ESWEEK 2024 Tutorial

Deploying Acoustic-Based Predictive AI for Machine Health using Model-Based Design

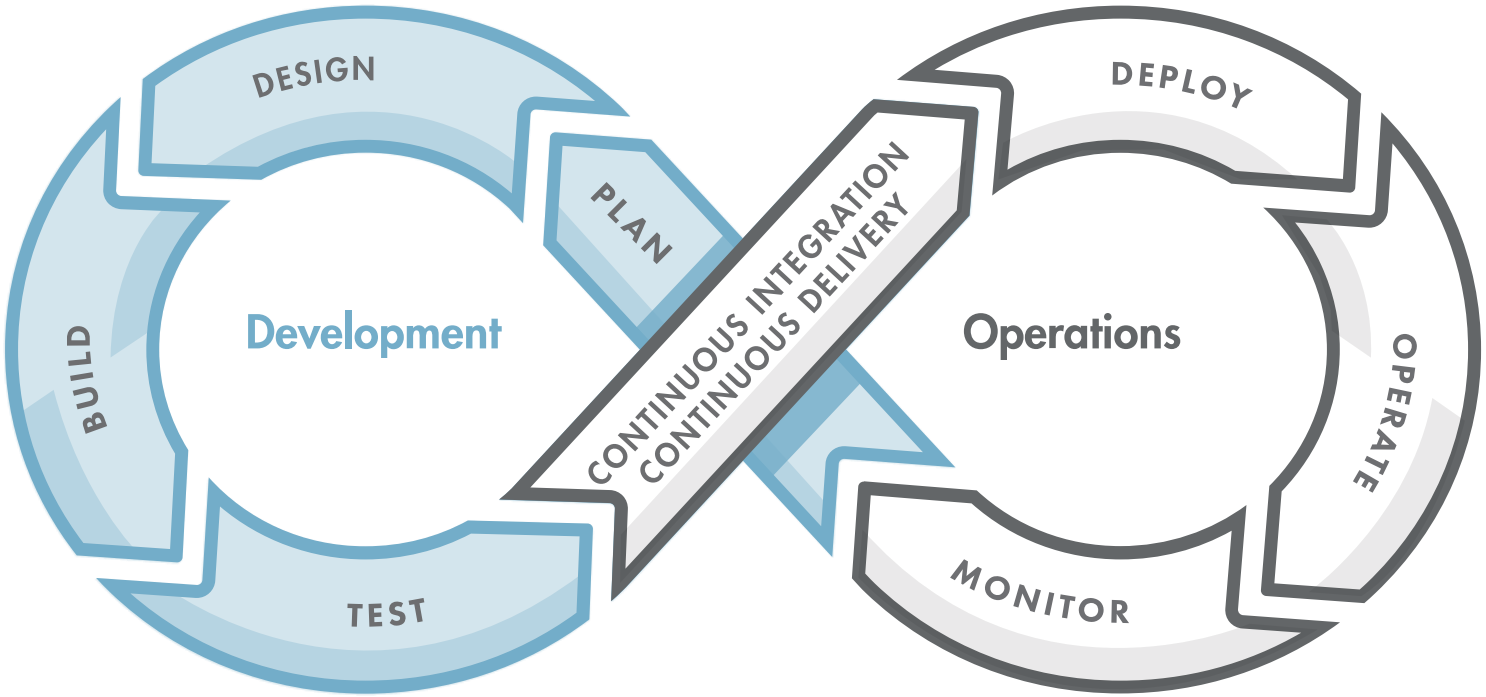
Brenda Zhuang, Akshay Rajhans, Tianyi Zhu

Sep 29, 2024

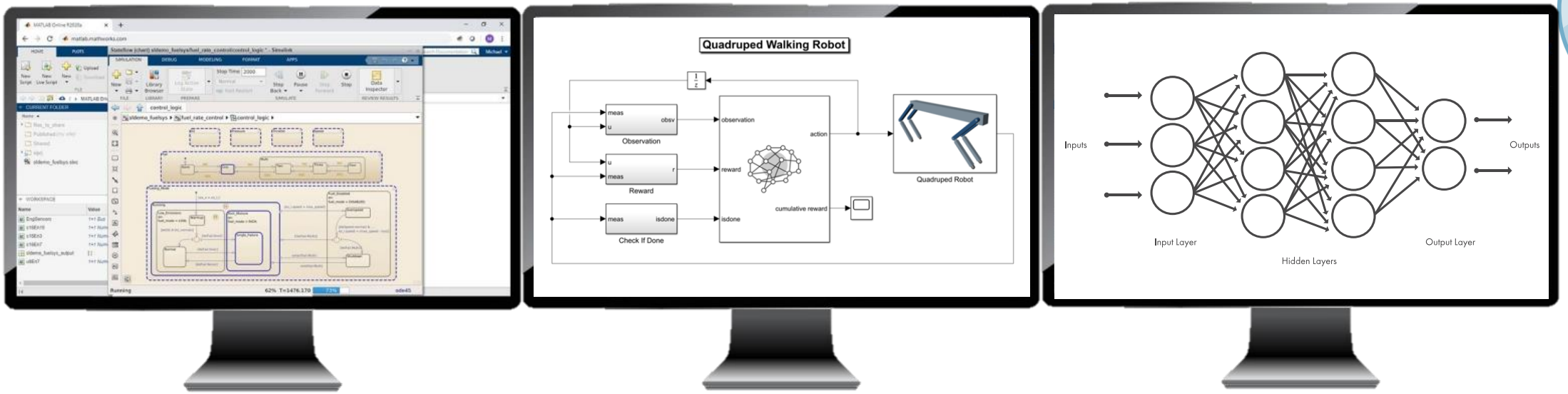
Preliminaries

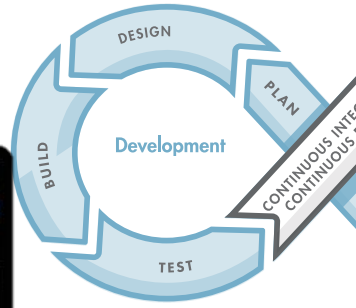
Model-Based Design and Digital Twins

Model-Based Design and Operation



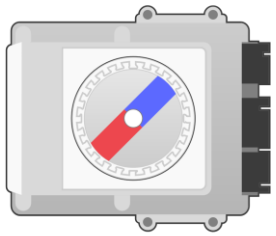
Code generation – auto-generate code from a model





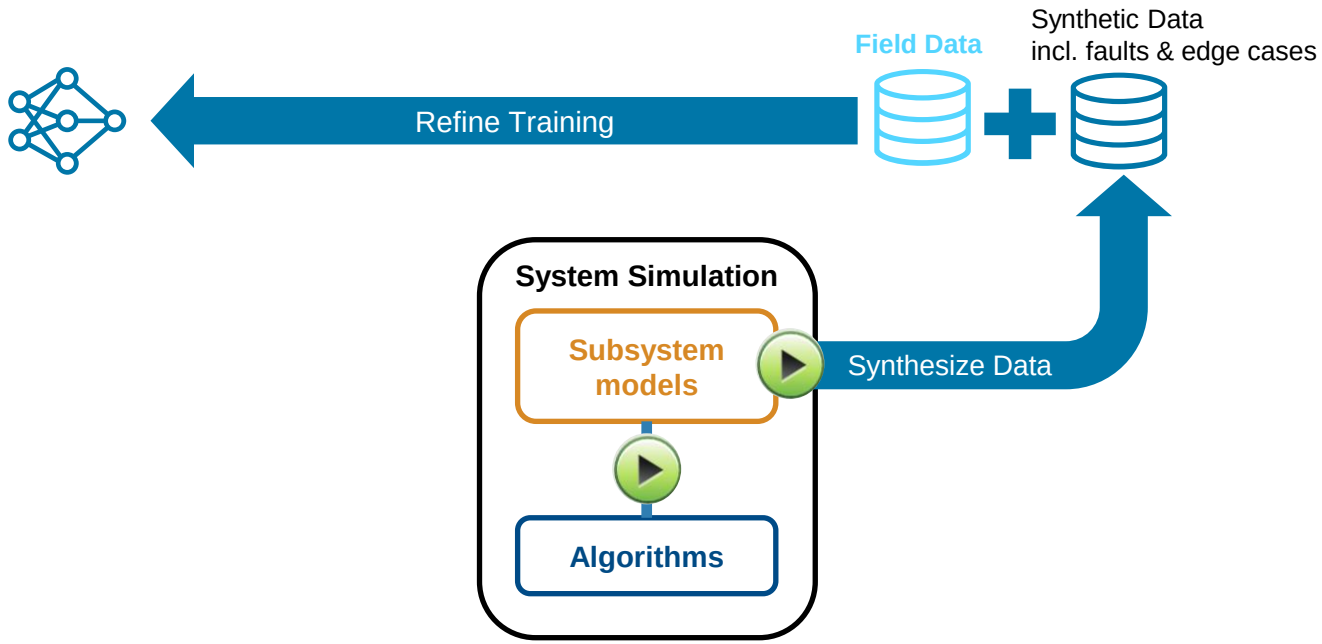
Unified internal representation

C Code | **C++ Code** | **HDL Code** | **GPU Code** | **PLC Code**

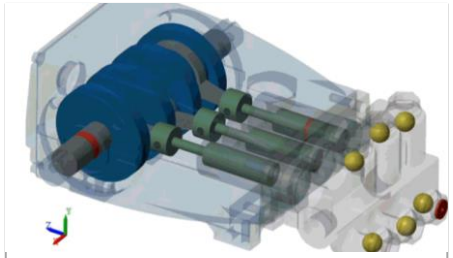


Model-Based Design and Operation with AI in the mix

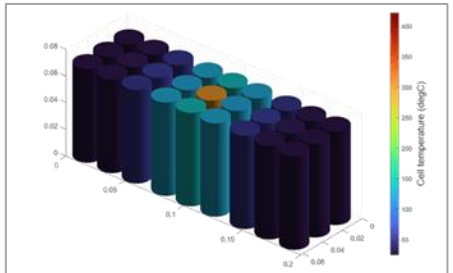
1 Simulations close data gaps and verification gaps for AI algorithms



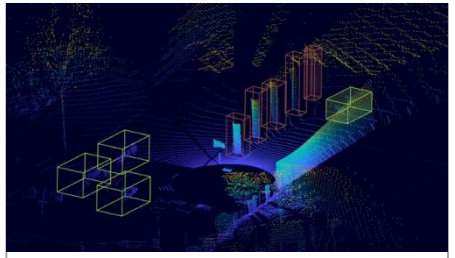
1 Simulations close data gaps and verification gaps for AI algorithms



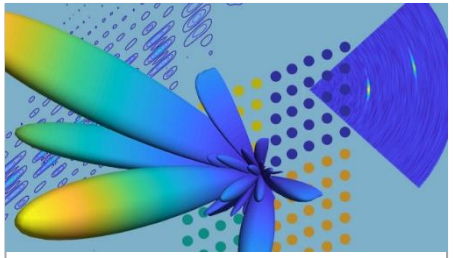
Predictive Maintenance
Anomaly Detection and Condition Monitoring



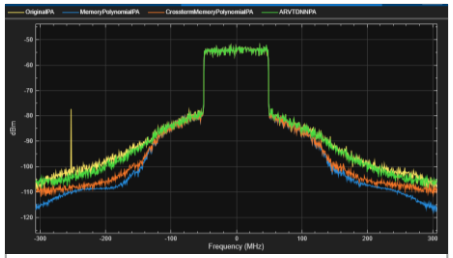
Electrification
Battery Management System and Grid Modernization



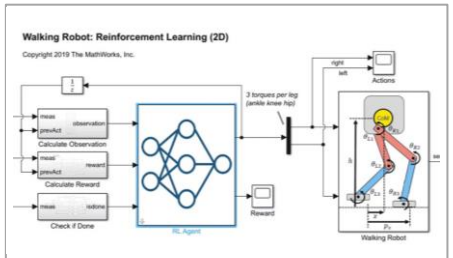
Lidar
3D Point Cloud Object Detection



Radar
Waveform Synthesis and Classification

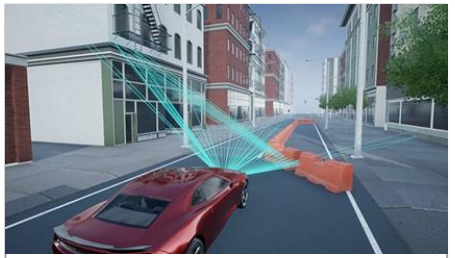


Wireless Comms
5G Channel Estimation and Accuracy Enhancement

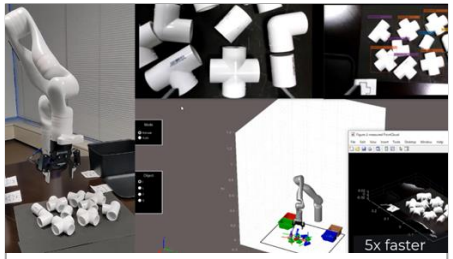


Control Systems
Reduced Order Modeling and AI-Based Controls

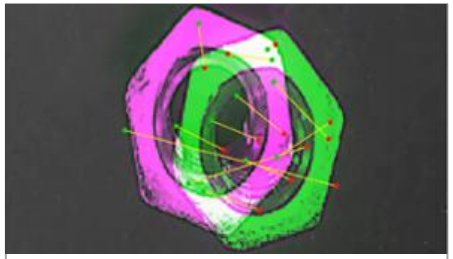
Aerospace Systems
Autonomous Navigation and Guidance



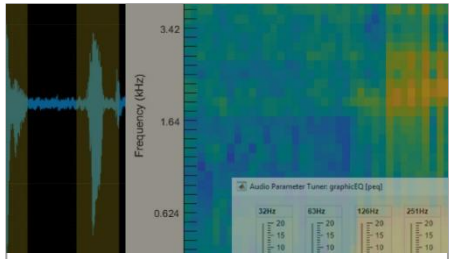
Automated Driving
Driving Scenario Data Generation



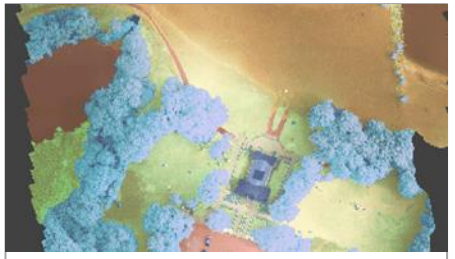
Robotics
Vision, Planning, and Manipulation



Industrial Automation
Visual Inspection and Defect Detection

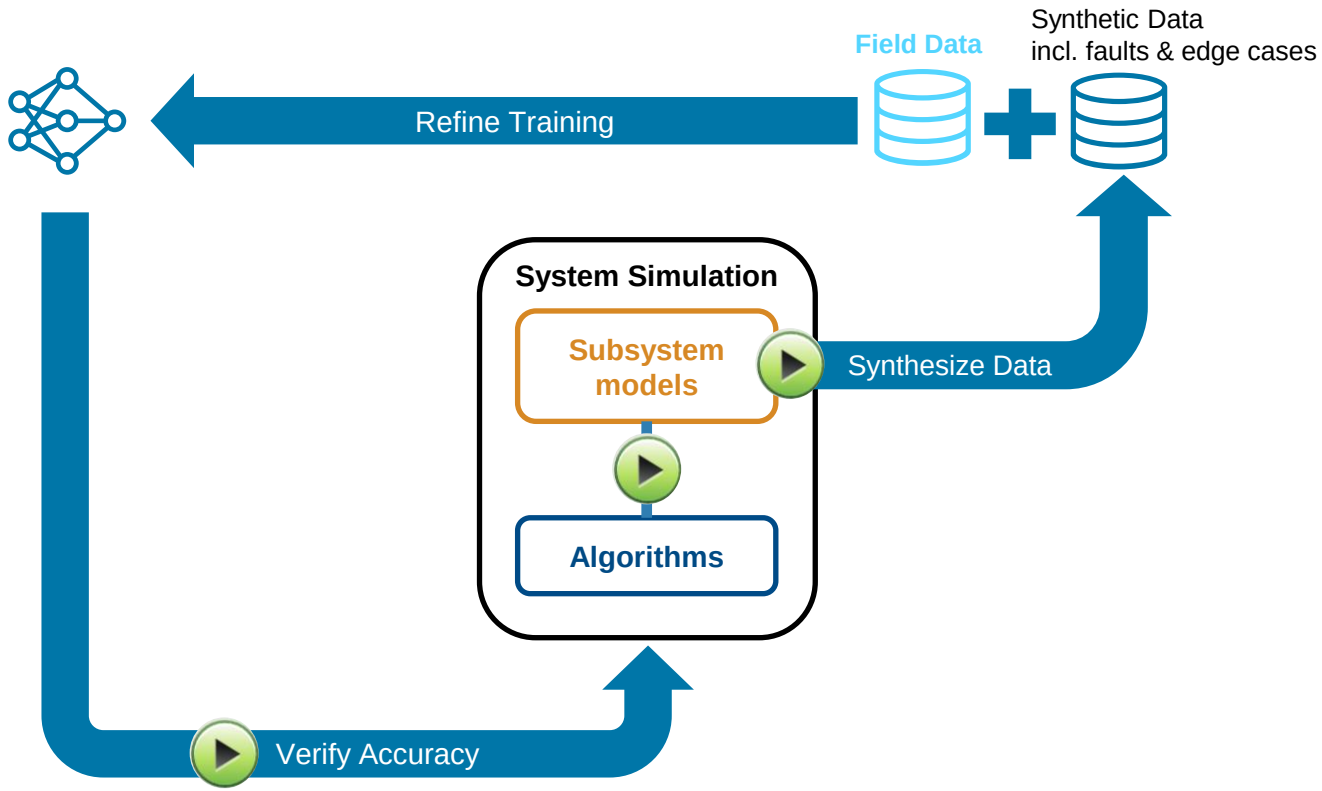


Audio
Speech Recognition and Fault Classification

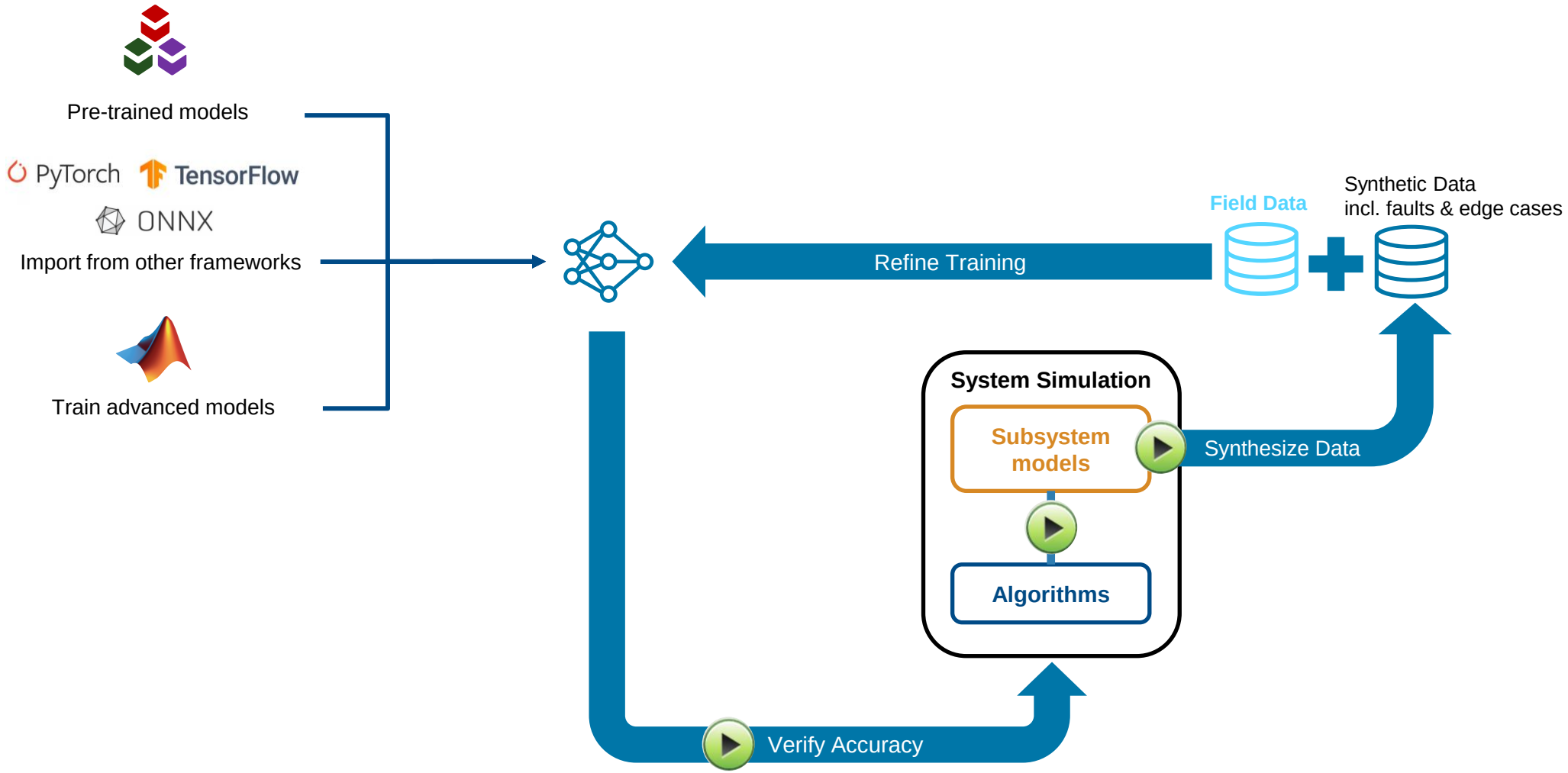


Geospatial Analysis
Image & Video Segmentation and Classification

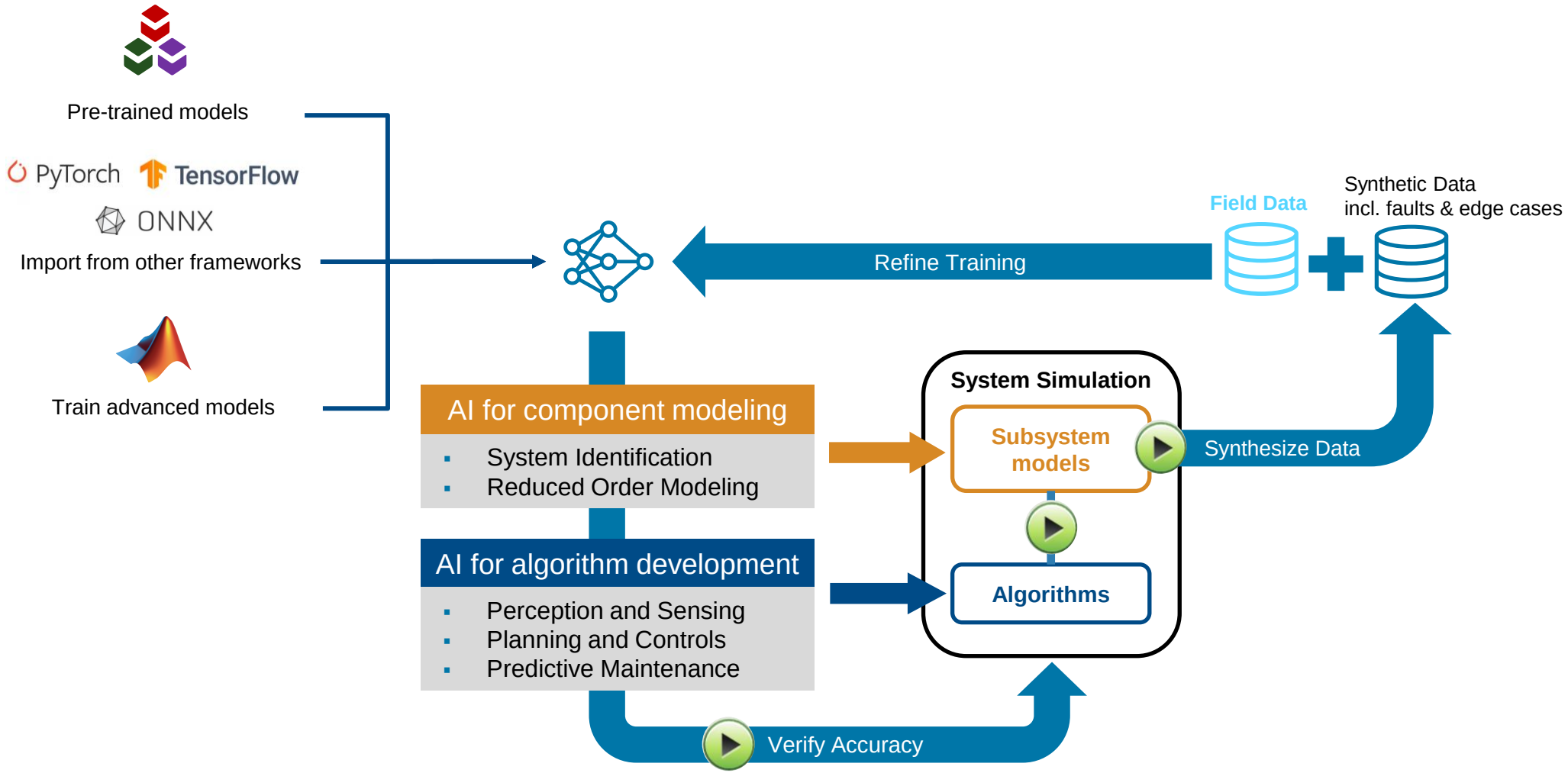
1 Simulations close data gaps and verification gaps for AI algorithms



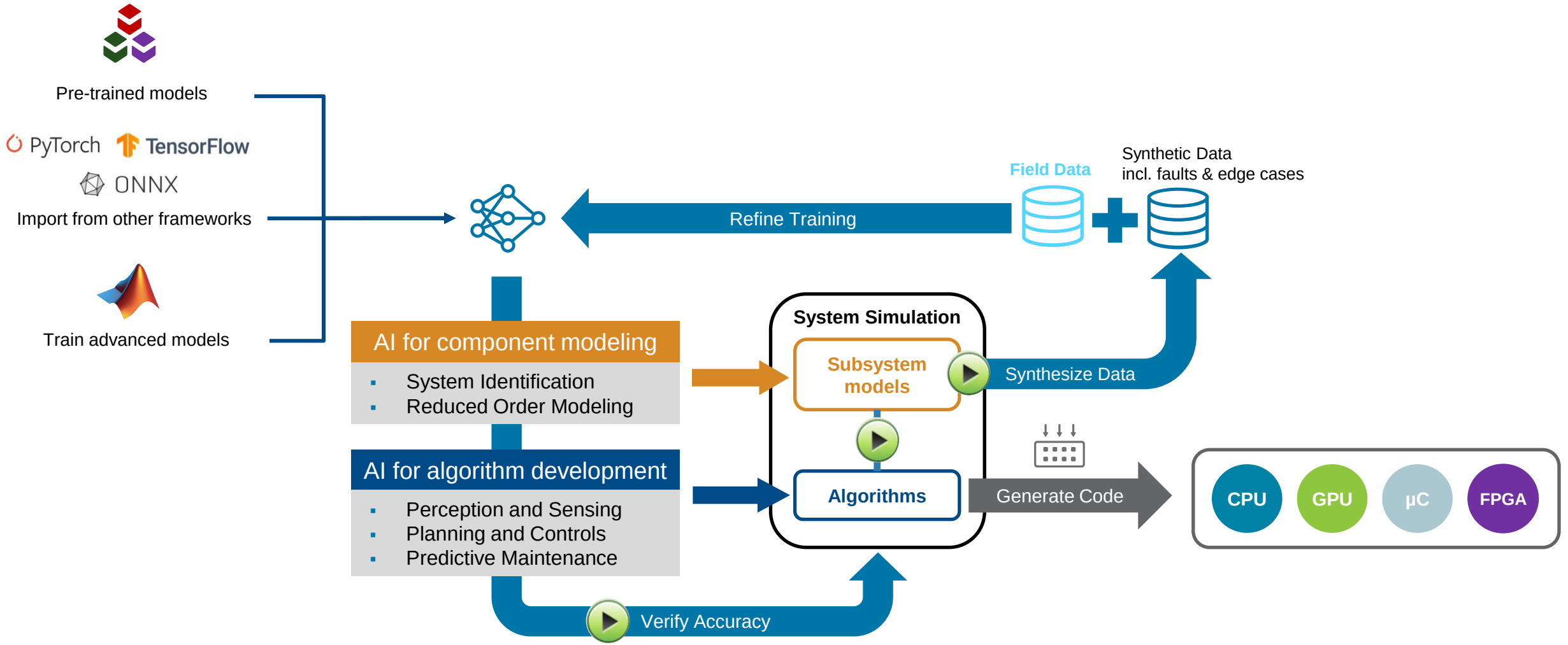
2 Spectrum of ways of leveraging state-of-the-art AI models



3 Incorporate AI as system components and algorithms

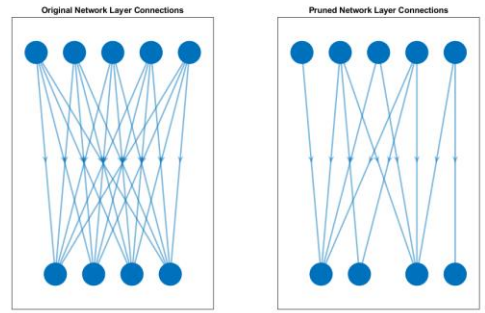


4 Deploy efficient AI to any processor with code generation

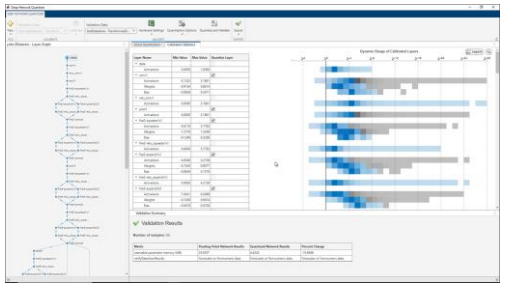


4 Deploy efficient AI to any processor with code generation

Model Compression



Structural



Data Type

Code Generation

CPU

Any CPU
No Library needed

oneDNN
Library

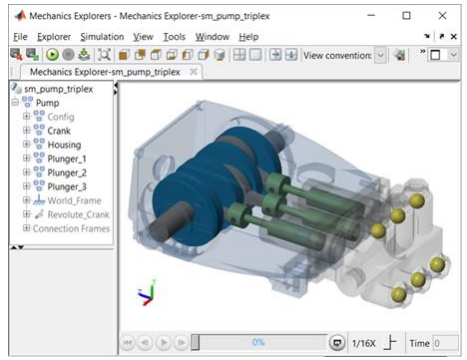
ARM Compute
Library

GPU

µC

FPGA

Today's Focus: Synthetic Fault Data Generation



Pre-trained models

PyTorch TensorFlow ONNX

Import from other frameworks

Train models

Refine Training

Field Data

Synthetic Data
incl. faults & edge cases

AI for component modeling

- System Identification
- Reduced Order Modeling

AI for algorithm development

- Perception and Sensing
- Planning and Controls
- Predictive Maintenance

System Simulation

Subsystem models

Synthesize Data

Algorithms

Generate Code

CPU GPU µC FPGA

Verify Accuracy

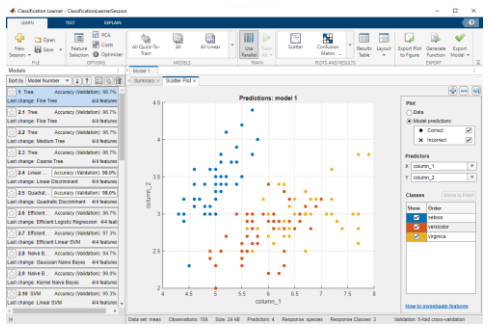
Today's Focus: Low-Code Machine Learning

Pre-trained models

PyTorch TensorFlow ONNX

Import from other frameworks

Train models



Refine Training

Field Data

Synthetic Data
incl. faults & edge cases

AI for component modeling

- System Identification
- Reduced Order Modeling

AI for algorithm development

- Perception and Sensing
- Planning and Controls
- **Predictive Maintenance**

System Simulation

Subsystem models

Algorithms

Synthesize Data

Generate Code

CPU GPU µC FPGA

Verify Accuracy

Prerequisites

To participate in the workshop, you will need:

1. A laptop
2. Google Chrome browser
3. A MathWorks Account

You will be provided with a temporary MATLAB workshop license that will give you access to all products used in the workshop, as well as the workshop exercise files.

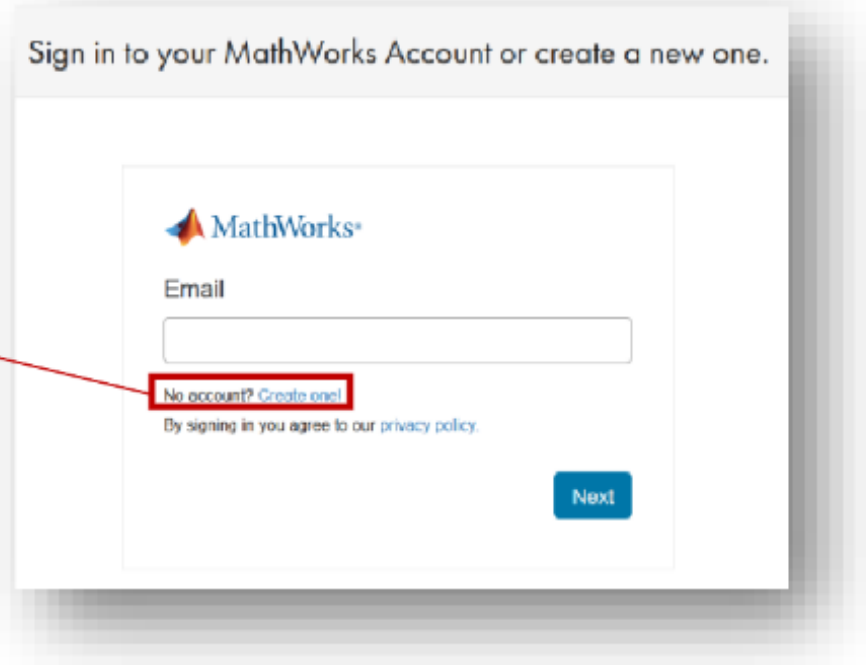
Step I. Set Up Your MathWorks Account

If you don't have a MathWorks account, you need to create one to get access to MATLAB Online and the material for this event. You will need access to your email on the machine you are using to create the account.

1. In Google Chrome, go to: https://www.mathworks.com/mwaccount/ and click **Create one!** next to No Account?:

   No account? Create one!

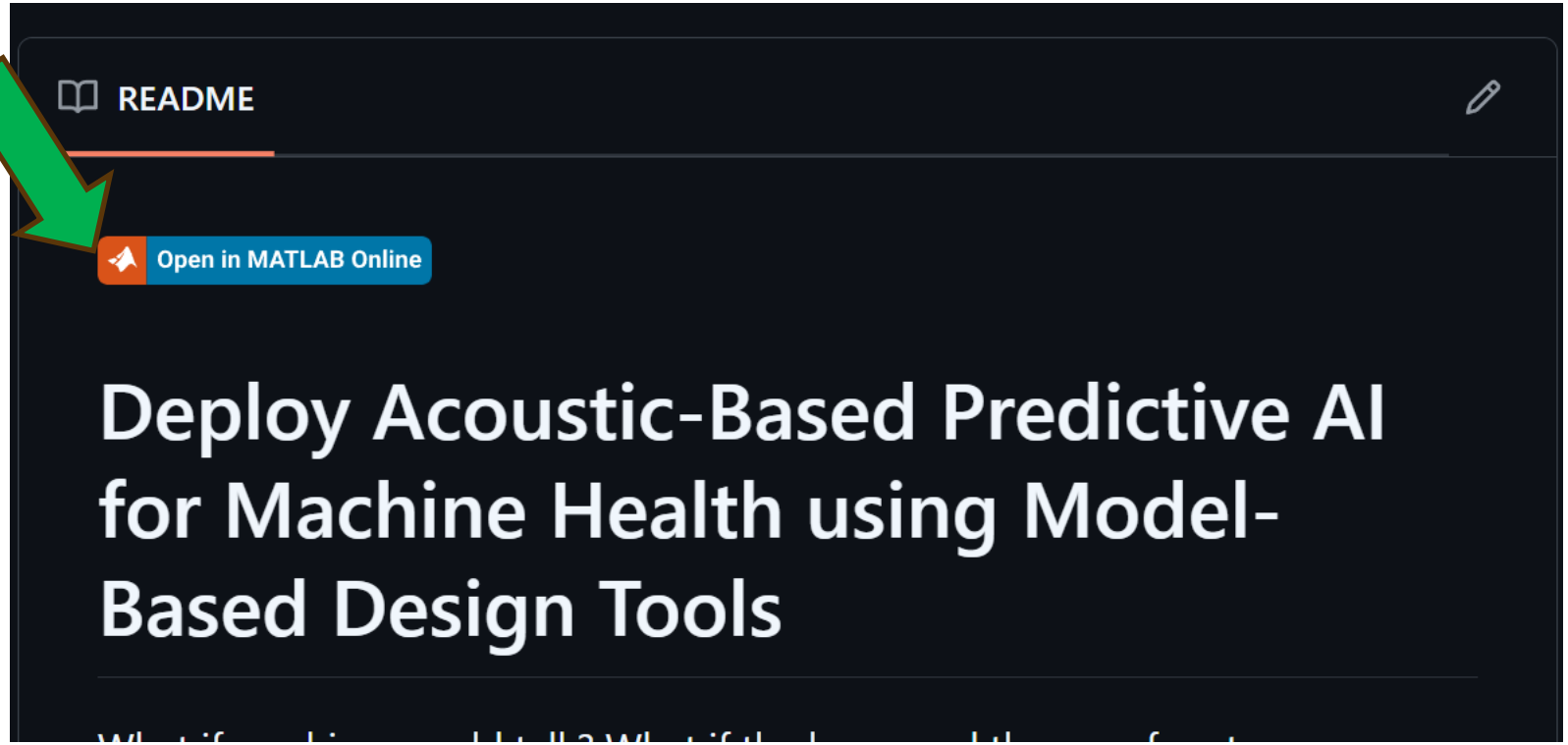
2. Fill out the form and click **Create**. Follow the directions for verifying your email address.
3. To complete your registration, click the link in the verification email and fill out the form.
   You must check the Online Services Agreement box.
   You may leave these fields blank: Activation Key or License Number, Sales rep contact, Associate with a license, Trial

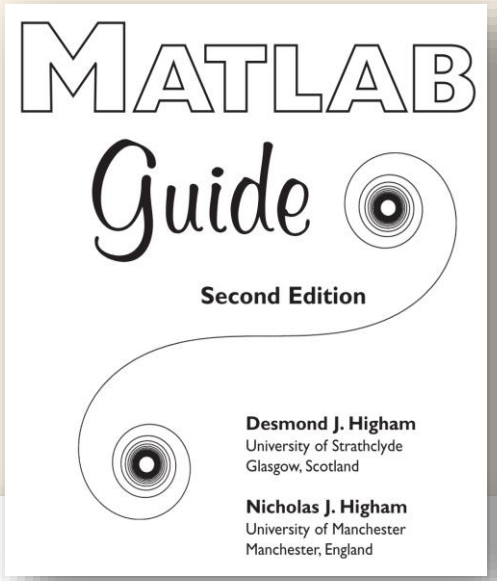


Access Exercise Files

MATLAB Online can port your workshop files from GitHub automatically. You will see a dialog pop-up for saving and opening the repository of files.

1. Exercise files are located at: https://github.com/Brenda-MW/ESWeek-Acoustic-AI-with-MBD
2. Directly launch the workshop links from the “Open in MATLAB Online” button in README.





For those of you that have not experienced MATLAB, we would like to try to show you what everybody is excited about . . . The best way to appreciate PC-MATLAB is, of course, to try it yourself.

— JOHN LITTLE and CLEVE B. MOLER, *A Preview of PC-MATLAB* (1985)

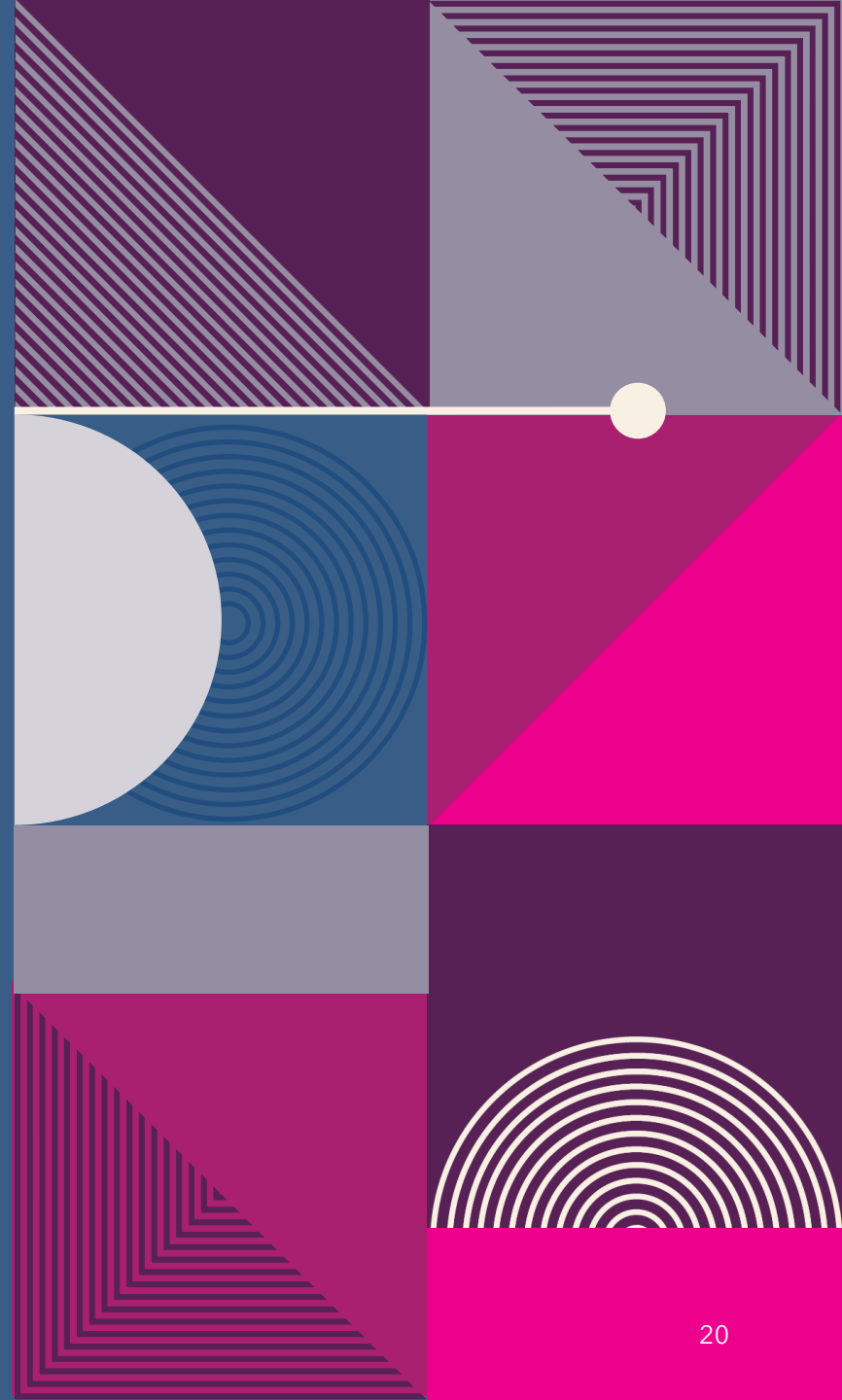
HANDS-ON: DEPLOY ACOUSTIC-BASED AI FOR MACHINE HEALTH

Straight from your browsers!

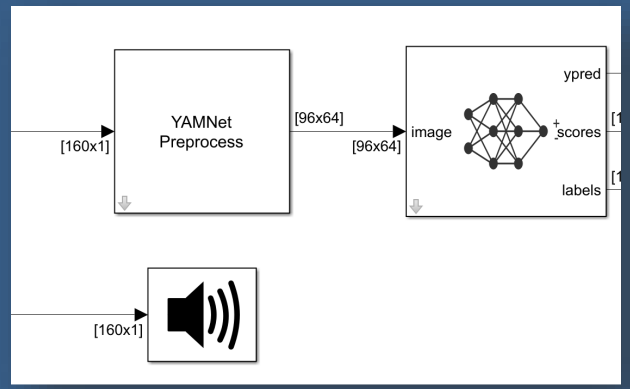
OBJECTIVES

Developing Edge AI poses challenges due to the need to **optimize performance** and **energy efficiency**, which requires an understanding of both AI and embedded systems.

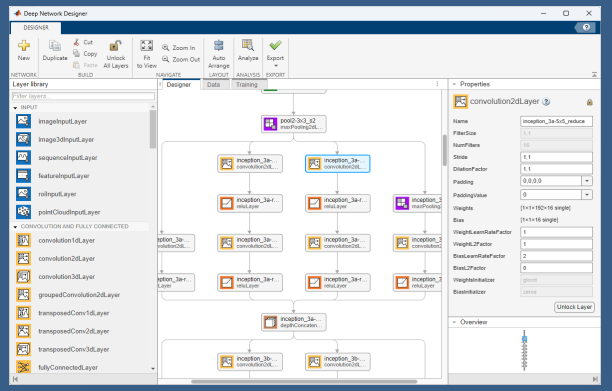
Model-Based Design (MBD) workflow from **MathWorks** simplifies this process, enabling efficient deployment of AI models to the edge devices.



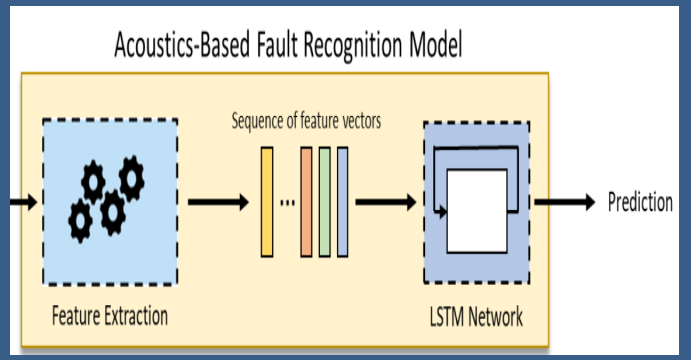
A look at today's Exercises



Exercise #1
Use pretrained model

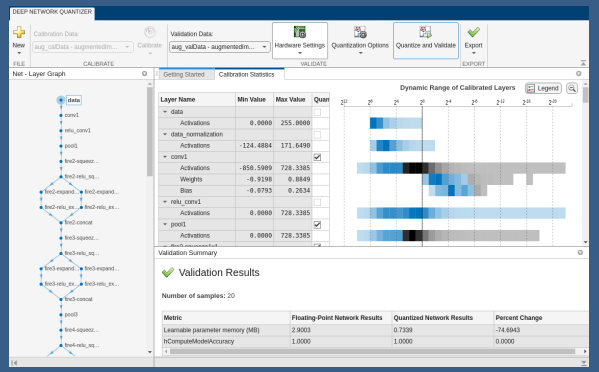


Exercise #2
Design AI Models

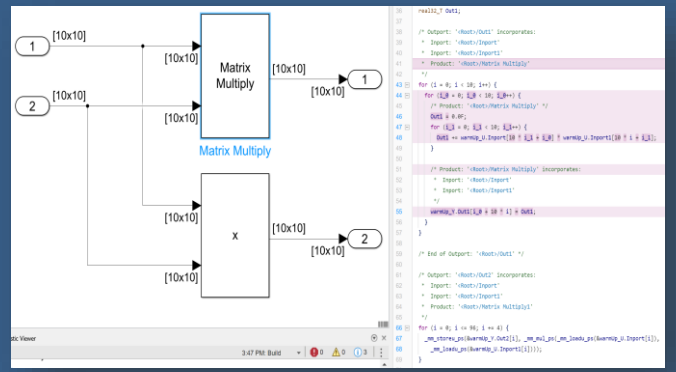


Exercise #3
Train models

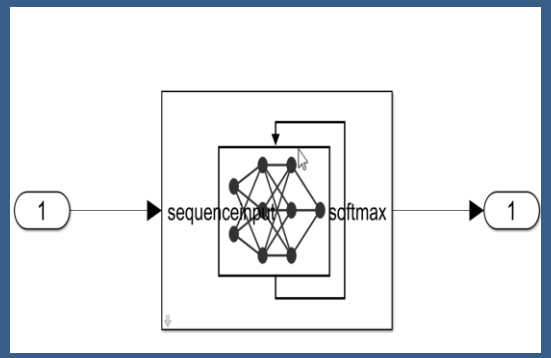
A look at today's example (Cont.)



Step 4
Optimize AI Model with Pruning and Quantization



Step 5
Deploy using Embedded C Code Generation



Step 6
Simulink as integration

Data representation from the physical world to neural networks is where the design starts

Tabular Data

ID	WC_TA	RE_TA	EBIT_TA	MVE_BVTD	S_TA	Industry	Rating
62394	0.013	0.104	0.036	0.447	0.142	3	BB
48608	0.232	0.335	0.062	1.969	0.281	8	A
42444	0.311	0.367	0.074	1.935	0.366	1	A
48631	0.194	0.263	0.062	1.017	0.228	4	BBB
43768	0.121	0.413	0.057	3.647	0.466	12	AAA
39255	-0.117	-0.799	0.01	0.179	0.082	4	CCC
62236	0.087	0.158	0.049	0.816	0.324	2	BBB
39354	0.005	0.181	0.034	2.597	0.388	7	AA
40326	0.47	0.752	0.07	11.596	1.12	8	AAA
51681	0.11	0.337	0.045	3.835	0.812	4	AAA

Time Series/ Text Data

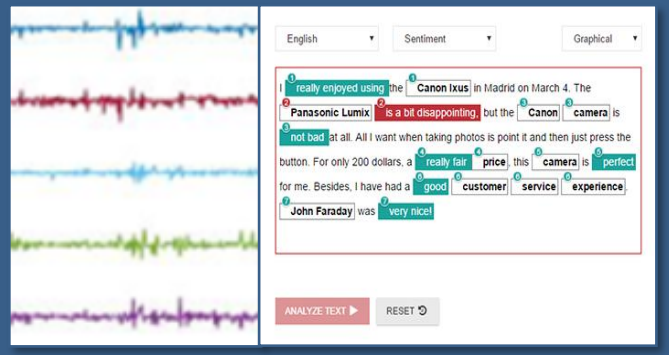
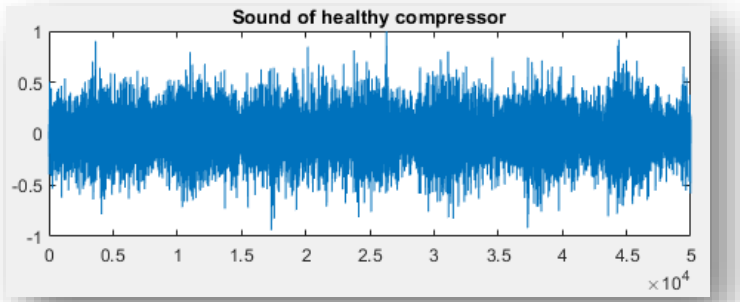
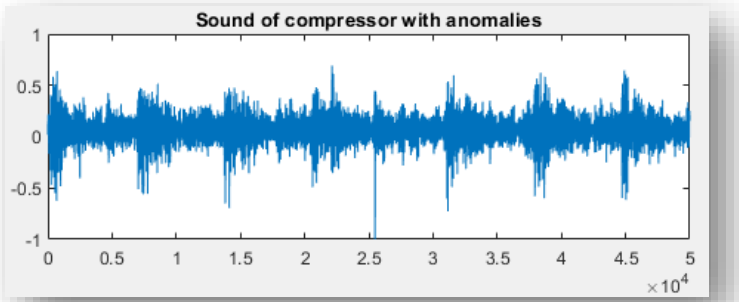


Image Data

How can I apply **transfer learning** to detecting faults in an air compressors based on their noise

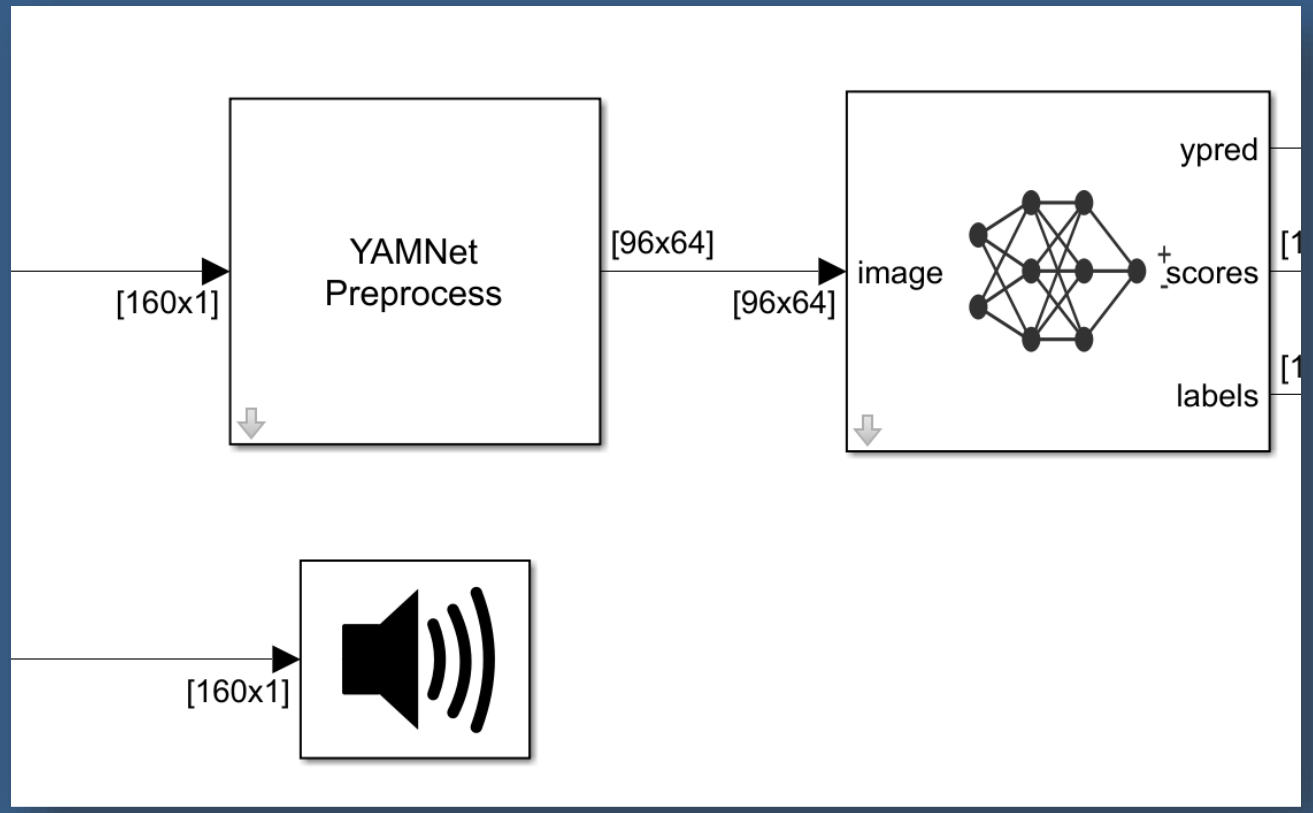
- Have dataset with labeled sound recordings
- One “healthy” class
- 7 different classes of faults
- 1800.wav files, 225 per class





[Example: Transfer Learning with Pretrained Audio Networks in Deep Network Designer]

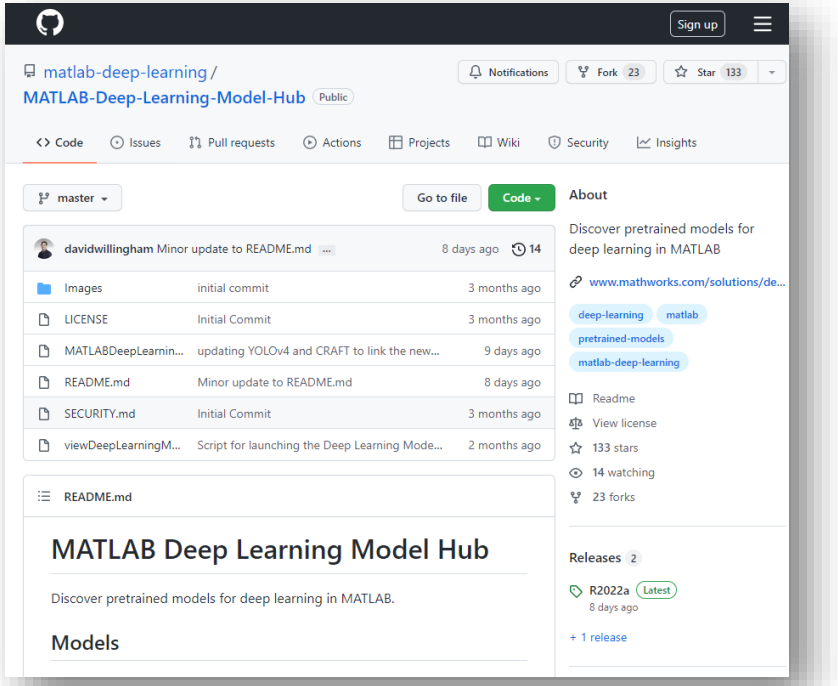
A look at today's Exercises



Exercise #1
Use pretrained model

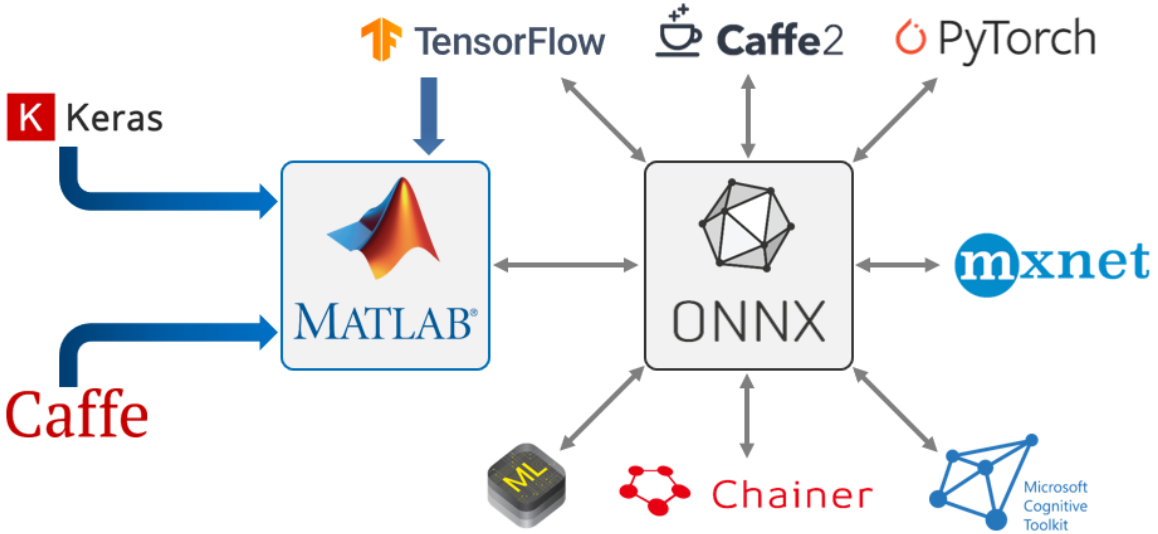
Finding a pre-trained deep learning network for Transfer Learning

- Find one directly in MATLAB



https://github.com/matlab-deep-learning/MATLAB-Deep-Learning-Model-Hub

- Import it from a known non-MATLAB repository



MATLAB R2022a

HOME PLOTS APPS LIVE EDITOR INSERT VIEW

Level A Update Cleanup Search Documentation Gabriele

New Open Save Compare Print Export FILE | Go To Find Bookmark NAVIGATE | Text Normal B I U M TEXT | Code Control Task Refactor CODE | Run Section Section Break Run and Advance Run to End SECTION | Run Step Stop RUN

C: ▸ Users ▸ gbunkhei ▸ OneDrive - MathWorks ▸ Documents ▸ MATLAB ▸ Examples ▸ R2022a ▸ deeplearning_shared ▸ TransferLearningWithAudioNetworkInDeepNetworkDesignerExample

Current Folder | Command History

Name	Size	Date Modified
Live Script		
Transf...	425 KB	07/04/2022 ...

Details

Workspace

Name	Value
ads	*1x1 audioData*
adsTest	*1x1 audioData*
adsTrain	*1x1 audioData*
adsValidation	*1x1 audioData*
datasetLocation	'C:\Users\gbun
downloadFolder	'C:\Users\gbun
tdsTrain	*1x1 Transform*
tdsValidation	*1x1 Transform*
url	'https://www.m

Live Editor - C:\Users\gbunkhei\OneDrive - MathWorks\Documents\MATLAB\Examples\R2022a\deeplearning_shared\TransferLearningWithAudioNetworkInDeepNetworkDesignerExample\TransferLearningWithAudioNetworkInDeepNetw...

TransferLearningWithAudioNetworkInDeepNetworkDesignerExample.mlx

Transfer Learning with Pretrained Audio Networks in Deep Network Designer

This example shows how to interactively fine-tune a pretrained network to classify new audio signals using Deep Network Designer.

Transfer learning is commonly used in deep learning applications. You can take a pretrained network and use it as a starting point to learn a new task. Fine-tuning a network with transfer learning is usually much faster and easier than training a network with randomly initialized weights from scratch. You can quickly transfer learned features to a new task using a smaller number of training signals.

This example retrains YAMNet, a pretrained convolutional neural network, to classify a new set of audio signals. This example requires Audio Toolbox™ and Deep Learning Toolbox™.

Load Data

Download and unzip the air compressor data set [1]. This data set consists of recordings from air compressors in a healthy state or one of 7 faulty states.

```
url = 'https://www.mathworks.com/supportfiles/audio/AirCompressorDataset/AirCompressorDataset.zip';
downloadFolder = fullfile(tempdir,'aircompressordataset');
datasetLocation = tempdir;

if ~exist(fullfile(tempdir,'AirCompressorDataSet'),'dir')
    loc = websave(downloadFolder,url);
    unzip(loc,fullfile(tempdir,'AirCompressorDataSet'))
end
```

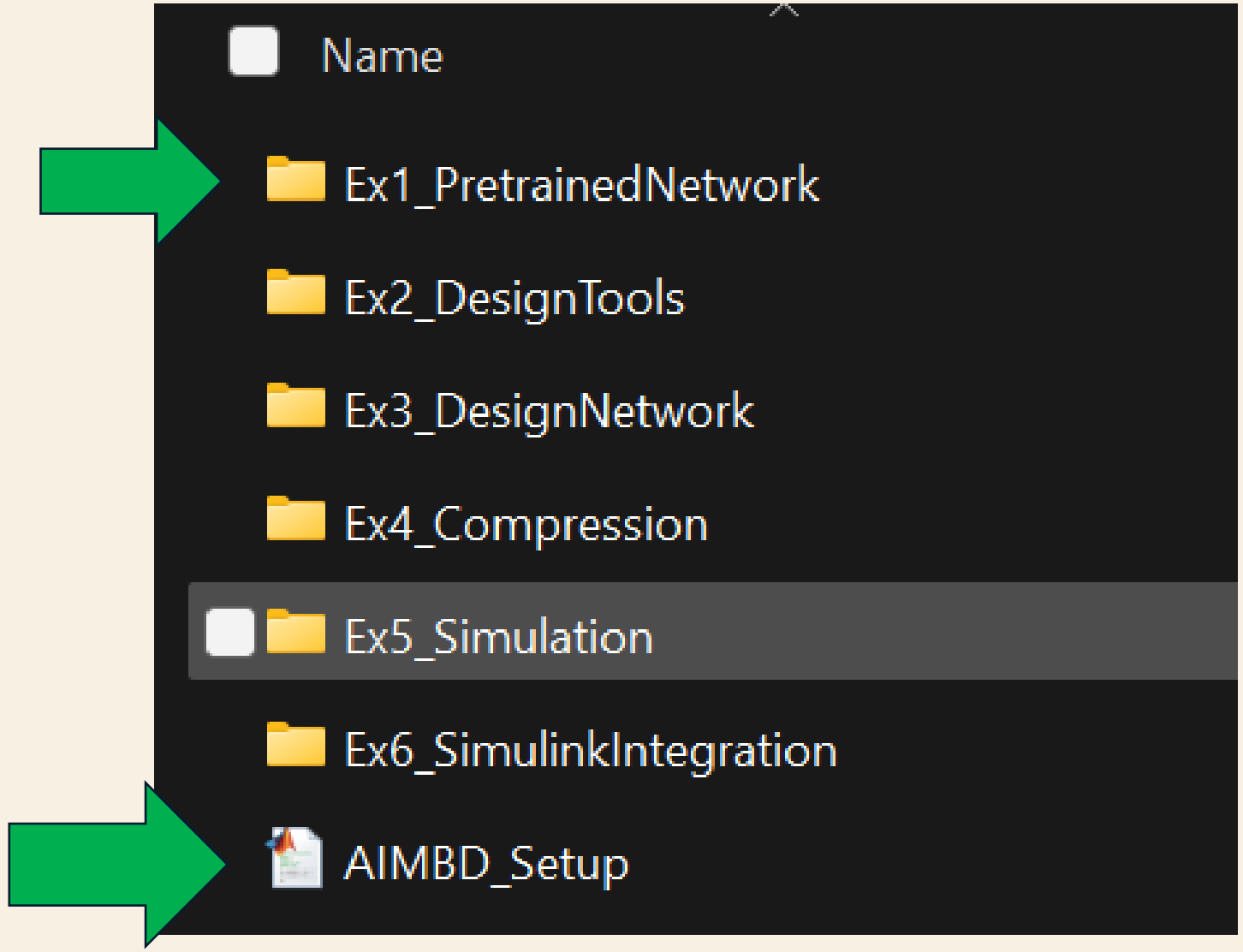
Create an `audioDatastore` object to manage the data and split it into training, validation, and test sets.

Command Window

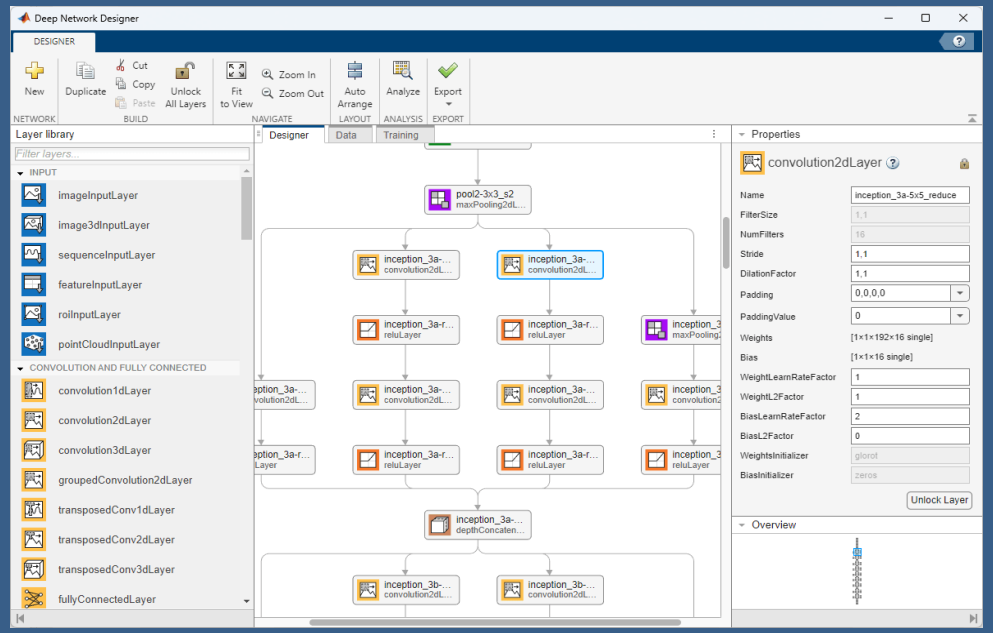
fx >>

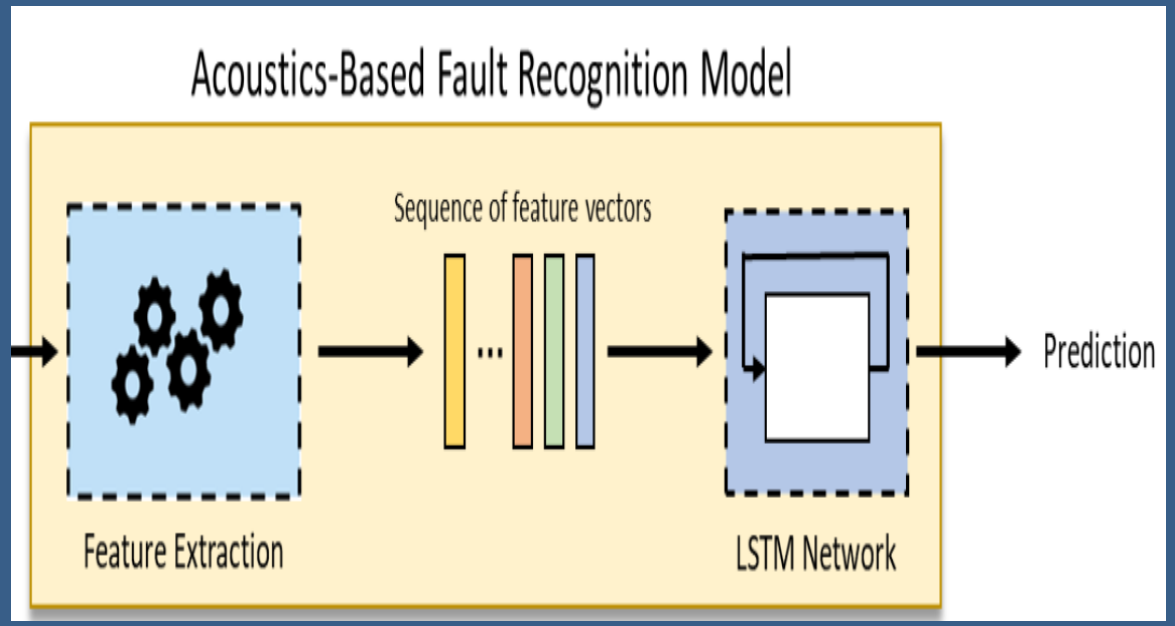
Zoom: 100% UTF-8 LF script

Let’s work on Exercise #1 in AcousticsAI_MBD



A look at today's Exercises

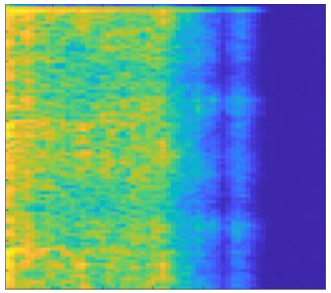


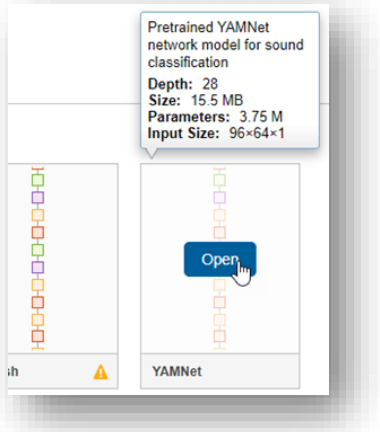


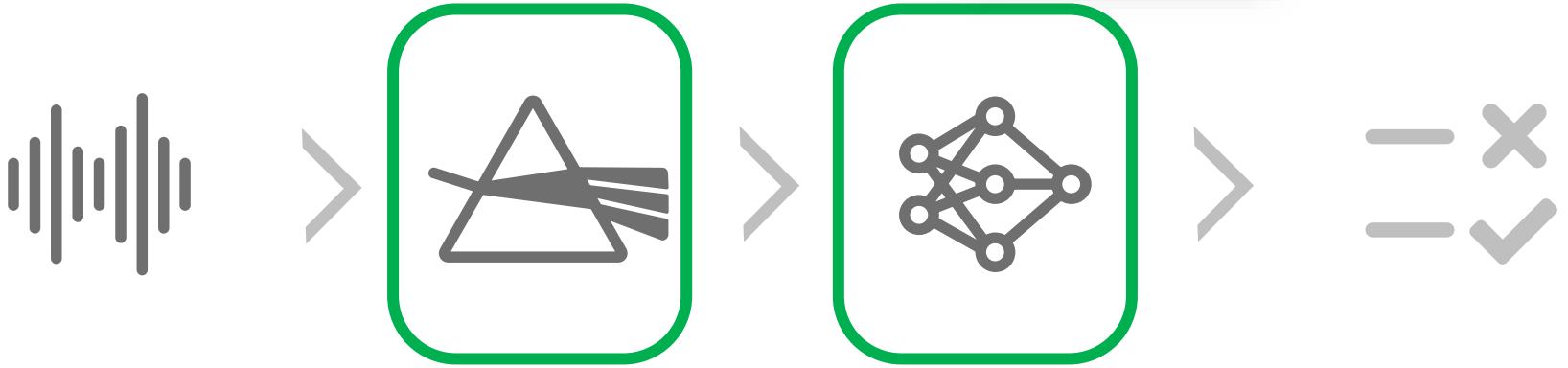
Exercise #2
Design AI Models

Exercise #3
Train models

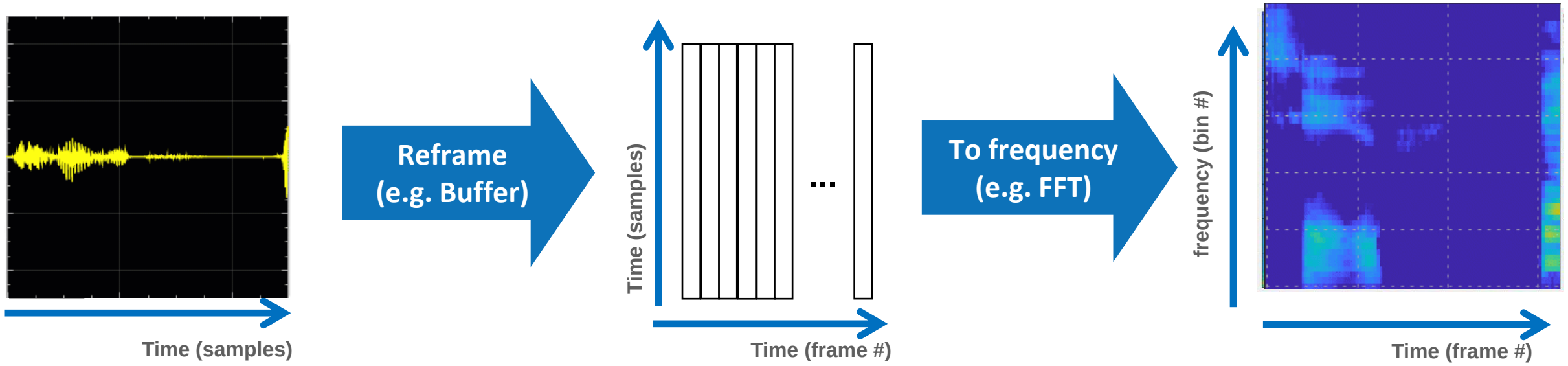
Deep networks most often don’t learn directly from raw signals





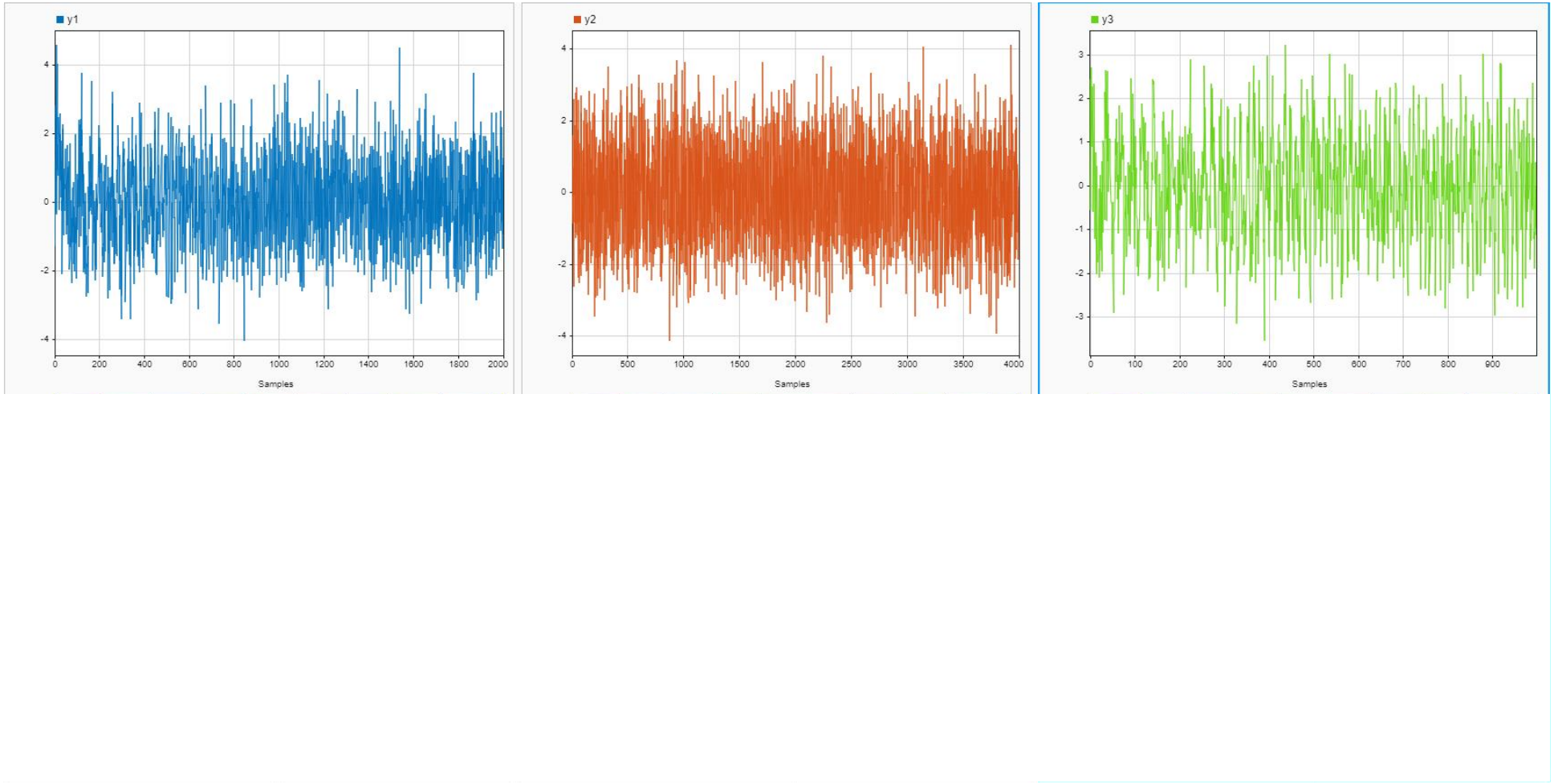


Time-frequency transformations are popular feature extraction methods



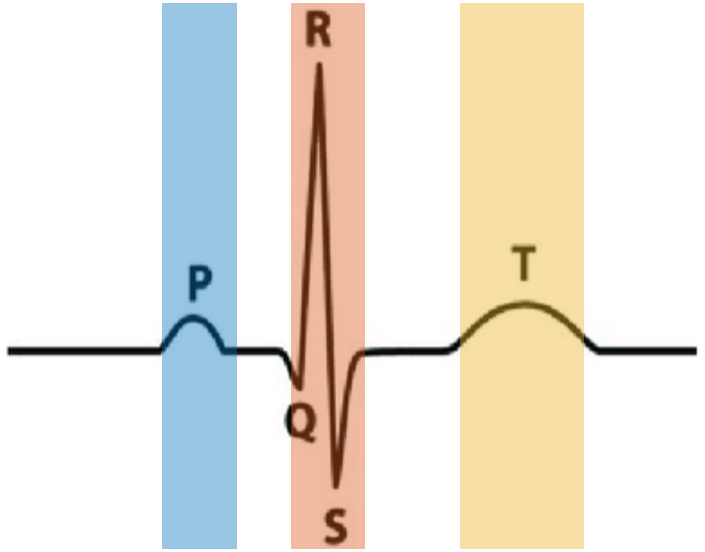
Time-frequency transforms make signal characteristics more evident

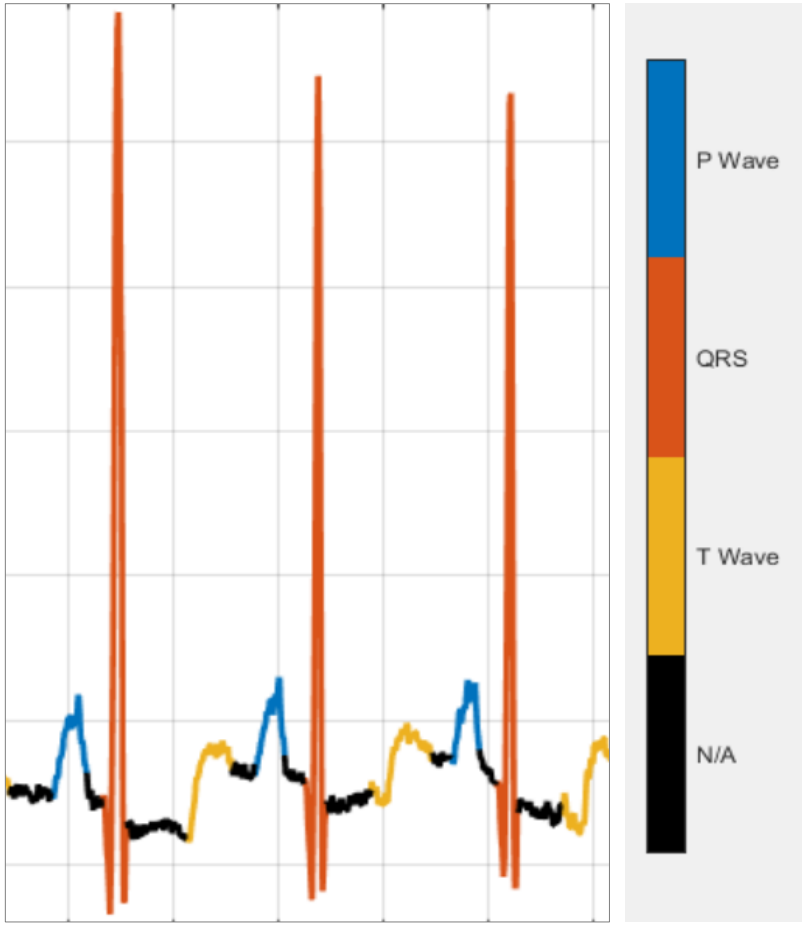
Time-frequency transform



How to use feature extraction to segment ECG signals?

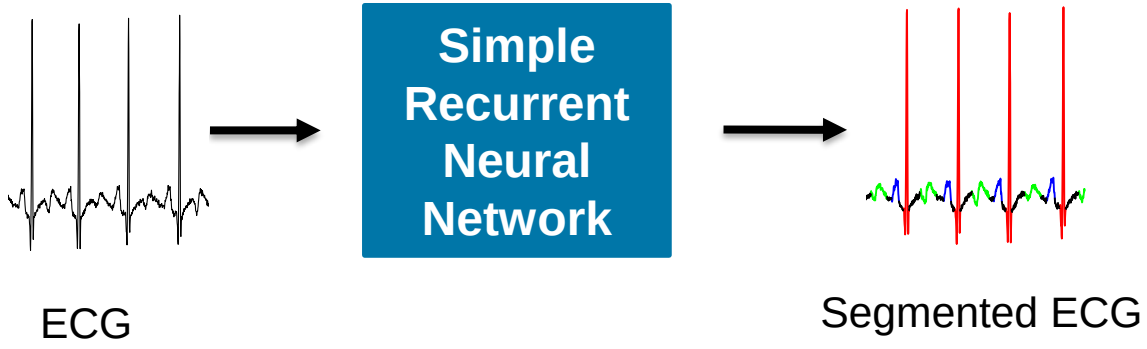
- Have dataset with signals labeled by cardiologists
- 3 types of wave events
- 210 ECG recordings (total ~15 minutes)

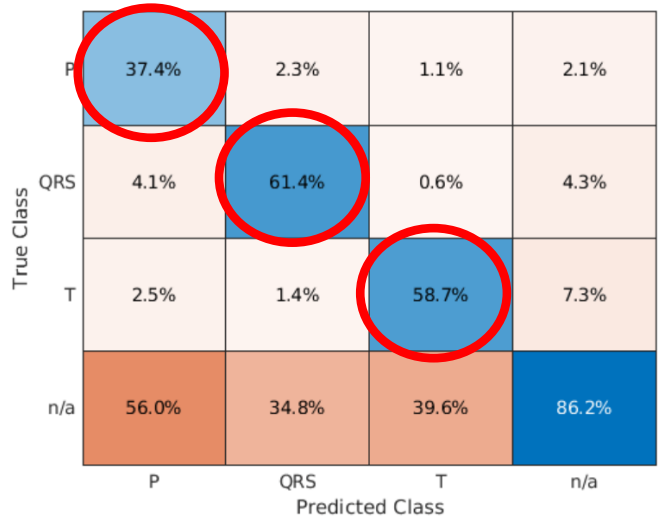


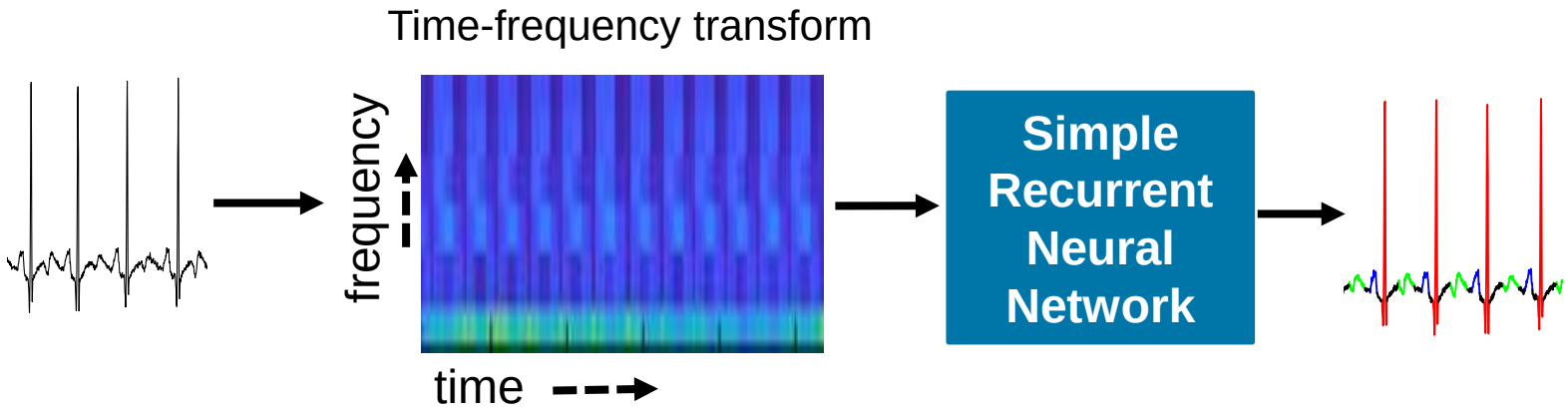


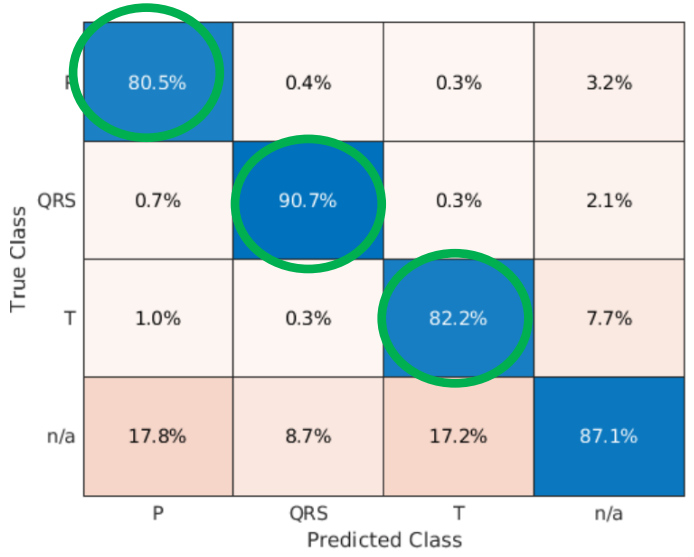
Example: Waveform Segmentation Using Deep Learning

Feature extraction allows getting high accuracy from simple AI models

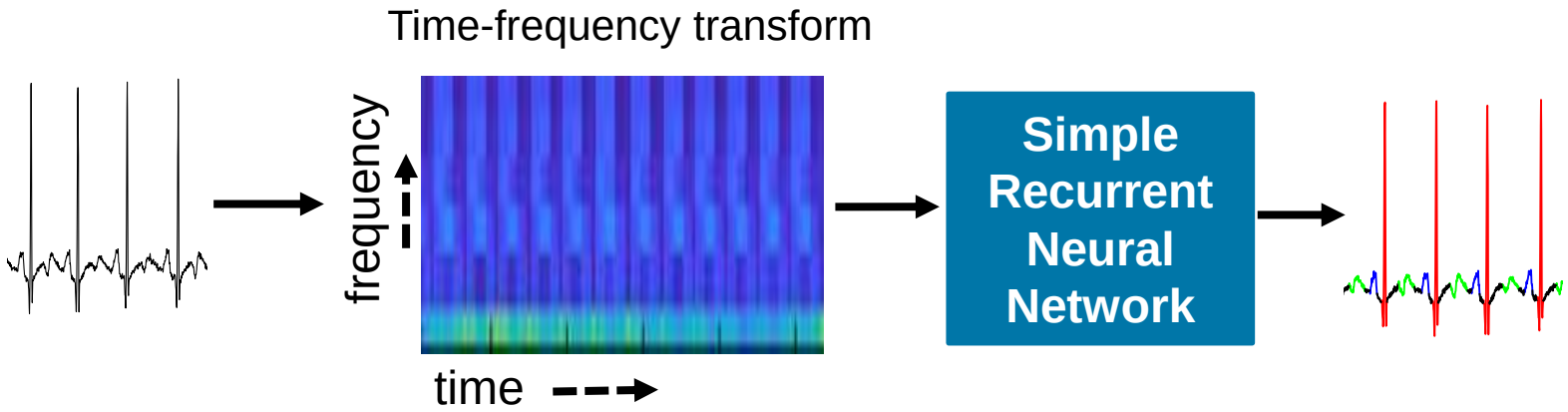


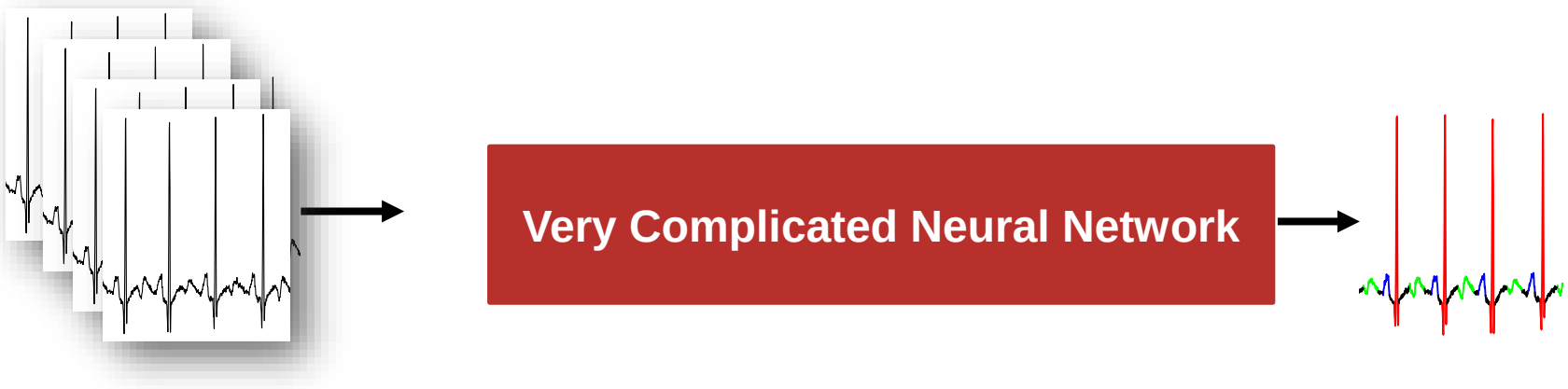


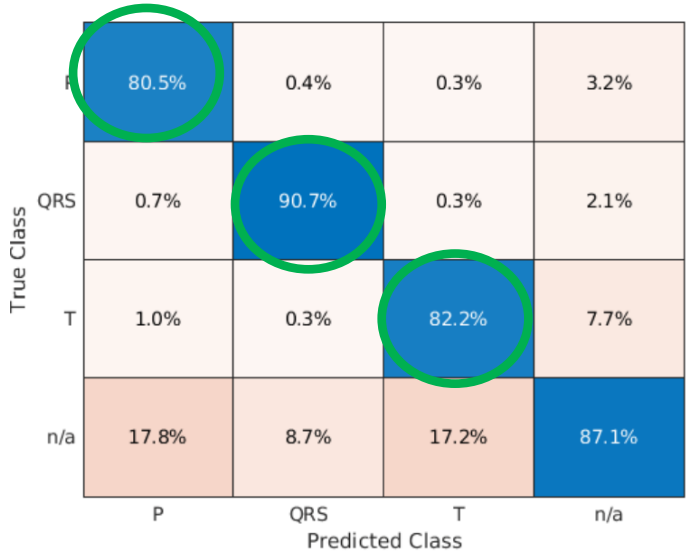




Feature extraction reduces model and data complexity

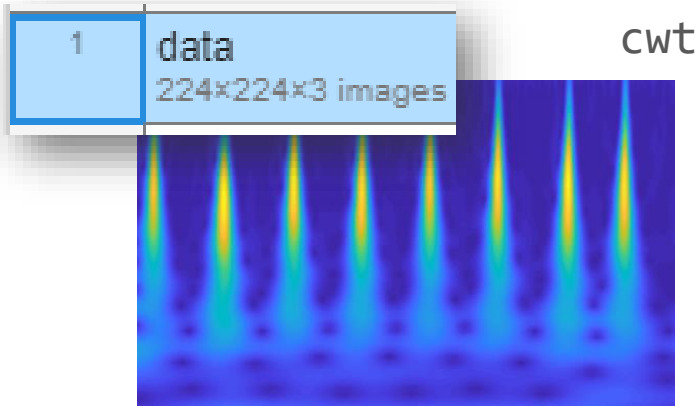






Domain experts are best placed to select feature extraction algorithm

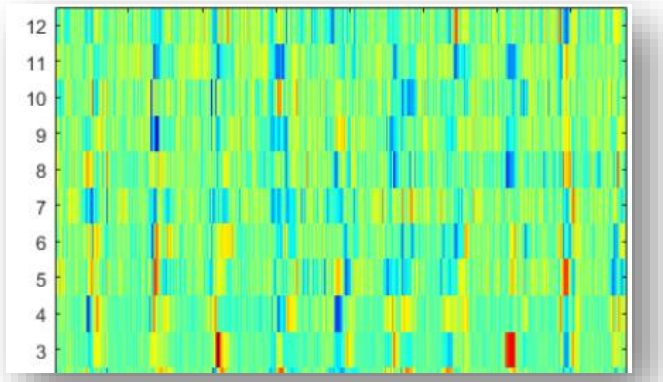
Model size, signal patterns



cwt

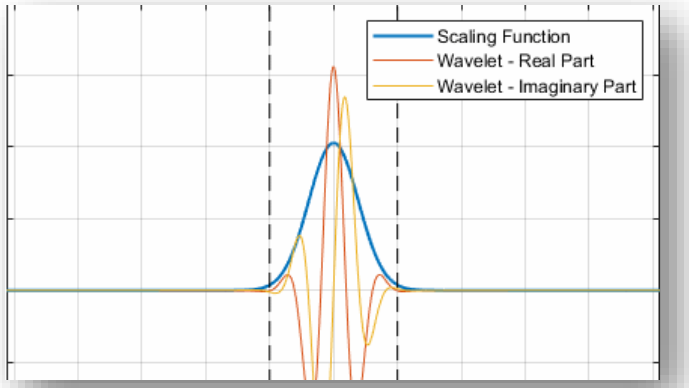
Application and signal type

mfcc



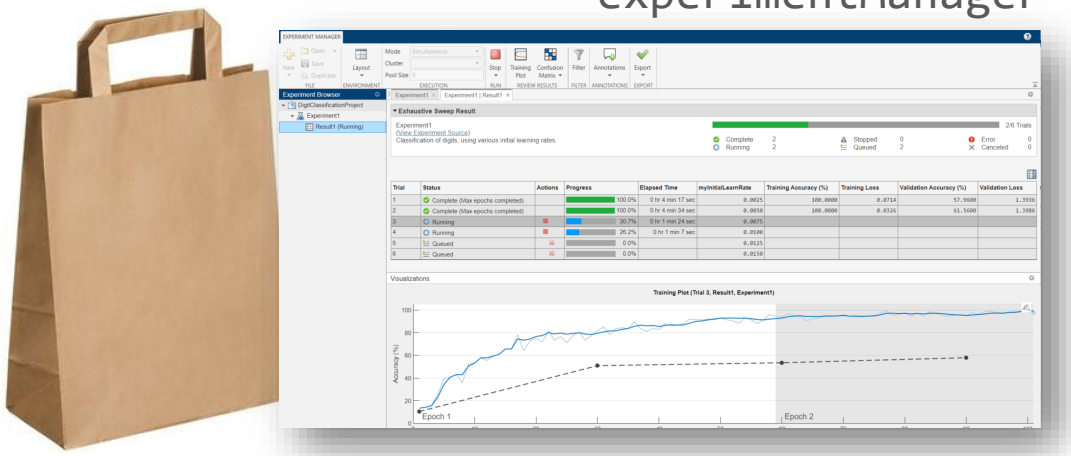
Automated methodology

waveletScattering



Test-based experiments

experimentManager



signalFrequencyFeatureExtractor

Feature Extraction

MathWorks Wins Geoscience AI GPU Hackathon

The following post is from Akhilesh Mishra, Mil Shastri and Samvith V. Rao from MathWorks here to talk about their participation and in a Geoscience hackathon. Akhilesh and Mil are Applications Engineers and Samvith is the Industry Marketing Manager supporting the Oil and Gas industry.

Background

SEAM (SEG Advanced Modeling Corp.) is a petroleum geoscience industry body that fosters collaborations among industry, government, and academia to address major Geological challenges. Their latest event was a hackathon (SEAM AI Applied Geoscience GPU Hackathon) that sought to explore the use of AI to improve both qualitative and quantitative interpretation of geophysical images of Earth's interior, and speed up the applications using NVIDIA GPUs.

A total of 7 teams participated from all over the world, including commercial companies (Chevron, Total, Petrobras) and a mix of industry and university students. Each team was assigned a mentor who is an expert geoscientist working for a top oil and gas company.

The Challenge

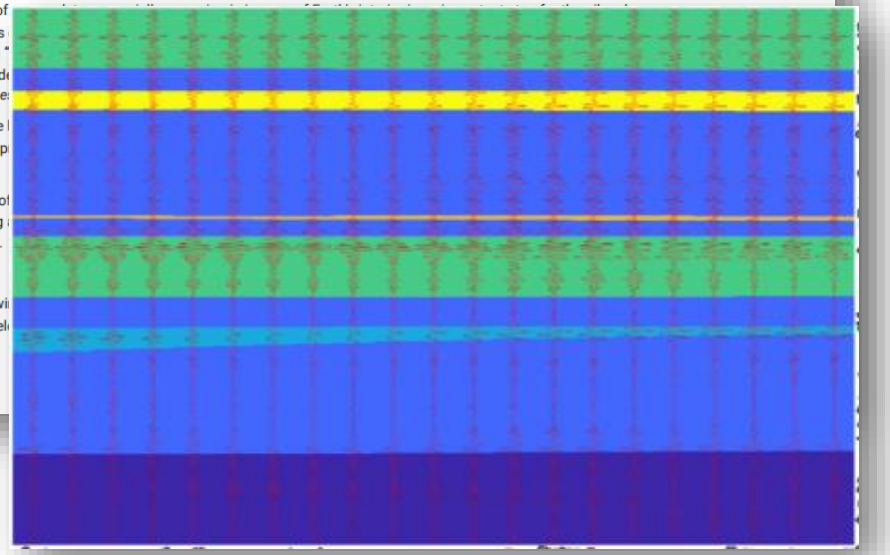
Geologic interpretation of
industry. Seismic images
summarized by the term "
and abandonment of unde
often called *seismic facie*

This process is still done
display. Successful interp
features.

The problem statement o
automatically, producing
up human interpretation.

The Data

We were given the followi
public and has been label



[MathWorks Deep Learning Blog Post]

Daihatsu Uses AI to Classify Engine Sounds

Challenge

Develop an AI solution that can judge the level of engine knocking sound, which only skilled workers could judge

Solution

Create classification models and easy-to-use interface with MATLAB, making it possible to examine features multiple times

Key Outcomes

- Performed knocking sound analysis with the same accuracy as skilled workers
- Increased AI expertise through MATLAB training
- Promoted visualization of AI and increased awareness of AI

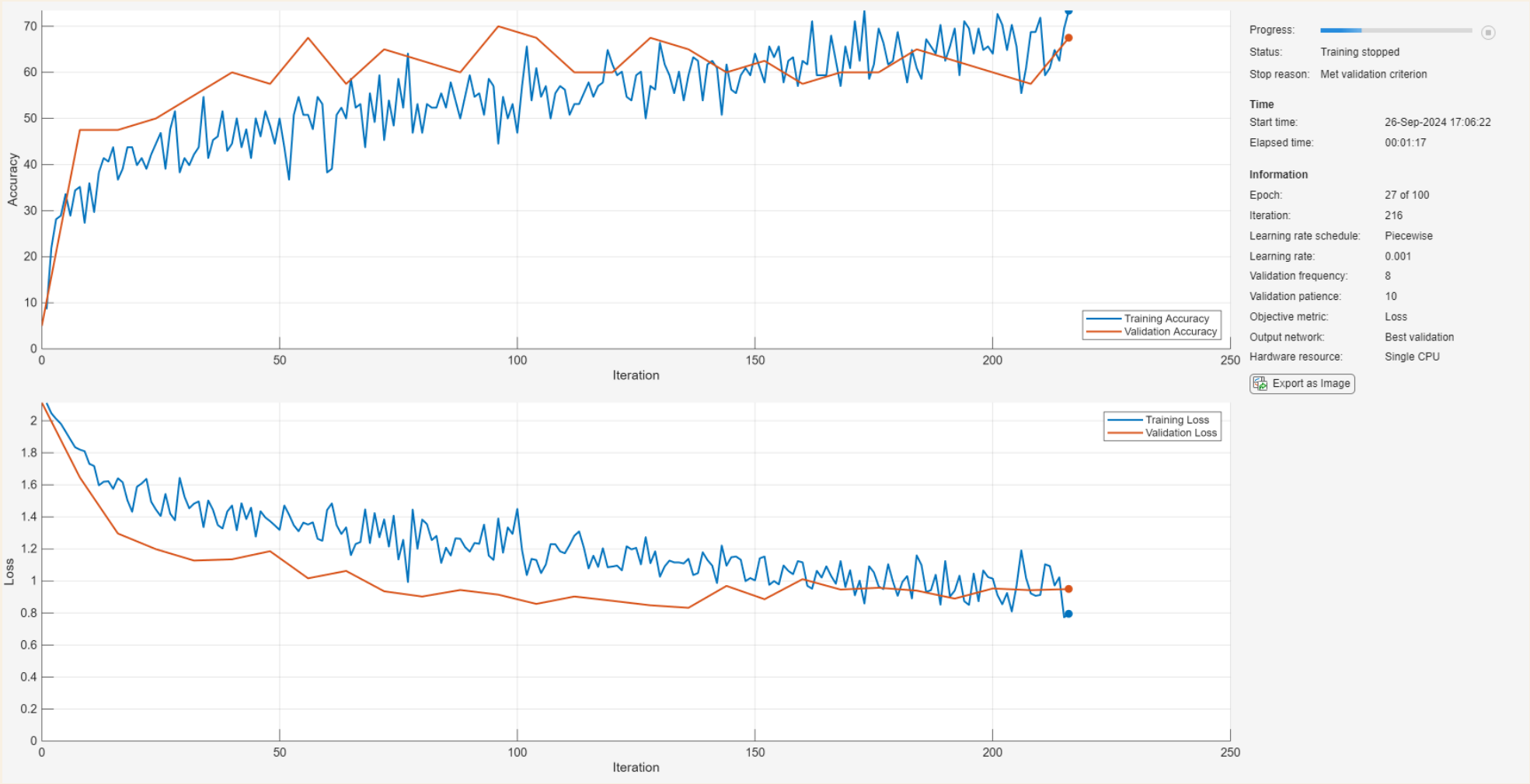
Daihatsu used AI to identify knocking sounds from its engines.

"Although we tried other programming languages, it was hard to implement. We decided to use MATLAB, which allows us to easily import the necessary data by dragging and dropping, and we could easily see the result by ourselves."
- Takuya Kumagae, Daihatsu Motor Co., Ltd.

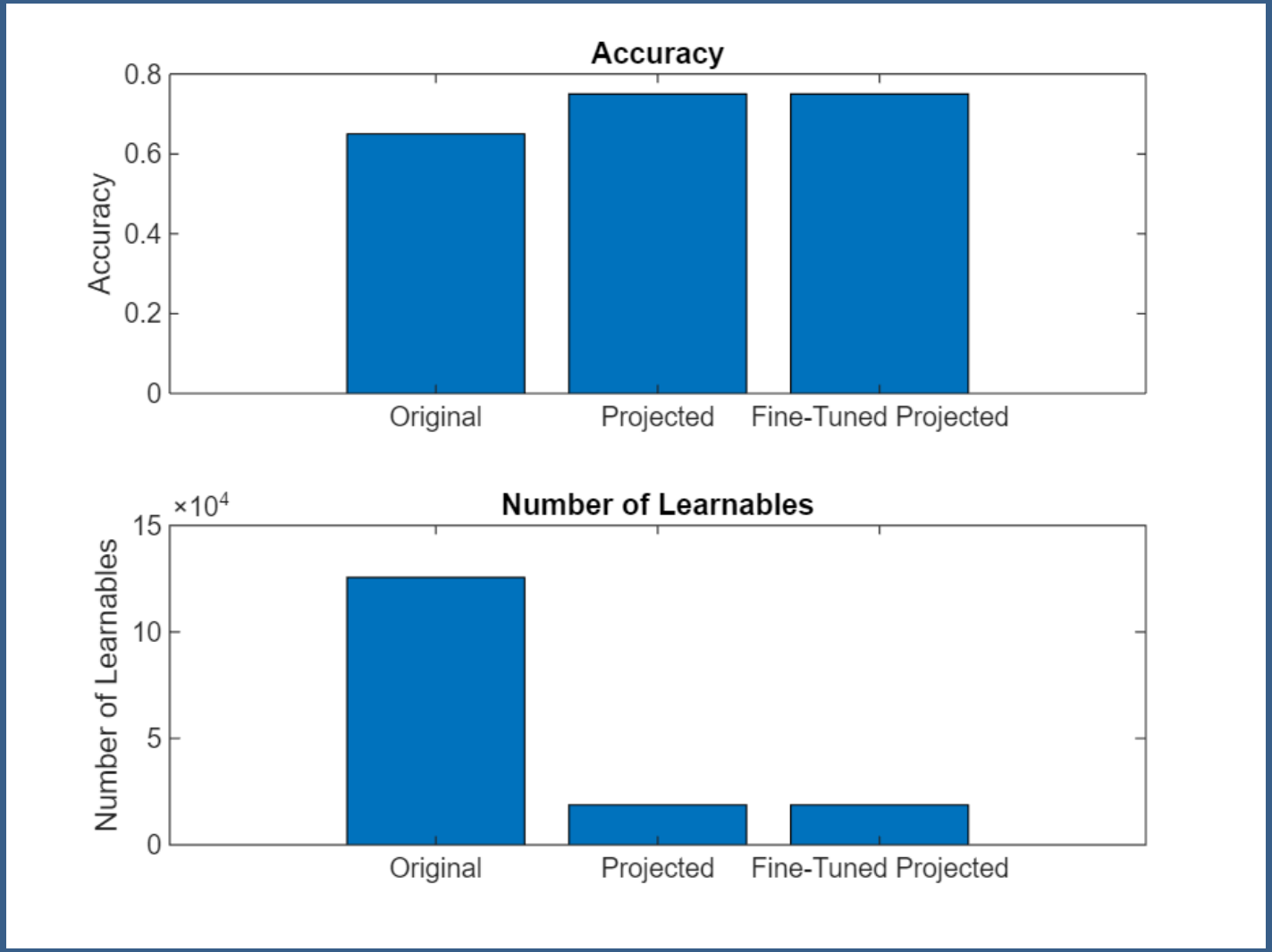
Link to case study

[Daihatsu User Story]

Let’s work on ex2 and ex3

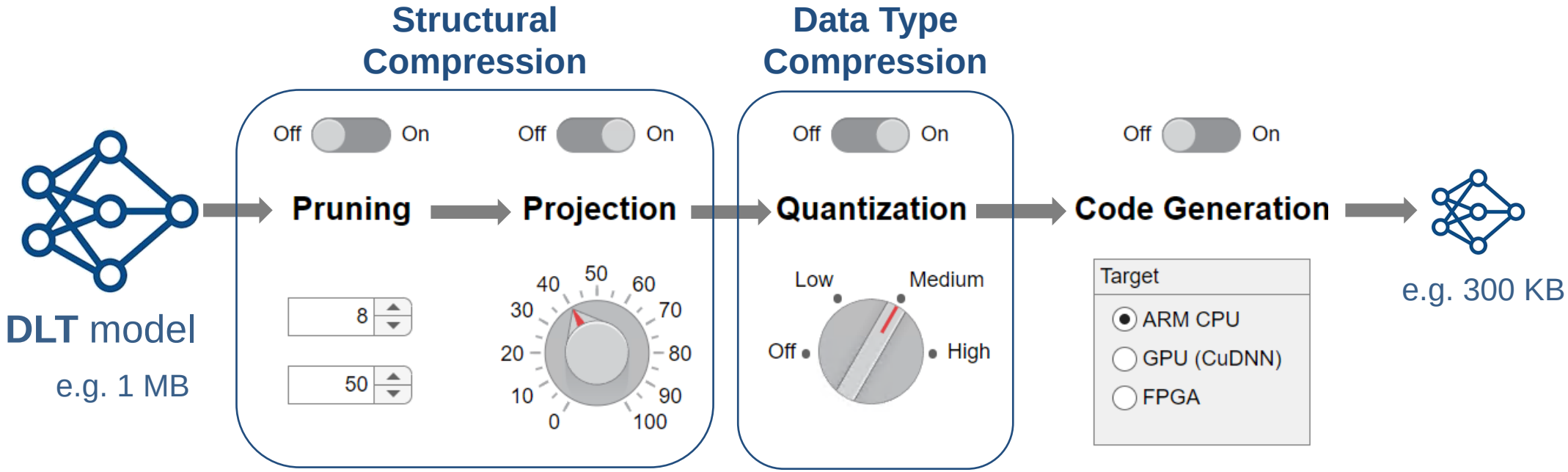


A look at today's Exercises

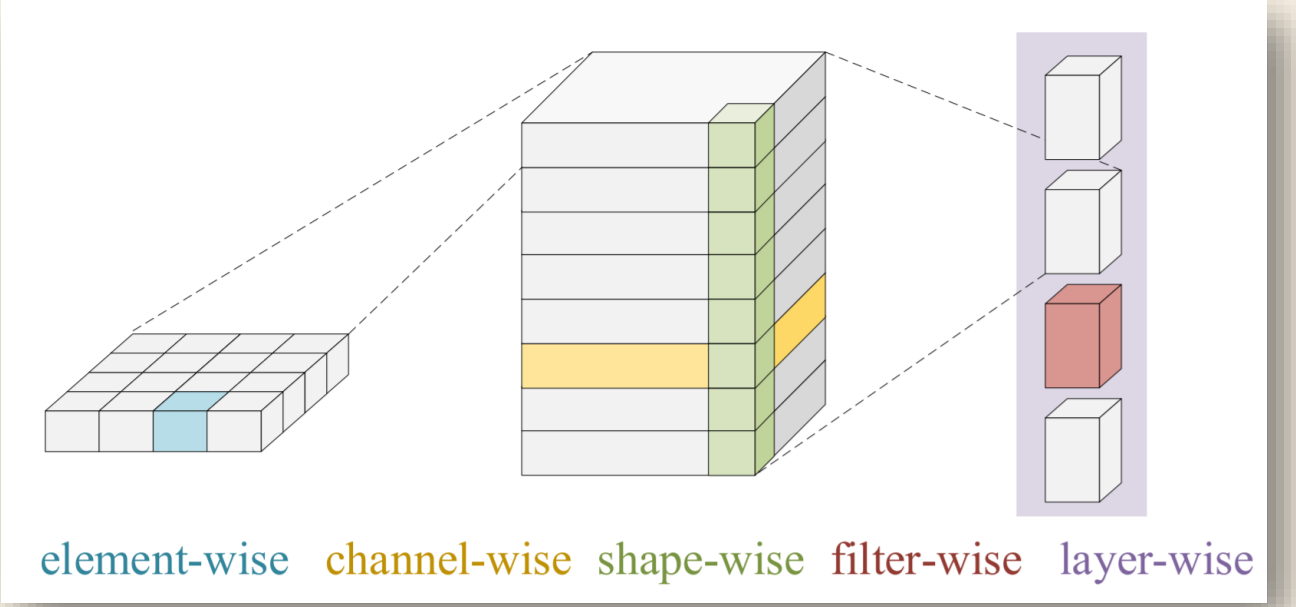


Step 4
Optimize AI Model with Pruning and Quantization

Reduce model footprint and accelerate inference of deep learning models



Structural compression aims to reduce the dimension to speed up inference computation



individual connections introduces sparsity

UNSTRUCTURED

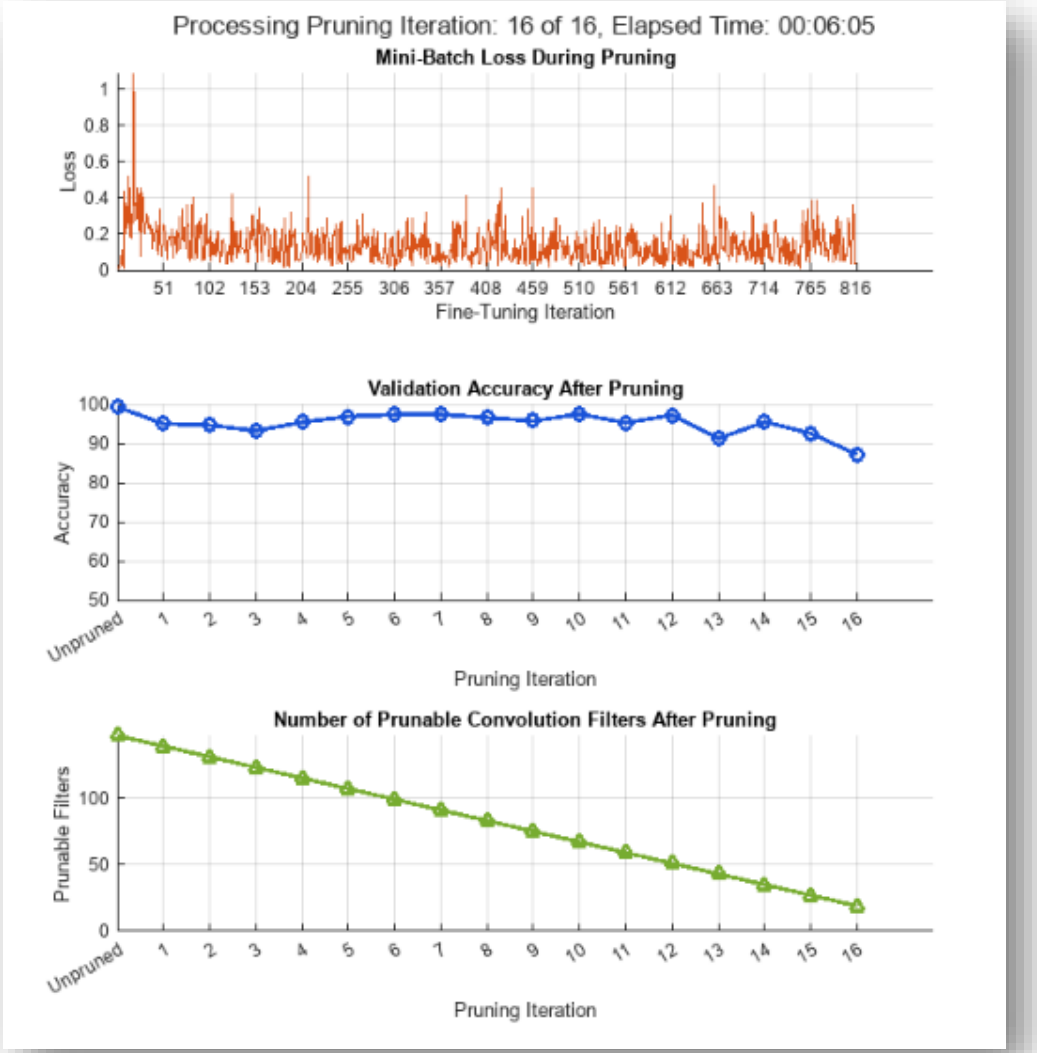
e.g. conv. filters, neurons in FC layer

STRUCTURED

Pruning and Quantization for Deep Neural Network Acceleration: A Survey. Liang et al. https://arxiv.org/pdf/2101.09671.pdf

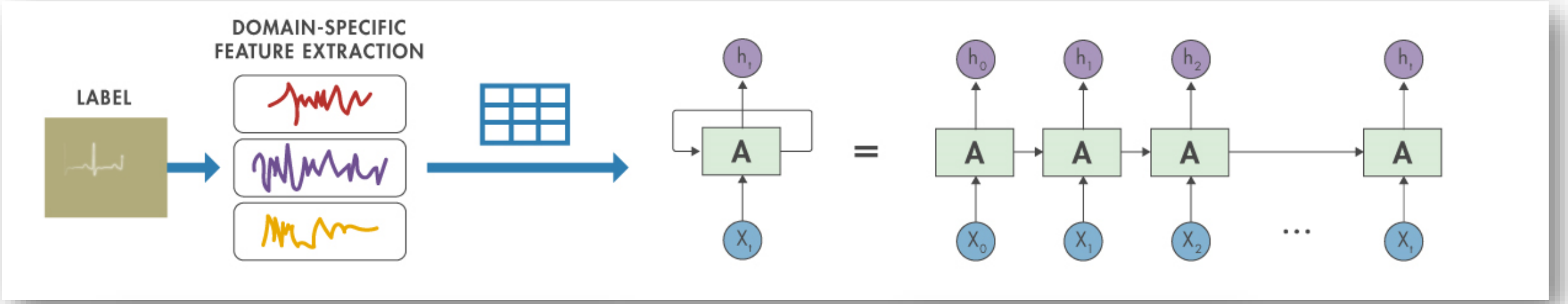
Taylor Pruning uses gradient score and eliminates number of filters in convolutional layers

- Gradient-based method to estimate filter “importance” using first-order Taylor expansion
- Prune less important filters to reduce model size while maintaining predictive power
- **STRUCTURED** approach
- Fine-tune pruned model with data



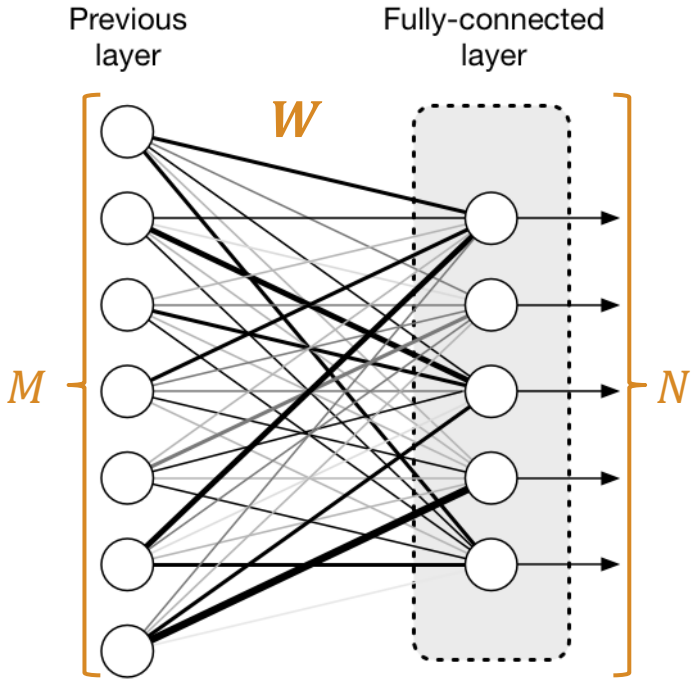
Structural pruning can reduce problem dimensions via projection into subspace

For example, a LSTM Networks



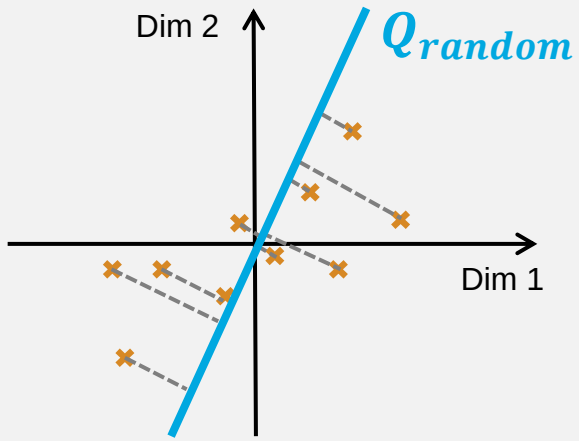
Two-step approach: Projection compression with neuron PCA

High-dimensional space of input and output neurons is underutilized



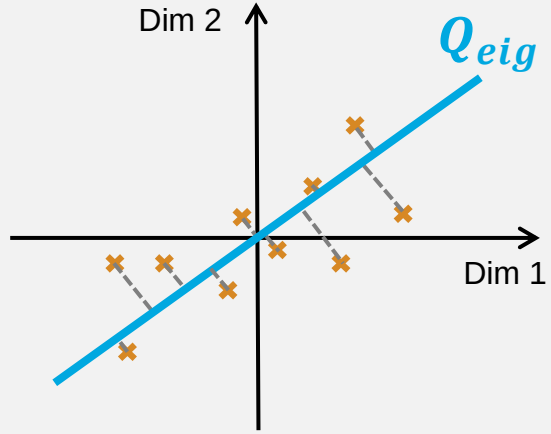
W is an N-by-M matrix

Dimensionality reduction via projection into subspace



projected layer

Minimize projection error via principal component analysis (PCA) of neurons



neuronPCA

Structural compression of LSTM layers can reduce model sizes

Compress the network.

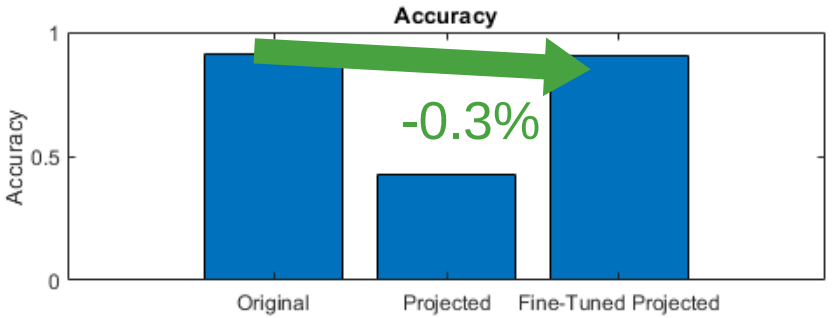
```
netProjected = compressNetworkUsingProjection(net,mbq);
```

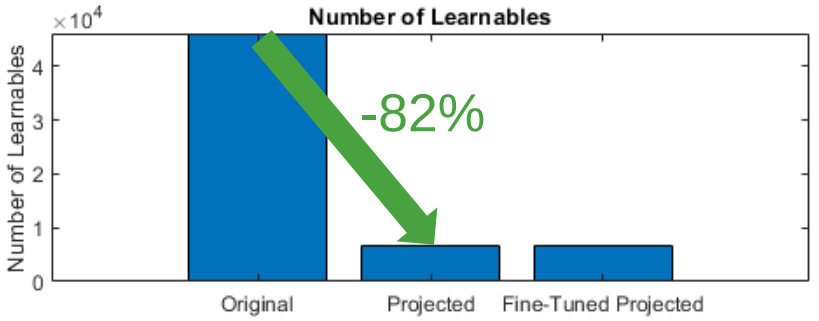
```
Compressed network has 82.4% fewer learnable parameters.
Projected layers explain on average 96.6% of layer activation variance.
```

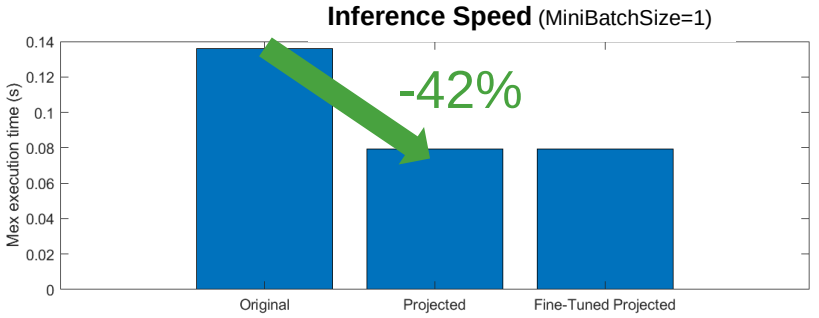
Optionally, precompute neuronPCA analysis for efficient experimentation:

```
npca = neuronPCA(netOriginal,mbqTrain,VerbosityLevel="steps");
```

```
Computing layer activations and covariance matrices...
Computing eigenvalues and eigenvectors...
neuronPCA analyzed 1 layers: "lstm"
```

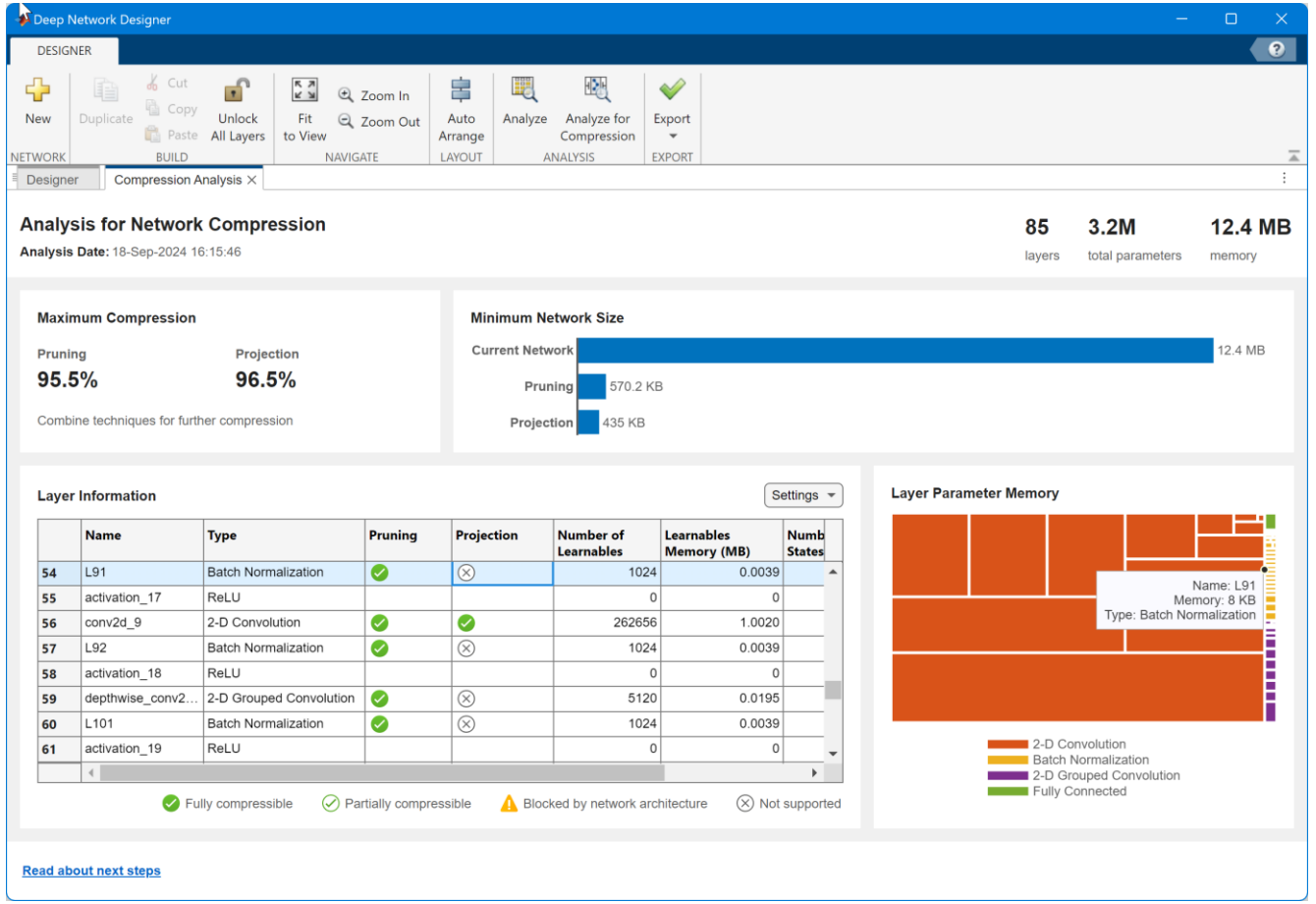






Doc Example: [Compress Neural Network Using Projection]
(Seq-2-One Classification on Japanese Vowels data set)

Give it a try with Ex 5



46

Quantization Theory made easy: Representation

Base 10, like dollars and cents

$$\begin{array}{ccccccc} \times & \times & \times & . & \times & \times \\ 10^{2} & 10^{1} & 10^{0} & & 10^{-1} & 10^{-2} \end{array}$$

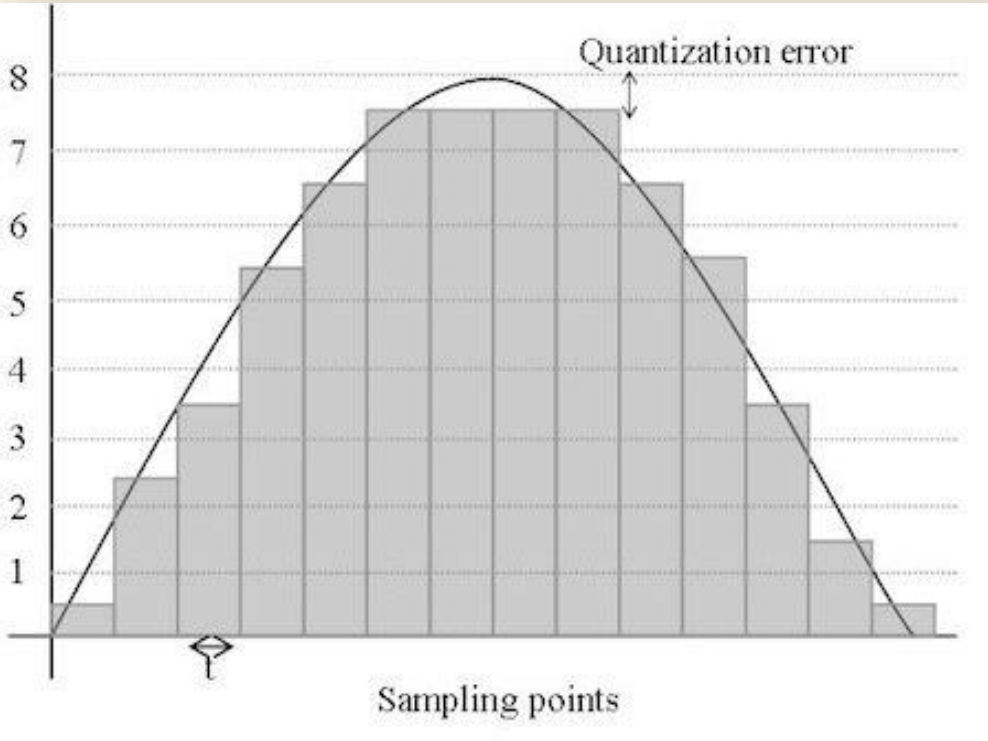
Base 2 fixed-point
total of 5 bits, unsigned

$$\begin{array}{ccccccc} \times & \times & \times & . & \times & \times \\ 2^{2} & 2^{1} & 2^{0} & & 2^{-1} & 2^{-2} \end{array}$$

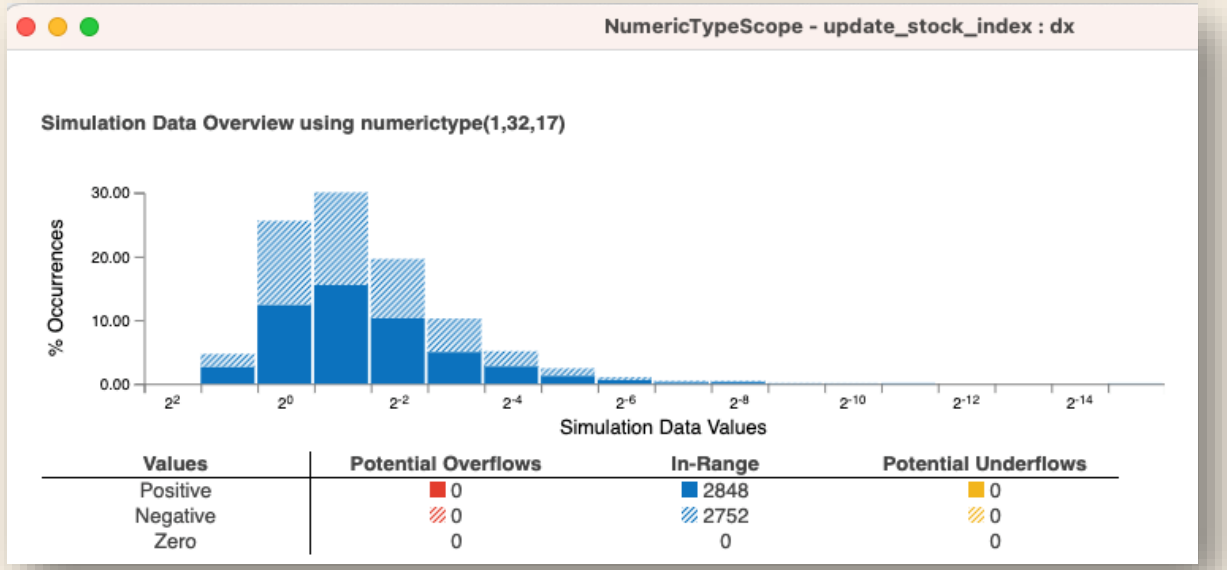
Programmatic example:

```
>> dx = fi(0.01, 1, 32, 10)

dx =       0.009765625
         DataTypeMode: Fixed-point: binary point scaling
           Signedness: Signed
           WordLength: 32
       FractionLength: 10


>> lsb(dx)
ans =      0.0009765625

>> range(dx)
ans =     -2097152          2097151.99902344

>> bin(dx)
ans =    '00000000000000000000000000001010'
                                       ^ ^
                                    2^-7 2^-9
```

Quantization Theory made easy: Errors



- **Overflow**
- **Underflow**
- **Quantization error**

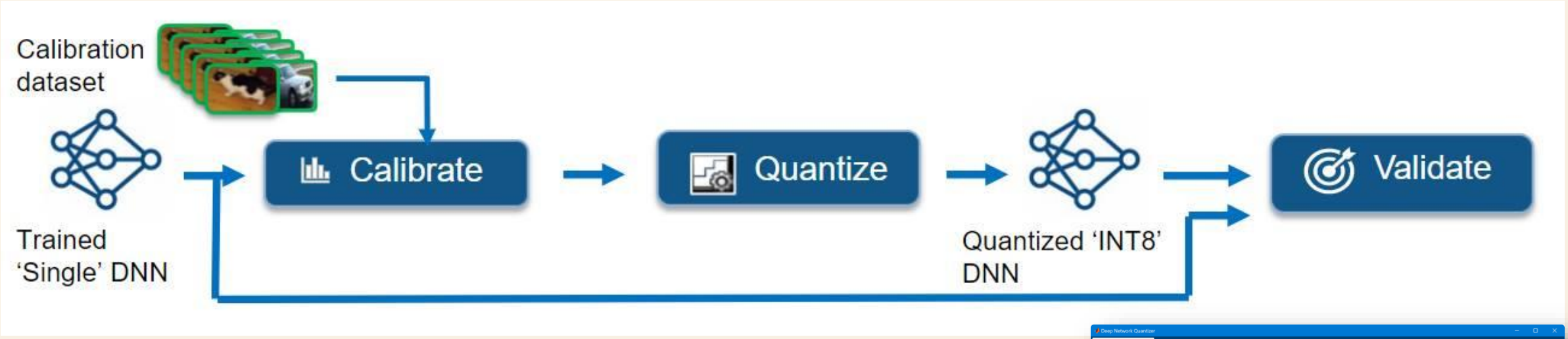


Quantization Theory made easy: Arithmetic

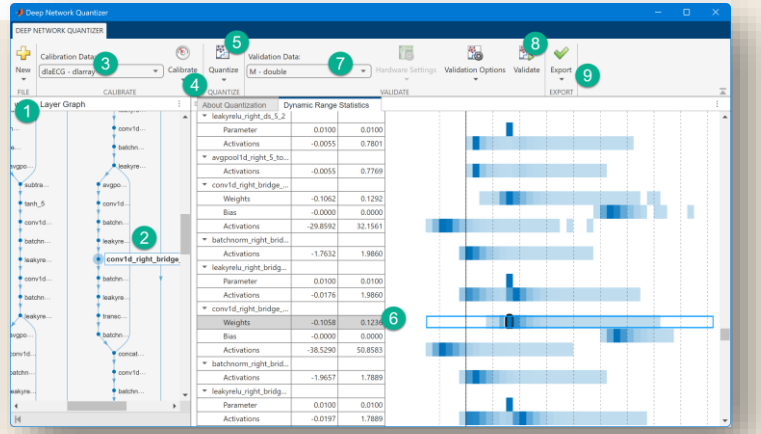
+	10^4-1=	9999	+	2^4-1=	1111 binary
	10^4-1=	9999		2^4-1=	1111 binary
		19998			11110

*** Full-precision result requires 1 digit of growth**

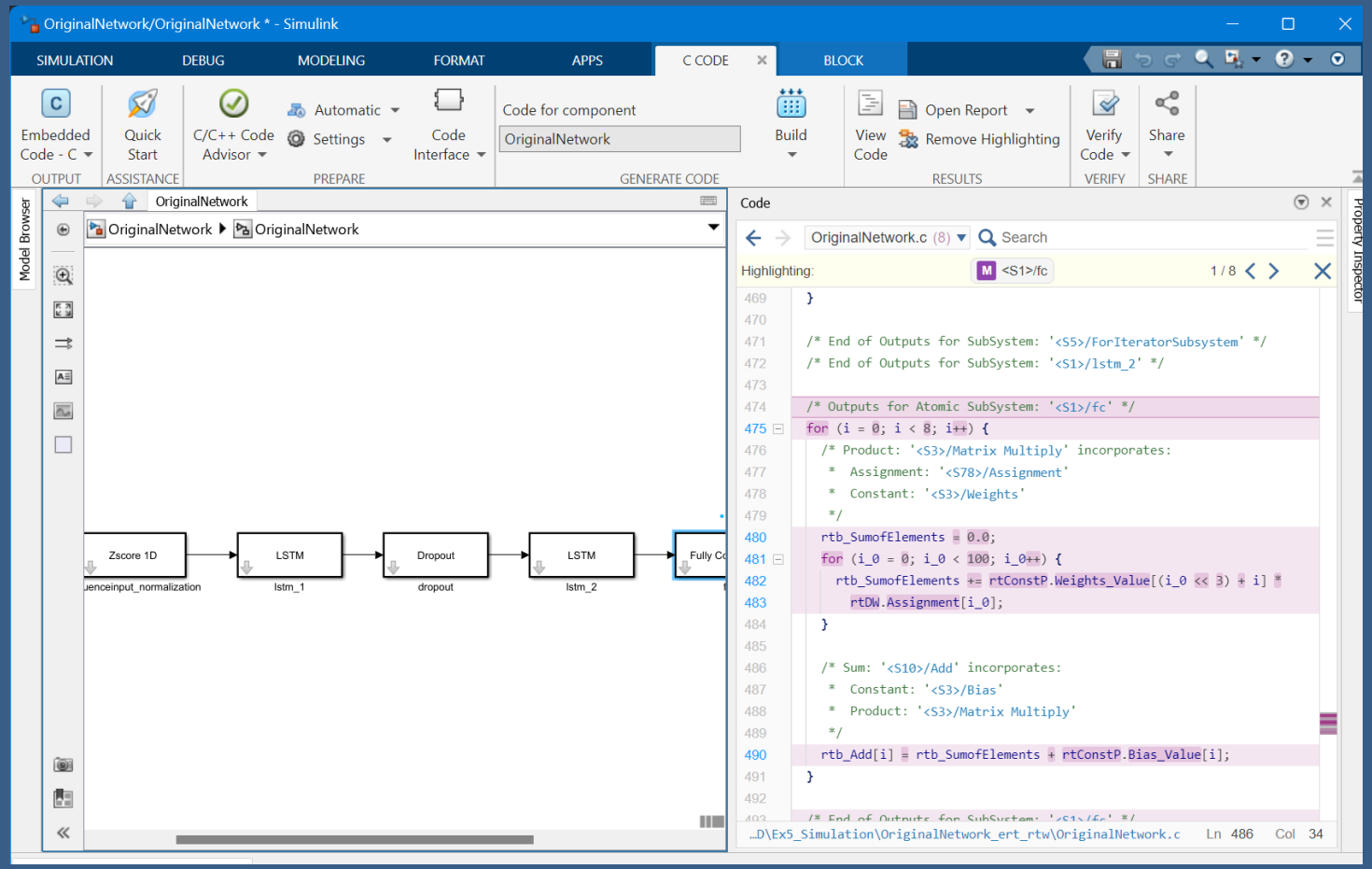
Quantizing a neural network is the balancing act among these factors



- Post-training quantization reduces the **memory requirements** for parameters and activation functions.
- The inference **runtime** can improve on device.



A look at today's Exercises



Step 5
Deploy using Embedded C Code Generation

Step 6
Approaches to integrate with Simulink

TACKLE THE CHALLENGES

The need to **optimize performance** and **energy efficiency** for AI in embedded systems.

Model-Based Design (MBD) workflow from **MathWorks** simplifies this process, enabling efficient deployment of AI models to the edge devices.