

Deploying PyTorch Models Where Failure Isn't an Option

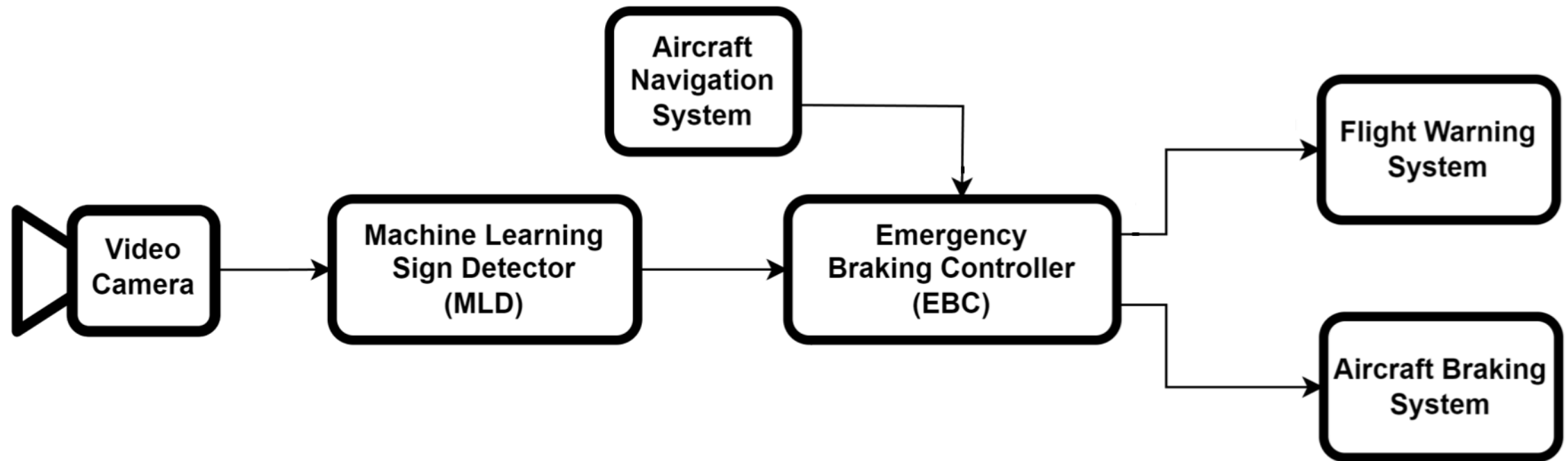


Ashvant Ram Selvam
Software Engineer,
MathWorks



Sarah Mohamed
Software Engineer,
MathWorks

What brings us to the conference?



Performance metrics are necessary, but not sufficient for safety-critical deployment

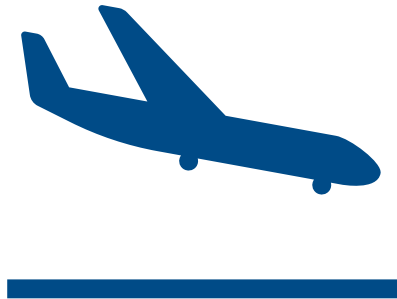
```
$ python train.py --model yolov8 --dataset runway_signs --epochs 200
[INFO] Epoch 1/200 loss: 4.21 mAP: 0.12
[INFO] Epoch 50/200 loss: 1.87 mAP: 0.61
[INFO] Epoch 100/200 loss: 0.98 mAP: 0.78
[INFO] Epoch 150/200 loss: 0.34 mAP: 0.91
[INFO] Epoch 200/200 loss: 0.28 mAP: 0.94
```

- ✓ mAP: 0.94
- ✓ F1: 0.91
- ✓ Inference latency: 12 ms

- ? Model works under all operating conditions
- ? System detects when model cannot be trusted
- ? Traceable requirements from safety objectives
- ? Evidence to pass regulatory review



**Software Considerations in
Aviation**



**Application to Emergency
Braking System**



**Rise of Verification and
Validation in AI**

Airworthy software must satisfy standards recognized by aviation authorities

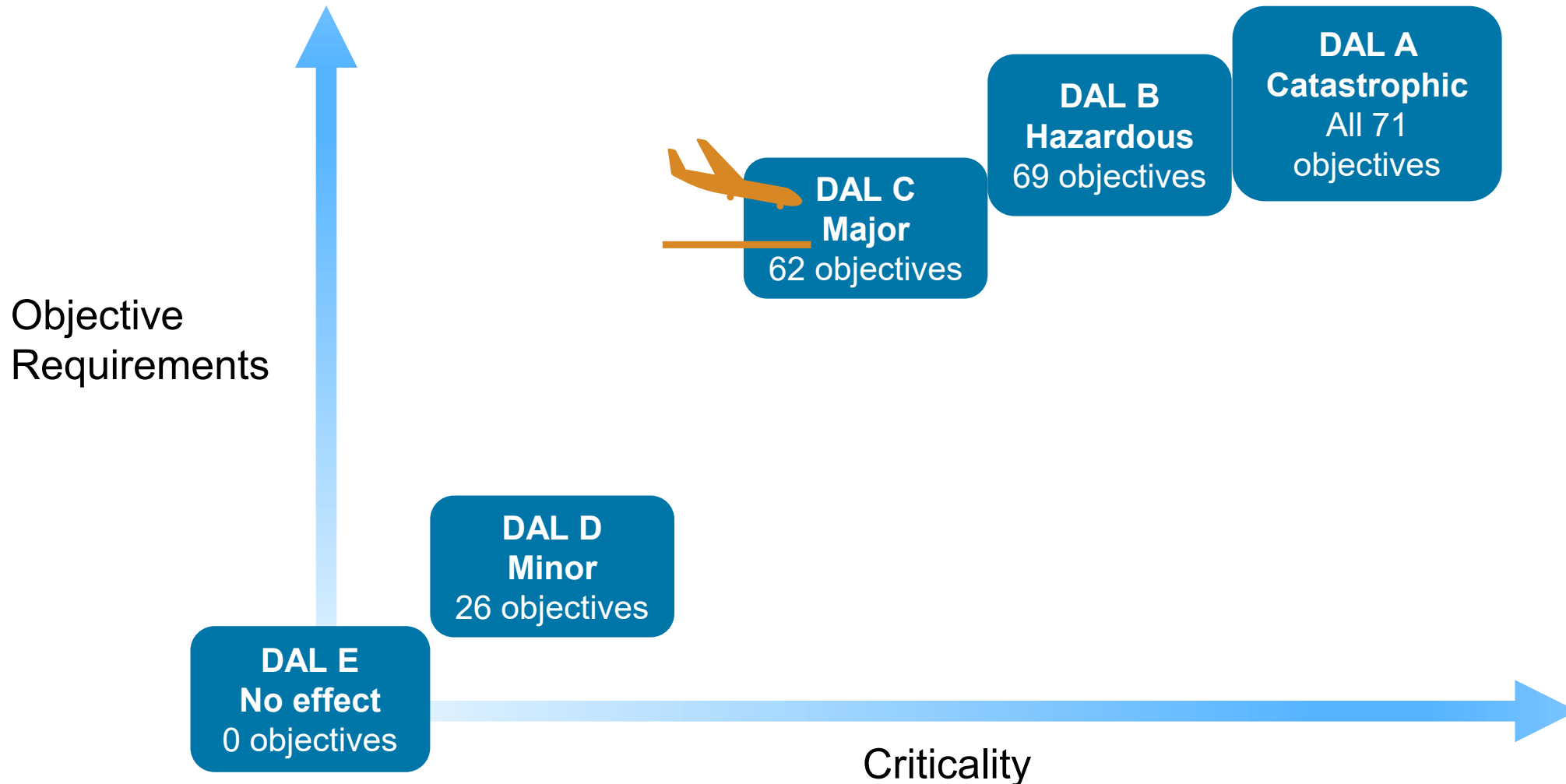
DO-178C

MB.6.3.4 Reviews and Analyses of Source Code

These review and analysis activities detect and report errors that may have been introduced during the software coding process. Primary concerns include correctness of the code with respect to the software requirements and the software architecture, and conformance to the Software Code Standards. These reviews and analysis activities are usually confined to the Source Code and confirm that the Source Code satisfies these objectives:

- 71 objectives across the development process
- Define **what** must be demonstrated, not necessarily **how**

The bar set by the standard increases based on potential consequences



Properties that apply to rules-based software don't always translate cleanly to AI models

Input Specification

</> Formally define valid inputs



Lack of established approaches to formally specify the space of all valid images

Traceability

</> Requirements trace to specific lines of code



Requirements can't be traced to individual weights

ARP-6983 is under development to extend DO-178C for AI. Concept papers are already publicly available

Given a system that includes a machine learning component, what evidence can we assemble to demonstrate it is safe?

- Certification objectives do not always prescribe exact methods, but we still need to demonstrate certain properties

What the standard expects

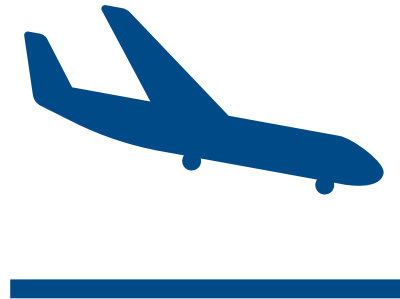
Executable object code is robust with respect to high-level requirements

What that means for a PyTorch model

Define concrete, testable requirements on the model. Show that it meets those requirements, and demonstrate that there are well-defined, deliberate responses to abnormal inputs.



Software Considerations in Aviation



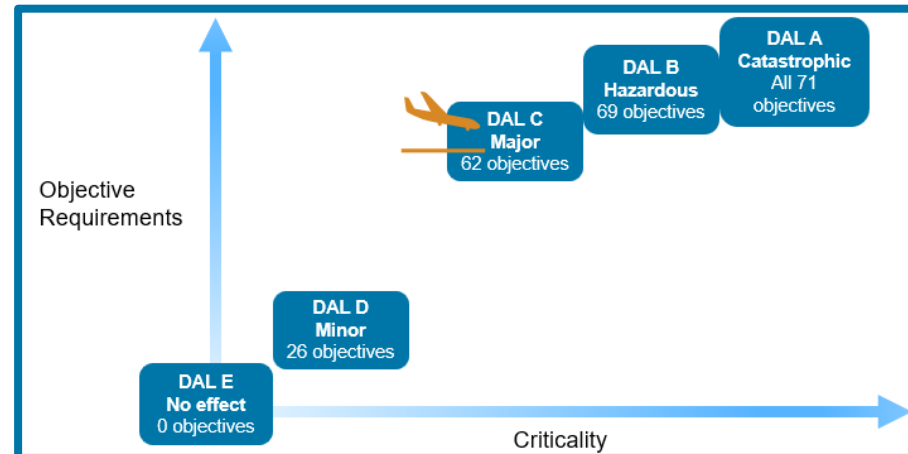
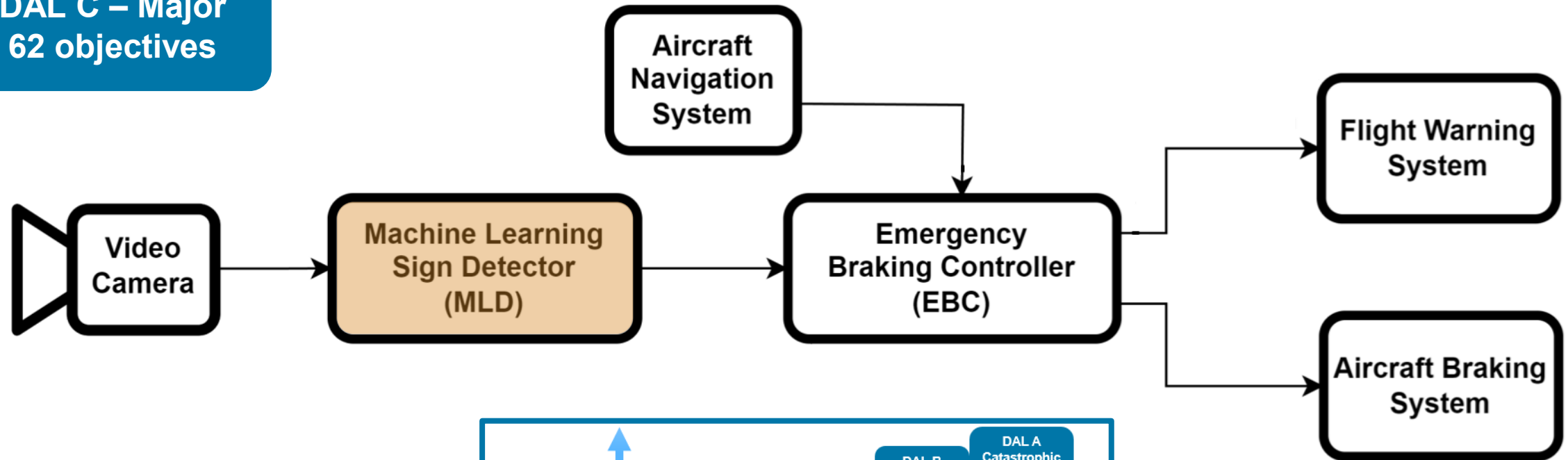
Application to Emergency Braking System



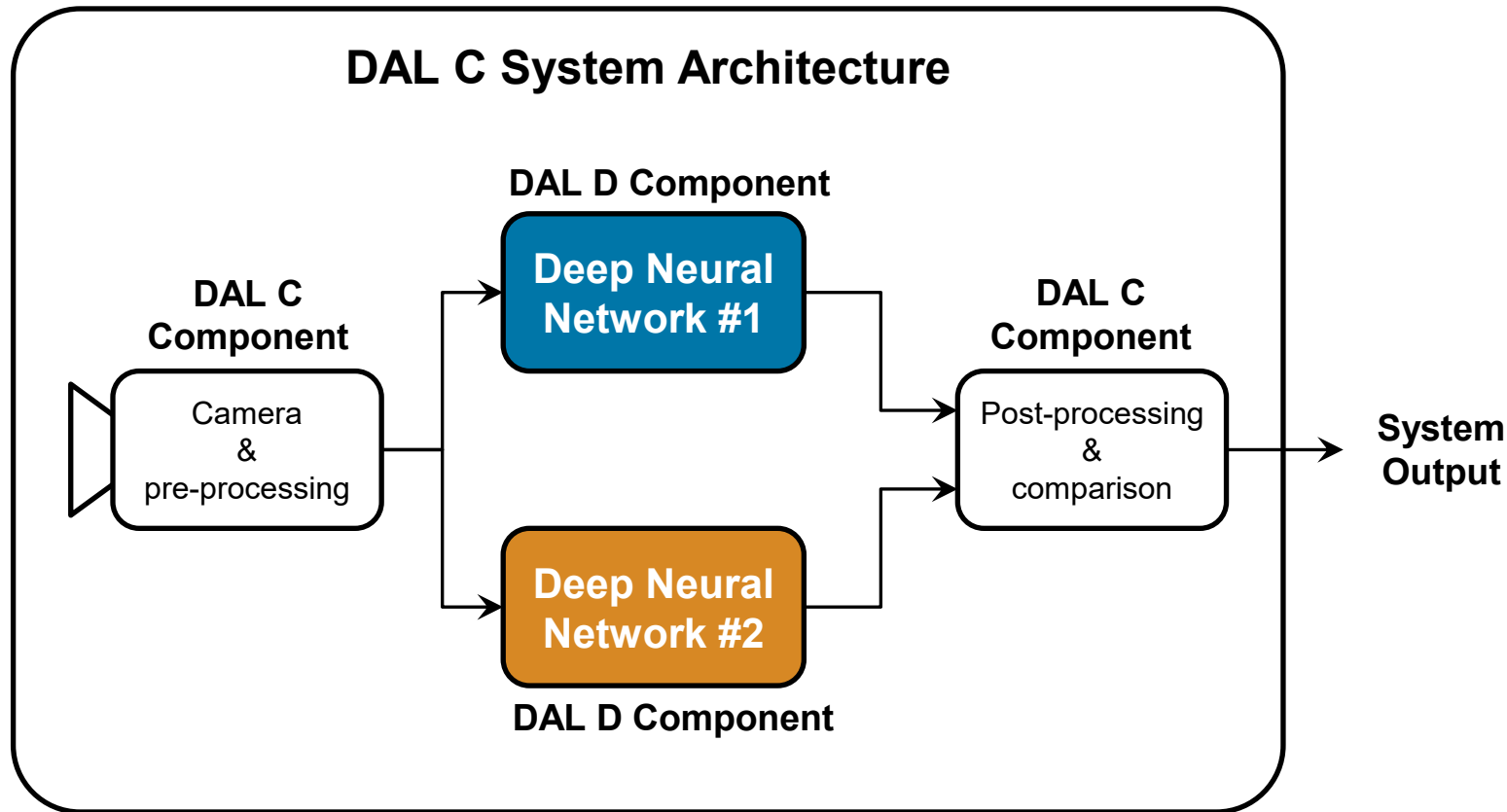
Rise of Verification and Validation in AI

Aircraft Emergency Braking System

DAL C – Major
62 objectives



Increasing assurance through dissimilar components

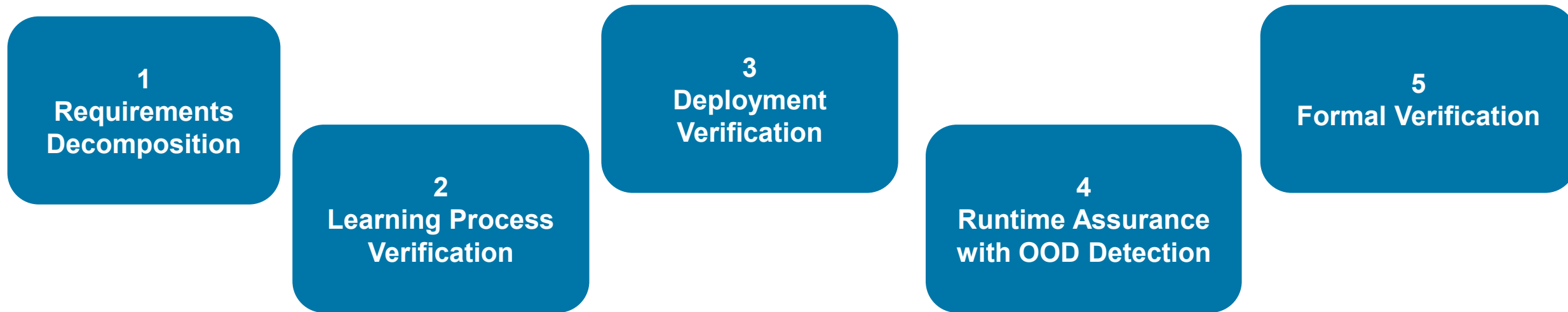


Axes of dissimilarity:

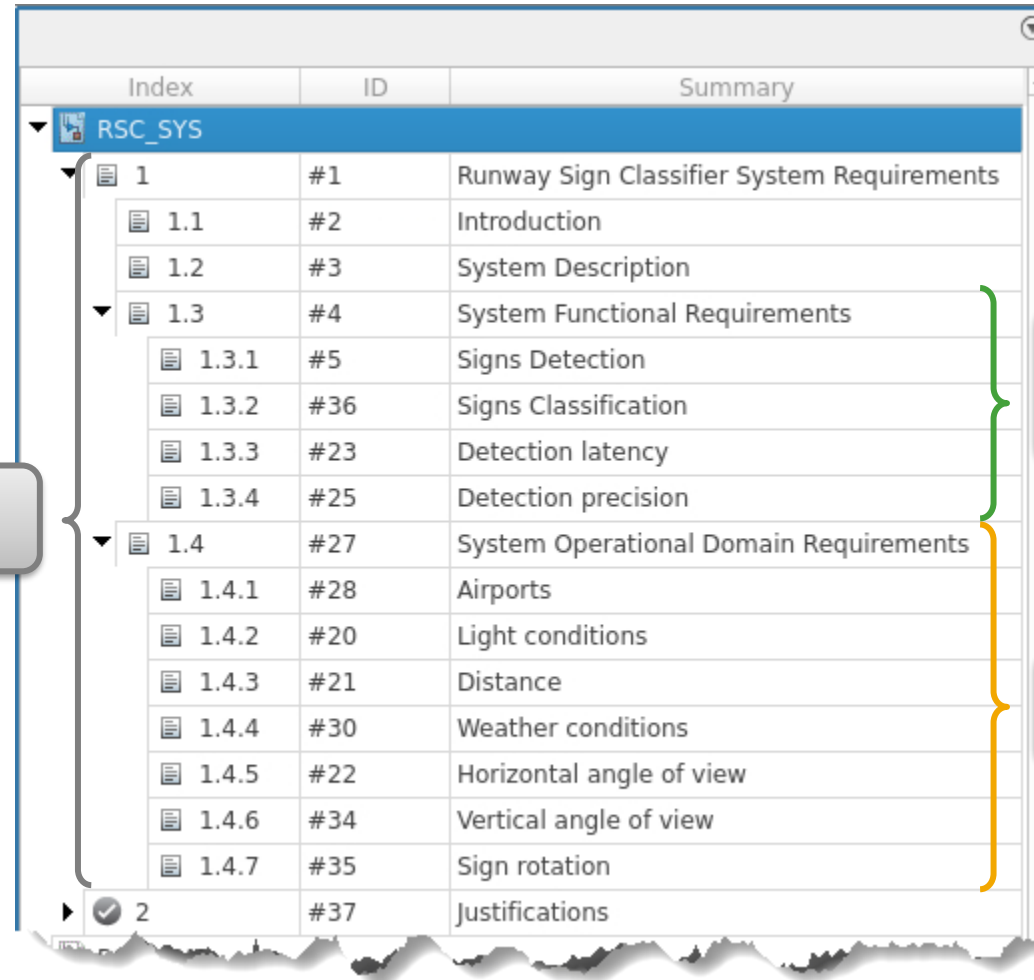
- Different development teams
- Different model architectures
- Different training frameworks
- Different datasets
- Different hardware

Fulfilling a safety requirement at DAL-D level of assurance

The system shall correctly detect runway signs under adverse weather conditions, including rain and reduced visibility



Define requirements for the system with testable, traceable properties



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

Machine Learning Component

Data

1
Requirements
Decomposition

Link system requirements to data requirements

The screenshot displays the MathWorks Requirements tool interface, illustrating the linkage between system requirements and data requirements.

Left Panel (System Requirements):

Index	ID	Summary
RSC_SYS		
1	#1	Runway
1.1	#2	Introduction
1.2	#3	System
1.3	#4	System
1.3.1	#5	Signs D
1.3.2	#36	Signs C
1.3.3	#23	Detecti
1.3.4	#25	Detecti
1.4	#27	System
1.4.1	#28	Airports
1.4.2	#20	Light co
1.4.3	#21	Distance
1.4.4	#30	Weather
1.4.5	#22	Horizon
1.4.6	#34	Vertical
1.4.7	#35	Sign rot
2	#37	Justifica
RSC_CMP		
RCS_DATA		
1	#1	RSC DN
1.1	#2	Introduc
1.2	#4	ML Data
1.2.1	KSFO	KSFO ai
1.2.2	KBOS	KBOS ai
1.2.3	KSAN	KSAN ai
1.2.4	FAIR	FAIR we

Middle Panel (Data Requirements):

Index	ID	Summary
1.1	#2	Introduction
1.2	#4	DNN requirements
2	#42	Justifications
RCS_DATA		
1	#1	RSC DNN Data Requirements
1.1	#2	Introduction
1.2	#4	ML Data Requirements
1.2.1	KSFO	KSFO airport
1.2.2	KBOS	KBOS airport
1.2.3	KSAN	KSAN airport
1.2.4	FAIR	FAIR weather condition
1.2.5	RAIN	RAIN weather condition
1.2.6	SNOW	SNOW weather condition
1.2.7	FOG	FOG weather condition
1.2.8	MRNG	MORNING time of the day
1.2.9	DUSK	DUSK time of the day
1.2.10	AFTN	AFTERNOON time of the day
1.2.11	DAWN	DAWN time of the day
1.2.12	DIST	Sign distance
1.2.13	ROT	Sign rotation
1.2.14	AGL	Elevation above ground level
1.2.15	SIDE	Lateral offset
2	#28	Justifications

Right Panel (Properties):

Properties

Type: Functional

Index: 1.2.4

Custom ID: FAIR

Summary: FAIR weather condition

Description | **Rationale**

Arial 10 B I U

The dataset shall include the images captured in FAIR weather condition

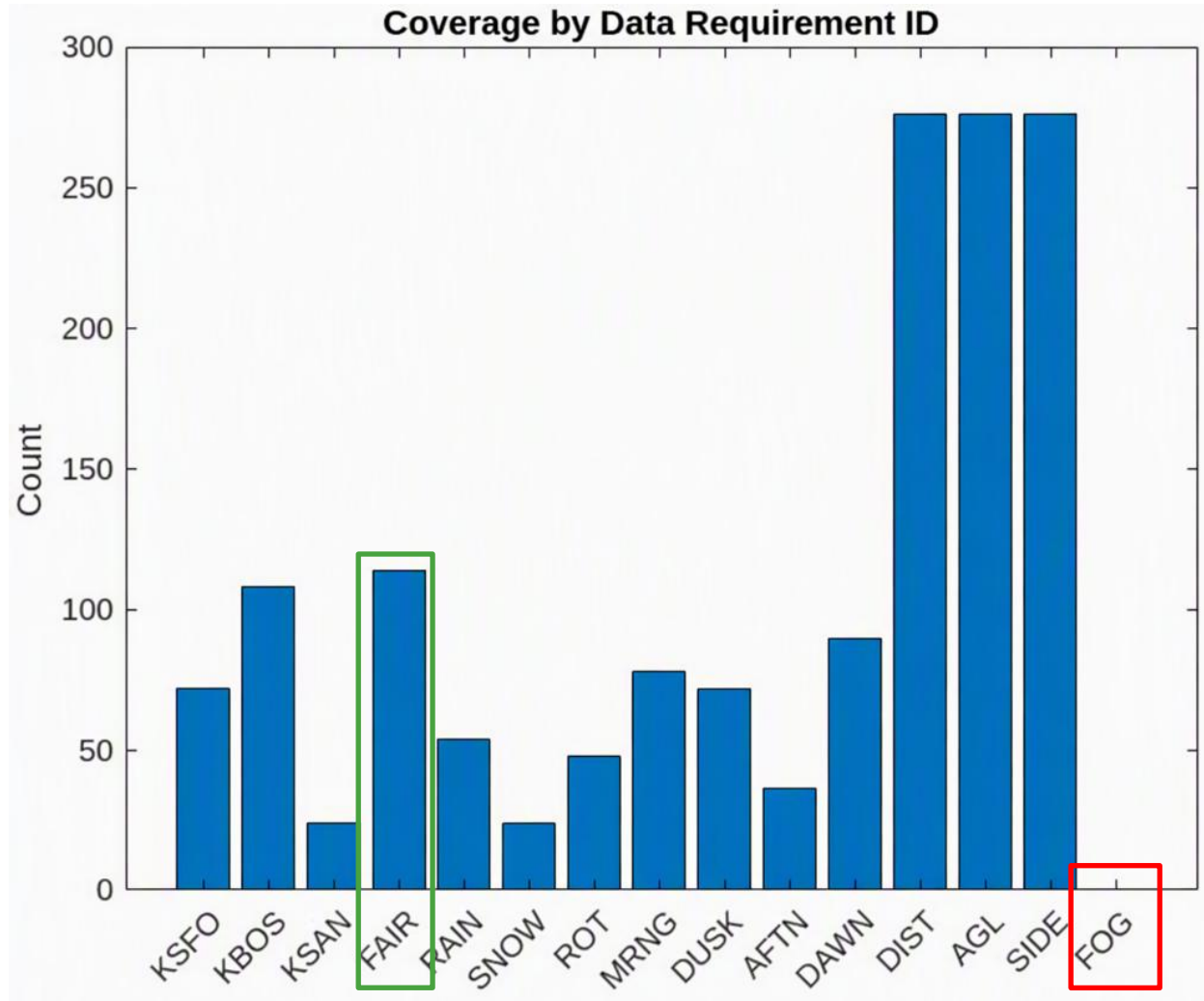
Keywords:

Revision information:

Links

- Implemented by: Dataset implementation
- Implements: Weather conditions
- Verified by: 739072.857.1 in dataTesting.m

Compute data coverage per data requirement



Examine data coverage in each of the operational conditions

Unfulfilled requirement on fog

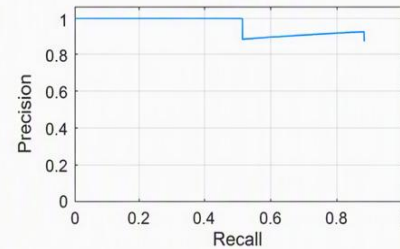
Verify the learning process

2 Learning Process Verification

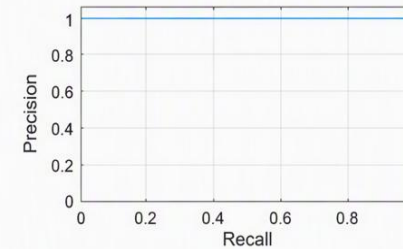
yolov2Detector
Test Image Sample



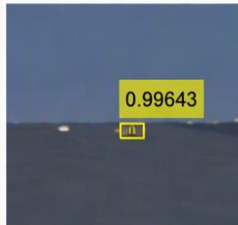
yolov2Detector
Average Precision = 0.85578
Class = FAIR



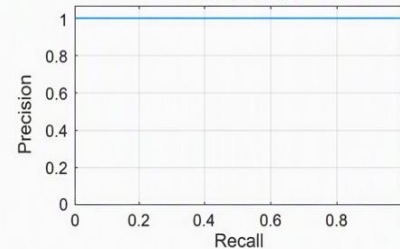
yolov2Detector
Average Precision = 1
Class = RAIN



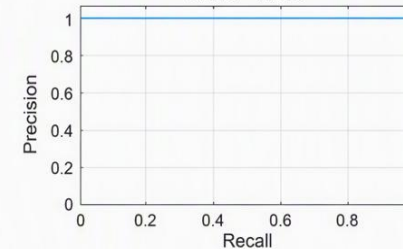
yolov3Detector
Test Image Sample



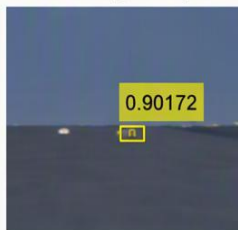
yolov3Detector
Average Precision = 1
Class = FAIR



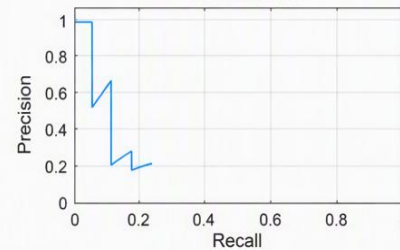
yolov3Detector
Average Precision = 1
Class = RAIN



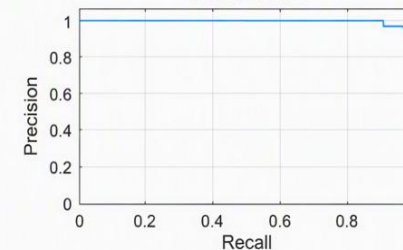
yolov4Detector
Test Image Sample



yolov4Detector
Average Precision = 0.13039
Class = FAIR



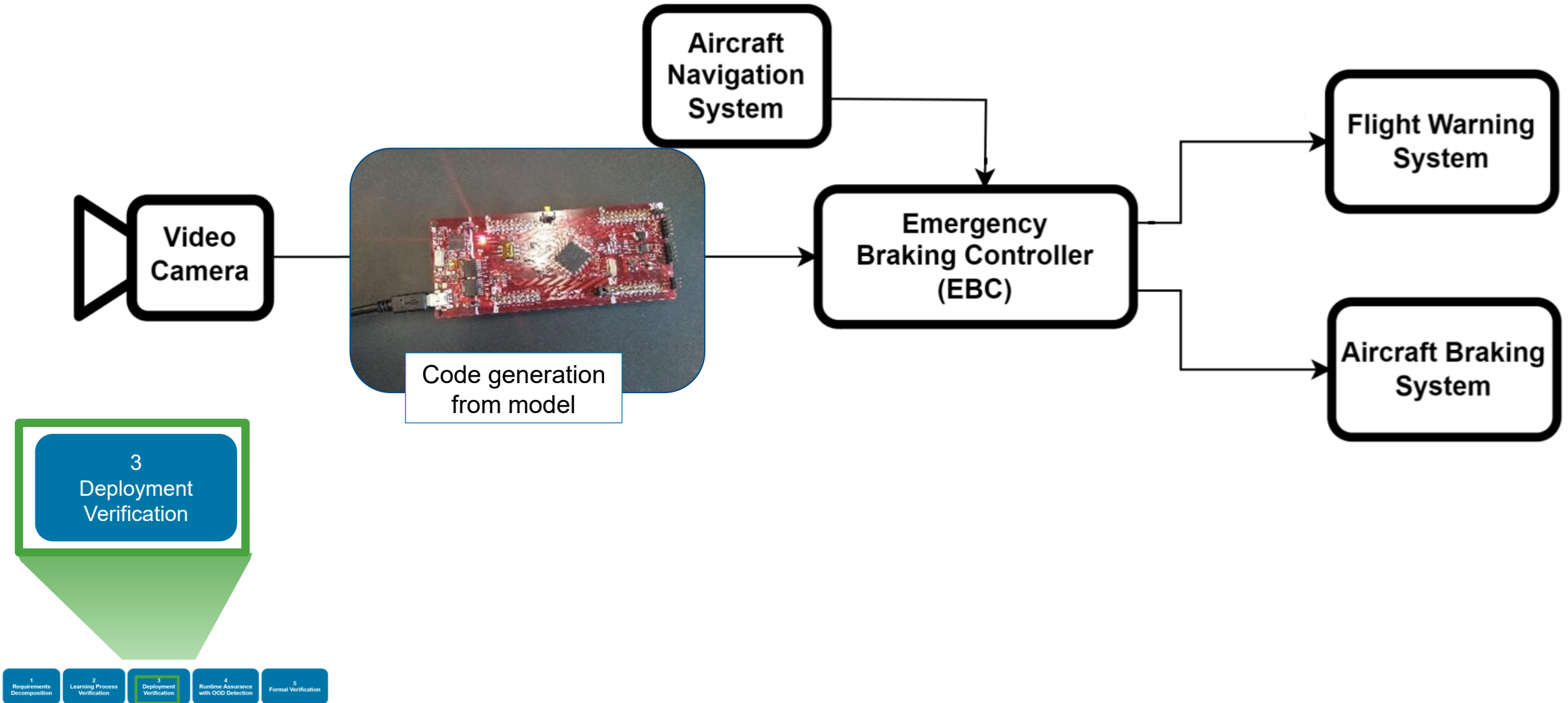
yolov4Detector
Average Precision = 0.97228
Class = RAIN



The ideal model delivers precision 1 at all recall levels.

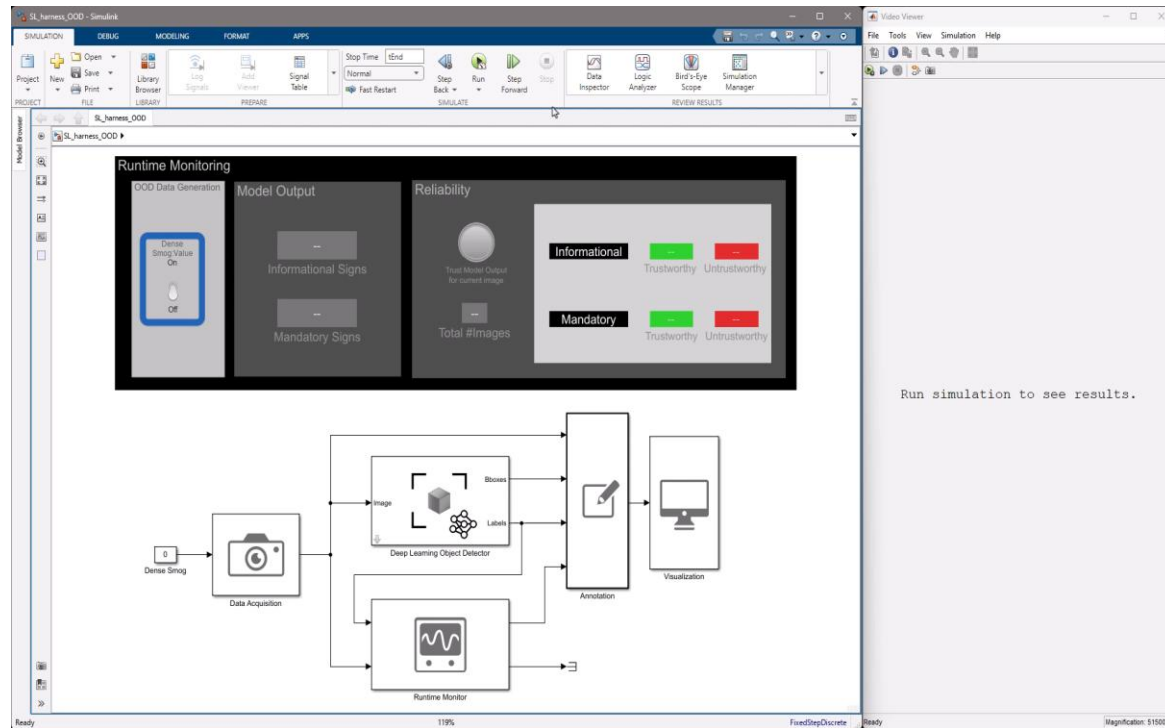
YOLO V3 meets the required average precision per class of at least 95%.

Processor-in-the-loop testing lets us test our system in a closed loop simulation

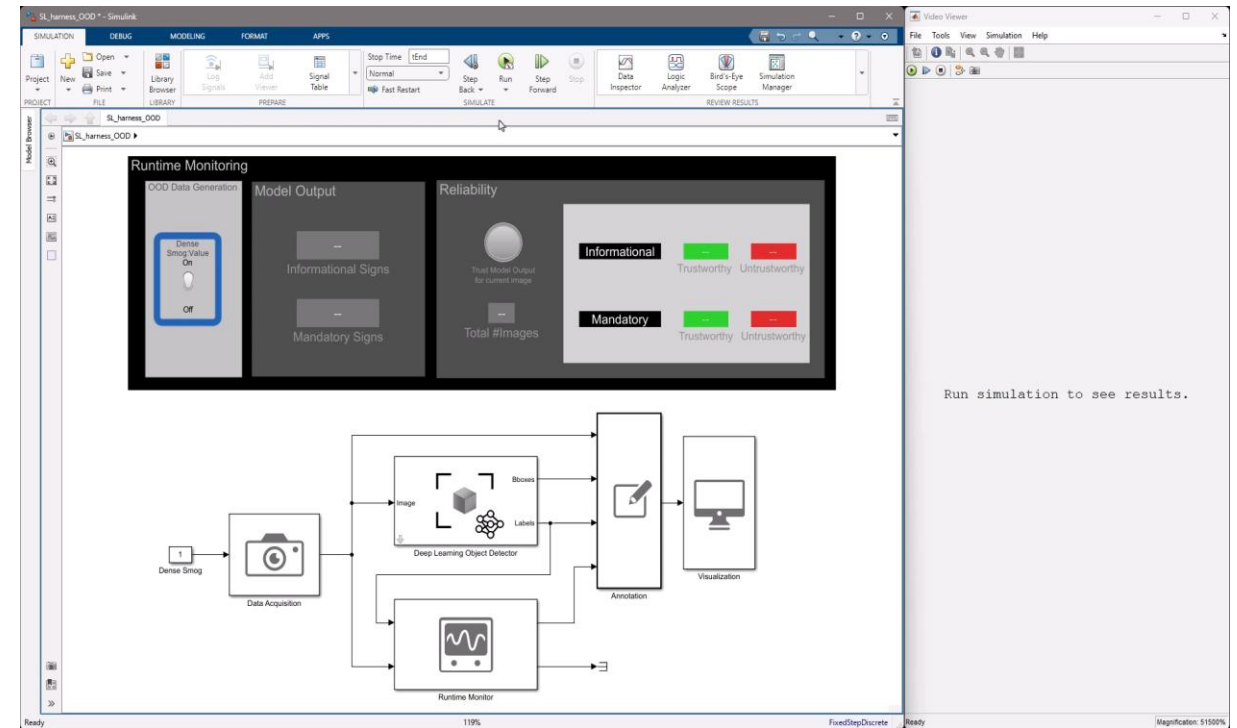


We might encounter conditions not represented in our development data

In-distribution ✓



Out-of-distribution ✗



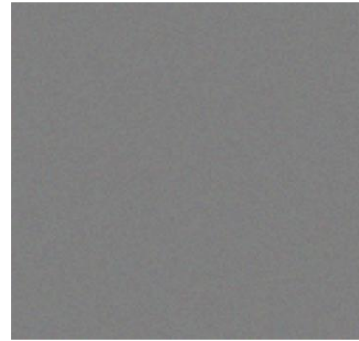
4
Runtime Assurance
with OOD Detection

Formal verification can be used to prove the absence of errors for all possible inputs

runway

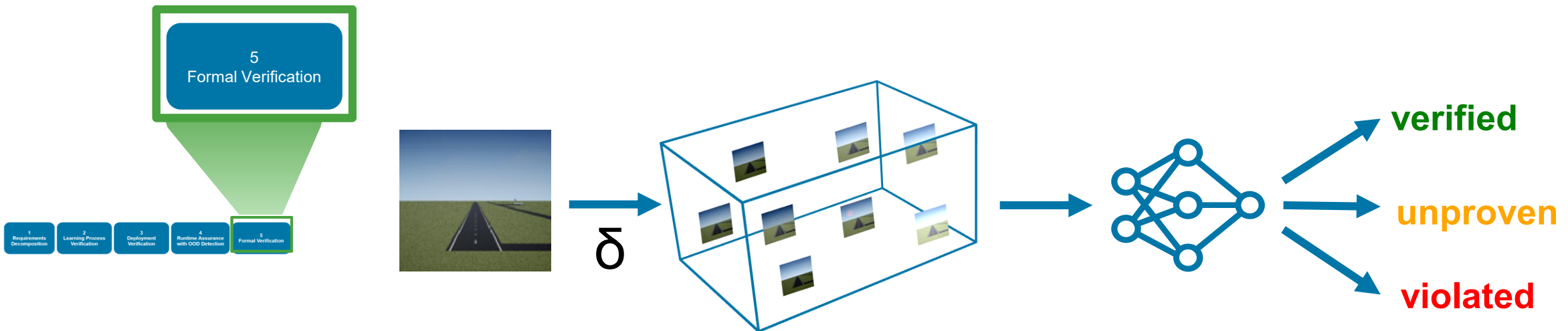


+



=

no runway



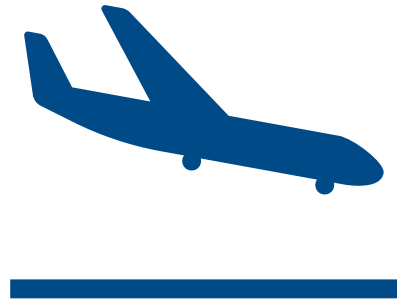
Fulfilling a safety requirement

The system shall correctly detect runway signs under adverse weather conditions, including rain and reduced visibility

Requirements Decomposition	Learning Process Verification	Deployment Verification	Runtime Assurance with OOD Detection	Formal Verification
Adverse weather conditions captured in data Bounding boxes stable within 5% noise	Model achieves at least 95% average precision under FAIR and adverse conditions	Processor-in-the-loop testing demonstrates correct system integration	System recognizes when conditions fall outside training distribution System falls back to manual human intervention when needed	Demonstrated robustness properties hold empirically



**Software Considerations in
Aviation**



**Application to Emergency
Braking System**



**Rise of Verification and
Validation in AI**

Industry and academia working together: VNN-COMP

- Verified Neural Networks Competition is one example of how industrial needs and research are working together to solve formal verification problems.
 - Alpha-beta-CROWN: leading verification algorithm and winner of VNN-COMP across several years
 - Industry contribute benchmarks and sponsor challenges to help advance research



Winner of International Verification
of Neural Networks Competitions
(VNN-COMP 2021 - 2026)

Open problems and questions

- Formal verification methods for vision-scale models
- Meaningful coverage metrics for neural networks
- Standard practices for what regulatory bodies will accept as sufficient evidence

MathWorks is contributing to evolving standards:

- ARP-6983 is under development to extend DO-178C for AI. Concept papers are already publicly available

Questions? Connect with us!



Ashvant Ram Selvam
aselvam@mathworks.com
<https://www.linkedin.com/in/ashvant>



Sarah Mohamed
smohamed@mathworks.com
<https://www.linkedin.com/in/sarah-m-mohamed>