MATLAB EXPO
FRANCE

8 octobre 2024 | Paris

Master Class : IA de confiance pour les systèmes critiques

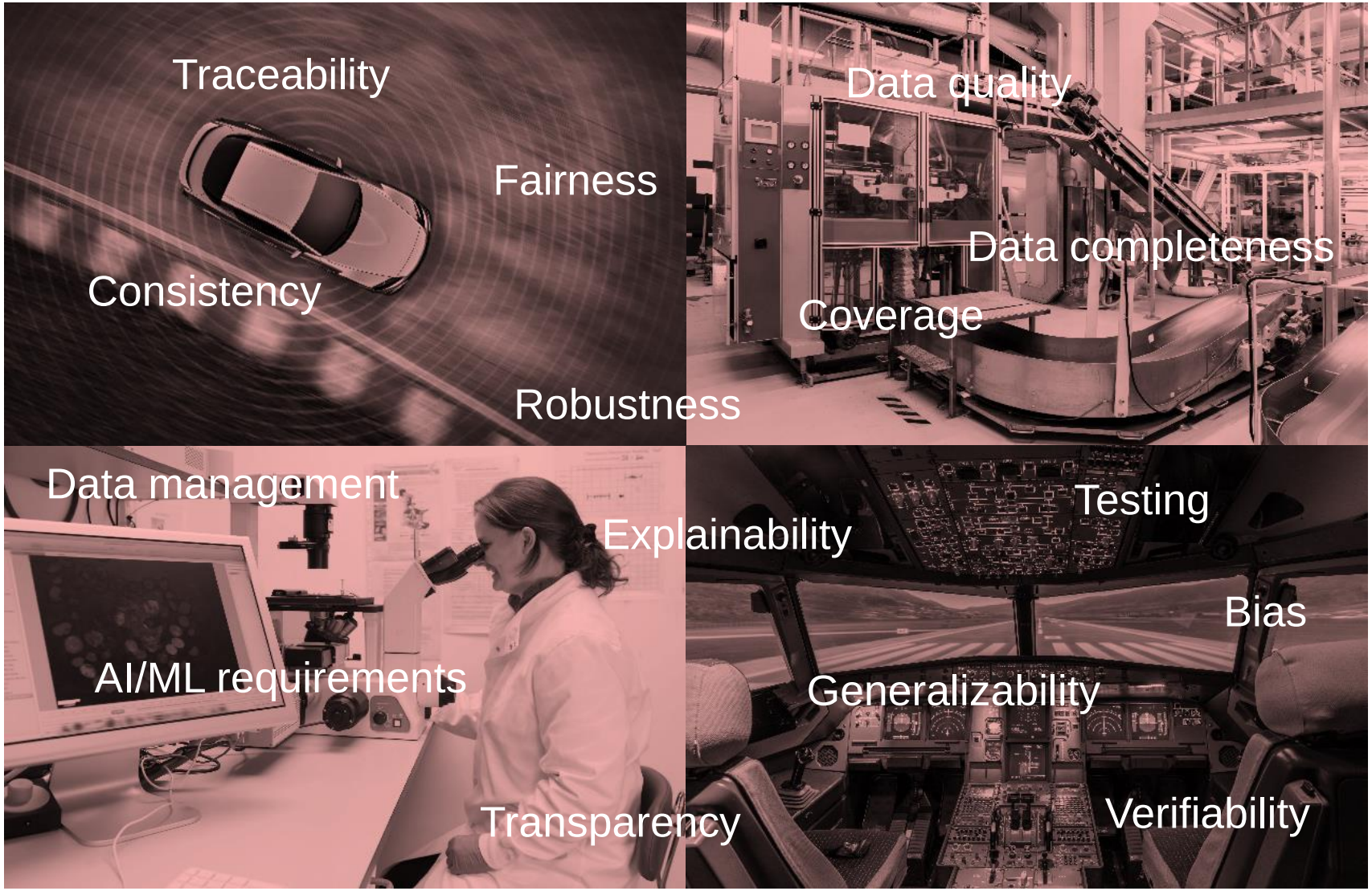
Moubarak Gado, MathWorks

Application Engineer

MathWorks

As AI use rises in production, there is a growing need to explain, verify and validate model behavior in safety-critical situations

Challenges in Verification and Validation of AI-enabled Systems



Critical System Certification

What is Certification?

Formal guarantee that the system meets **safety** and reliability standards

Software Certification standards

Ensure that the software development follows **state-of-the-art processes**

examples

Aviation

DO-178C

Automotive

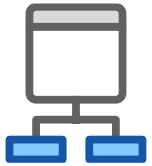
ISO 26262

Medical Devices

EC 62304

Railway:

EN 50128



What do standards cover?

Verification & Validation:
ensure systems meet **requirements**

Lifecycle management:
from design to maintenance

Criticality classification:
failure impact-based

Traceability & documentation & testing

Industries are making progress on verifying AI in systems through whitepapers, standards and planning

Automotive

← ICS ← 43 ← 43.040 ← 43.040.10

ISO/CD PAS 8800

Road Vehicles — Safety and artificial intelligence

Abstract

This document defines safety-related properties and risk factors impacting the insufficient performance and malfunctioning behaviour of Artificial Intelligence (AI) within a road vehicle context. It describes a framework that addresses all phases of the development and deployment lifecycle. This includes the derivation of suitable safety requirements on the function, considerations related to data quality and completeness, architectural measures for the control and mitigation of failures, tools used to support AI, verification and validation techniques as well as the evidence required to support an assurance argument for the overall safety of the system.

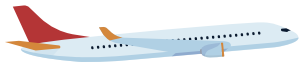
General information

Status : Under development

Edition : 1

Technical Committee : ISO/TC 22/SC 32 Electrical and electronic components and general system aspects

ICS : 43.040.10 Electrical and electronic equipment | 43.040.15 Car informatics. On board computer systems



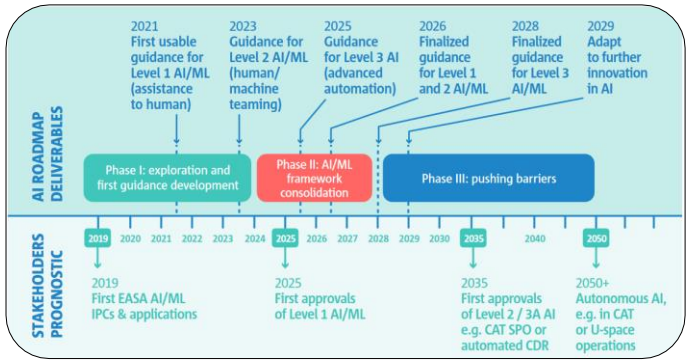
Aerospace

SAE INTERNATIONAL EUROCAE

WIP 2023-06-26

Process Standard for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing AI ARP6983

This document discusses guidelines for the development of Aircraft Systems leveraging AI capabilities, taking into account the overall aircraft operating environment and functions. This includes validation of requirements and verification of the design implementation for certification and product assurance and guidelines with the assessment of safety. It provides practices for showing compliance with the regulations and serves to assist a company in developing and meeting its own internal standards by considering the guidelines herein.



European Aviation Safety Agency AI Roadmap

Medical Devices

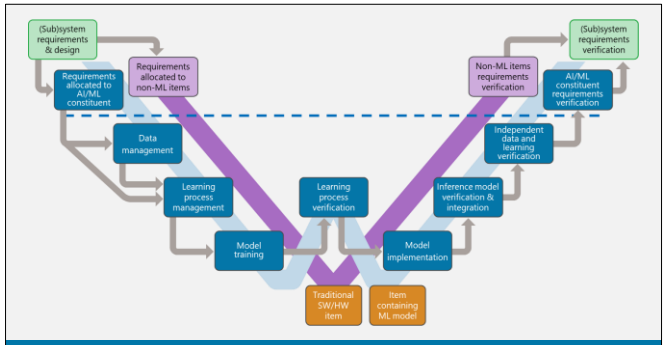
FDA Search Menu

← Software as a Medical Device (SaMD)

Artificial Intelligence and Machine Learning (AI/ML)-Enabled Medical Devices

October 5, 2022 update: 178 Artificial Intelligence and Machine Learning (AI/ML)-Enabled Medical Devices were added to the list below. With this update, the FDA has also added the ability to download the list as an Excel file.

Agenda

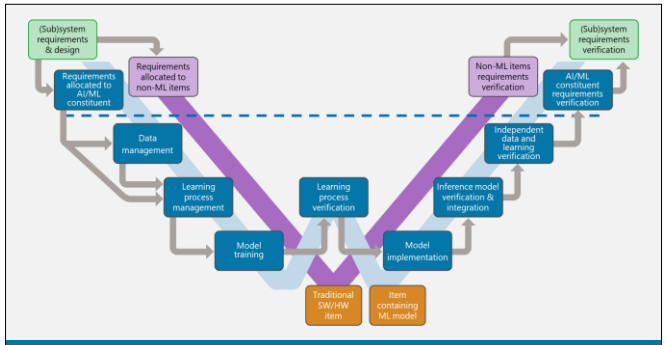


AI Certification in Airborne Systems: Overview

Certifying DAL D A Case Study: Runway Sign Classifier

Towards DAL C

Agenda



AI Certification in Airborne Systems: Overview

Certifying DAL D A Case Study: Runway Sign Classifier

Towards DAL C

Why Certifiable AI in Aviation?

Runway detection

Cockpit voice recognition

Predictive maintenance

Collision avoidance

Single pilot operation

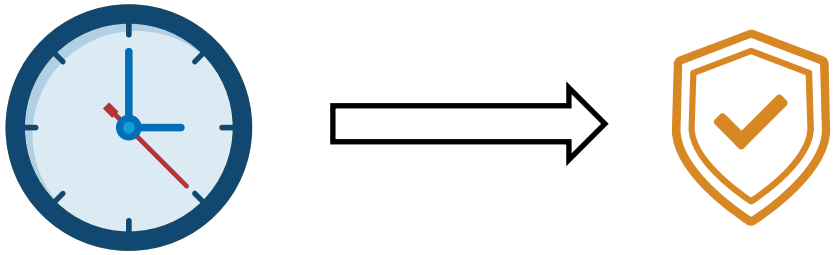
Autonomous flight

What is DO-178C?

DO-178C: Aviation software standard published by the Radio Technical Commission for Aeronautics (RTCA)

- Guidance for software design in aircraft, helicopters, UAVs, and spacecraft.
- Defines **five** software levels based on safety impact.

Level	Failure condition
A	Catastrophic
B	Hazardous
C	Major
D	Minor
E	No Safety Effect



Long Service History
(1982-2023)

Built trust

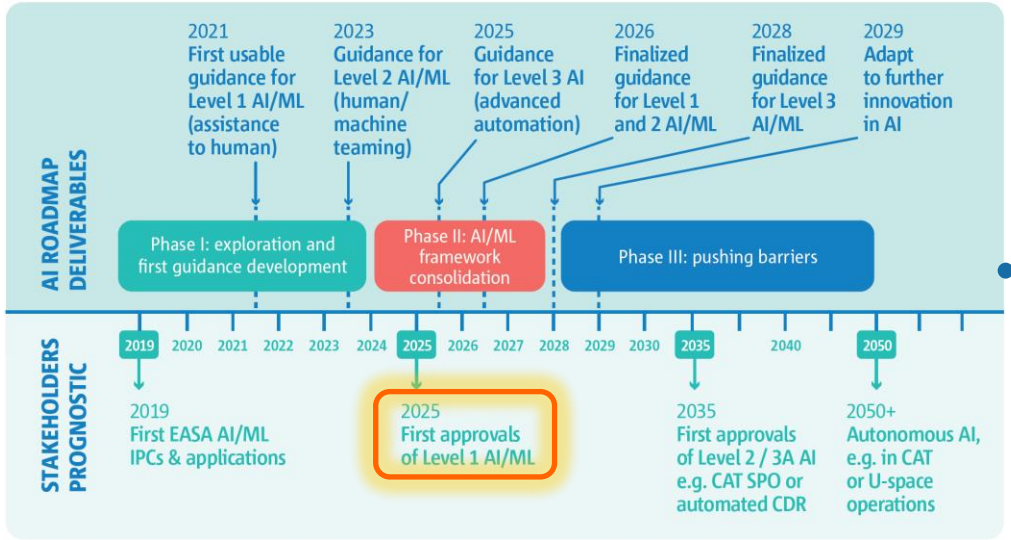
Why is Machine Learning Certification a problem?

Several DO-178C objectives cannot be directly applied to software with a Machine Learning component.

How can we certify Machine Learning component against DO-178C?

How can we build trust in new verification methods for Machine Learning?

Industry and Regulators are making significant progress for Machine Learning Certification in Aerospace



Sources: easa.europa.eu, eurocae.net, faa.gov

MathWorks is playing an active role in working Group (EUROCAE WG-114 / SAE-34) for ARP6983

Established in 2019. Over 500 members

MathWorks

EASA European Aviation Safety Agency

FEDERAL AVIATION ADMINISTRATION

AIRBUS

BOEING

WIP 2023-06-26

Process Standard for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing AI ARP6983

This document discusses guidelines for the development of Aircraft Systems leveraging AI capabilities, taking into account the overall aircraft operating environment and functions. This includes validation of requirements and verification of the design implementation for certification and product assurance and guidelines with the assessment of safety. It provides practices for showing compliance with the regulations and serves to assist a company in developing and meeting its own internal standards by considering the guidelines herein.

In progress

Complementary to DO-178C

Expected to be published

Traditional Software vs Machine Learning Systems

Traditional Software

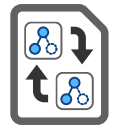
Code defines behavior

Inputs and outputs are clear

Code execution paths are known and fixed

Testing + High code coverage are key to minimize risk of bugs

code

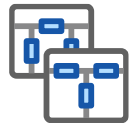


Machine Learning Systems

Rely on Learned parameters

Robustness Dependent on:

Training Data



Learning Algorithms

Generalization

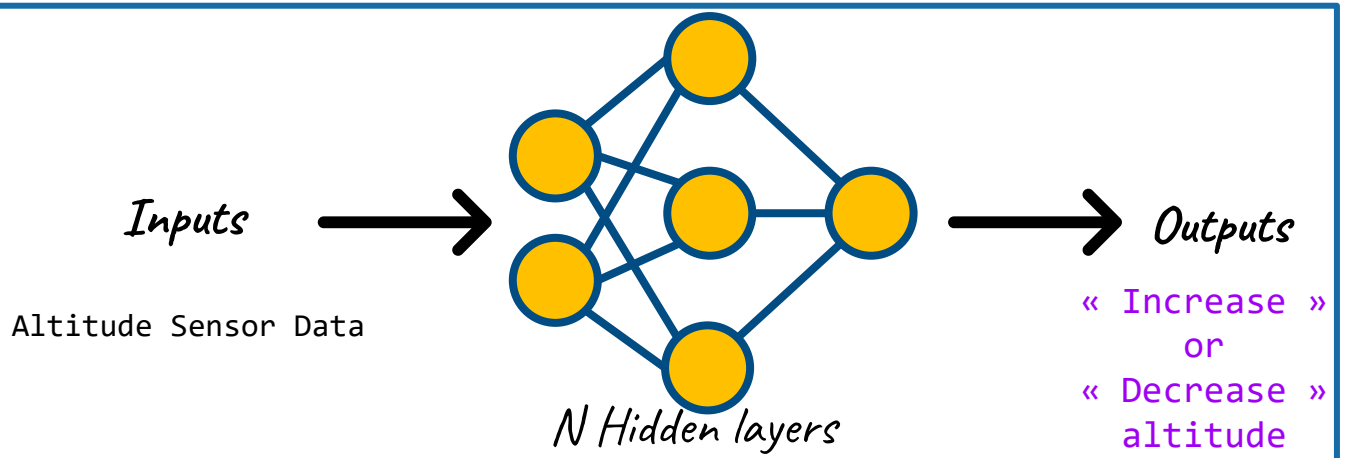
Certification focus Shift

How big is the gap between Machine Learning and traditional software development?

Requirement: *"Maintain altitude change within ±10 feets."*

```
if (altitudeSensor > 10) && (altitudeSensor > sensorValue)
    command = "decrease_altitude";
else
    command = "increase_altitude";
end
```

Rule-based



Data-driven

Explainability

Traceability

Coverage

DO-178C: 15 out of 71 incompatible objectives*

(*) AIR6988 “Artificial intelligence in aeronautical systems. statement of concerns.” EUROCAE, Tech. Rep., 2021

Can we certify Machine Learning now?

Step #1: **DAL D** ML Workflow

- Black-box approach
- Based on existing standards
  - DO-178C, ARP4754, DO-254

Step #2: **DAL C** ML Workflow

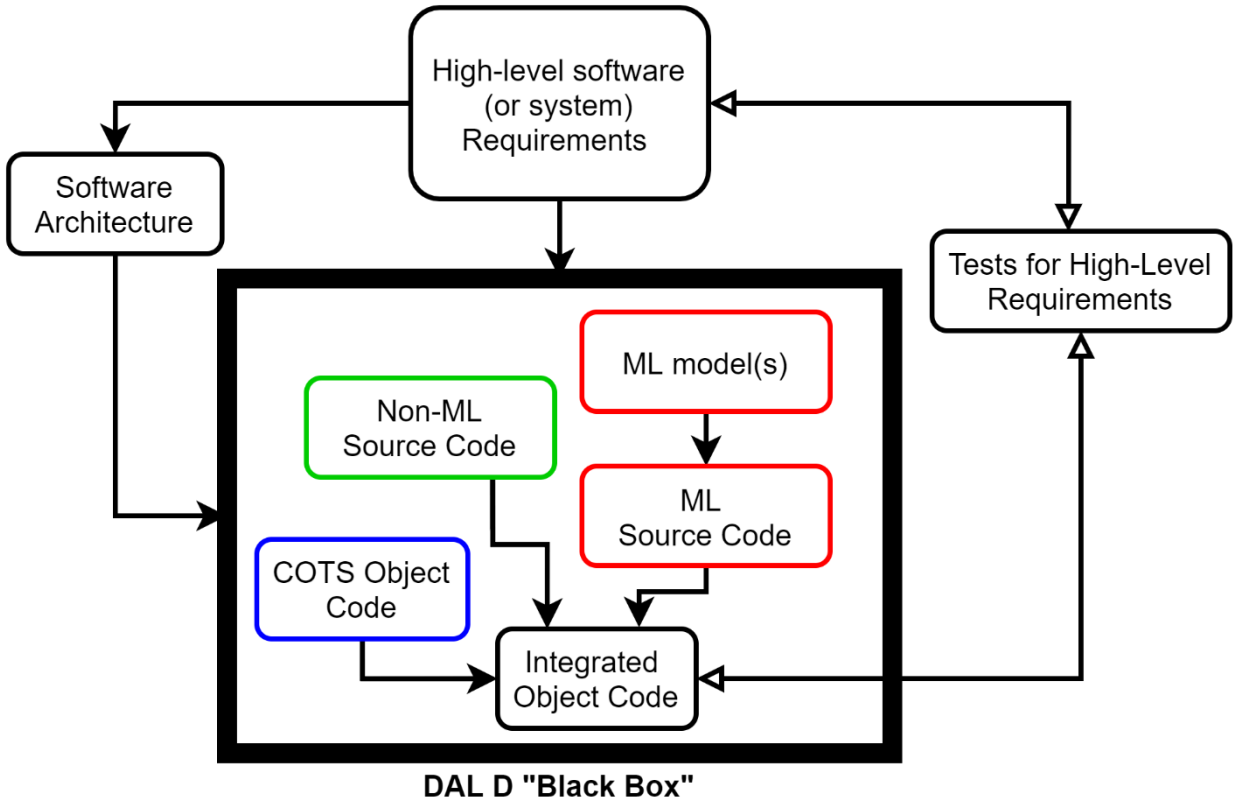
- Existing standards +
- Architectural mitigation
- ML-specific novel VnV

Incremental Certification Approach for Low-Criticality ML Systems*

Failure Category	Software Level (DAL)	DO-178C Objectives
Catastrophic	DAL A	• 71 objectives • 30 require independence
Hazardous	DAL B	• 69 objectives • 18 require independence
Major	DAL C Step #2	**62 objectives** 5 require independence
Minor	DAL D Step #1	**26 objectives** 2 require independence
No Safety Effect	E	• No required objectives

* K. Dmitriev, J. Schumann and F. Holzapfel, "Toward Certification of Machine-Learning Systems for Low Criticality Airborne Applications," *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, San Antonio, TX, USA, 2021

DAL D Black Box Approach

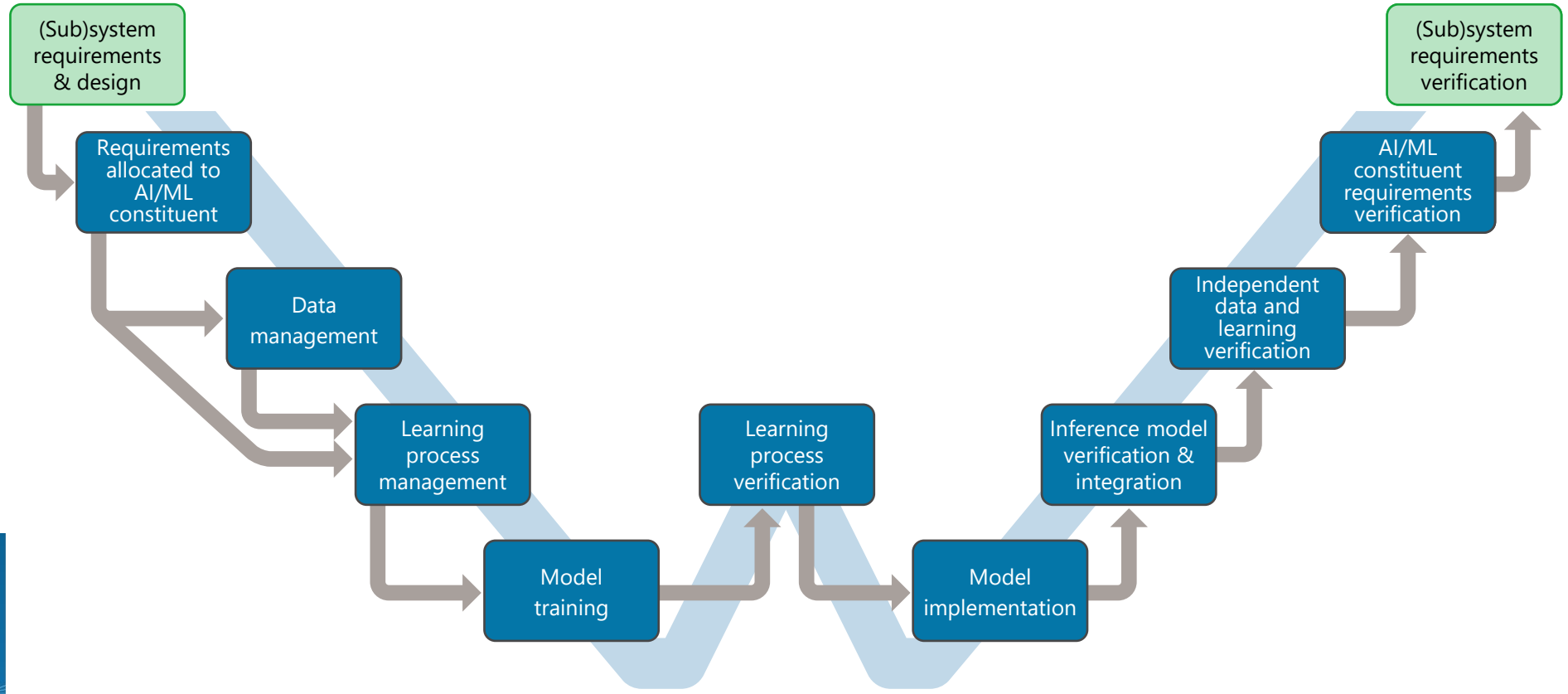


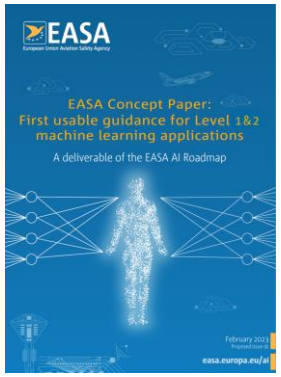
- DAL D: the origin of the source and object code does not matter
- Black-box verification satisfies all objectives

DO-178C Compliance Analysis

	DO-178C / DO-331 Objectives	Software Levels A	B	C	D	E	Analysis
1	High-level requirements (HLR) are developed.	x	x	x	x		In the proposed ML workflow, a trained NN is provided from system level to software life cycle as *Design Model*, from which source code can be directly developed. In this case, as discussed in DO-331 MB.6.1.3, "the guidance related to high-level requirements should be applied to the requirements from which the model is developed." This objective can be achieved by developing system requirements specifying functional, operational, performance, and other applicable characteristics in traditional (e.g., textual) form using the applicable DO-178C guidelines for high-level requirements.
2	Derived high-level requirements are defined and provided to the system processes, including the system safety assessment.	x	x	x	x		In the same manner as for the previous objectives, to achieve this objective, DO-178C guidelines for derived high-level requirements can be applied to system level requirements (from which then ML model is developed) expressed in traditional form.
3	Software architecture is developed.	x	x	x	x		The software architecture defines the software components and their interfaces to enable appropriate grouping of the software functions. Given that a typical ML component (NN) is composed of simple sequential arithmetic operations, we assume that there is no need for an architectural breakdown of the ML function and it can be arranged as a single ML component. However, this objective still applies to traditional software components, which integrate with ML component and can be achieved using the applicable DO-178C guidelines for software architecture.
4	Low-level Requirements are developed.	x	x	x			This objective does not apply to Level D and is excluded from the analysis.
5	Derived low-level requirements are defined and provided to the system processes, including the system safety assessment.	x	x	x			This objective does not apply to Level D and is excluded from the analysis.
6	Source Code is developed	x	x	x			This objective does not apply to Level D and is excluded from the analysis.
7	Executable Object Code and Parameter Data Item (PDI) Files, if any, are produced and loaded in the target computer.	x	x	x	x		Activities for production and loading of executable object code and PDI files are not different between traditional software and ML components, DO-178C guidelines can be directly applied to achieve this objective.

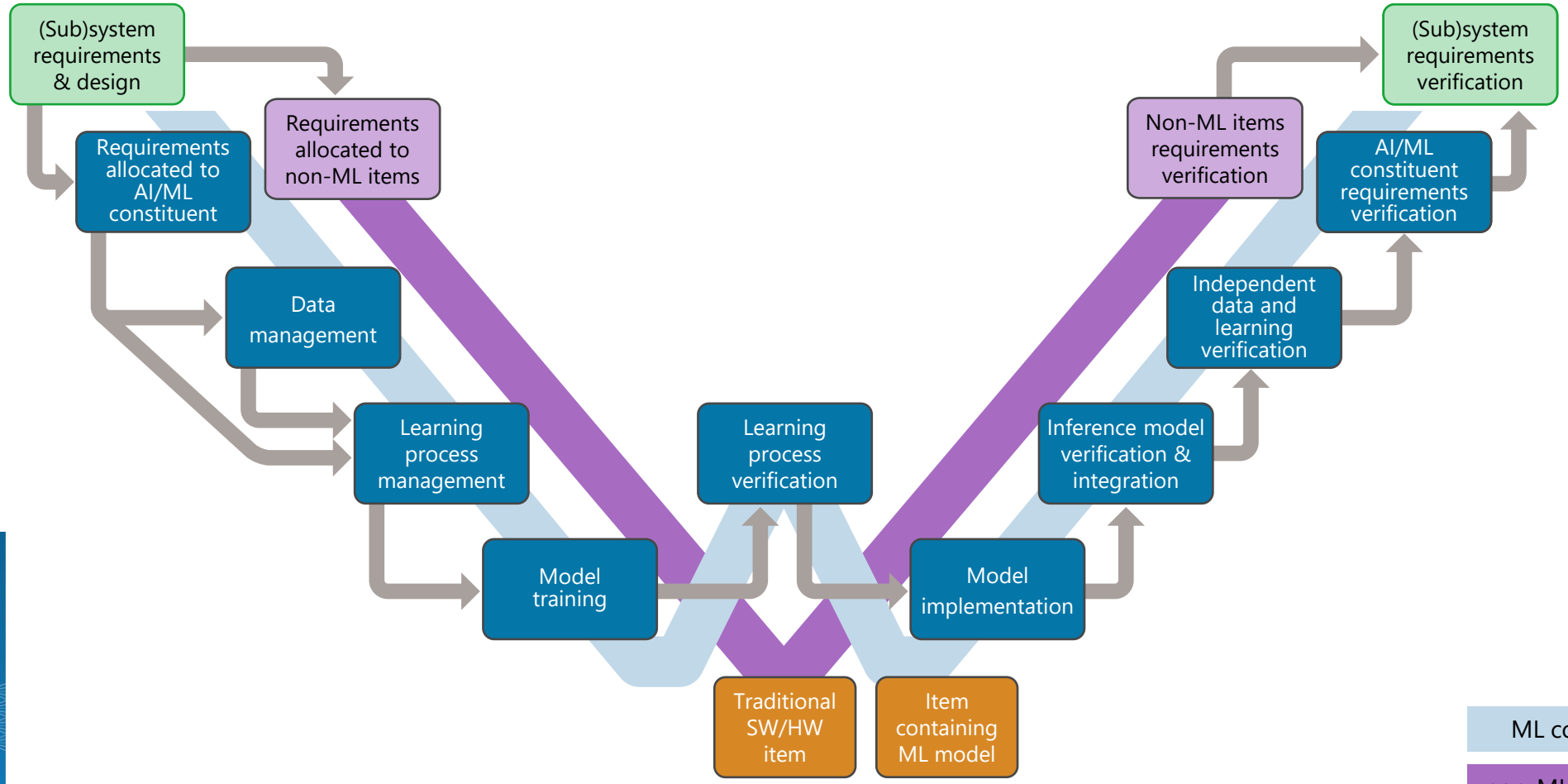
W-shaped development process adapting the classical V-shaped cycle to ML applications

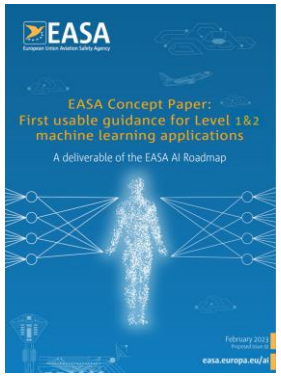




Source: EASA Concept Paper Proposed ISSUE 02 'First usable guidance for Level 1&2 machine learning applications

W-shaped development process can coexist with V-shaped cycle for non-ML components





Source: EASA Concept Paper Proposed ISSUE 02 'First usable guidance for Level 1&2 machine learning applications

Agenda

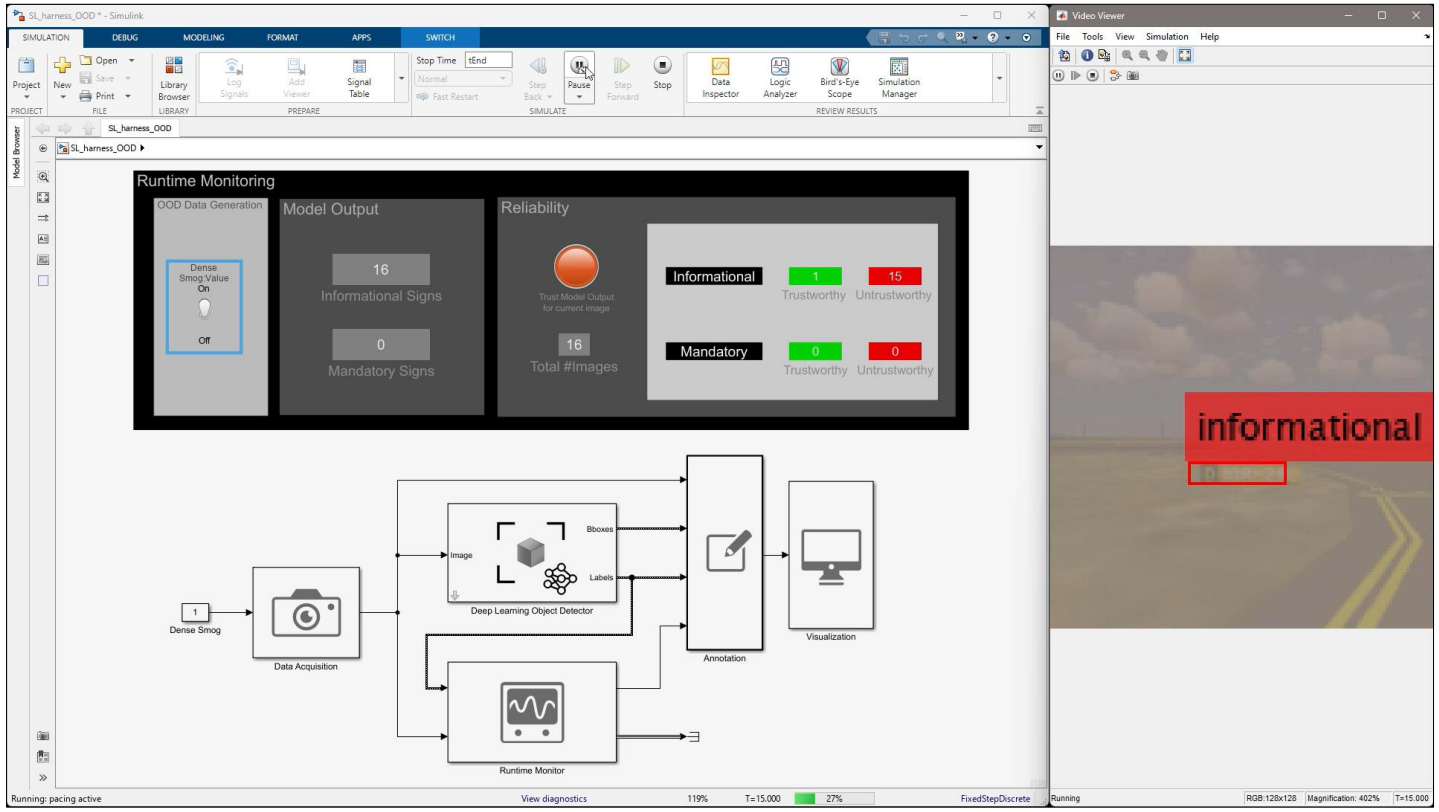
AI Certification in Airborne Systems: Overview

Certifying DAL D A Case Study: Runway Sign Classifier

Towards DAL C

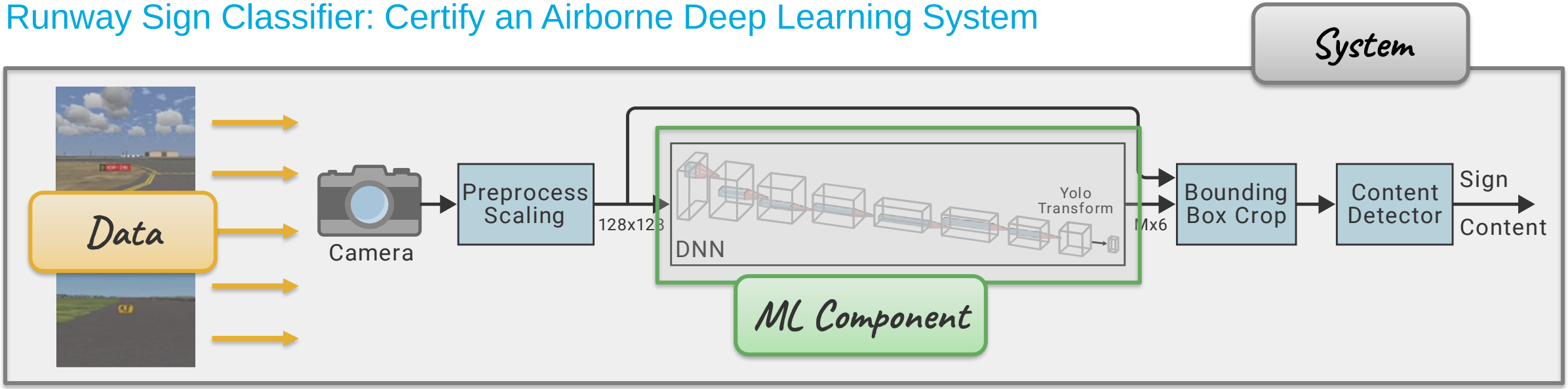
Case Study

Runway Sign Classifier: Certify an Airborne Deep Learning System

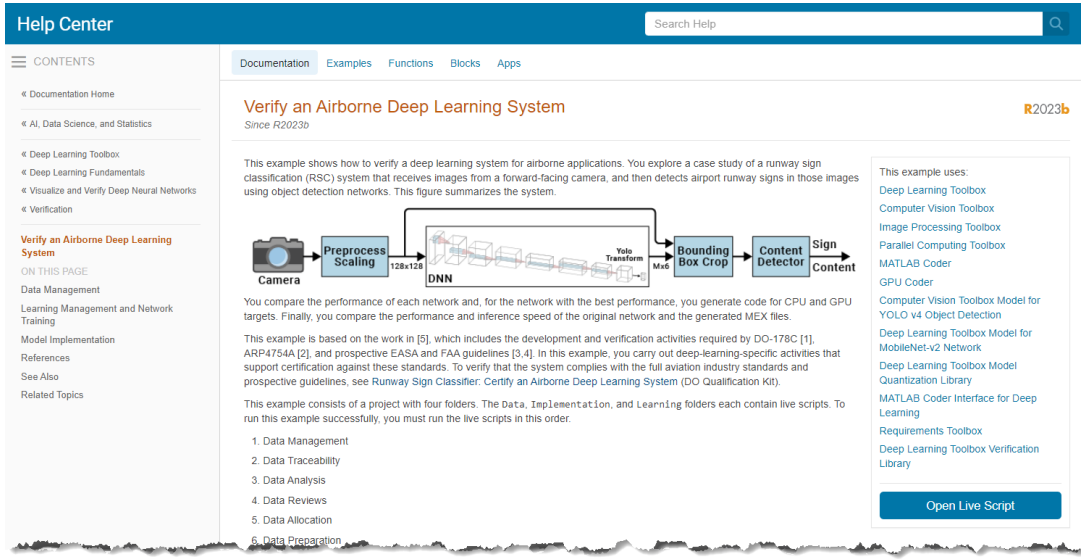


Case Study

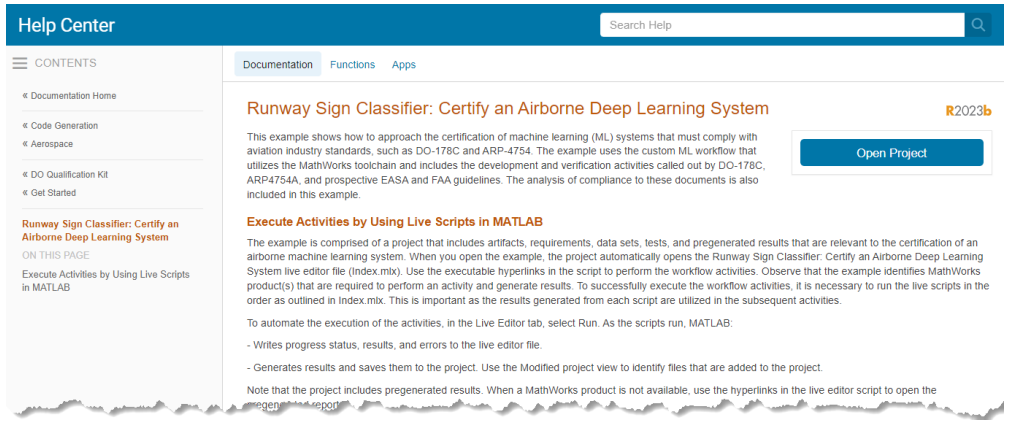
Runway Sign Classifier: Certify an Airborne Deep Learning System



Deep Learning Toolbox Example

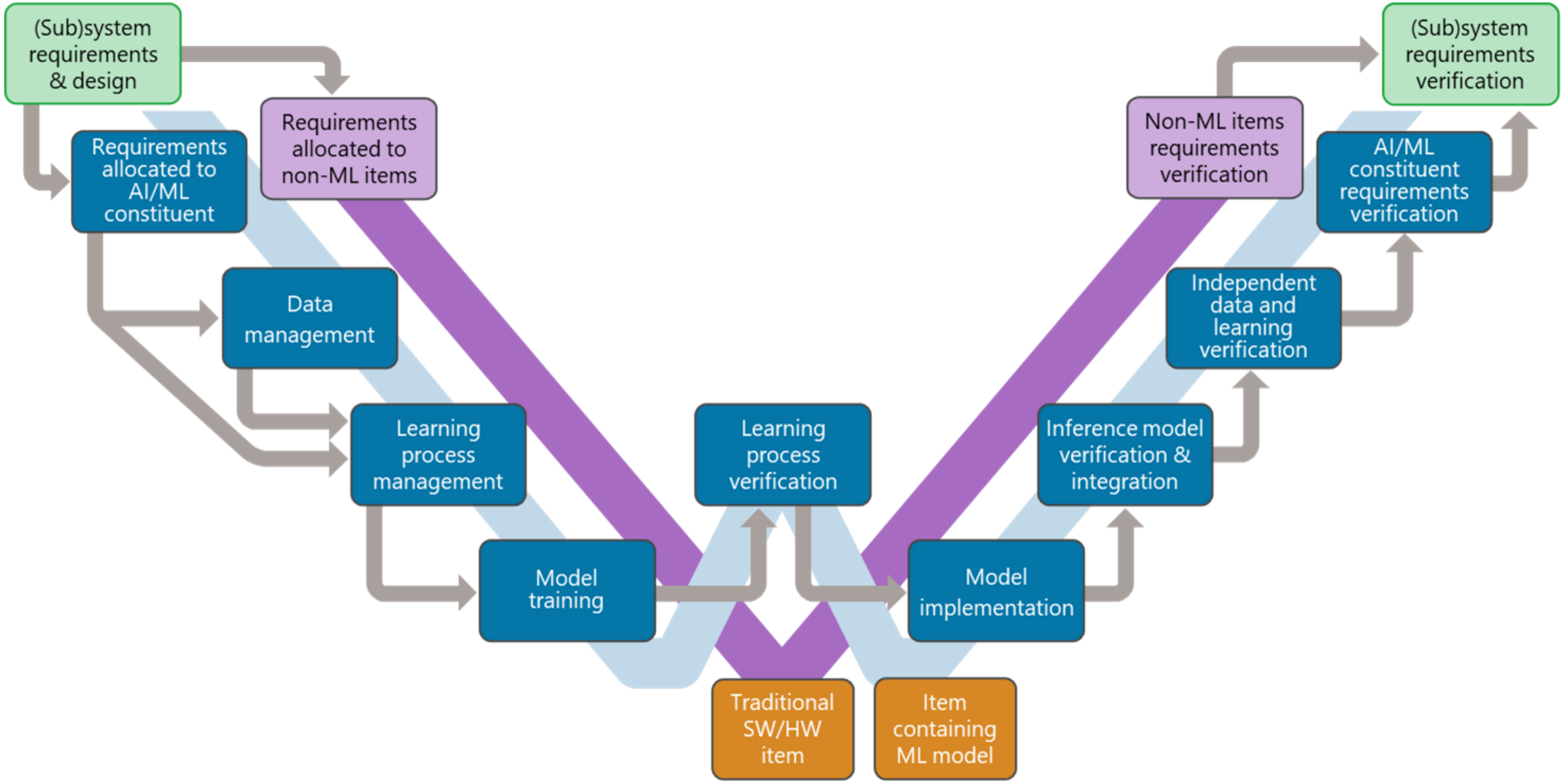


DO Qualification Kit Example

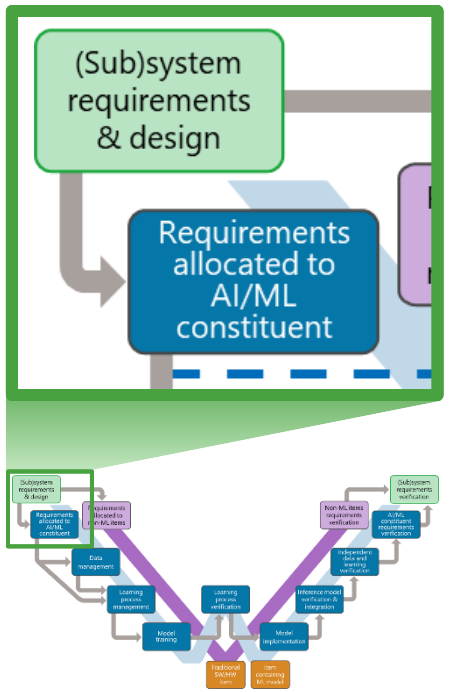


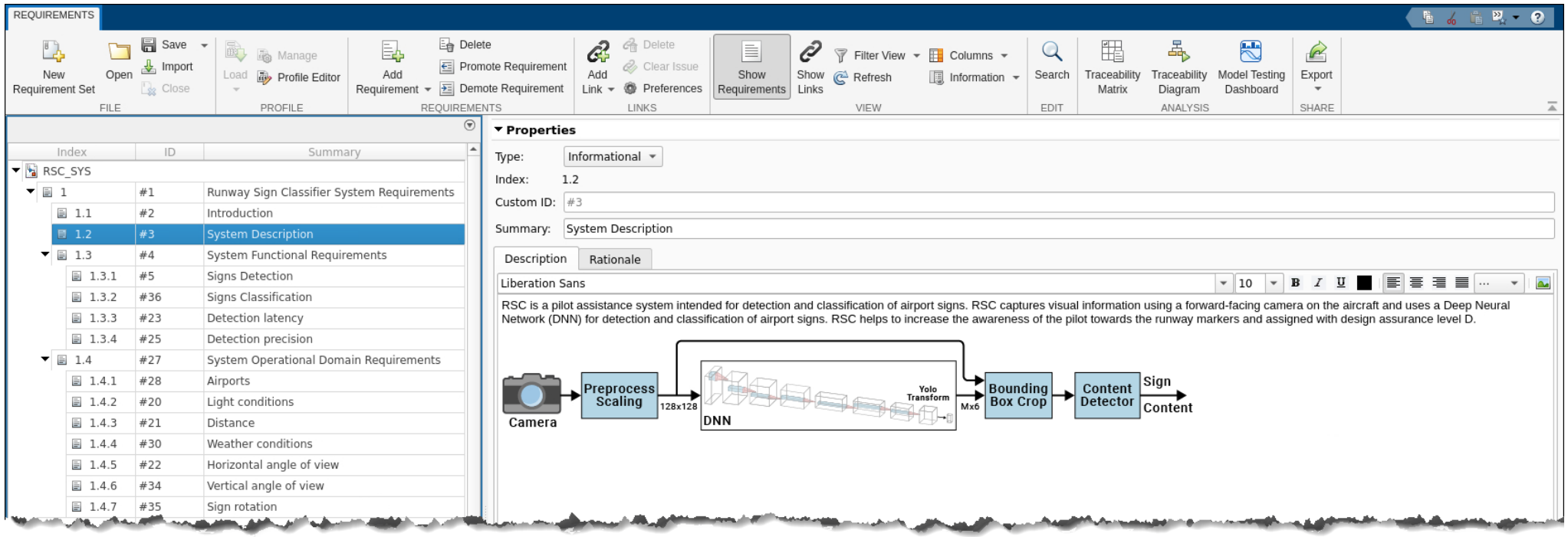
Case Study

Runway Sign Classifier: Certify an Airborne Deep Learning System

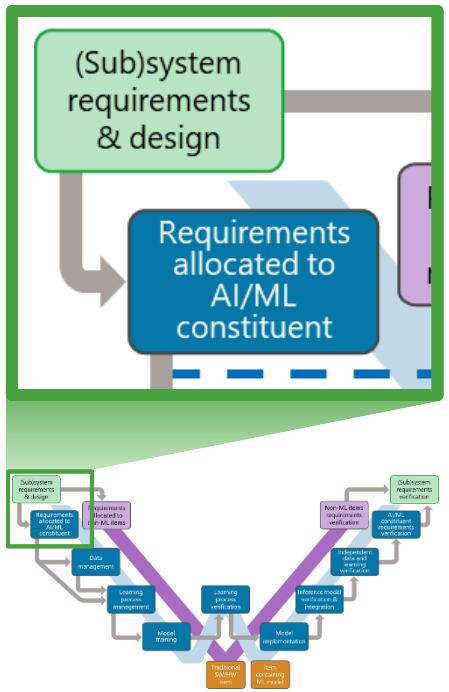


Define system requirements and allocate them to the AI constituent





Define system requirements and allocate them to the AI constituent



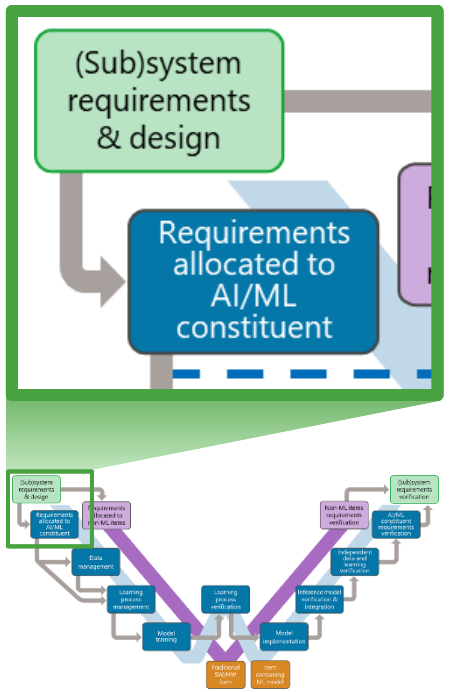
Index	ID	Summary
RSC_SYS		
1	#1	Runway Sign Classifier System Requirements
1.1	#2	Introduction
1.2	#3	System Description
1.3	#4	System Functional Requirements
1.3.1	#5	Signs Detection
1.3.2	#36	Signs Classification
1.3.3	#23	Detection latency
1.3.4	#25	Detection precision
1.4	#27	System Operational Domain Requirements
1.4.1	#28	Airports
1.4.2	#20	Light conditions
1.4.3	#21	Distance
1.4.4	#30	Weather conditions
1.4.5	#22	Horizontal angle of view
1.4.6	#34	Vertical angle of view
1.4.7	#35	Sign rotation
2	#37	Justifications

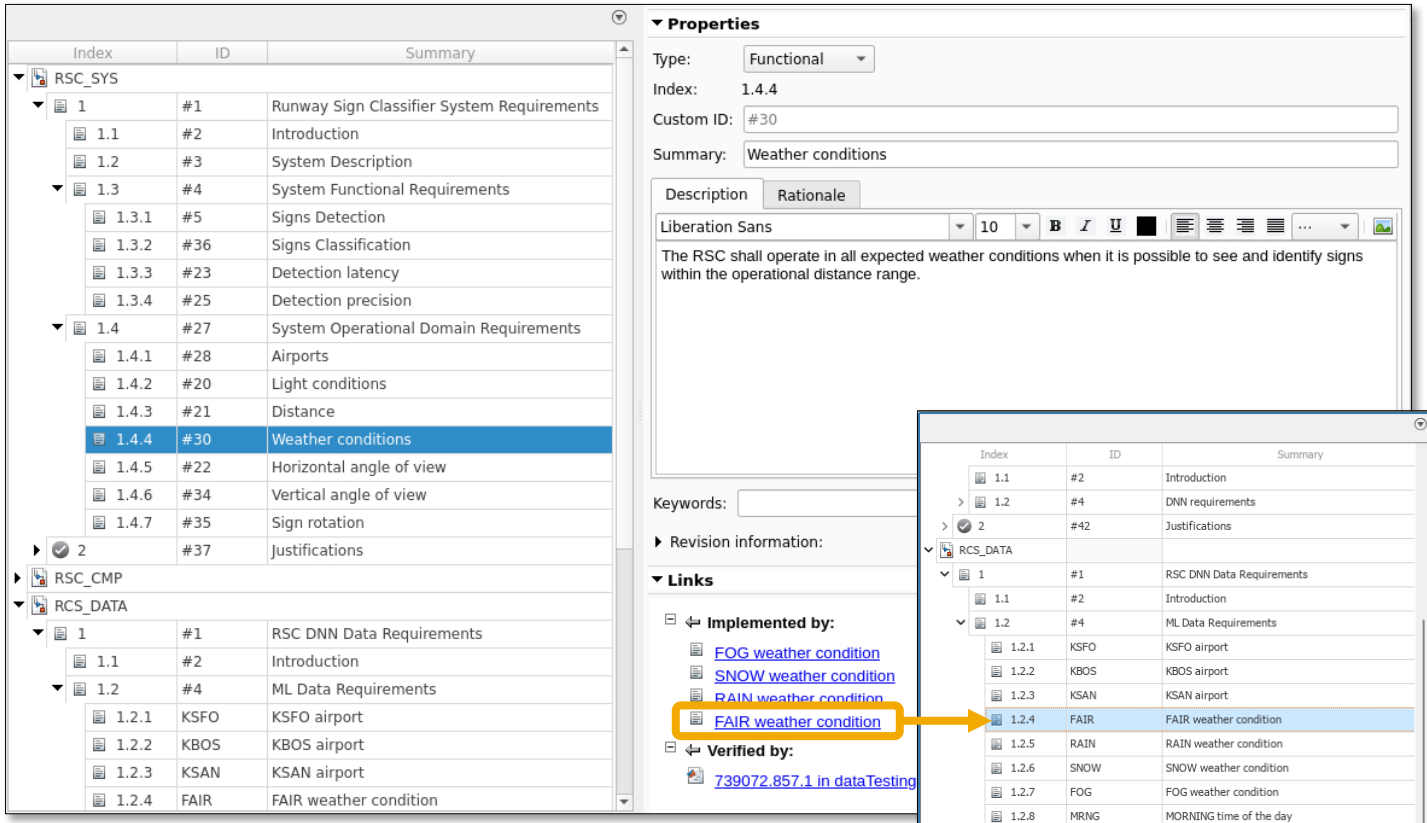
System

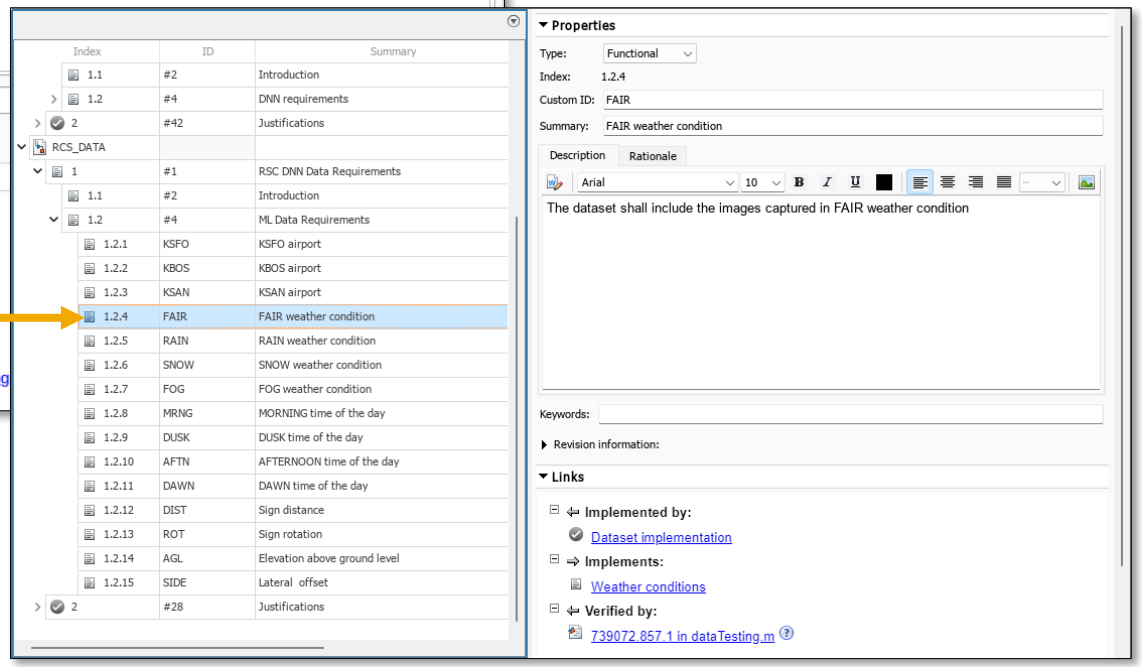
ML Component

Data

Requirements Toolbox allows you link system requirements to data requirements

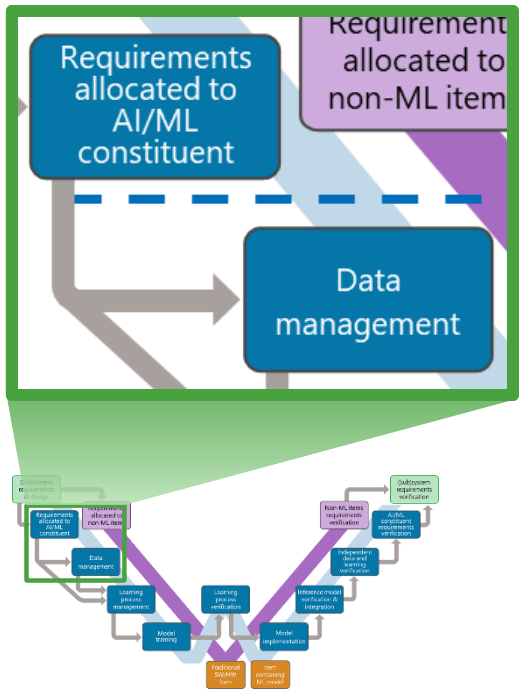






Map data requirements to your MATLAB data and review data in Image Labeler App

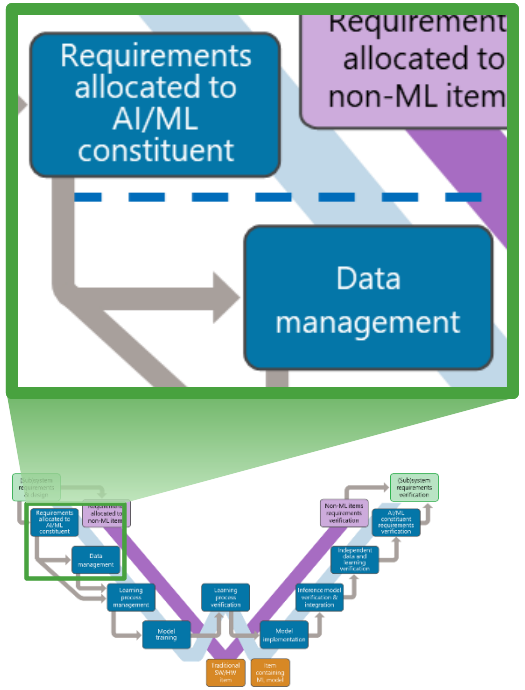
Data Requirement ID	Datastore	Datastore Size	Datastore Link
{'KSFO'}	{1×1 matlab.io.datastore.ImageDatastore}	72	{["Open Datastore"]}
{'KBOS'}	{1×1 matlab.io.datastore.ImageDatastore}	108	{["Open Datastore"]}
{'KSAN'}	{1×1 matlab.io.datastore.ImageDatastore}	24	{["Open Datastore"]}
{'FAIR'}	{1×1 matlab.io.datastore.ImageDatastore}	114	{["Open Datastore"]}
{'RAIN'}	{1×1 matlab.io.datastore.ImageDatastore}	54	{["Open Datastore"]}
{'SNOW'}	{1×1 matlab.io.datastore.ImageDatastore}	24	{["Open Datastore"]}
{'FOG' }	{1×1 matlab.io.datastore.ImageDatastore}	48	{["Open Datastore"]}
{'MRNG'}	{1×1 matlab.io.datastore.ImageDatastore}	78	{["Open Datastore"]}
{'DUSK'}	{1×1 matlab.io.datastore.ImageDatastore}	72	{["Open Datastore"]}
{'AFTN'}	{1×1 matlab.io.datastore.ImageDatastore}	36	{["Open Datastore"]}
{'DAWN'}	{1×1 matlab.io.datastore.ImageDatastore}	90	{["Open Datastore"]}
{'DIST'}	{1×1 matlab.io.datastore.ImageDatastore}	276	{["Open Datastore"]}
{'AGL' }	{1×1 matlab.io.datastore.ImageDatastore}	276	{["Open Datastore"]}
{'SIDE'}	{1×1 matlab.io.datastore.ImageDatastore}	276	{["Open Datastore"]}
{'ROT' }	{0×0 double }	0	{0×0 double }

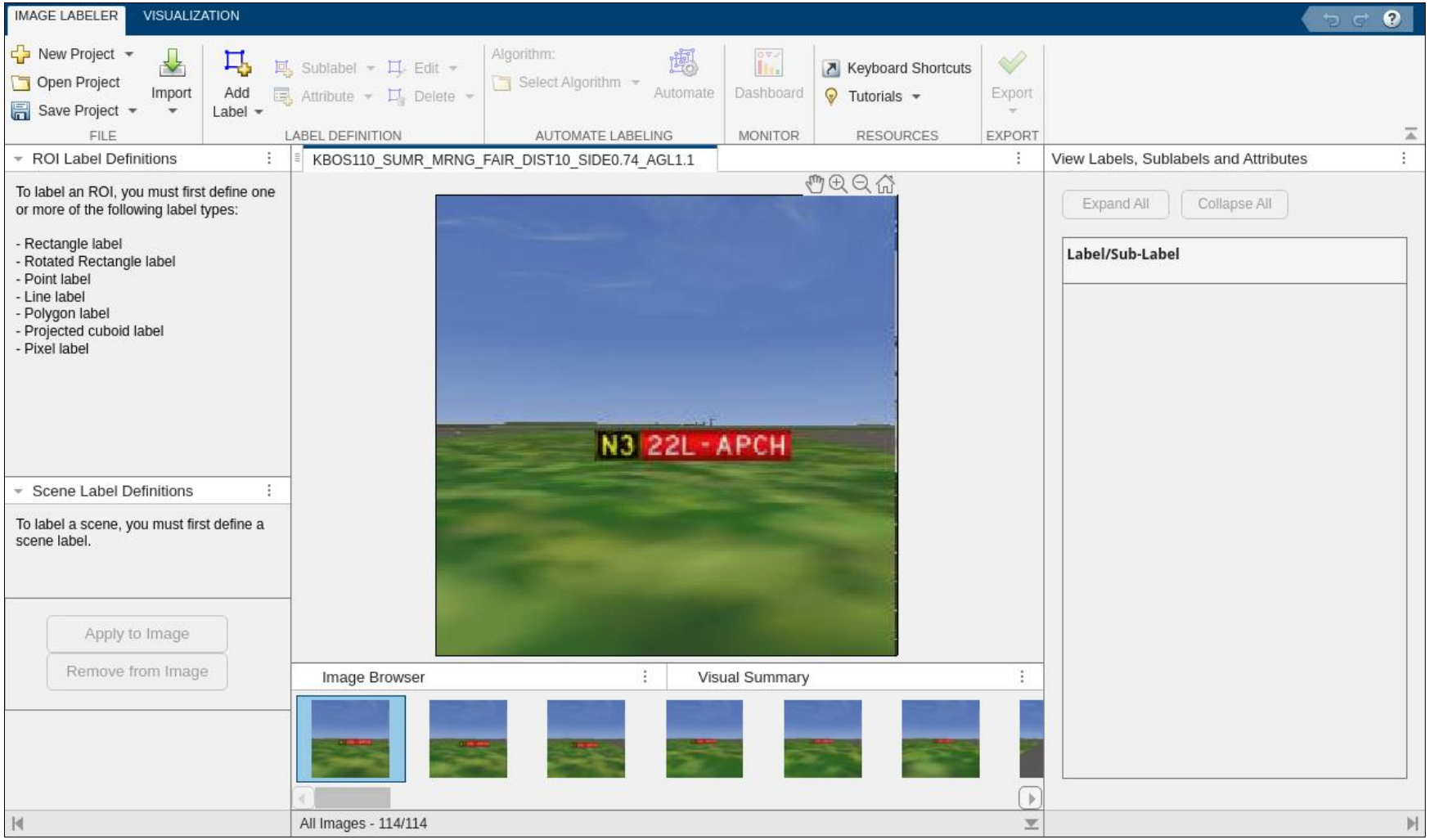


Hyperlinks open the datastores ready to view in the Image Labeler App for review

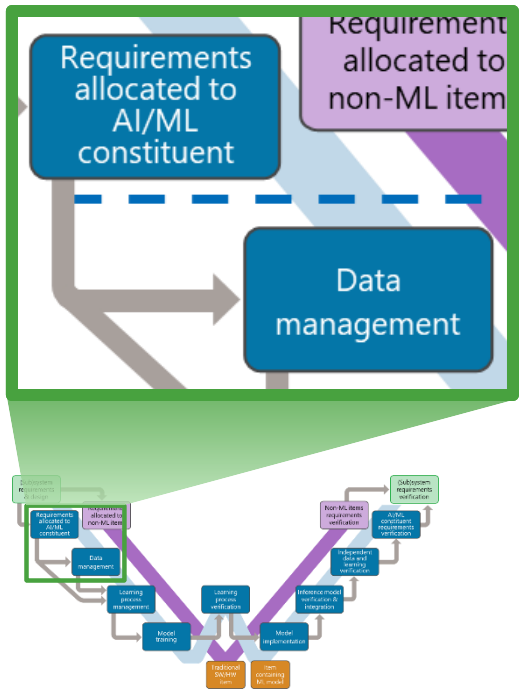
Image Labeler

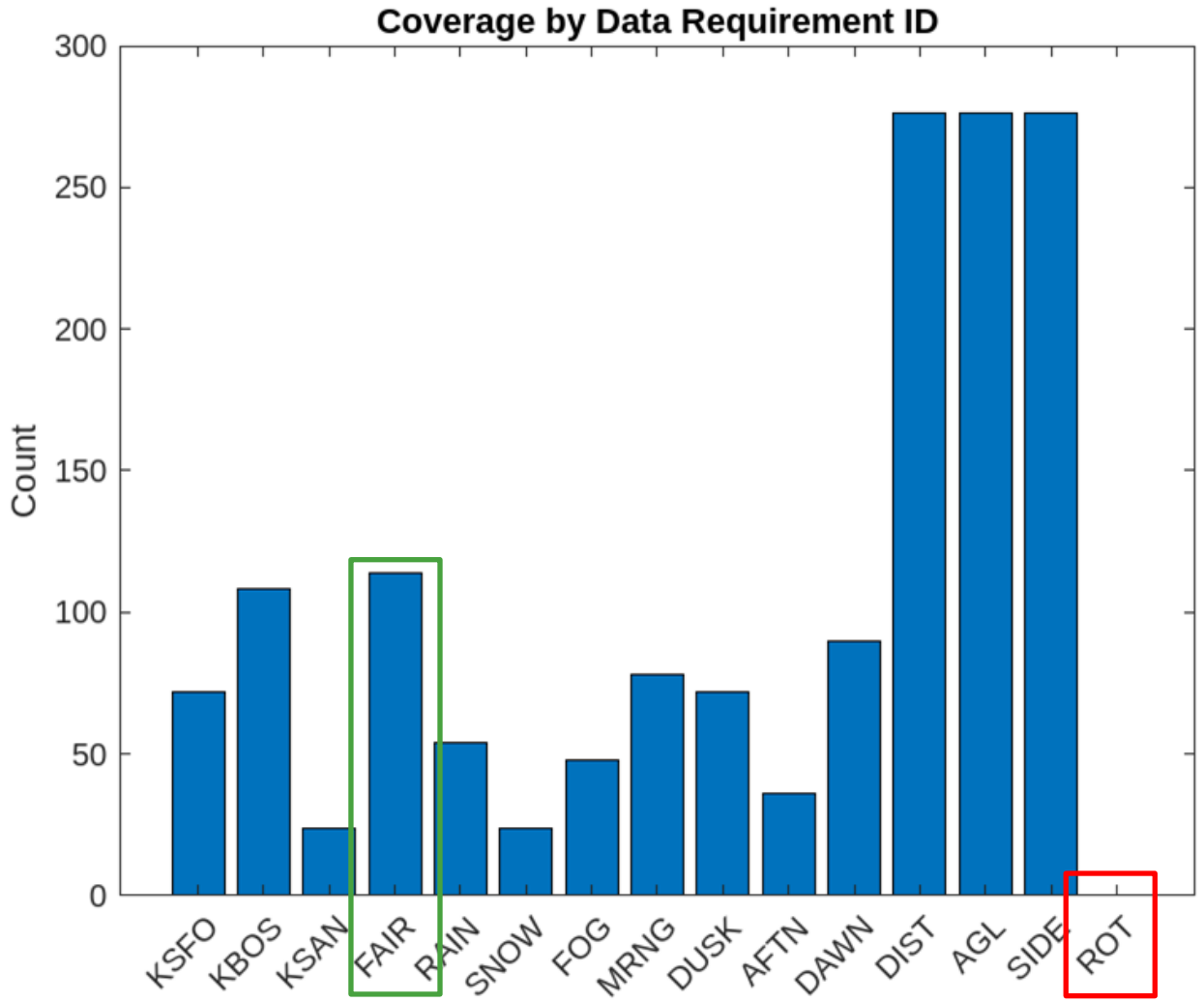
Map data requirements to your MATLAB data and review data in Image Labeler App





Compute data coverage per data requirement





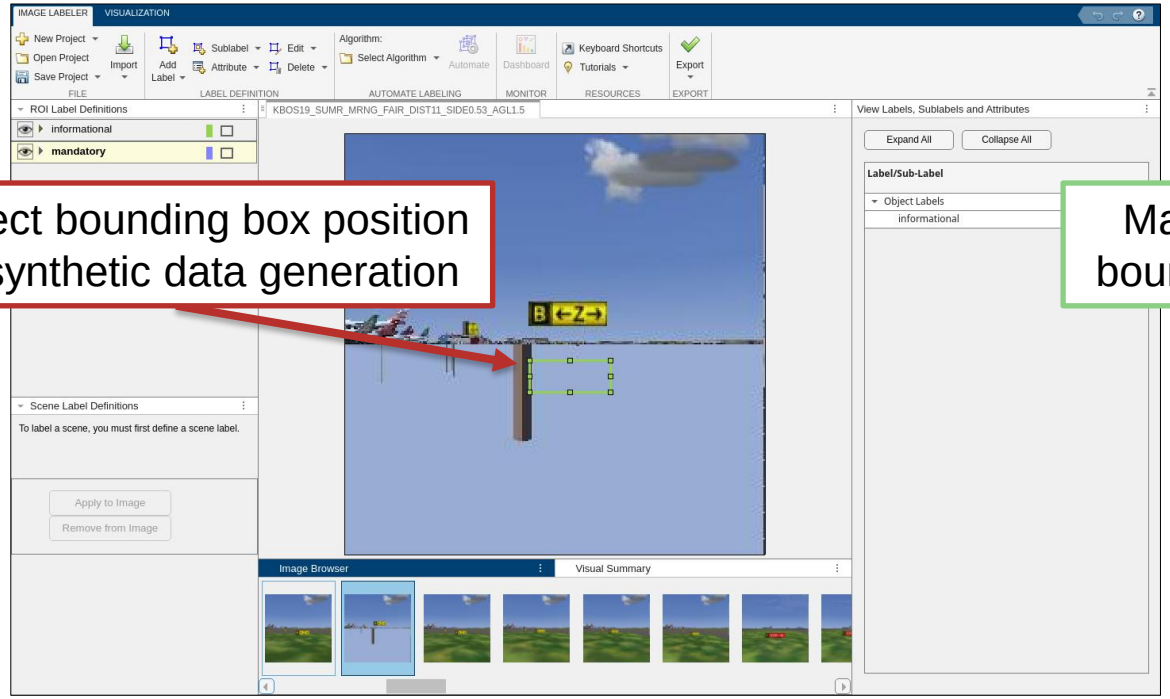
Examine data coverage in each of the operational conditions

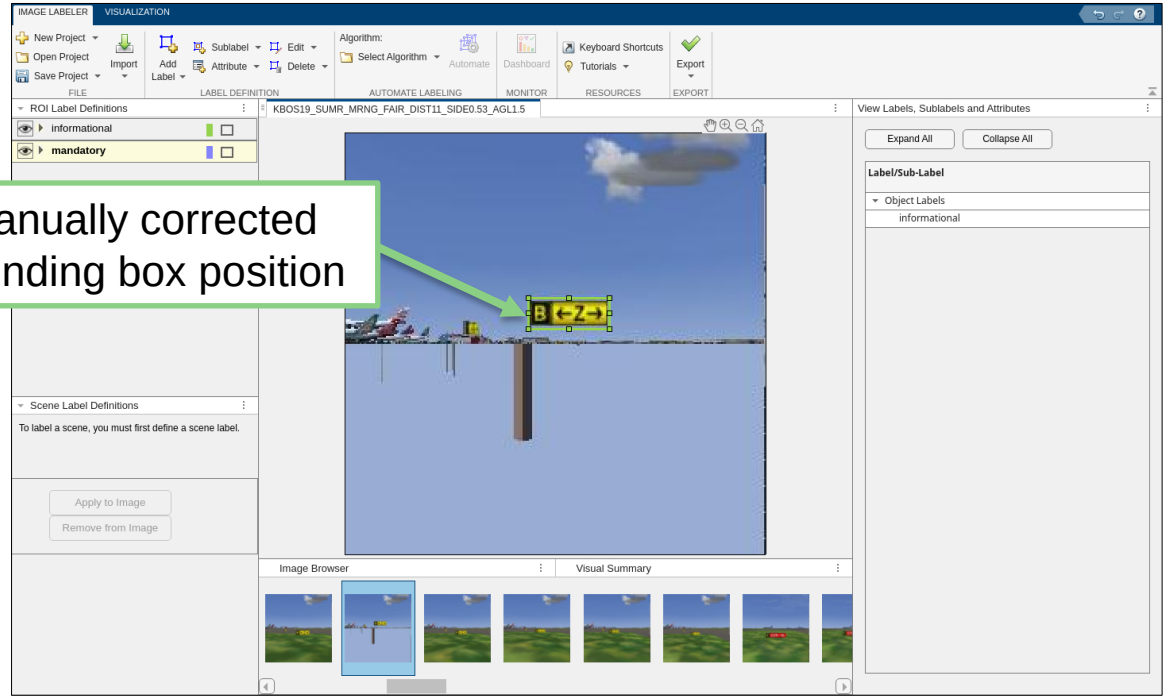
Unfulfilled requirement on sign rotation

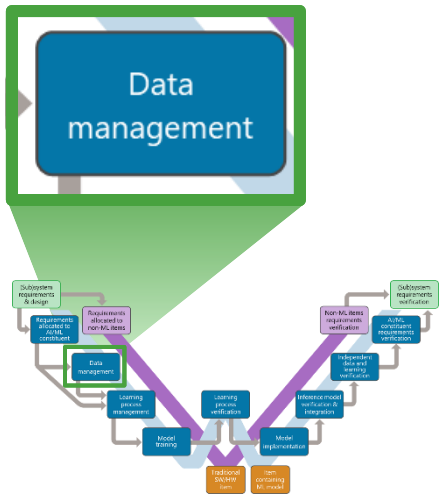
Review Data Correctness

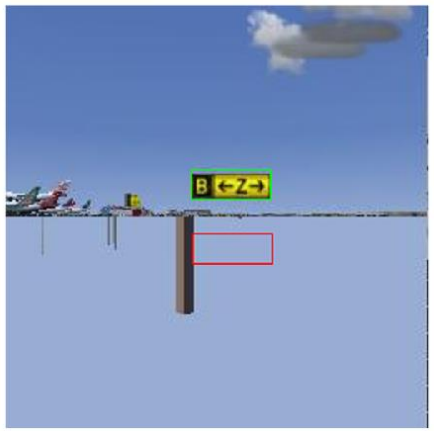
Incorrect bounding box position from synthetic data generation

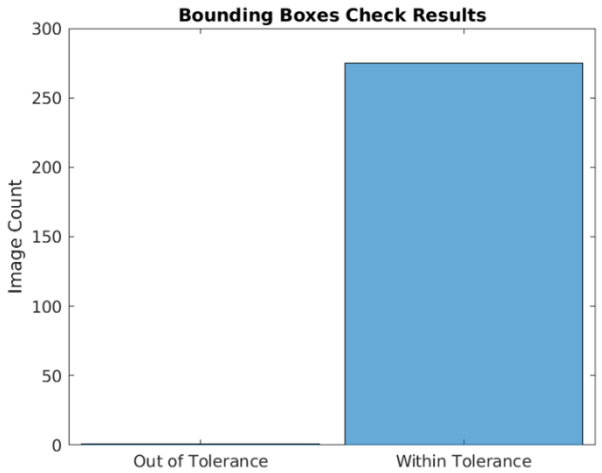
Manually corrected bounding box position











Dataset Percentage Correctness = 99.6377%

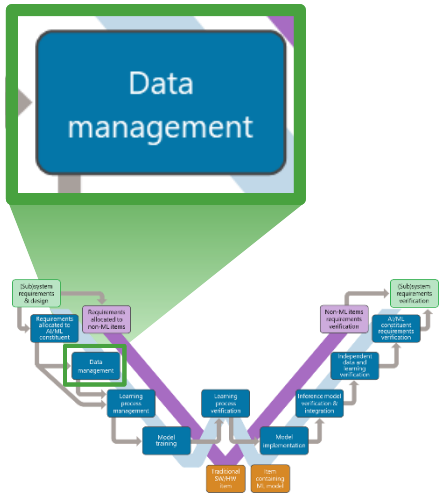
Data Augmentation

Boost accuracy by transforming data

Add variety without extra samples

Examples:

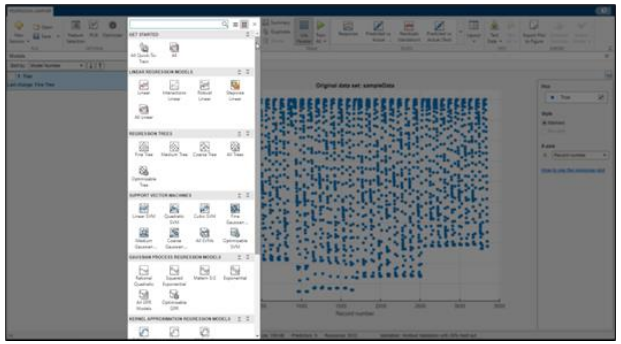
Rotation
Flipping
Translation
Noise injection



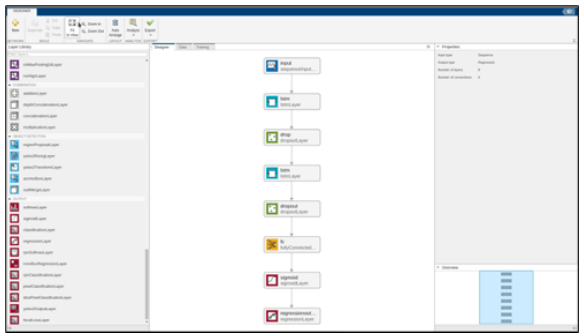
Color jitter augmentation in HSV space

Random scaling by 50 percent

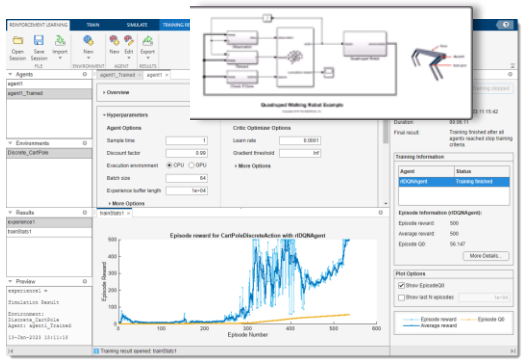
Increase productivity using Apps for design and analysis



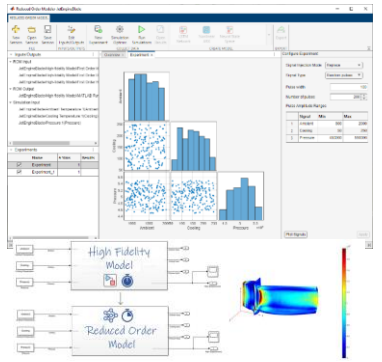
Machine Learning Apps
Train machine Learning Models



Deep Network Designer
Build, visualize, and edit deep learning networks

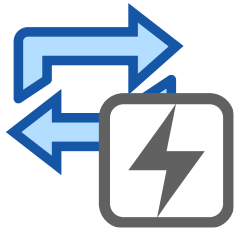


Reinforcement Learning Designer
Design, train, and simulate agents for existing environments

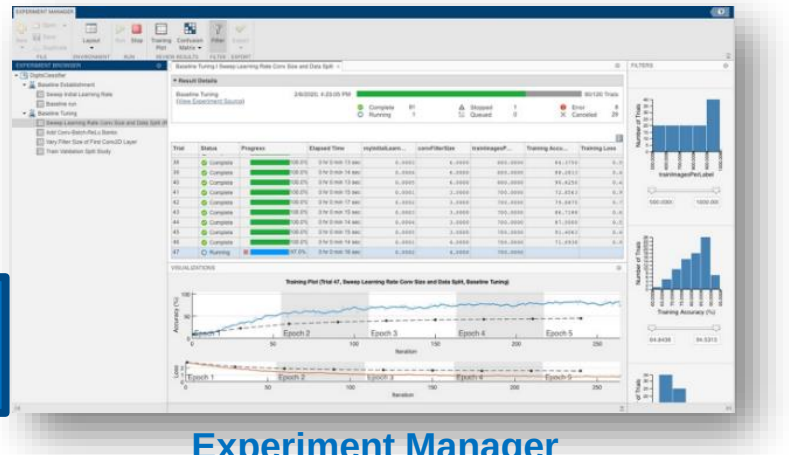


Reduced Order Modeler App
Create and train data-driven ROM of subsystems (including those with high-fidelity 3rd party tools)

Design your AI model

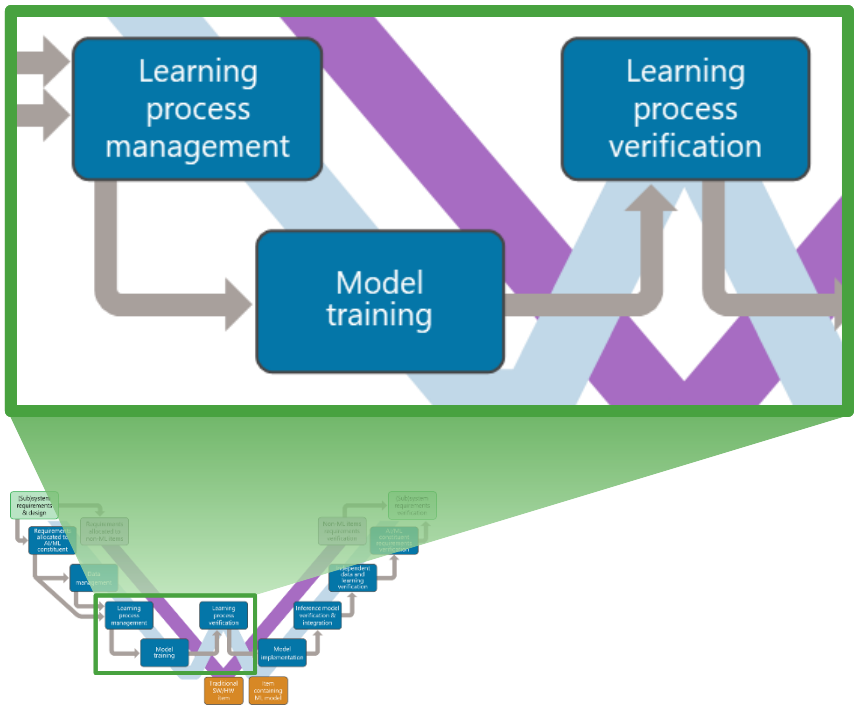


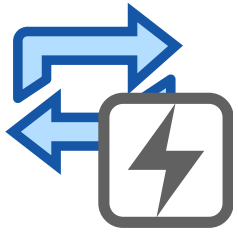
Track training progress and performance for different experiments



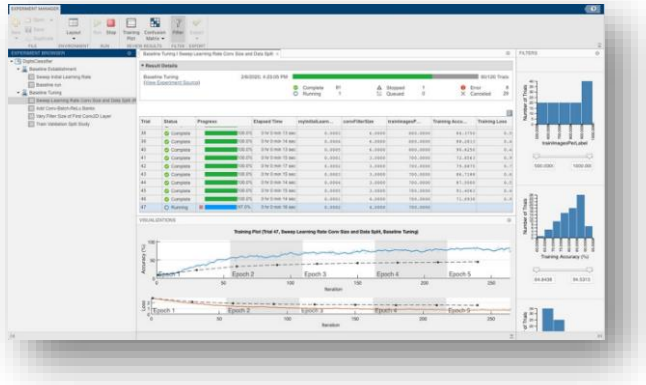
Experiment Manager
Run multiple experiments, analyze and compare results, optimize your AI model

Manage, train and verify the learning process

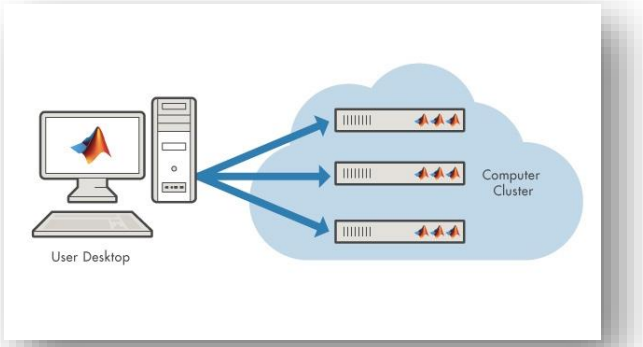




Iterate faster with productivity apps



Accelerate model training with GPUs and the cloud



Manage, train and verify the learning process

MLA_04 Learning Management

Previous Chapter: Data Management

You must run all the scripts in the Data Management chapter before you can run the scripts that comprise this chapter. T
previous chapter. You must also run the scripts in this chapter in this order:

1. Learning Management
2. Estimate Anchor Boxes
3. YOLO v2 Train
4. YOLO v3 Train
5. YOLO v4 Train
6. Model Evaluation

Model Family Selection and Training

Based on the requirements for the RSC DNN component, use the YOLO family of fast and compact detection DNNs for
networks, see Object Detection.

Anchor boxes are important parameters in deep learning object detectors such as Faster R-CNN and YOLO. You can es
scale, and number of anchor boxes impact the efficiency of the training and ultimately the accuracy of the detectors. Thi
boxes hyperparameter:

Estimate Anchor Boxes

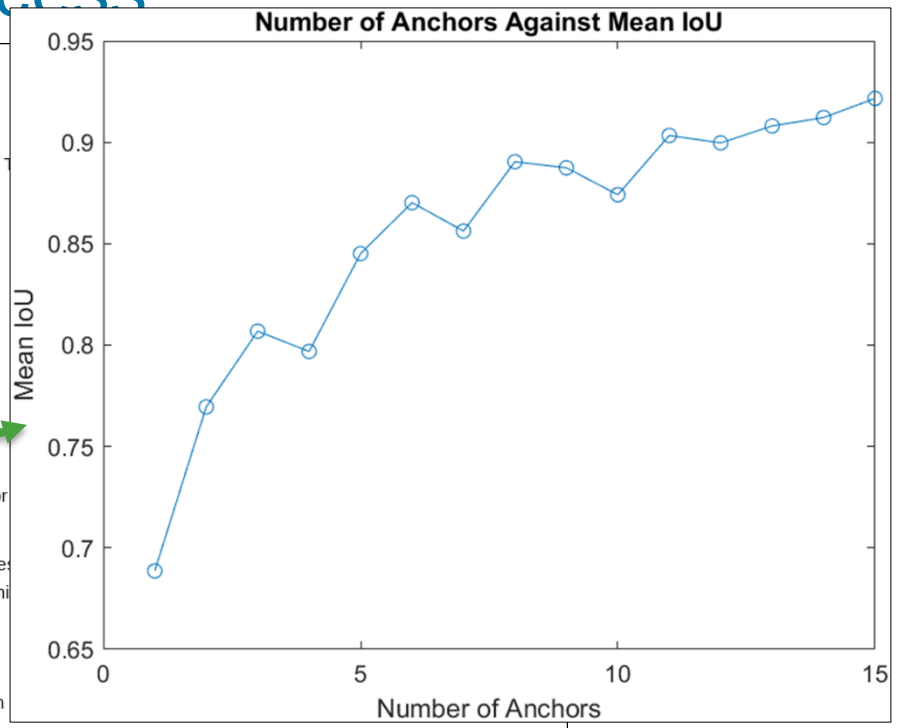
These Live Scripts show how to train YOLO v2, v3, and v4 networks for further evaluation in the context of runway sign

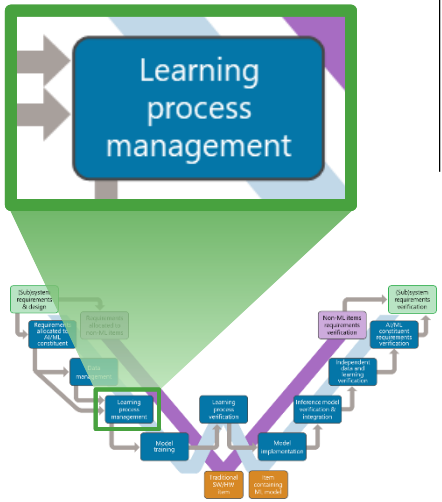
Train YOLO v2 Network

Train YOLO v3 Network

Train YOLO v4 Network

To save time, load pretrained YOLO v2, v3, and v4 object






```
trainingOptionsRSC = trainingOptions("adam", ...
    ExecutionEnvironment="auto", ...
    InitialLearnRate=0.001, ...
    MiniBatchSize=16, ...
    MaxEpochs=40, ...
    Shuffle="every-epoch", ...
    LearnRateDropFactor=0.1, ...
    LearnRateDropPeriod=10, ...
    LearnRateSchedule="piecewise", ...
    BatchNormalizationStatistics="moving",...
    ResetInputNormalization=false, ...
    VerboseFrequency=30, ...
    ValidationData=validationDatastore, ...
    Plots="training-progress");
```

Manage, train and verify the learning process

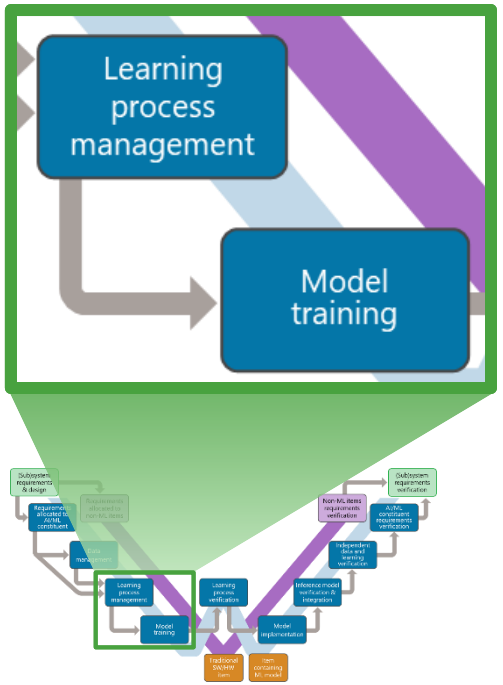
Different ways to train
DL networks in MATLAB

Using high level APIs

```
trainnet
trainYOLOv3ObjectDetector
trainYOLOv4ObjectDetector
...
```

Using Custom Training loop

Write your own training loop

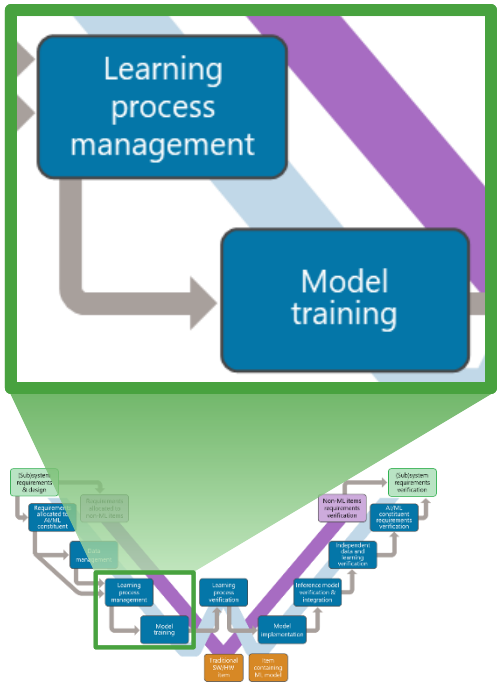


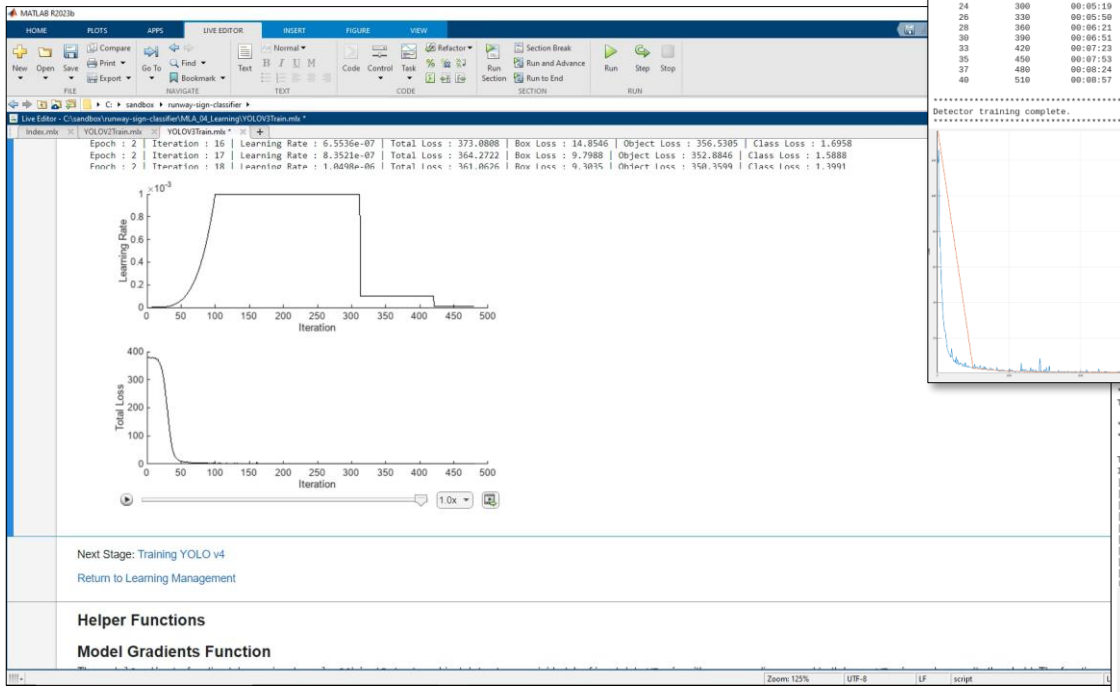

```
doTraining = true;
if doTraining
    yolov4Detector = trainYOLOv4ObjectDetector(trainingDatastore,yolov4Detector,trainingOptionsRSC);
    fileName = fullfile(matlab.project.rootProject().RootFolder,"Learning","yolov4DetectorRSC");
    save(fileName,"yolov4Detector");
end
```

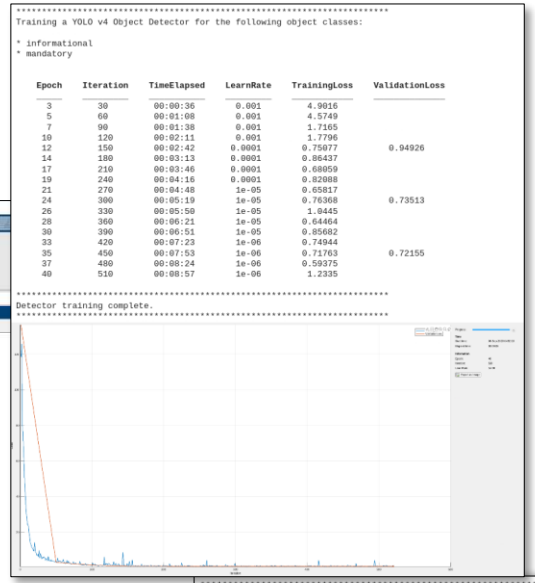
```
*************************************************************************
Training a YOLO v4 Object Detector for the following object classes:

* informational
* mandatory

    Epoch    Iteration    TimeElapsed    LearnRate    TrainingLoss    ValidationLoss
    _____    _________    ___________    _________    ____________    ______________

        1            1        00:00:02        0.001          87.253             98.96
        3           30        00:00:28        0.001          5.1062
        4           50        00:00:40        0.001          3.1646             3.175
```

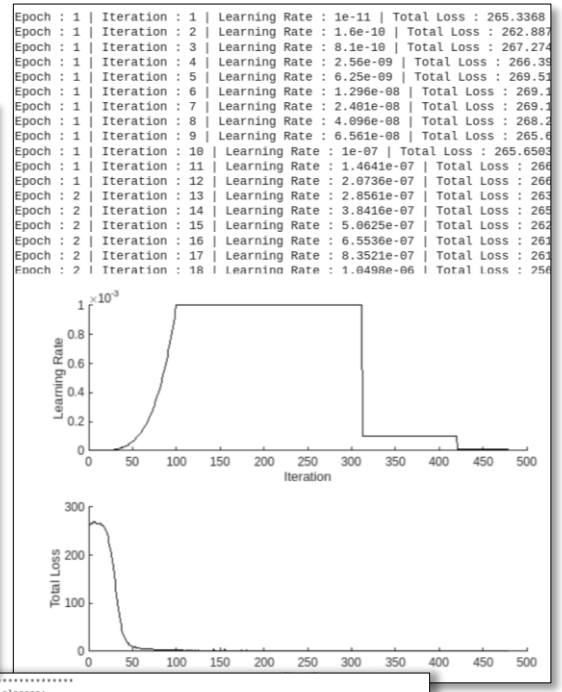
```
if doTraining
    fig = figure;
    [lossPlotter,learningRatePlotter] = configureTrainingProgressPlotter(fig);
    iteration = 0;
    for epoch = 1:MaxEpochs

        reset(mbqTrain);
        % Shuffle the training data at every epoch.
        shuffle(mbqTrain);

        while(hasdata(mbqTrain))
            iteration = iteration + 1;

            [XTrain,YTrain] = next(mbqTrain);

            % Evaluate the model gradients and loss.
            [gradients,state,lossInfo] = dlfeval(@modelGradients,yolov3Detector,XTrain,YTrain,PenaltyThreshold);

            % Apply L2 regularization.
            gradients = dlupdate(@(g,w) g + L2Regularization*w,gradients,yolov3Detector.Learnables);

            % Determine the current learning rate value.
            currentLR = piecewiseLearningRateWithWarmup(iteration,epoch,InitialLearnRate,WarmupPeriod,MaxEpochs);

            % Update the detector learnable parameters using the SGDM optimizer.
            [yolov3Detector.Learnables,Velocity] = sgdmupdate(yolov3Detector.Learnables,gradients,Velocity,currentLR);

            % Update the state parameters of the network.
            yolov3Detector.State = state;

            % Display the training progress.
            displayLossInfo(epoch,iteration,currentLR,lossInfo);

            % Update the training plot with new points.
            updatePlots(lossPlotter,learningRatePlotter,iteration,currentLR,lossInfo.totalLoss);
        end
    end
```

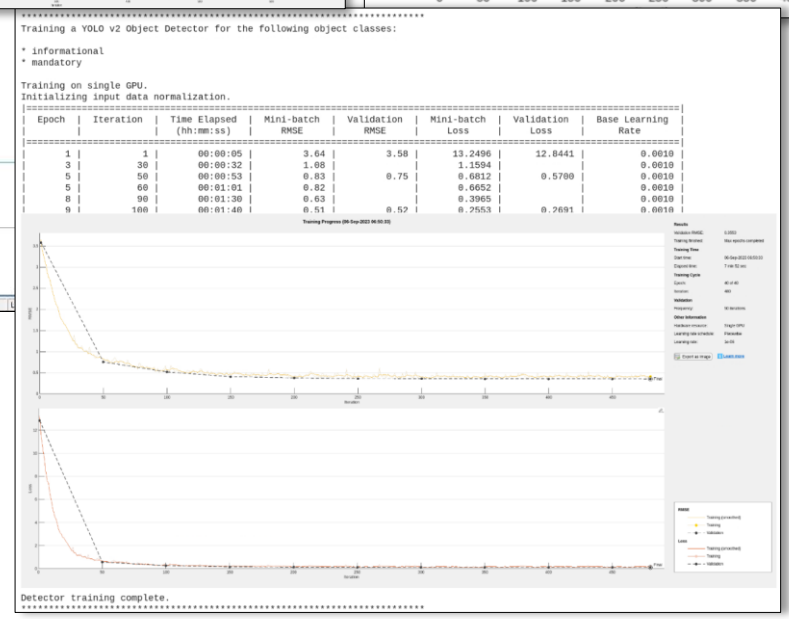
Manage, train and verify the learning process



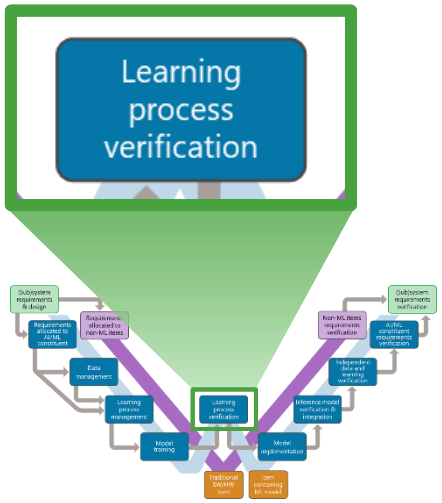




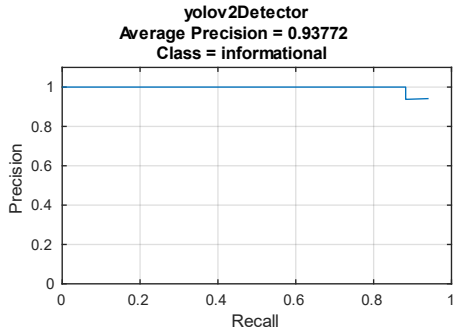




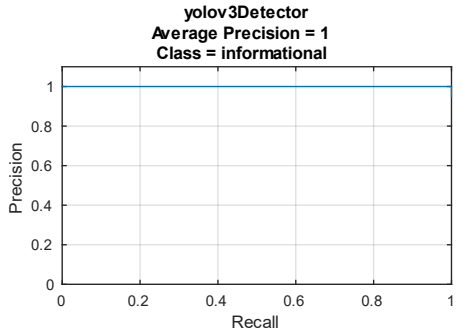
Manage, train and verify the learning process



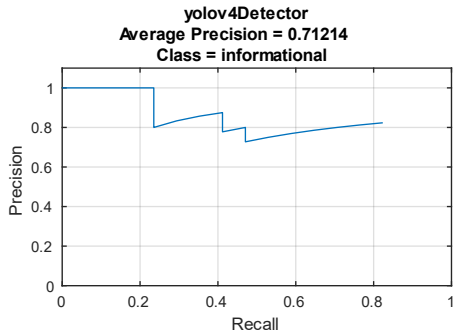
yolov2Detector
Test Image Sample



yolov3Detector
Test Image Sample



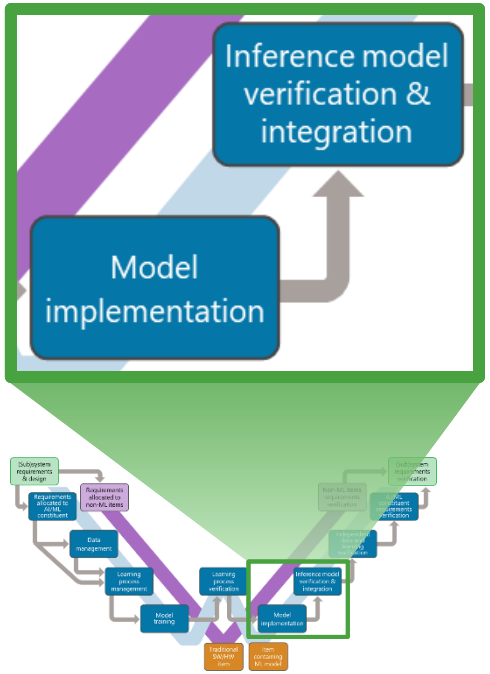
yolov4Detector
Test Image Sample

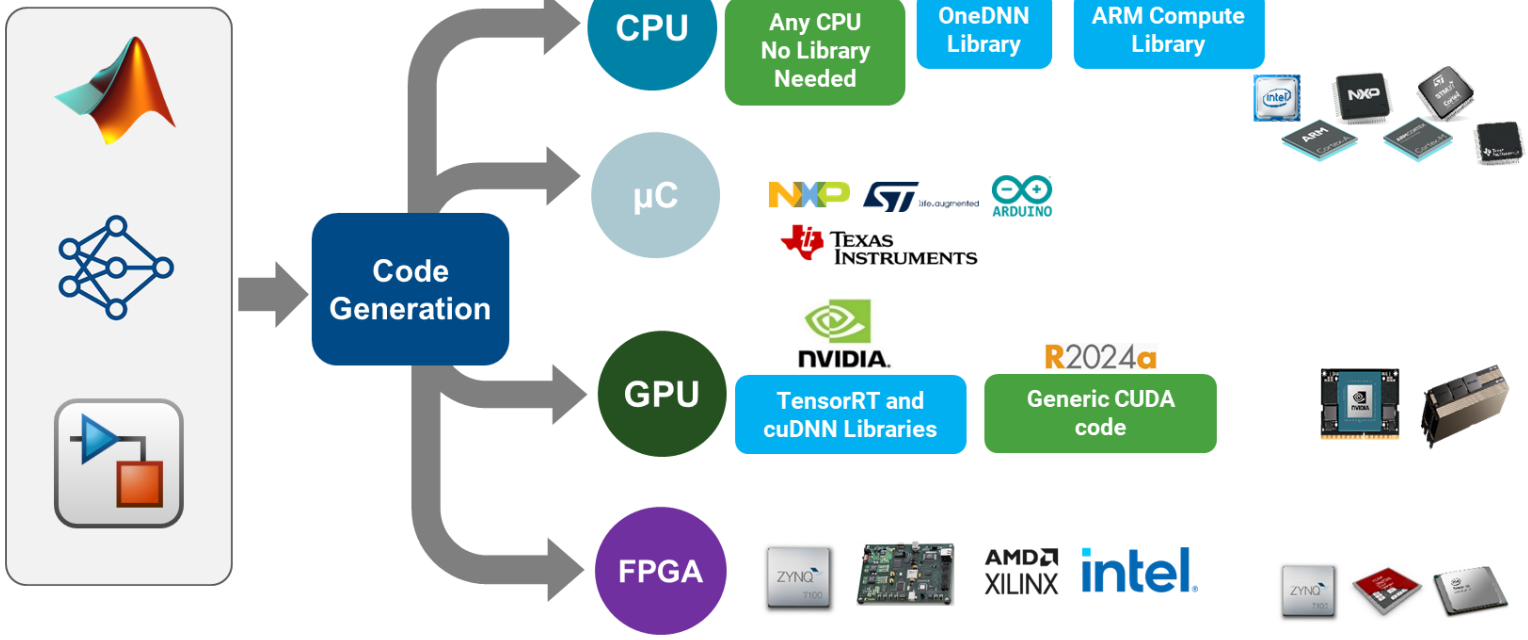


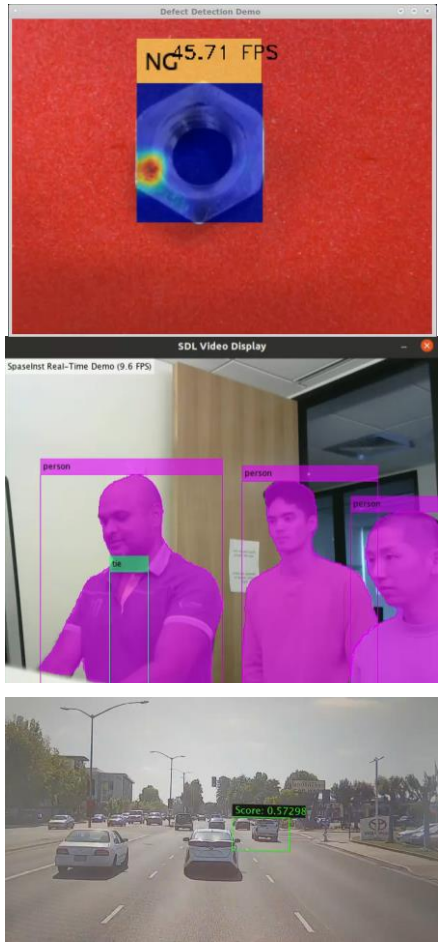
The ideal model delivers precision 1 at all recall levels.

YOLO V3 meets the required average precision per class of at least 95%.

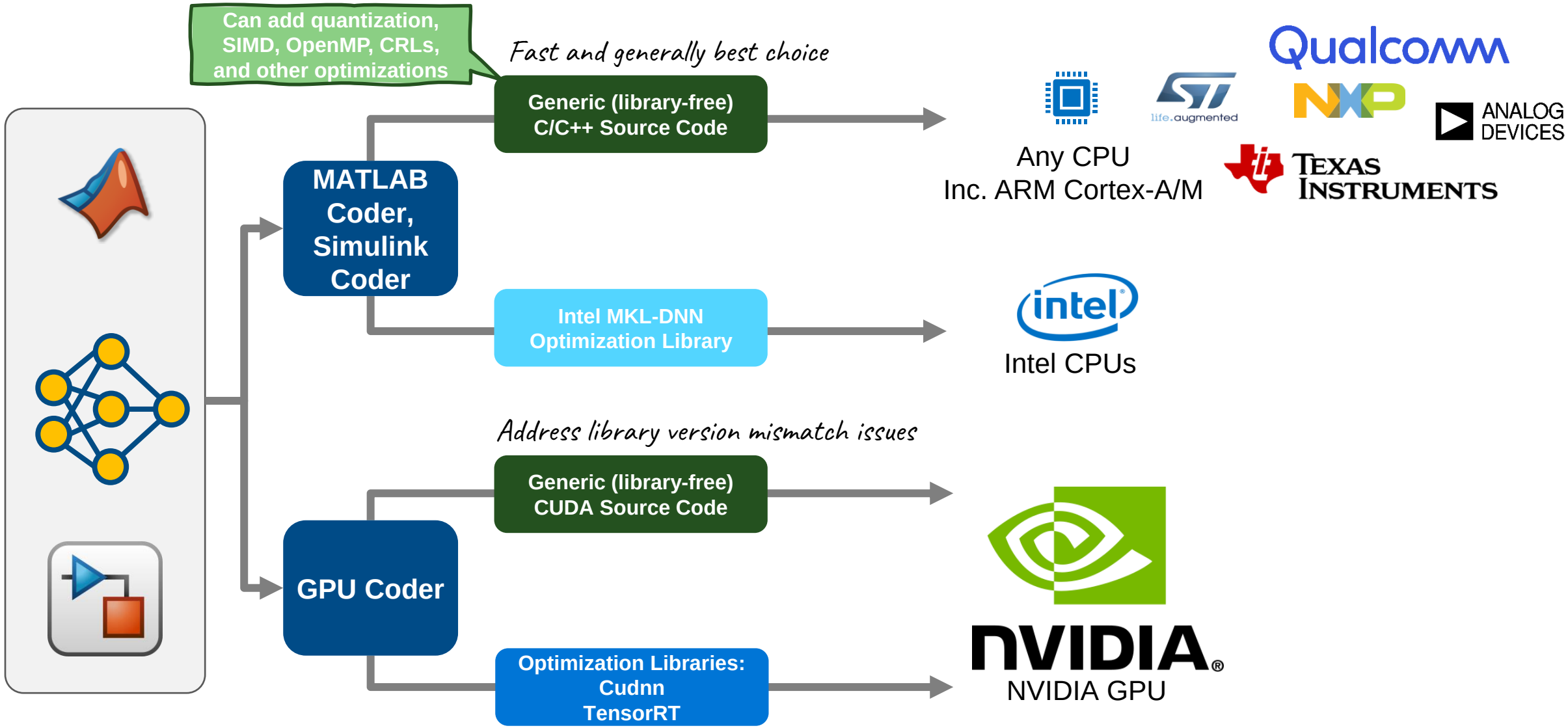
Deploy to targets





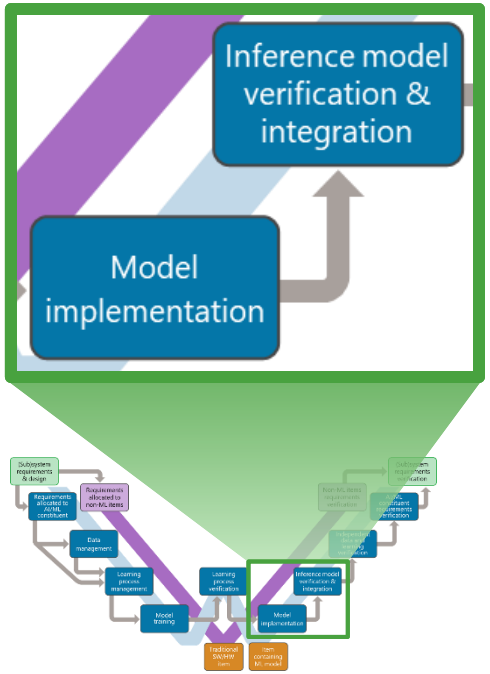


Deep Learning C/C++ Code Generation



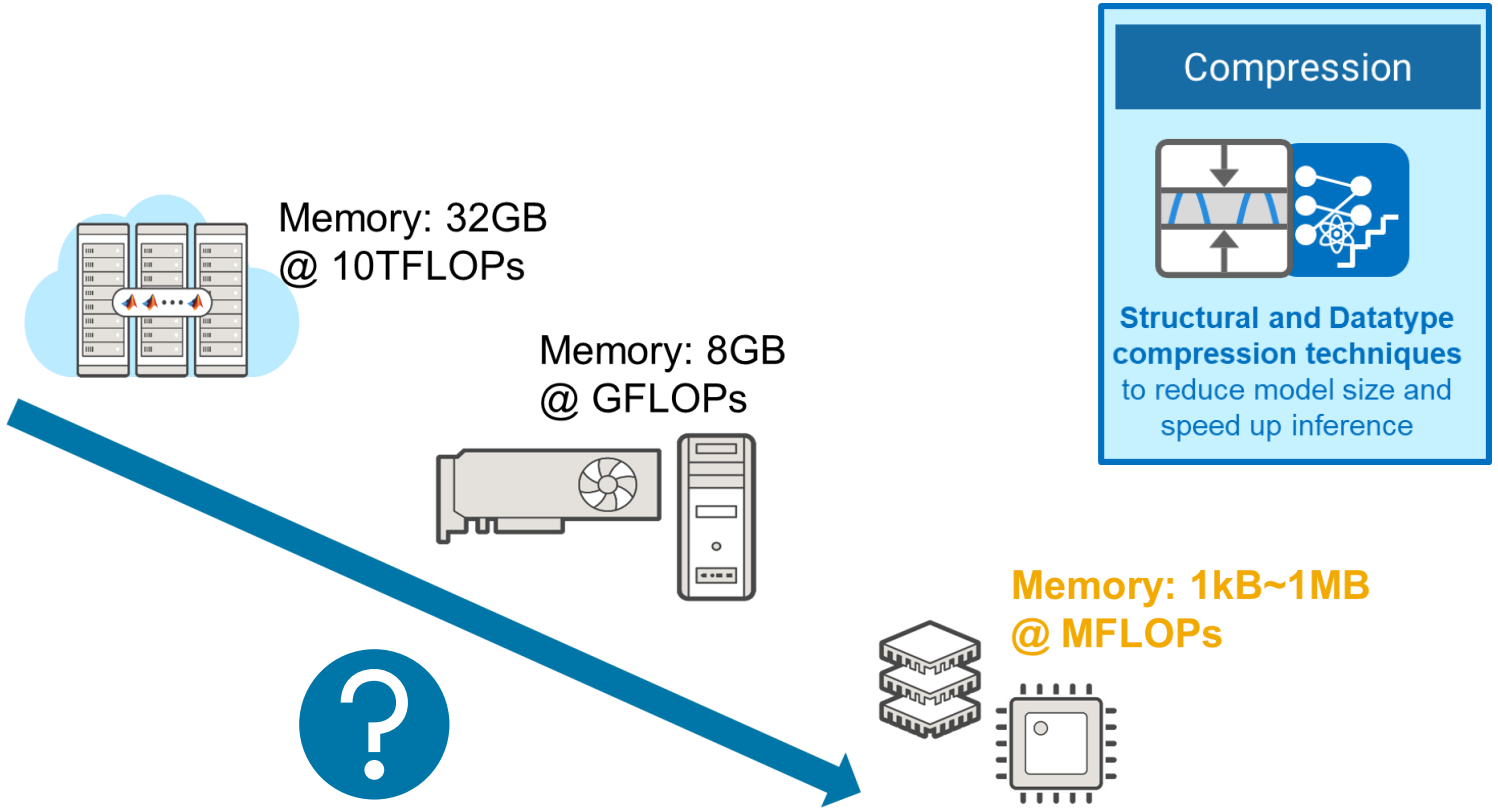
Flexible library deployment for a given model

```
>> analyzeNetworkForCodegen(yolov3Detector.Network)
```



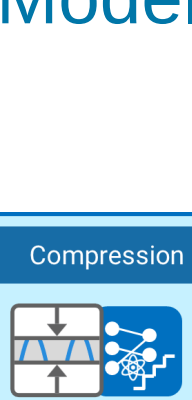
	Supported
none	"Yes"
arm-compute	"Yes"
mkldnn	"Yes"
cudnn	"Yes"
tensorrt	"Yes"

How to optimized performance in hardware constrained environment?

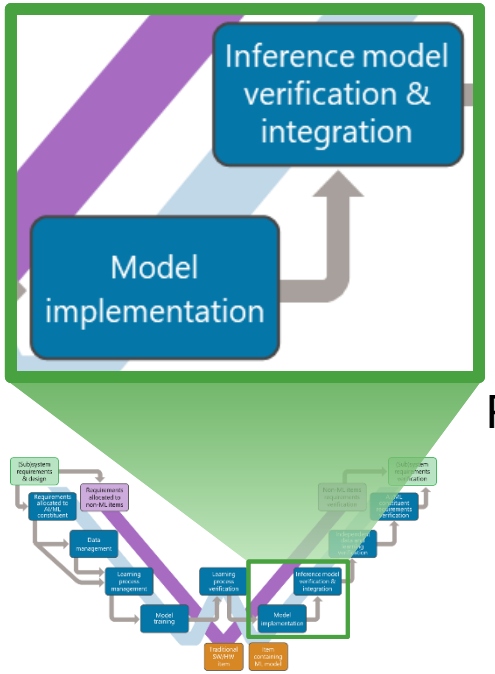


Model Compression

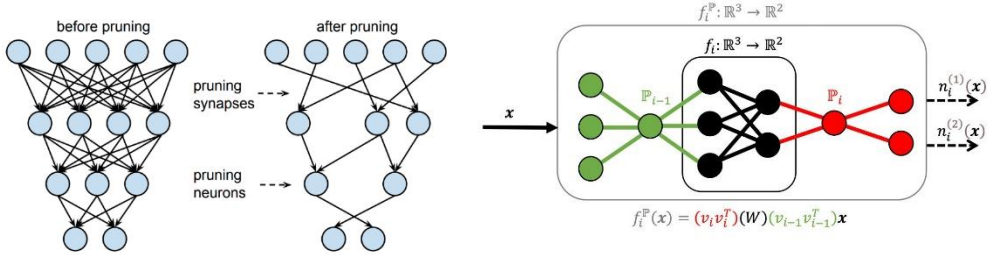
Compression



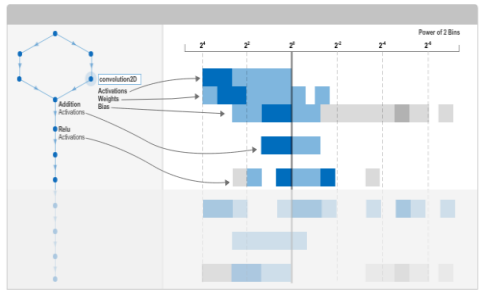
Structural and Datatype compression techniques to reduce model size and speed up inference



Structural Compression



Datatype Compression



Code generation

Pruning → **Projection** → **Quantization**

Removing unimportant parts of networks

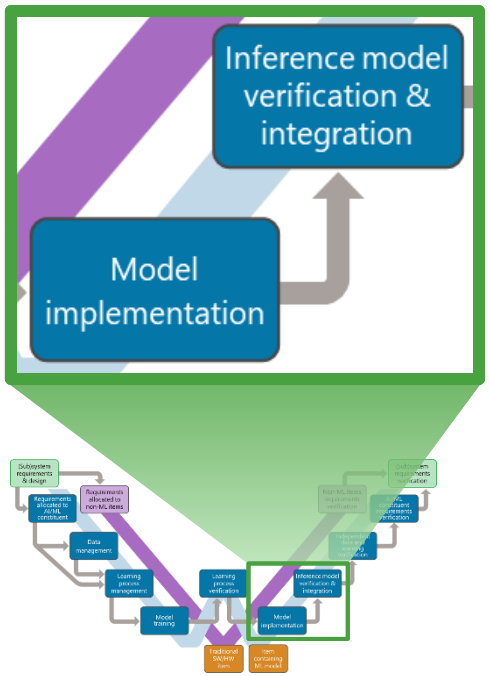
Project learnable parameters into a lower dimensional space

Conversion or parameters to lower precision datatypes (bfloat16, int8)

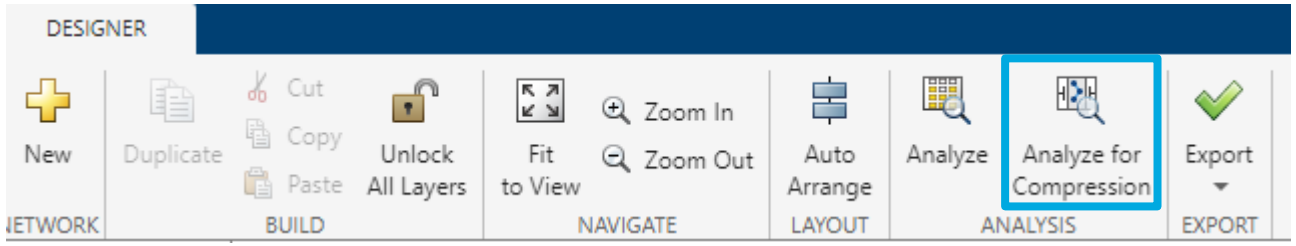
Model Compression

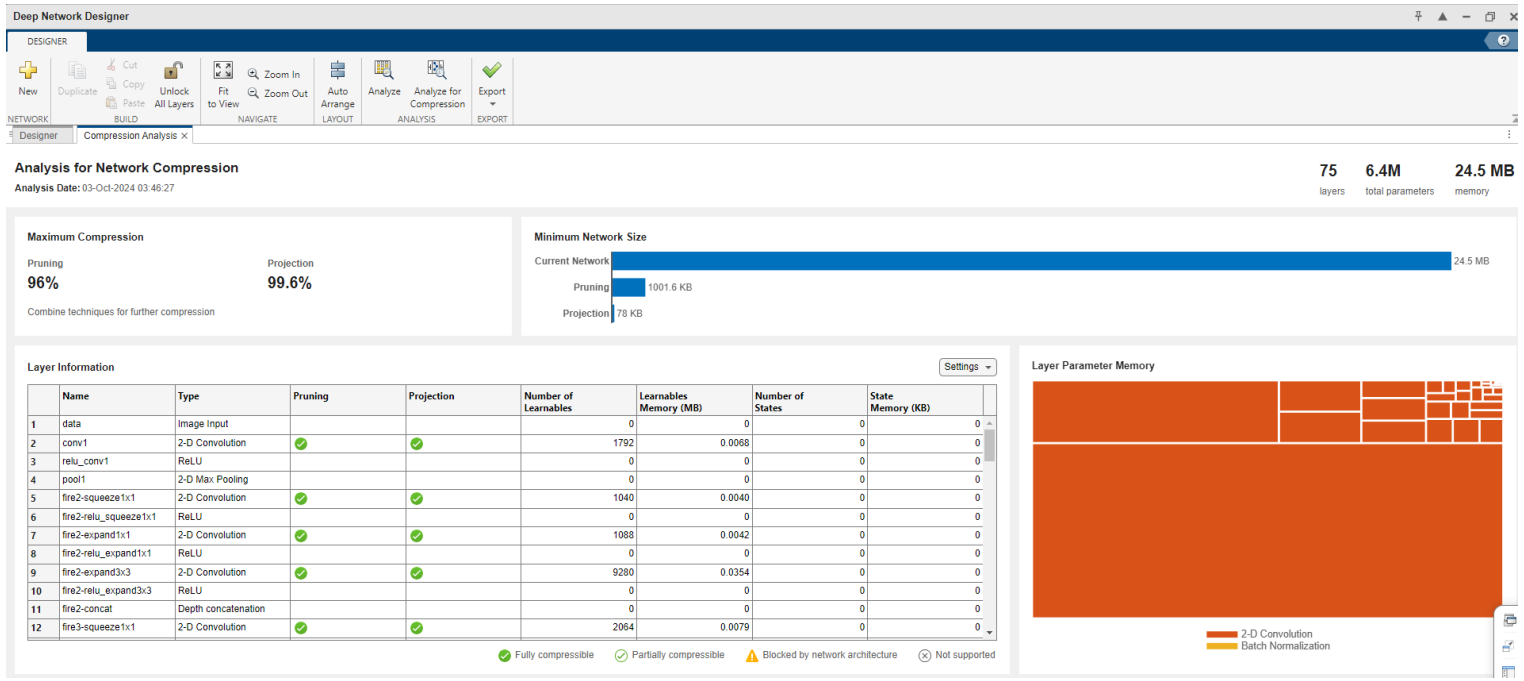
Analyze network for Compression

- *Maximum possible memory reduction*
- *Pruning and projection support*
- *Ability to prune individual layers*
- *Layer memory*

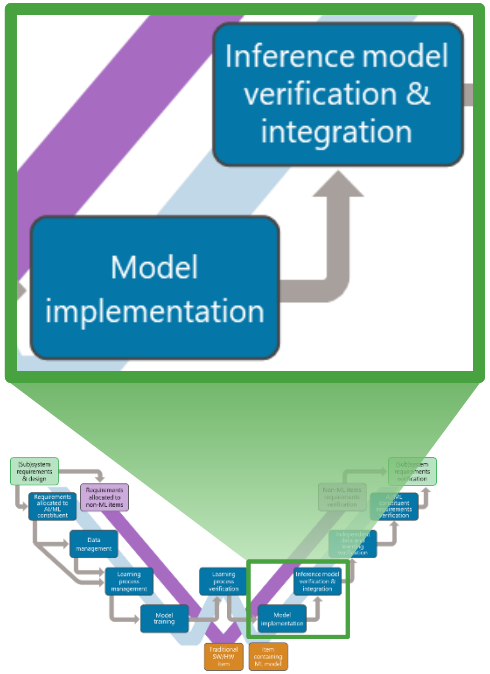


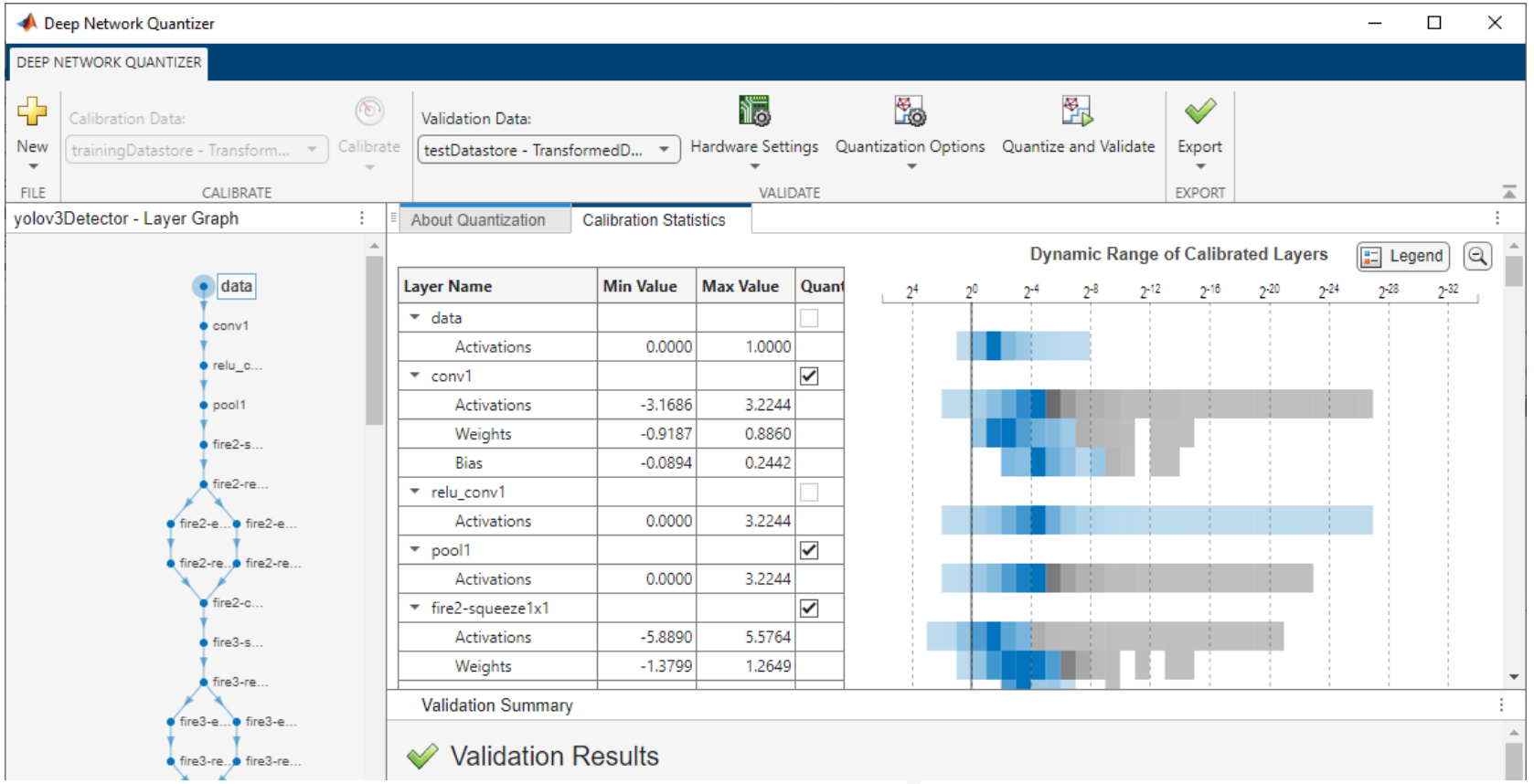

```
>>deepNetworkDesigner(yolov3Detector.Network)
```





Quantization, compression and code generation






```
valResults.Statistics
```

ans = *2×2 table*

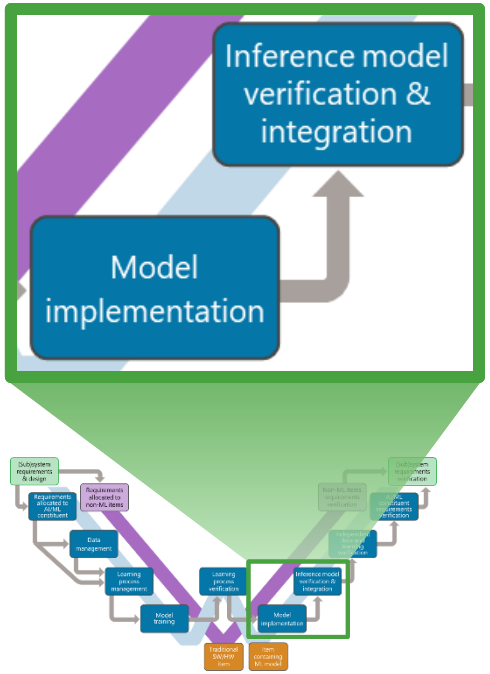
	NetworkImplementation	LearnableParameterMemory(bytes)
1	'Floating-Point'	25635668
2	'Quantized'	6422036

```
valResults.MetricResults.Result
```

ans = *2×2 table*

	NetworkImplementation	MetricOutput
1	'Floating-Point'	[1;1]
2	'Quantized'	[1;0.9667]

Quantization, compression and code generation



Generate `int8` CUDA Code

After you train and evaluate the detector, you can generate CUDA code using GPU Coder software. For more details, see Generate `INT8` Code for Deep Learning Networks.

```
cfg = coder.gpuConfig("mex");
cfg.TargetLang = "C++";
```

Check the computing capability of the GPU.

```
gpuInfo = gpuDevice;
cc = gpuInfo.ComputeCapability;
```

Create a deep learning code generation configuration object.

```
cfg.DeepLearningConfig = coder.DeepLearningConfig("cudnn");
```

Set `DataType` to `"int8"` and set the `CalibrationResultFile` property. `int8` precision requires a CUDA GPU with a minimum computing capability of 6.1, 6.3, or higher.

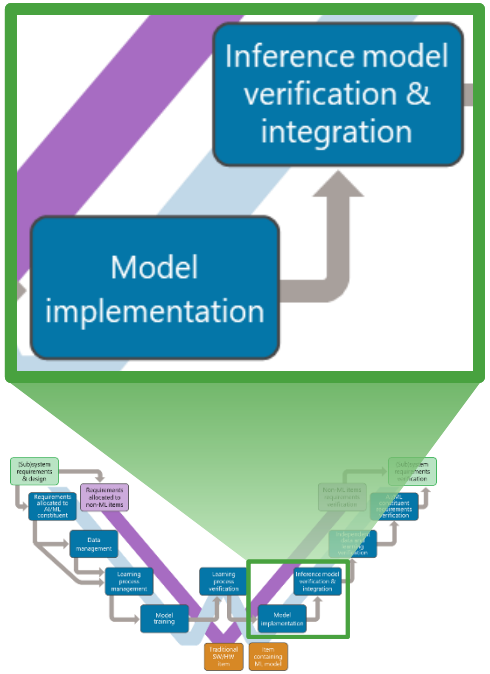
```
cfg.GpuConfig.ComputeCapability = cc;
cfg.DeepLearningConfig.DataType = "int8";
cfg.DeepLearningConfig.CalibrationResultFile = "dlquantObj.mat";
```

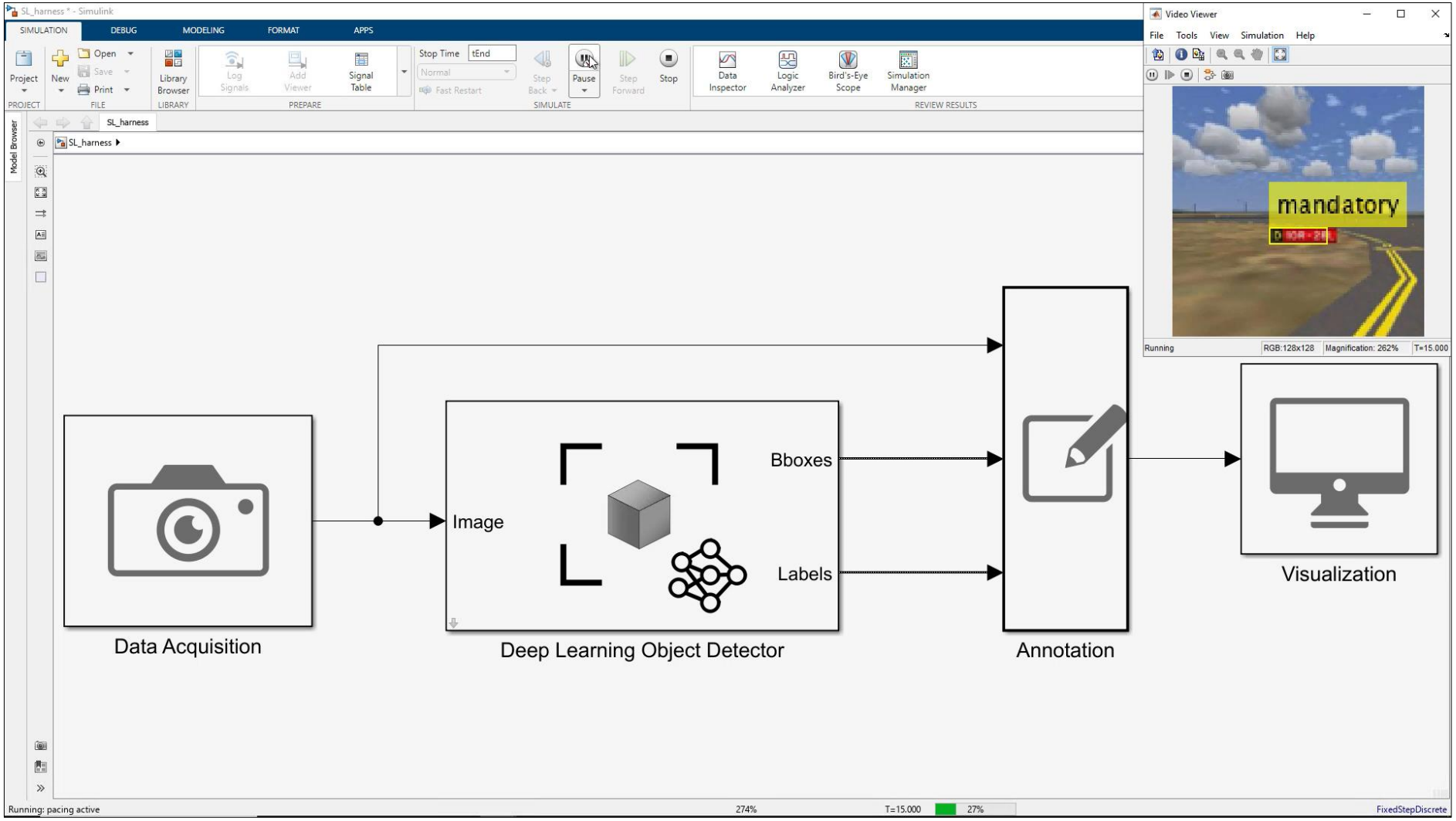
Generate CUDA code by using the `codegen` function. You can view the resulting code generation report by clicking **View Report** under the `codegen` function call after you run this section. The report opens in the Report Viewer window. If the code generator detects errors or warnings during code generation, the report describes the issues and provides links to the problematic MATLAB code. For more information, see Code Generation Reports.

```
inputSize = [128 128 3];
X = ones(inputSize,"uint8");
codegen -config cfg coderDetect -args {X} -o gpuDetect -d gpucodegen -report
```

```
Code generation successful: View report
```

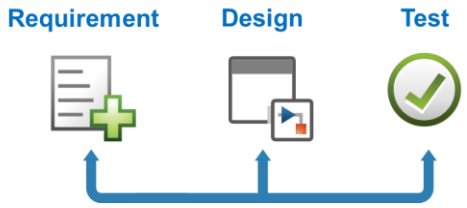
Integrate your AI model with Simulink

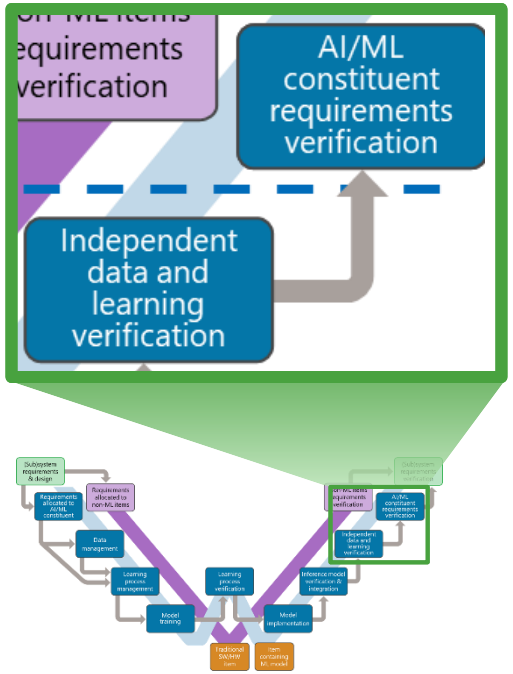


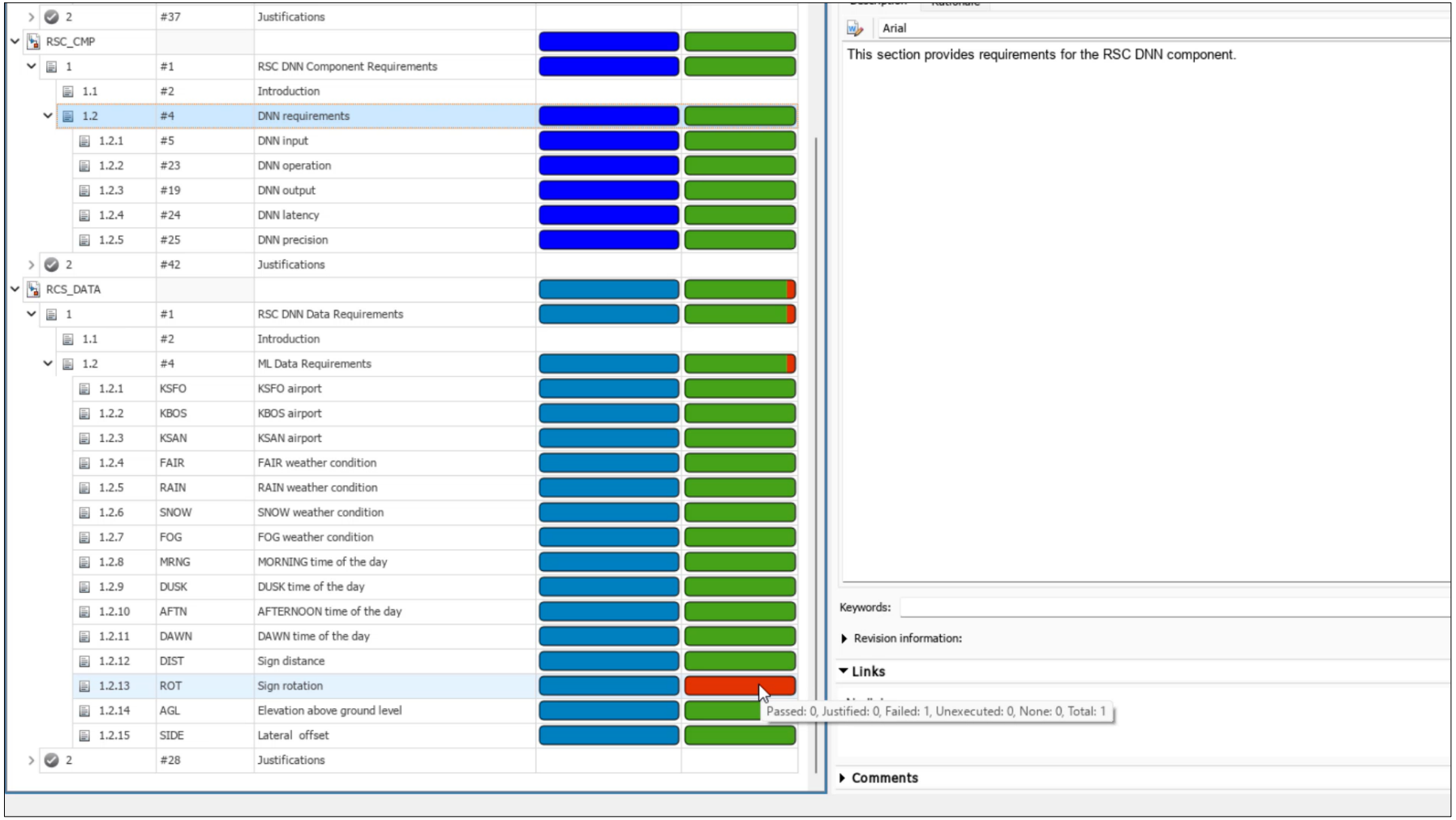


Requirements Verification through testing

Establish Traceability between Requirements, Design, and Tests







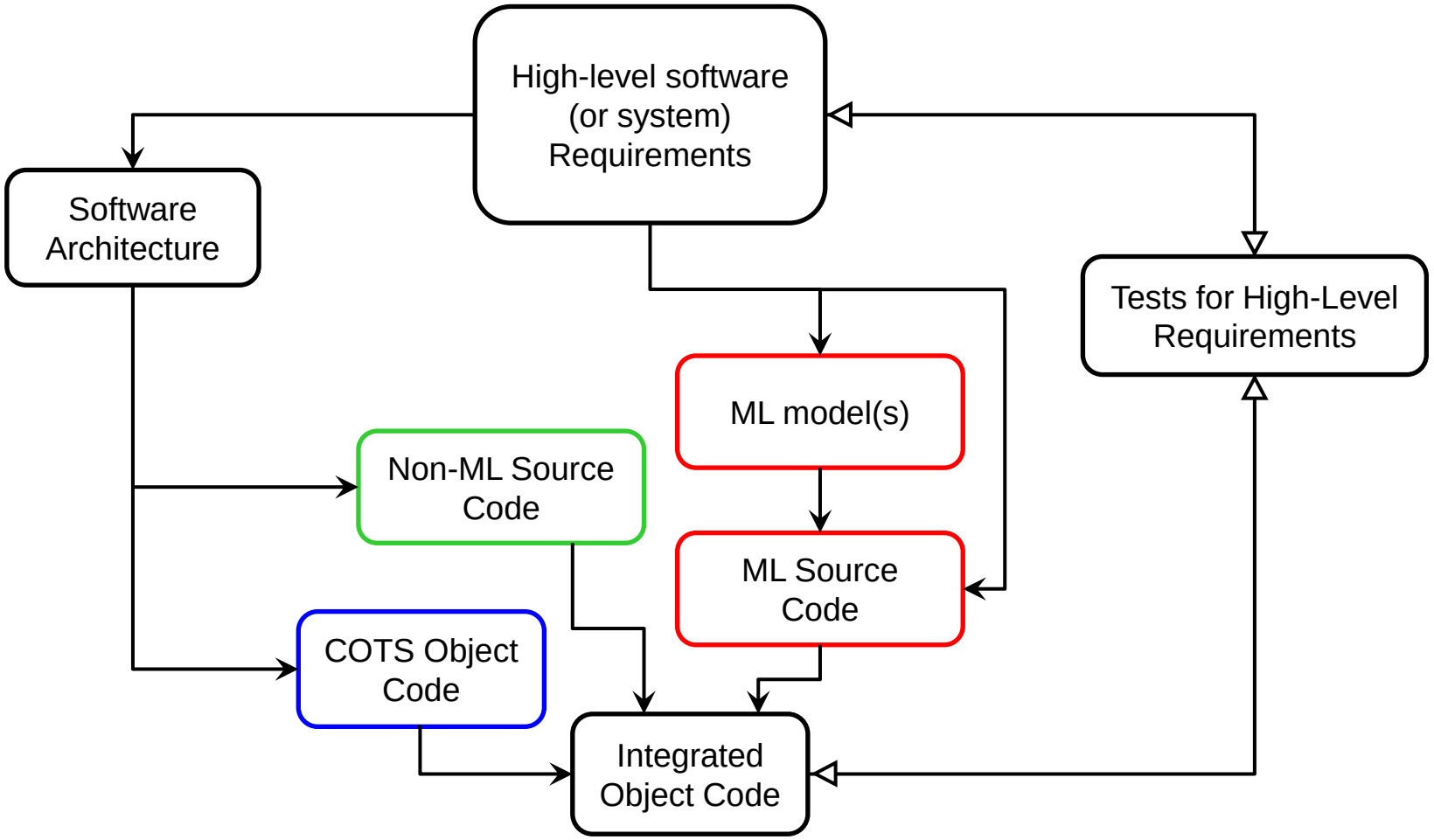
Agenda

AI Certification
in Airborne Systems:
Overview

Certifying DAL D
A Case Study:
Runway Sign Classifier

Towards DAL C

DAL-C Certification requires a deeper understanding of your AI model



DAL C

Verification and Validation for Deep Neural Networks

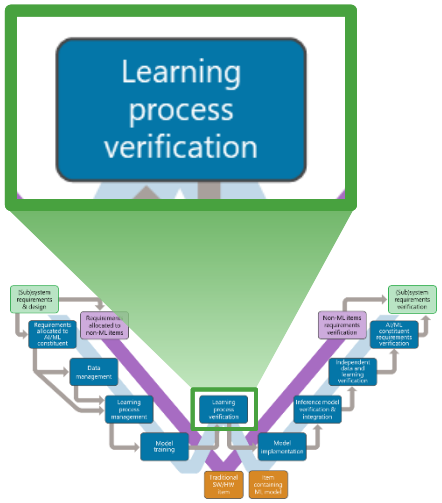
Use Formal Verification Methods to Ensure Robustness

Deep Learning Toolbox Verification Library

by MathWorks Deep Learning Toolbox Team STAFF

Verify and test robustness of deep learning networks

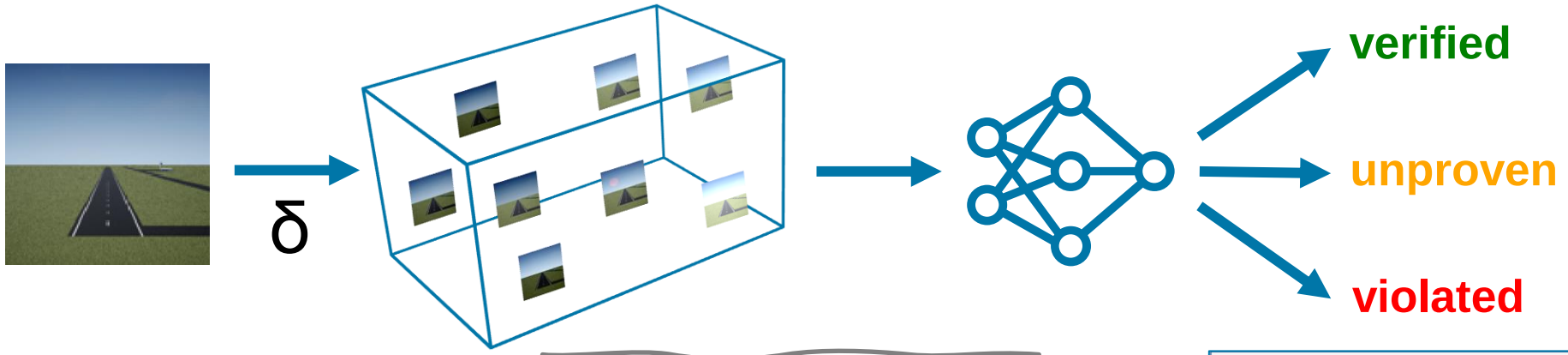
https://www.mathworks.com/help/deeplearning/verification.html

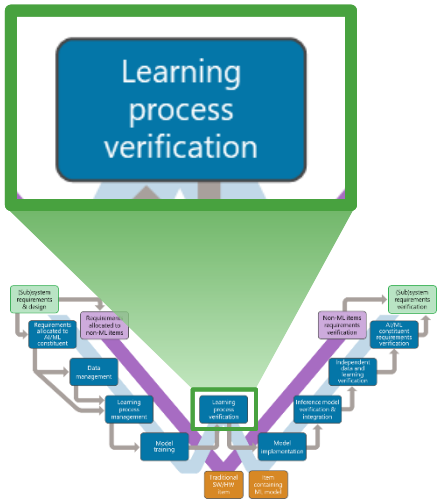


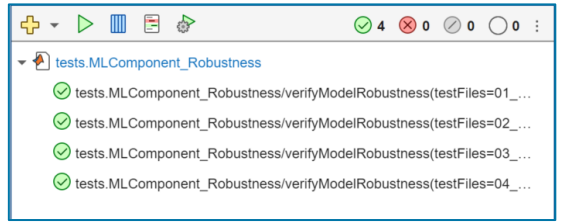
Provide Safety Guardrails for Handling Unseen Data

Verify robustness of deep learning networks

Use Formal Verification Methods to Ensure Robustness







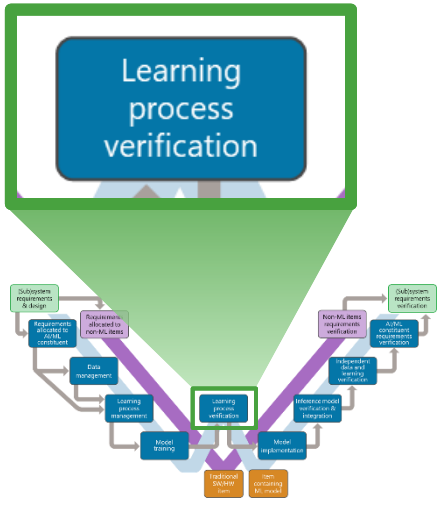
(for regression) `estimateNetworkOutputBounds` (for classification) `verifyNetworkRobustness`

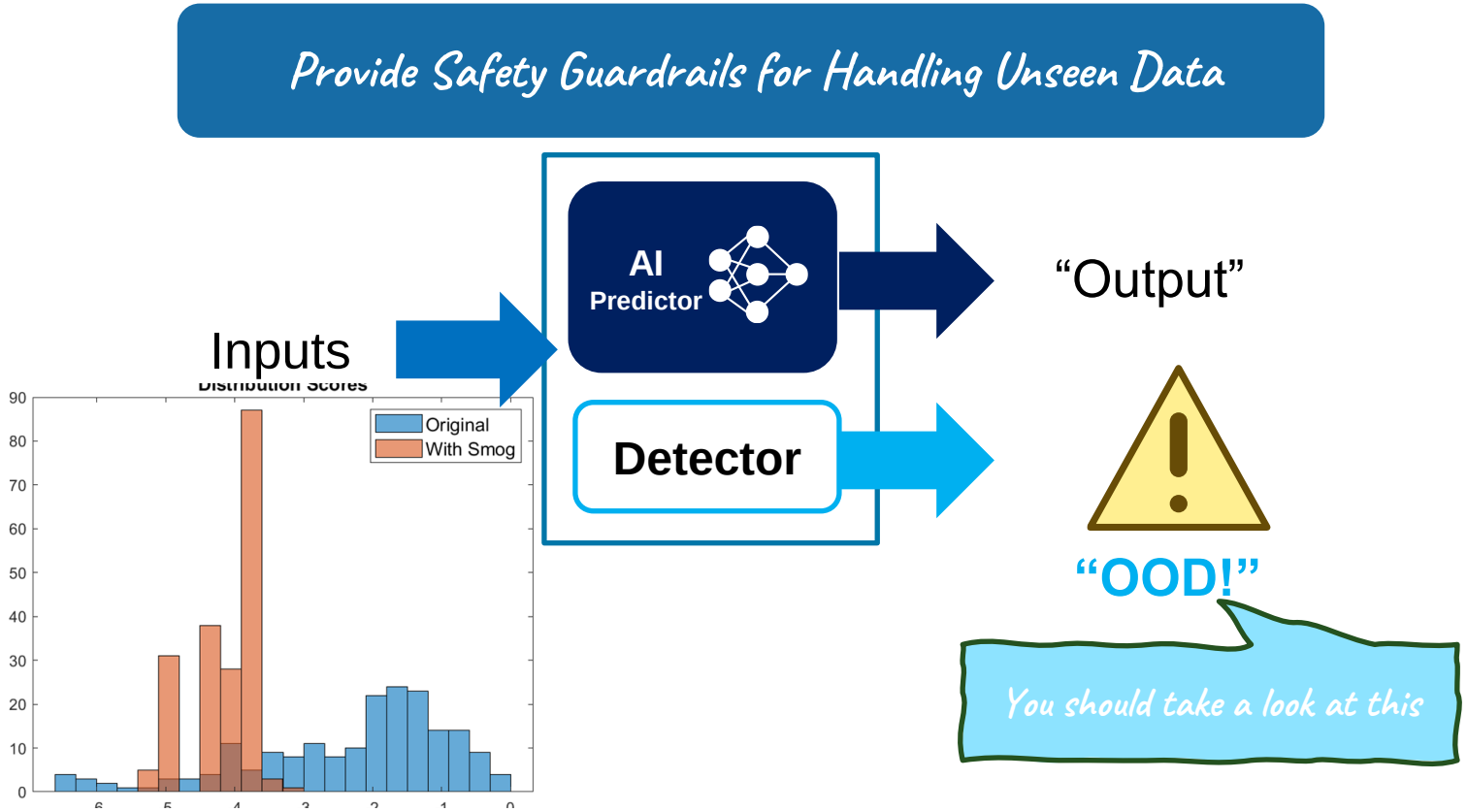
```
dlX = dlarray(X,'BC');
XLower = dlX-perturbation;
XUpper = dlX+perturbation;
[YLower,YUpper] = estimateNetworkOutputBounds(trainedNetwork,XLower,XUpper);
YLower = extractdata(YLower)';
YUpper = extractdata(YUpper)';
maxDeviation = max([abs(YLower-Y_pred);abs(YUpper-Y_pred)]);
```

Handling unseen data

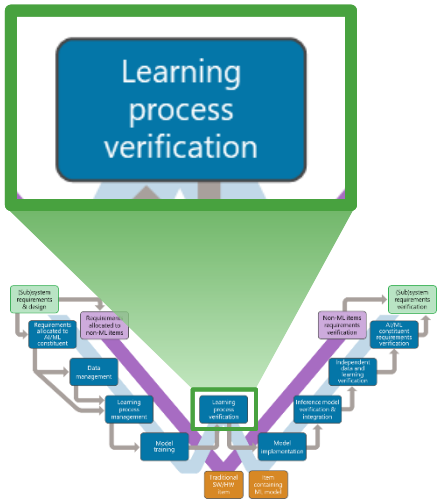
What if the AI model receives inputs that **it has never seen before**?

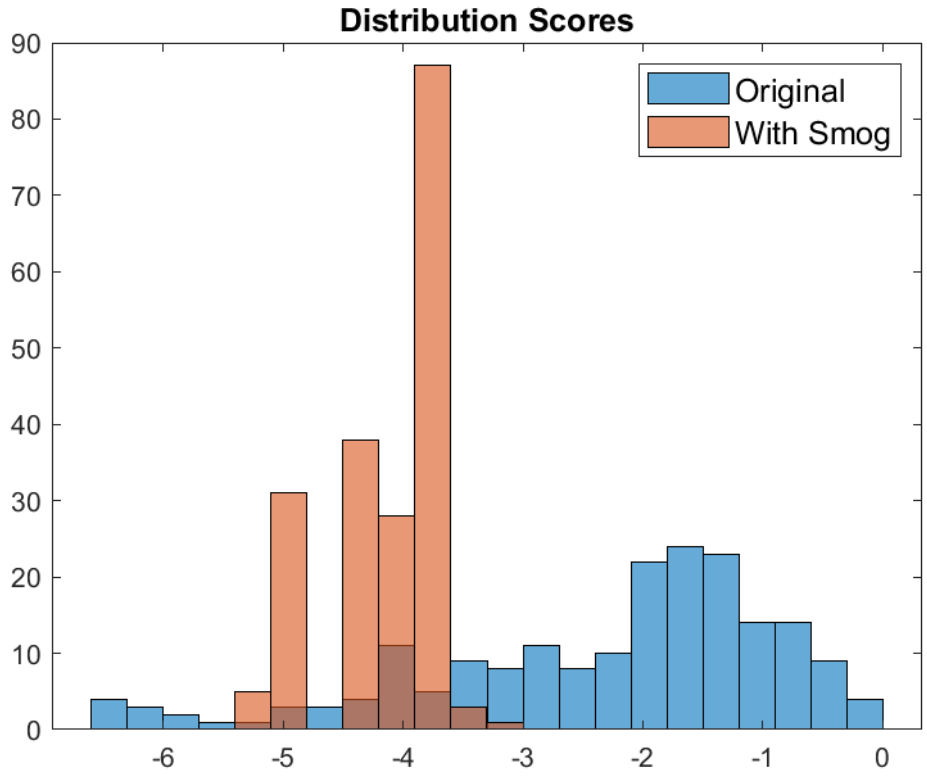
Can we **detect and flag outputs** from **out-of-distribution** (OOD) data?



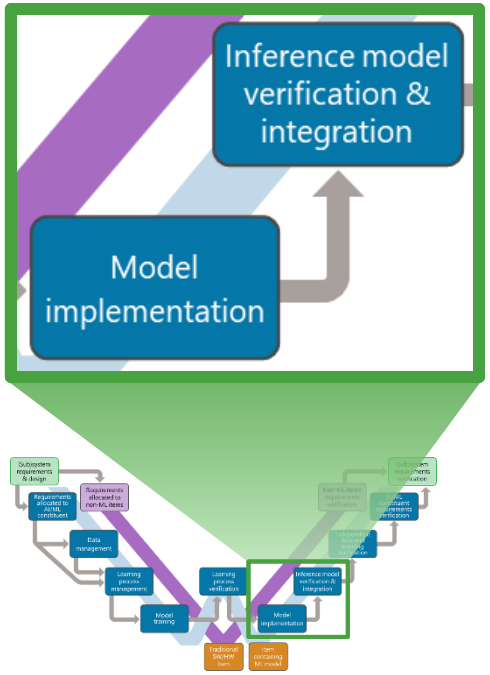


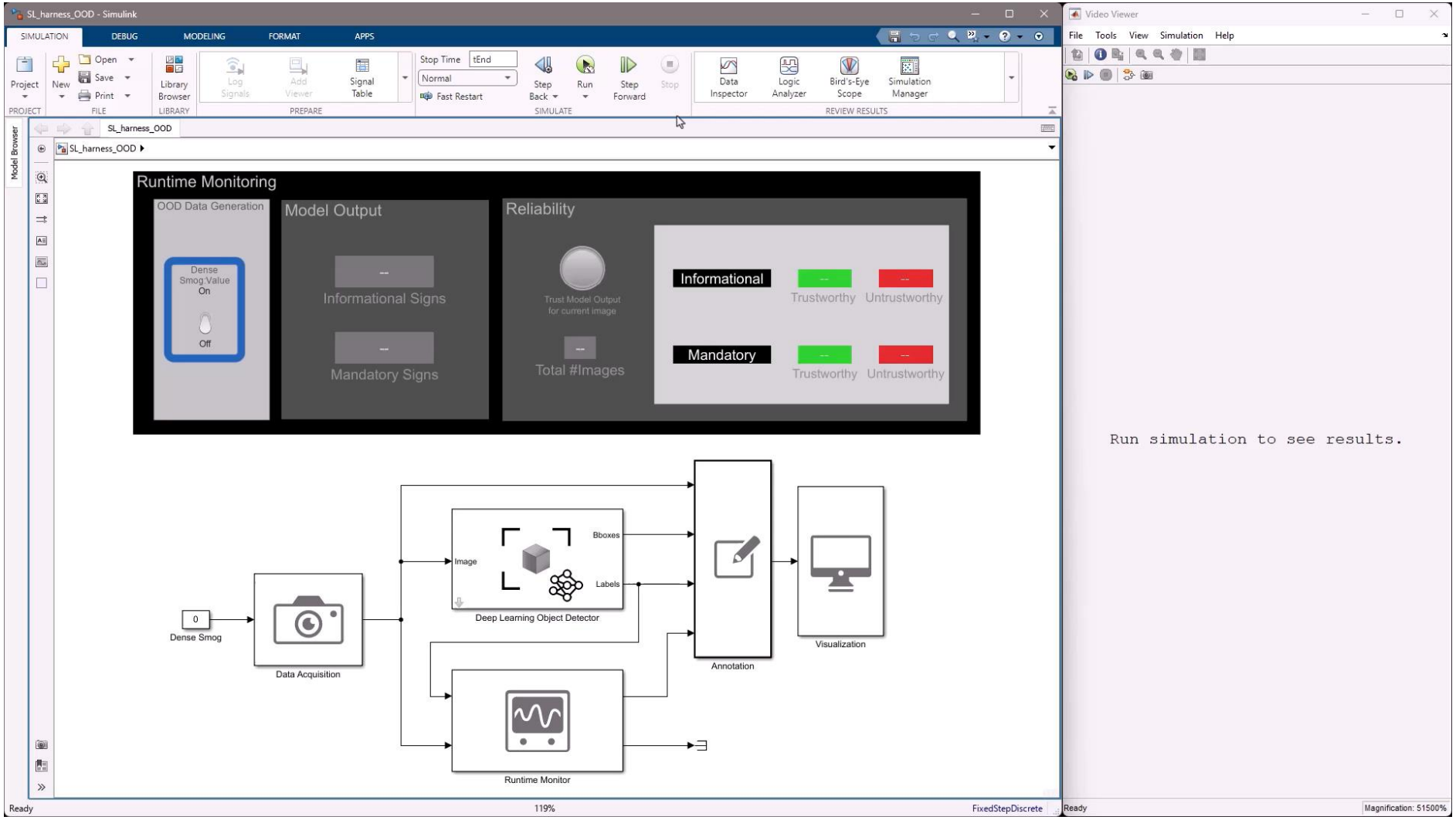
Identify unknown examples to the model and reject or transfer to a human for safe handling



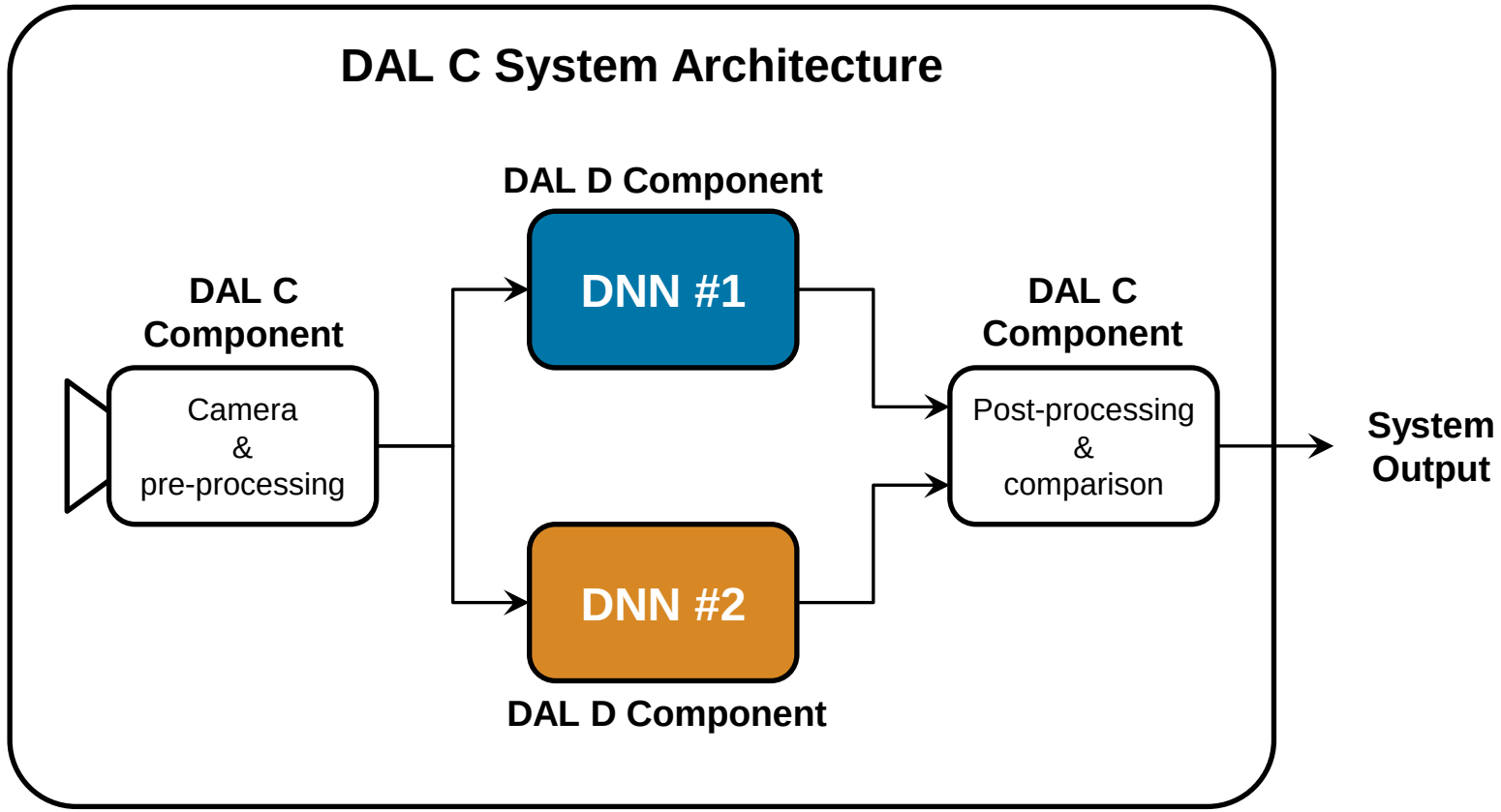


Integrate your AI model with Simulink with a Runtime Monitoring System

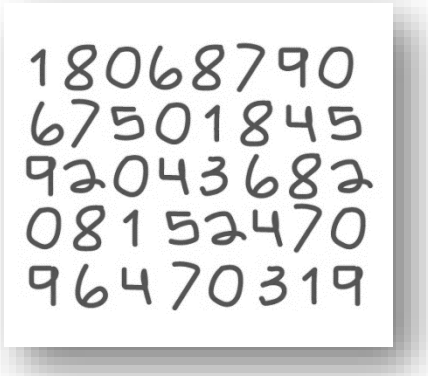




Architectural Mitigation through dissimilar DAL D Components

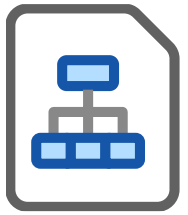


How to Achieve Dissimilarity?

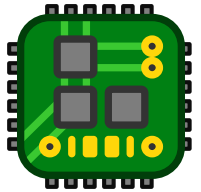


Data

ML Framework

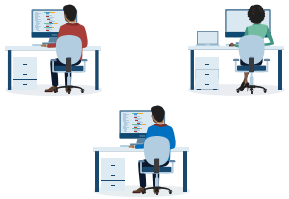


Model Architecture



Hardware

Software

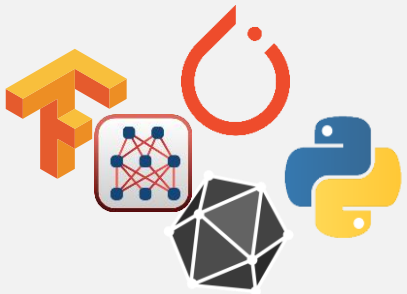


Human Independence

Common Mode Analysis (ARP4761)

Access AI models from other colleagues and the broader AI community

Interoperate with 3rd party framework

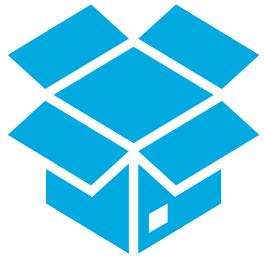


`importNetworkFromTensorFlow`
`importNetworkFromPyTorch`
`importNetworkFromONNX`

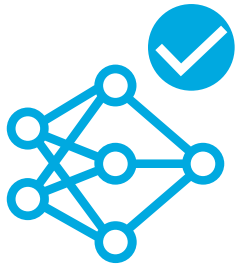
Convert a Deep Learning TensorFlow / PyTorch / ONNX model

Coexecute a TensorFlow or PyTorch or any Python model from MATLAB

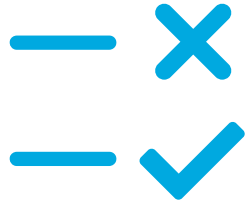
Perspectives: Three-pronged approach to explain, verify, and test AI-driven systems



White Box modeling

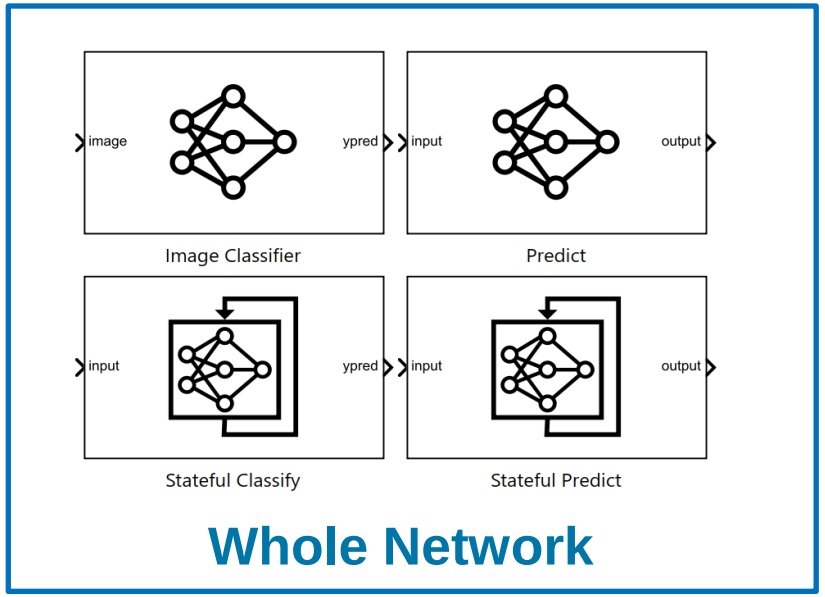


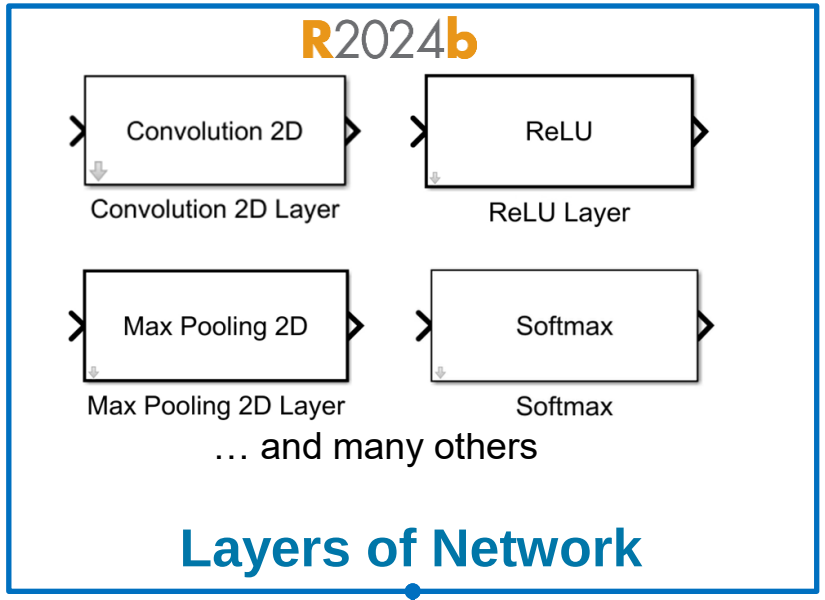
AI Verification

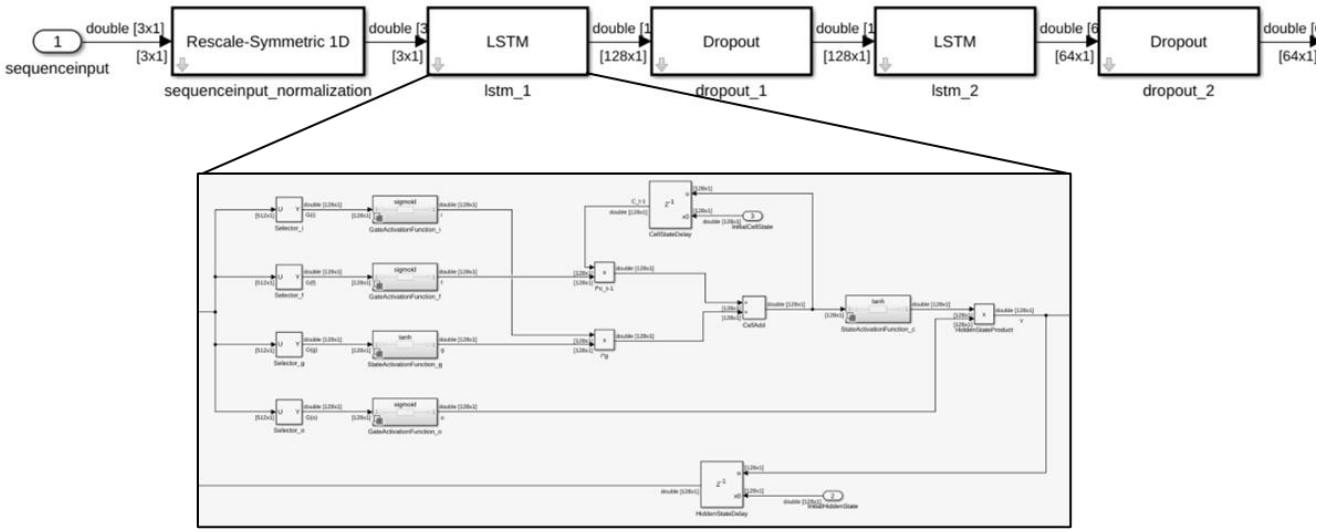


Simulation & Testing

Deep Learning: Whole Network and Layer Blocks



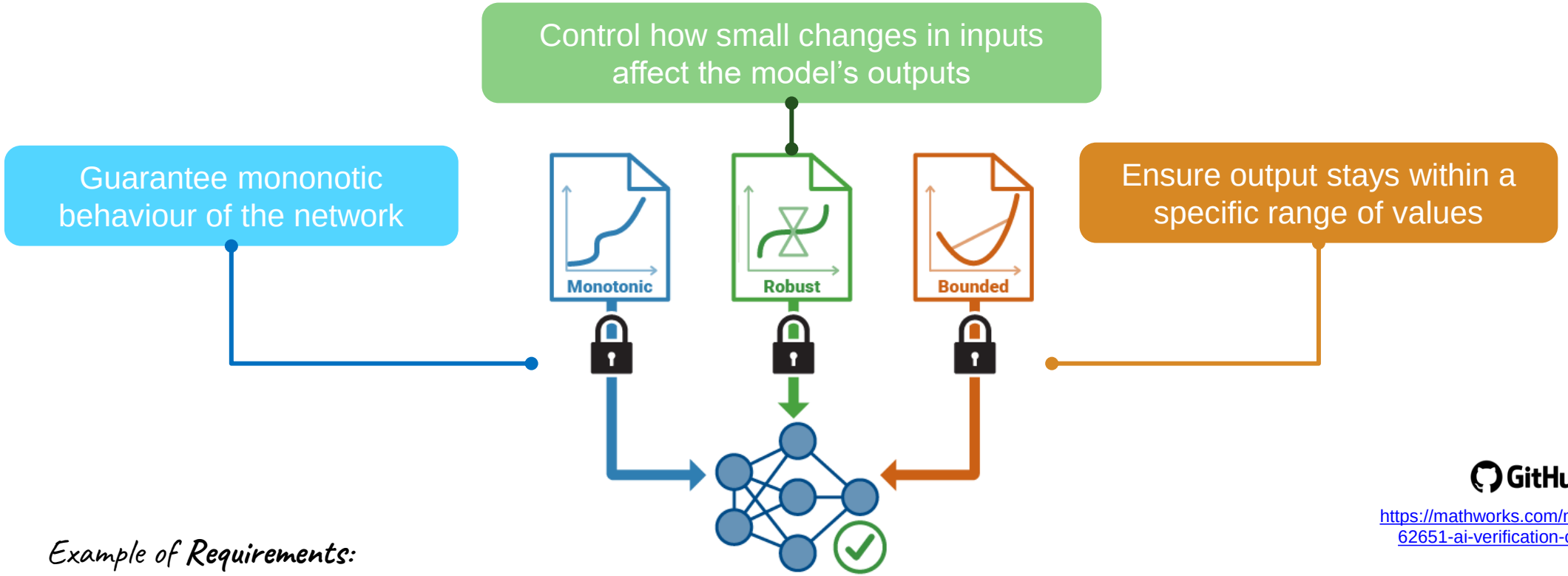




- *Visualization of the network*
- *Easy debugging*
- *Simulation: Continuous and discrete states*
- *C/C++ Code generation*
- *Simulink/Embedded Coder features for code customization and optimization*
- *AUTOSAR classic and adaptive*
- *Traceability of generated code*

How to interpret System Requirements as Network Constraints and enforce it in the learning process?

Constrained deep learning is an advanced approach to train deep neural networks by incorporating domain-specific constraints



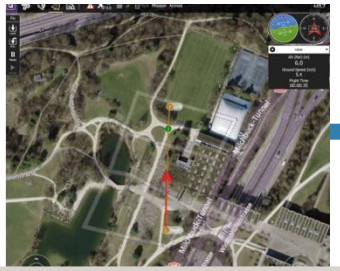
GitHub R2024a

https://mathworks.com/matlabcentral/fileexchange/162651-ai-verification-constrained-deep-learning

Example of ***Requirements:***

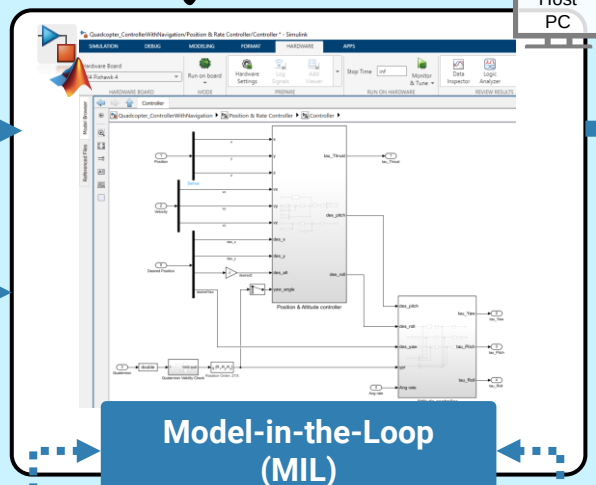
1. *Battery state-of-charge must increase in time as the battery charges*
2. *Steering angle change should not exceed a predefined rate (degrees per second) during automated lane correction*
3. *The aircraft airspeed must stay within operational limits to avoid structural damage (e.g., between minimum stall speed and maximum safe speed).*

Test AI-based systems with scenario-based testing



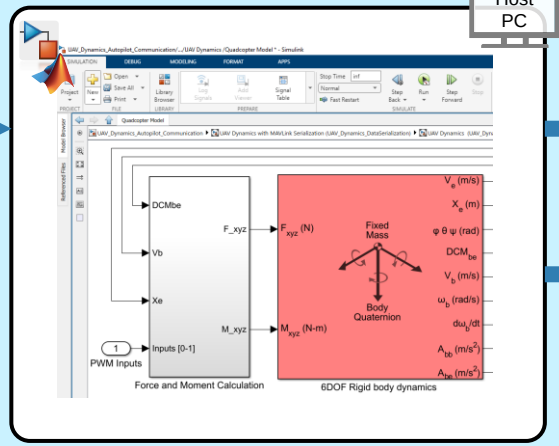
Ground Control Station

Flight Controller



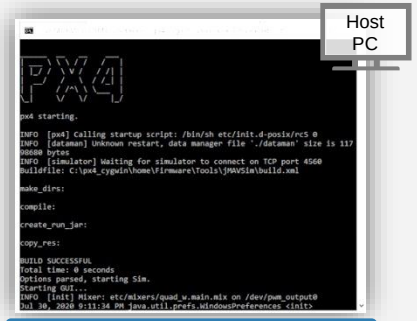
Model-in-the-Loop (MIL)

Plant Model



Scenario Simulation

C++ EXE on Host PC

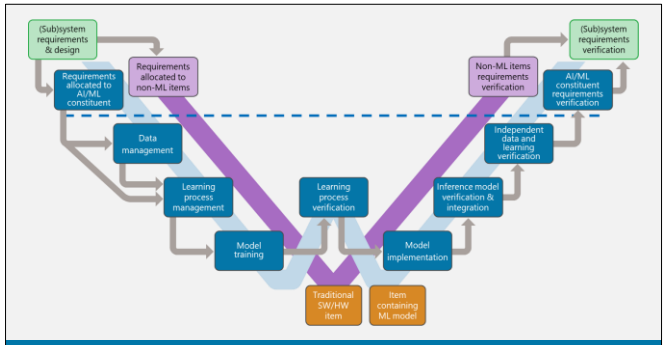


Software-in-the-Loop (SIL)

Deploy on Hardware

Hardware-in-the-Loop (HIL)

Key Takeaways



Need to incorporate AI in production in safety-critical situations; Certification Standards are in progress

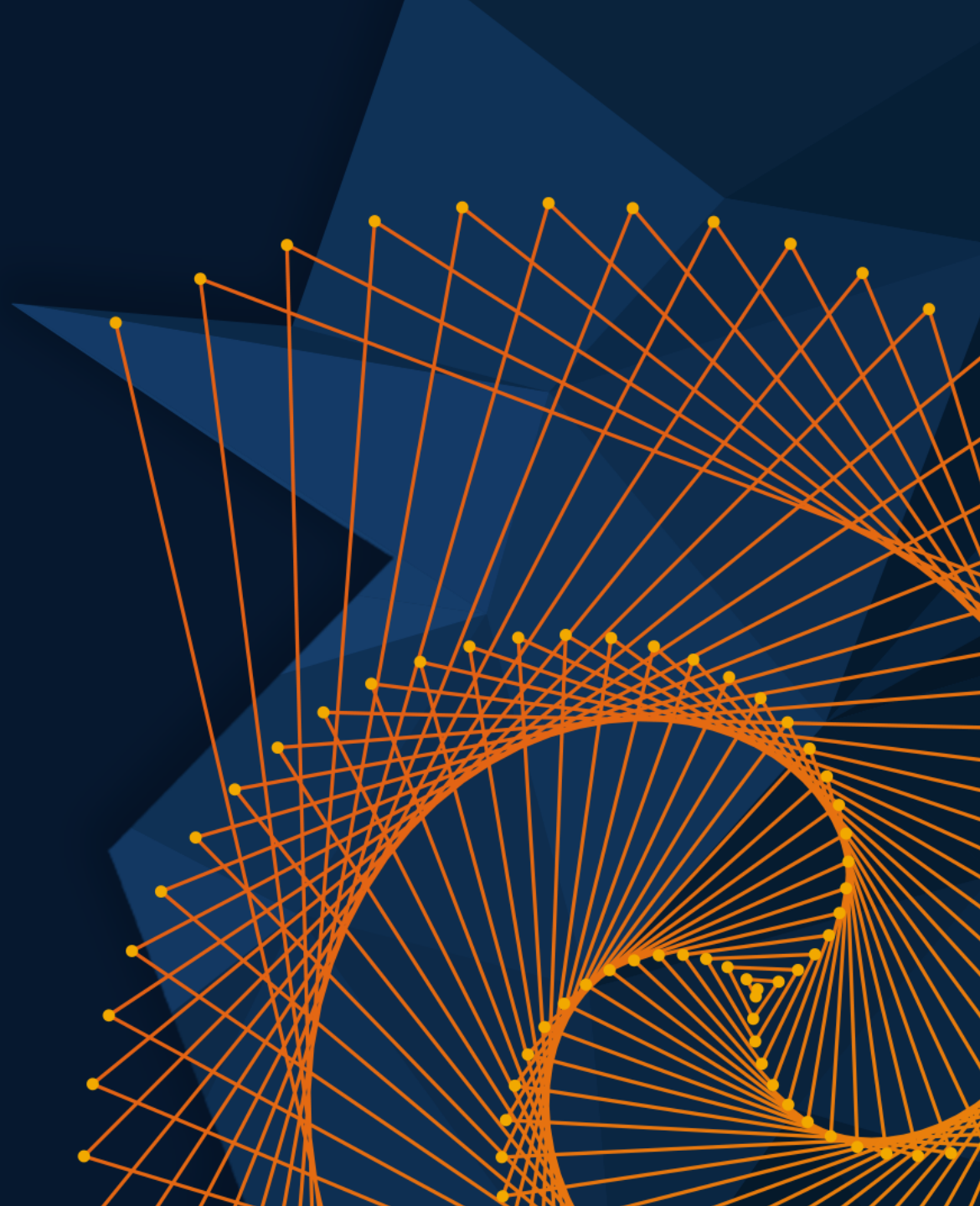
End-to-End Case Study to certify a DAL D Runway Sign Classifier Component

Some methods established, others in research; DAL C can be addressed today through Architectural Mitigation

We're here to help you navigate AI certification and verification challenges. Let's discuss your projects!

MATLAB EXPO
FRANCE

Merci!



MathWorks®