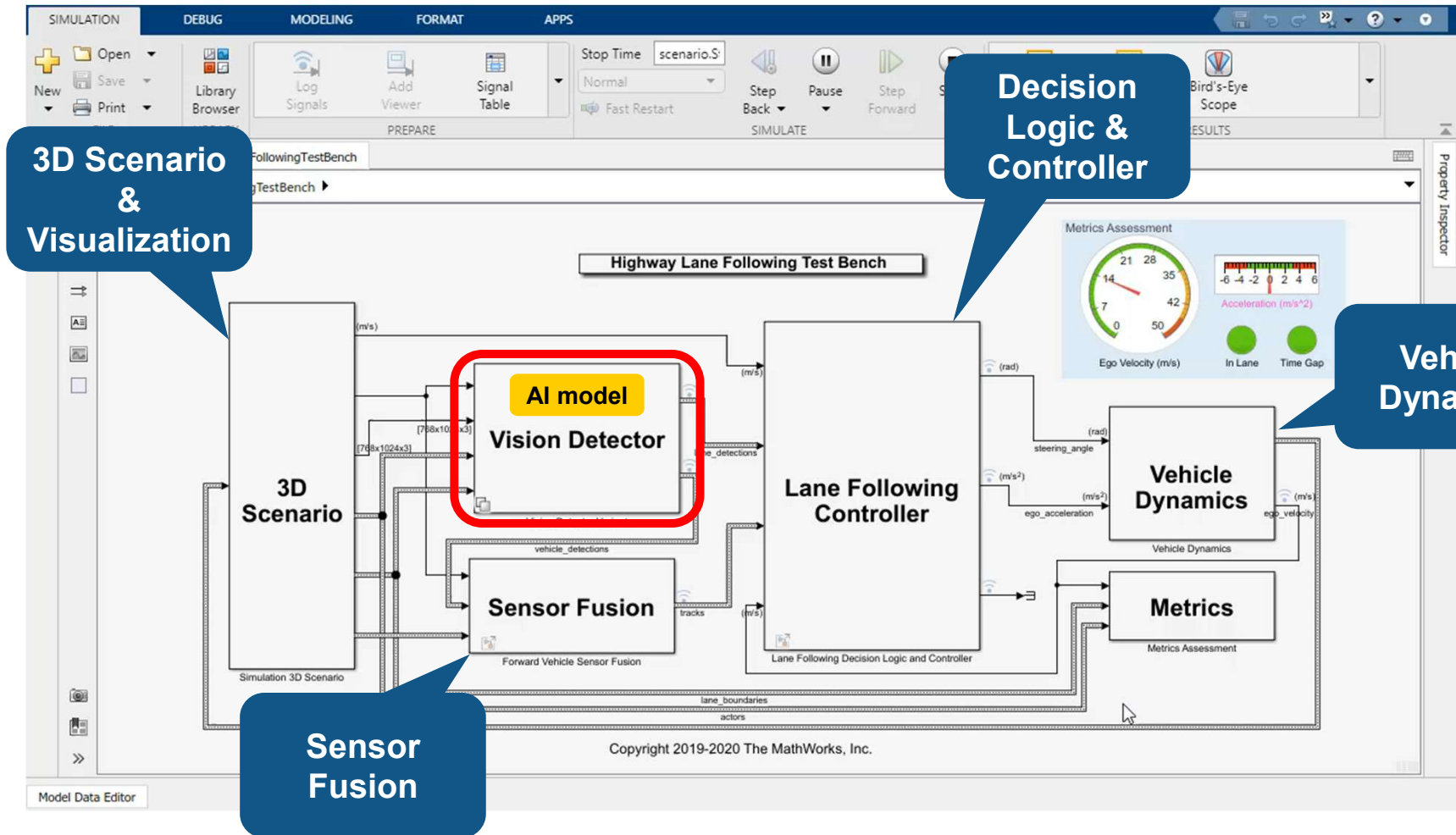


MATLAB Function

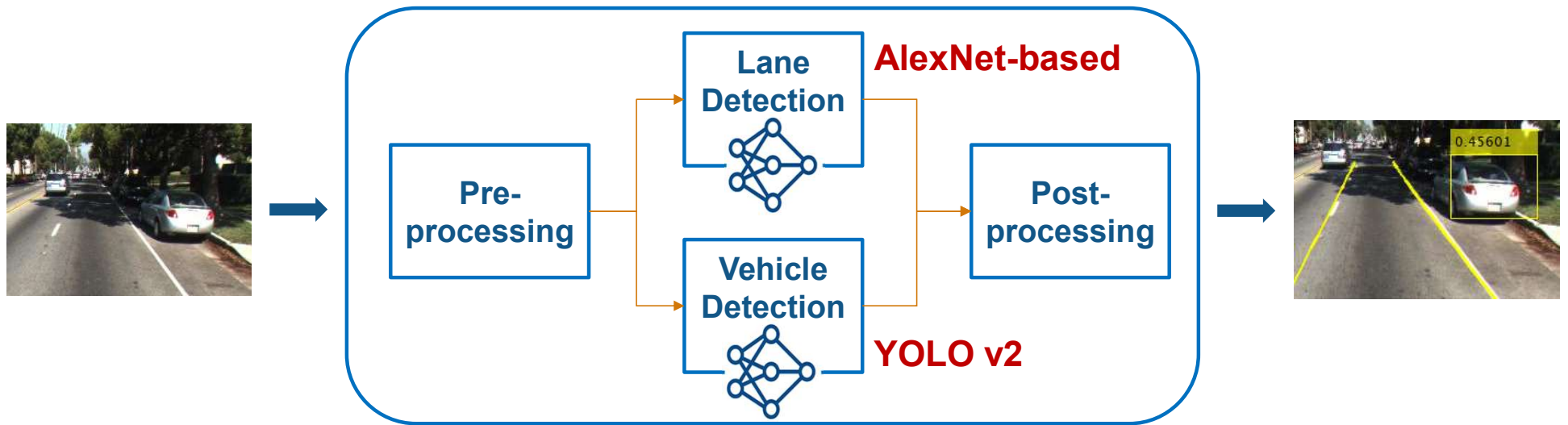
Deep Learning Networks in Simulink



Highway Lane Following Model



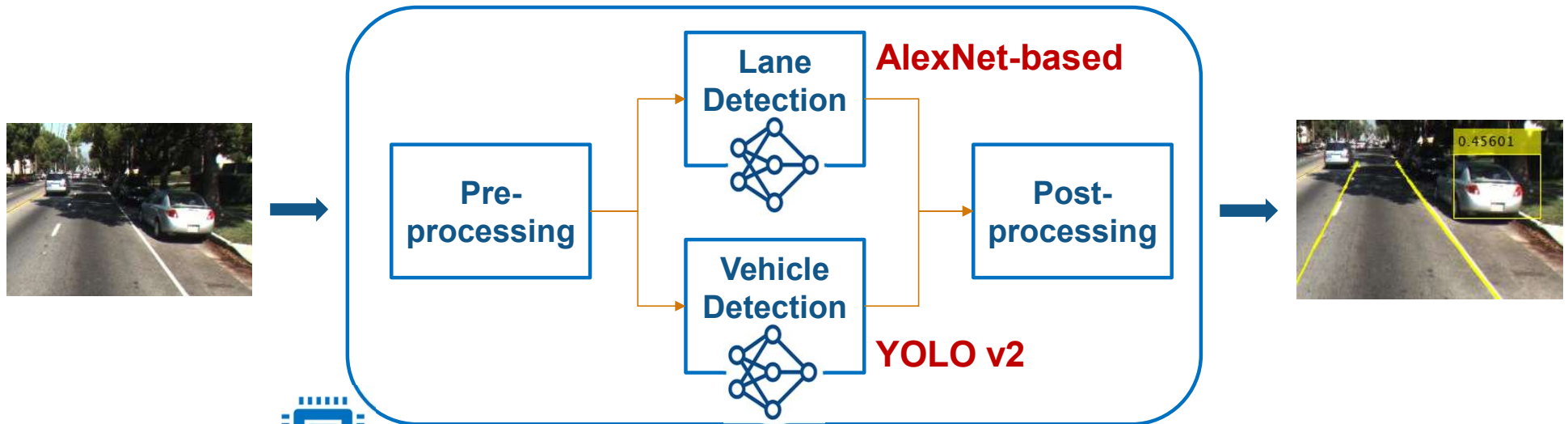
Lane and Vehicle Detection



Workflow:

- 1) Run simulation on desktop CPU
- 2) Run simulation on desktop GPU and generate CUDA code
- 3) Generate CUDA code and run on Jetson AGX Xavier

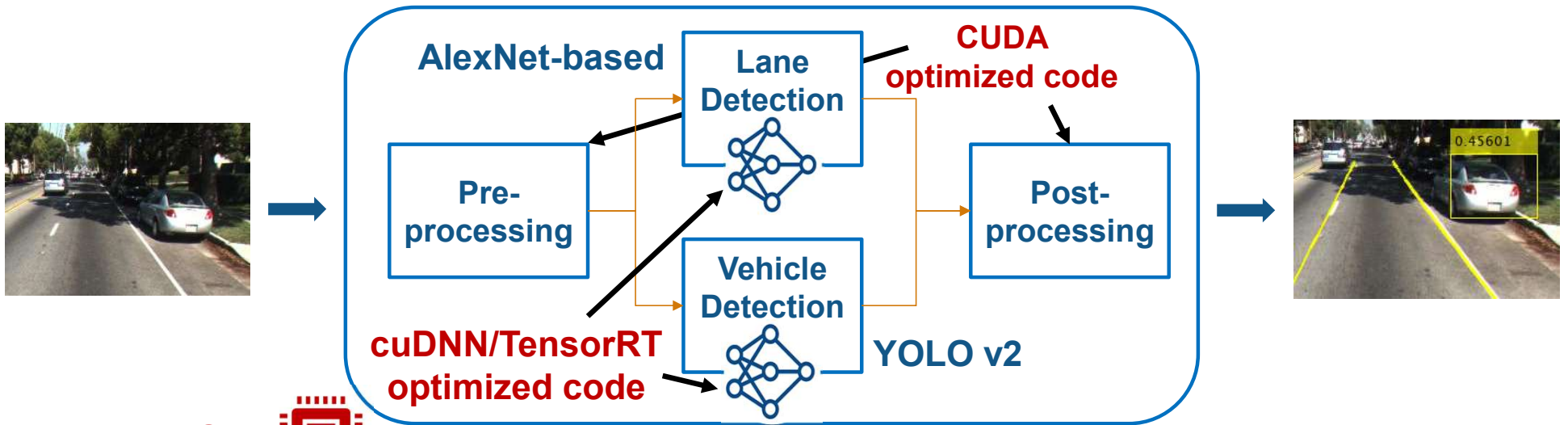
(1) Run Simulation on Desktop CPU



Intel CPU 



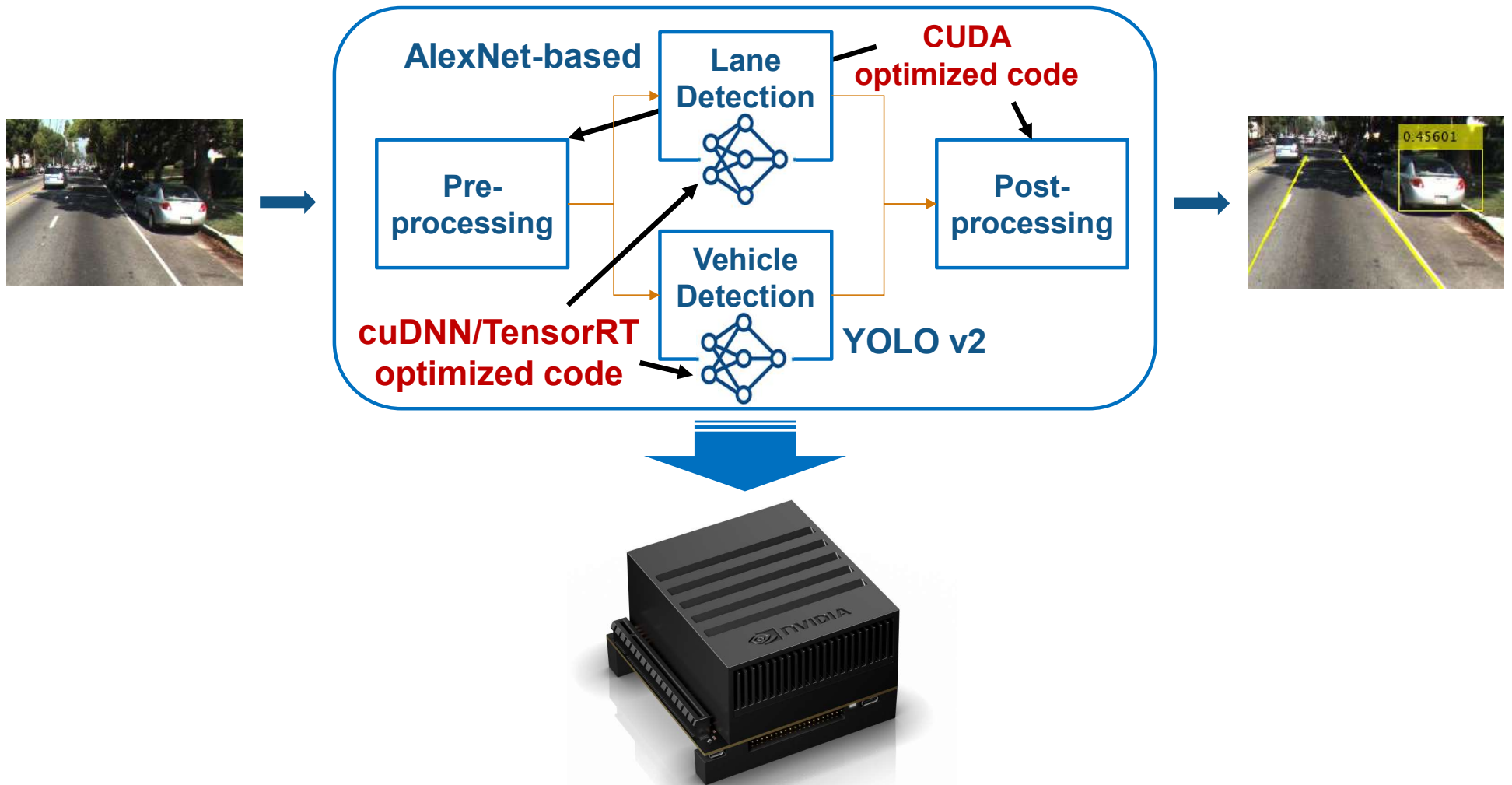
(2) Run Simulation on Desktop GPU and Generate CUDA code



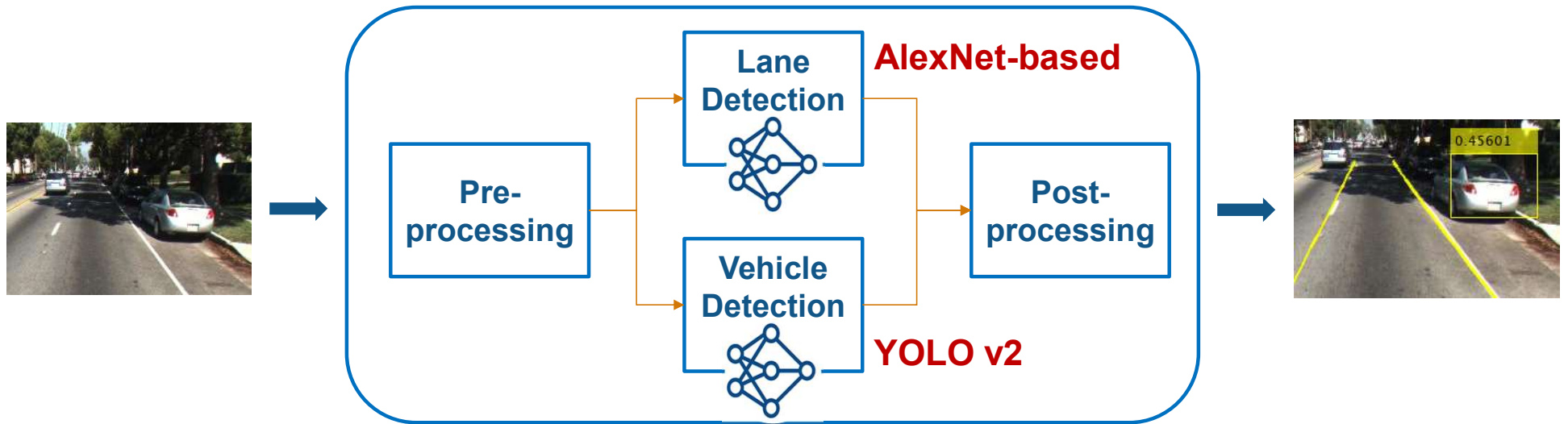
NVIDIA GPU 



(3) Generate CUDA Code and Run on Jetson AGX Xavier



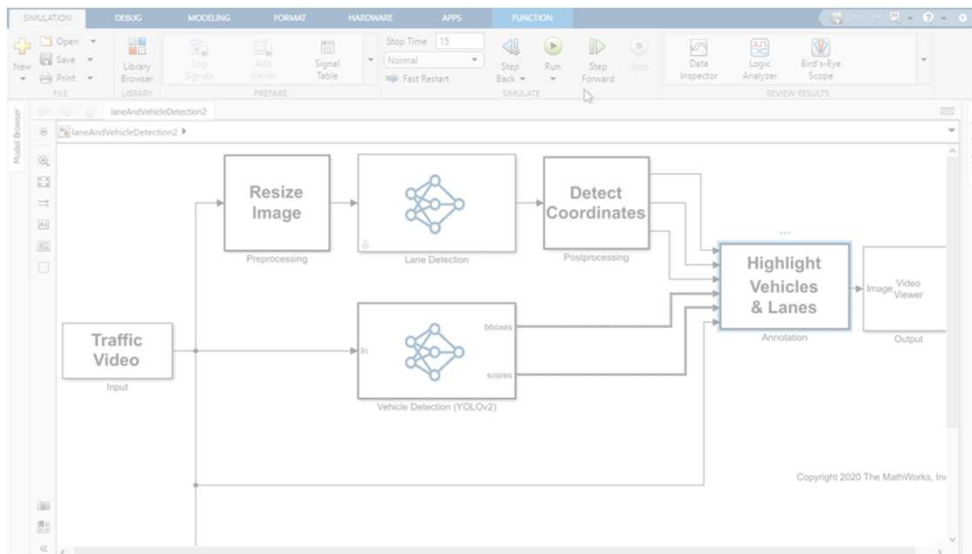
Lane and Vehicle Detection



- 1) Running on CPU & GPU
- 2) ~7X faster running generate code on desktop GPU vs CPU
- 3) Generate CUDA code and run on Jetson AGX Xavier

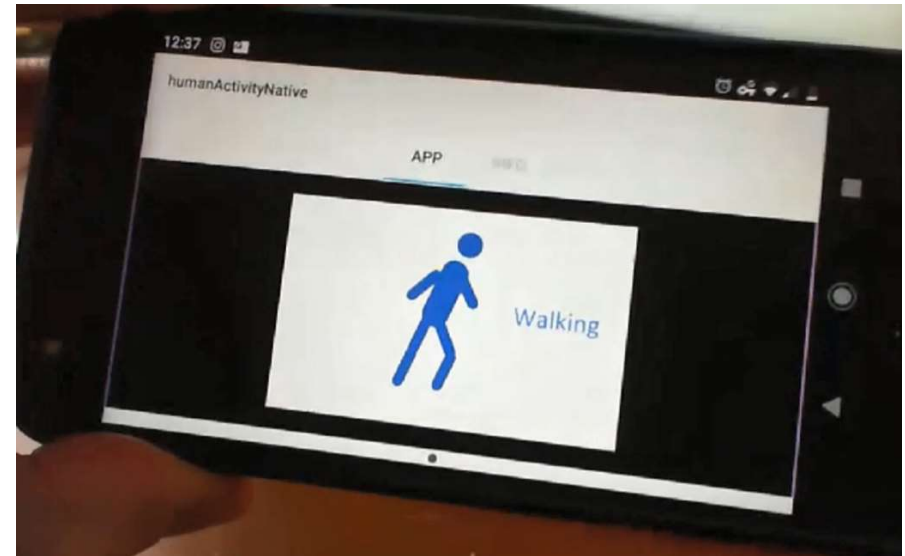
Today's Objective: How to Build AI Functionality into your Systems

Deep Learning in Simulink



Demo: Lane and vehicle detection

Traditional Machine Learning in Simulink



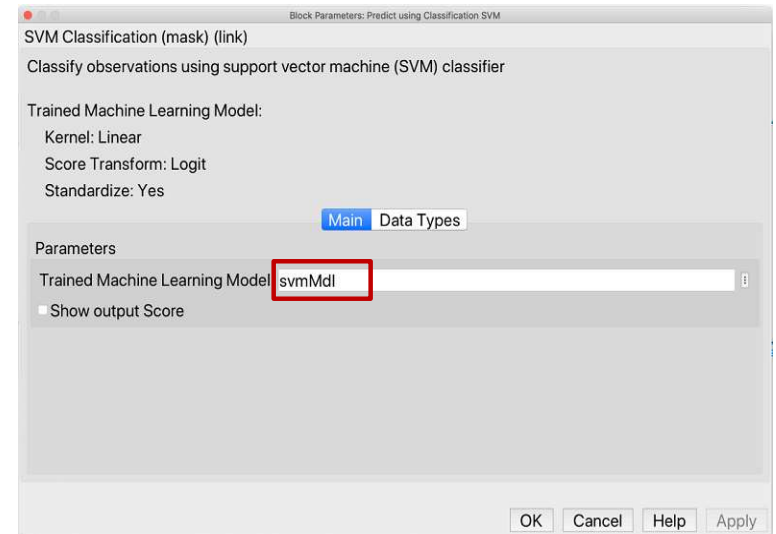
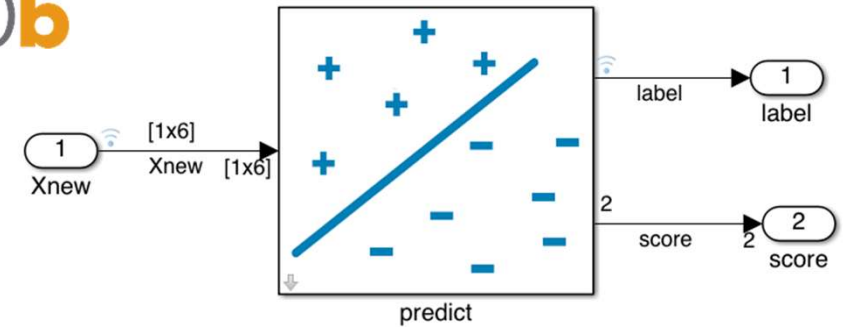
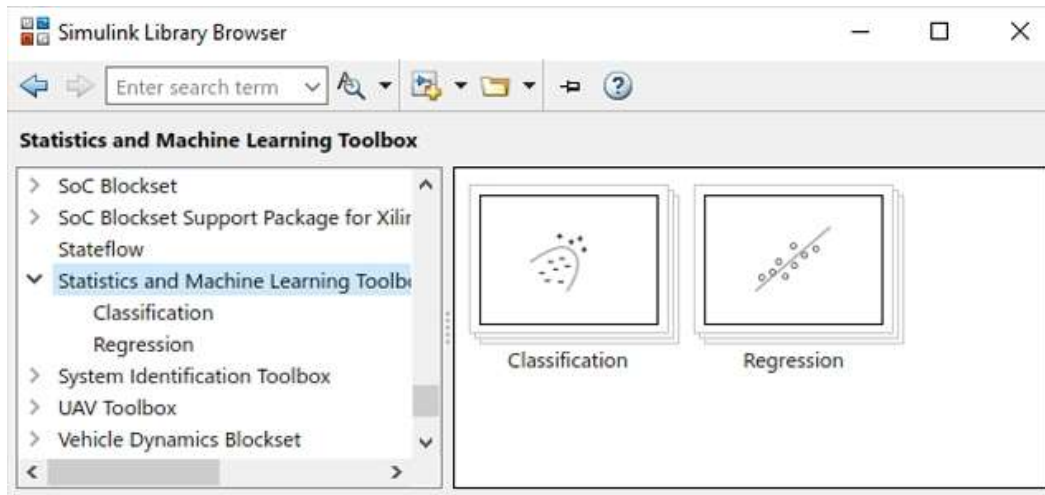
Demo: Human activity recognition

What to use? Deep Learning vs. Machine Learning

	Deep Learning	Machine Learning
Popular among Practitioners:	Convolutional Neural Network (CNN)	Linear Models - Decision Trees
	Long-Short Term Memory (LSTM)	Support Vector Machines
	Generic Adversarial Network (GAN)	Gaussian Process Regression
Types of data:	Images / Video Signal - Text	Sensor Numeric
Requirements:	Lots of (labelled) data	Moderate amounts of data
	Performance computing / GPU	

How to Integrate Machine Learning?

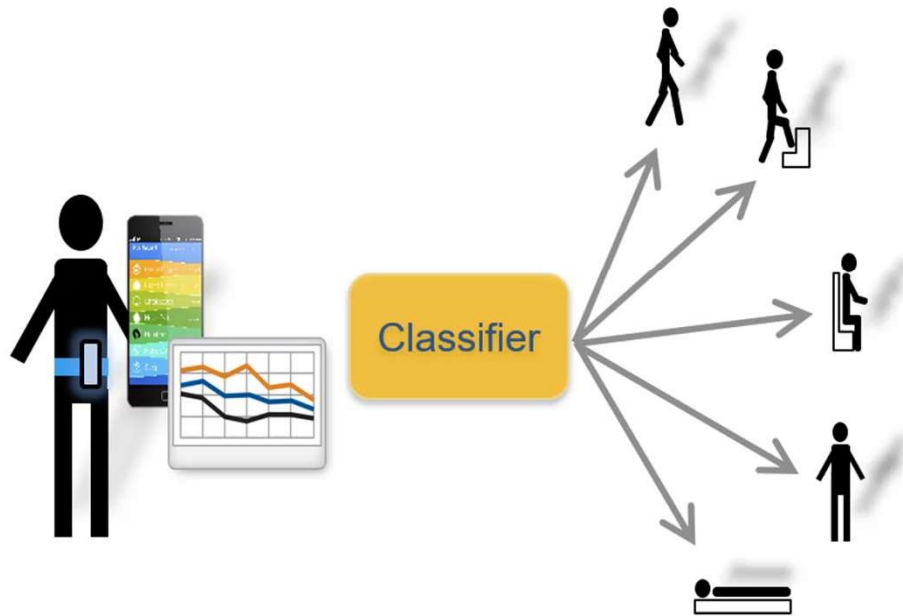
Built-in Machine Learning blocks **R2020b**



MATLAB Function Blocks

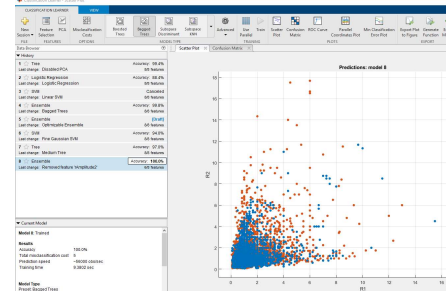
- Preprocessing
- Feature Extraction

Human Activity Learning using Smartphones

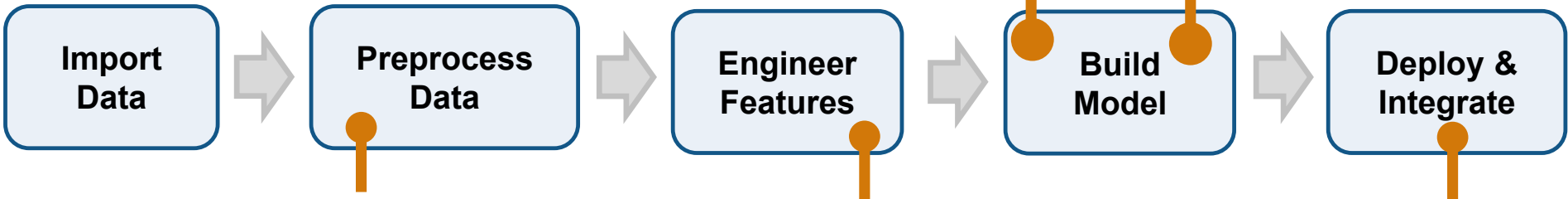
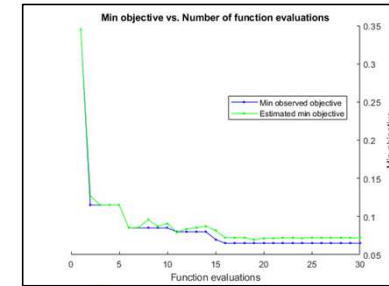


Machine Learning Workflow and Tools

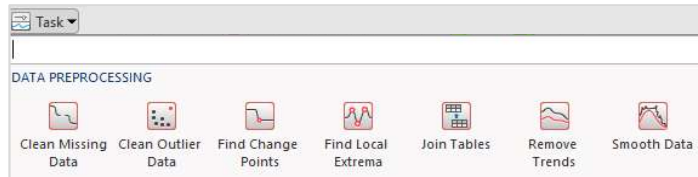
Interactive Model Training



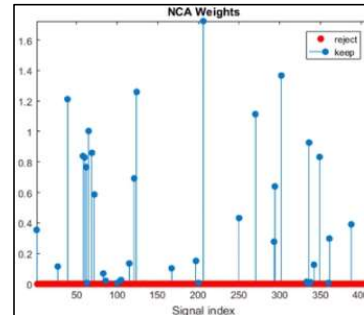
Automated Model Optimization



Clean Messy Raw Data



Feature Selection



Automated Code Generation

MATLAB code

```

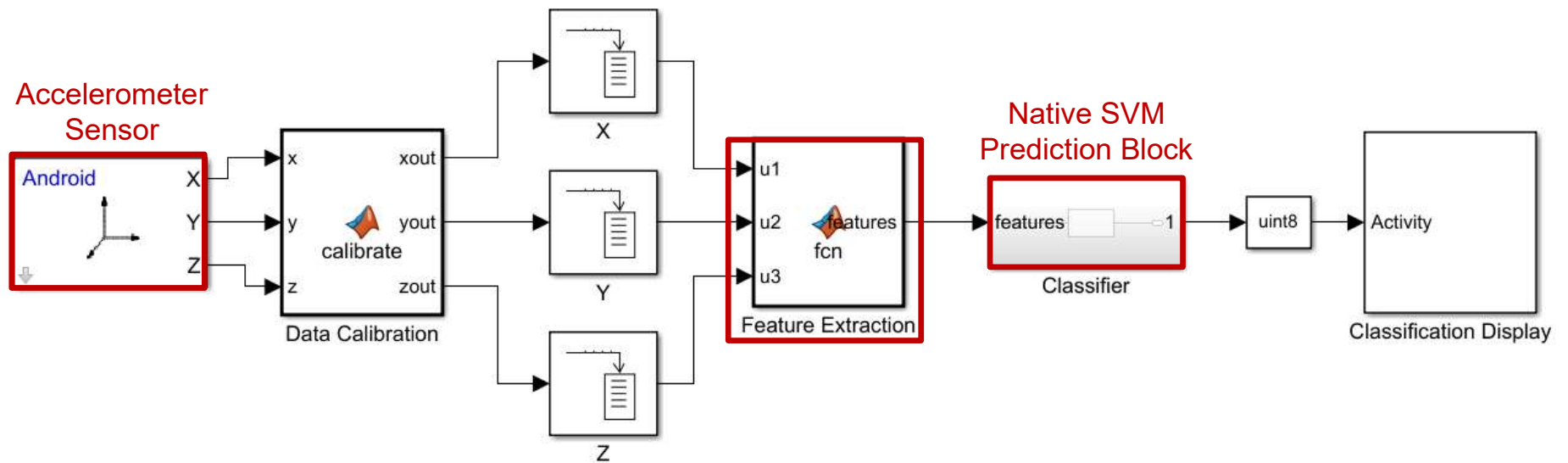
function label = classifyIonosphere(X) %codegen
%classfytionosphere Classify Ionosphere based on pre-trained SVM model
mdl = loadLearnerForCoder('SVMIonosphere');
label = predict(mdl, X);
end
  
```

C code

```

/* Define 'predict' */
16 static emrInfo emrPRD = { 4, /* lcolNo */
16 /* colNo */
16 "C:\Users\johesse\Documents\temp\feature"
16 };
19
20 /* Function Definitions */
21 void classifyIonosphere(classifyIonosphereStep
22 const real_T X[11394], cell_wrap_0_label[155]
23 {
24 real_T to_alpha[90];
25 real_T exp1_temp[14];
  
```

Demo: Human Activity Recognition



Key Takeaway: Increased performance and functionality

Integrate AI into Simulink models for Complex Systems

- Test overall design in simulation
- Implement using code generation

Build AI models using Interactive Apps and Examples

- in Deep Learning / Statistics and Machine Learning Toolboxes
- ... Or integrate models developed by your colleagues