

Setup for Workshop

Requirements:

- MathWorks Account
- MATLAB Online: use supported browser (Google Chrome or Microsoft Edge)

- Get the workshop materials: https://github.com/resjoehr/AMLD_2025
- Access the MATLAB Online license: <https://tinyurl.com/AMLD-2025>

The screenshot shows a GitHub README page for the "MathWorks Workshop at AMLD 2025: Developing and Deploying AI Algorithms on GPU Accelerated Hardware". The page includes a title, a section for "Access to the workshop materials", and a button labeled "Open in MATLAB Online". A red arrow points to this button, and the text "Click here" is overlaid in red.

README

MathWorks Workshop at AMLD 2025: Developing and Deploying AI Algorithms on GPU Accelerated Hardware

Access to the workshop materials

Click on the button below to download the materials.

Open in MATLAB Online

Click here



Developing and Deploying AI Algorithms on GPU Accelerated Hardware

Workshop at Applied Machine Learning Days, 14.02.2025

Speakers



Res Jöhr

Senior Application Engineer (Universities)
resjoehr@mathworks.com



Sébastien Dupertuis

Senior Application Engineer



Franziska Albers

Senior Application Engineer (Universities)

Agenda

14:00 - 14:30 **Overview of AI Deployment Workflow**



Res JÖHR, Switzerland

14:30 - 15:30 **Data Preparation and Model Development**



Franziska ALBERS, Germany

15:30 - 16:00 **Coffee break**

16:00 - 16:45 **Model Verification and Code Generation**



Sébastien DUPERTUIS, Switzerland



Res JÖHR, Switzerland

16:45 - 17:30 **Model Deployment to GPU Accelerated Hardware**



Sébastien DUPERTUIS, Switzerland



Res JÖHR, Switzerland

AMLD 2025: Let's get to know one
another!



<https://forms.office.com/r/ZdHD3p4hpS>

Technical Setup

```
>> answer = Do you have a MathWorks account?  
  
>> If answer==true  
    Follow the short intro;  
else  
    Please go to mathworks.com/login and register;  
end  
  
>> Access workshop materials after introduction
```

About MathWorks

We at MathWorks believe in the importance of engineers and scientists. They increase human knowledge and profoundly improve our standard of living.

We created MATLAB and Simulink to help them do their best work.



6500+ staff

34 offices worldwide



Privately held
and profitable every year

MATLAB is used to design the products we rely on every day



Aerospace and Defense



Automotive



Biological Sciences



Biotech and Pharmaceutical



Communications



Electronics



Energy Production



Financial Services



Industrial Machinery



Medical Devices



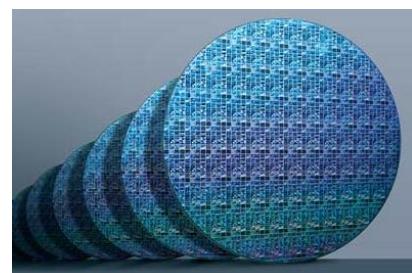
Process Industries



Neuroscience



Railway Systems

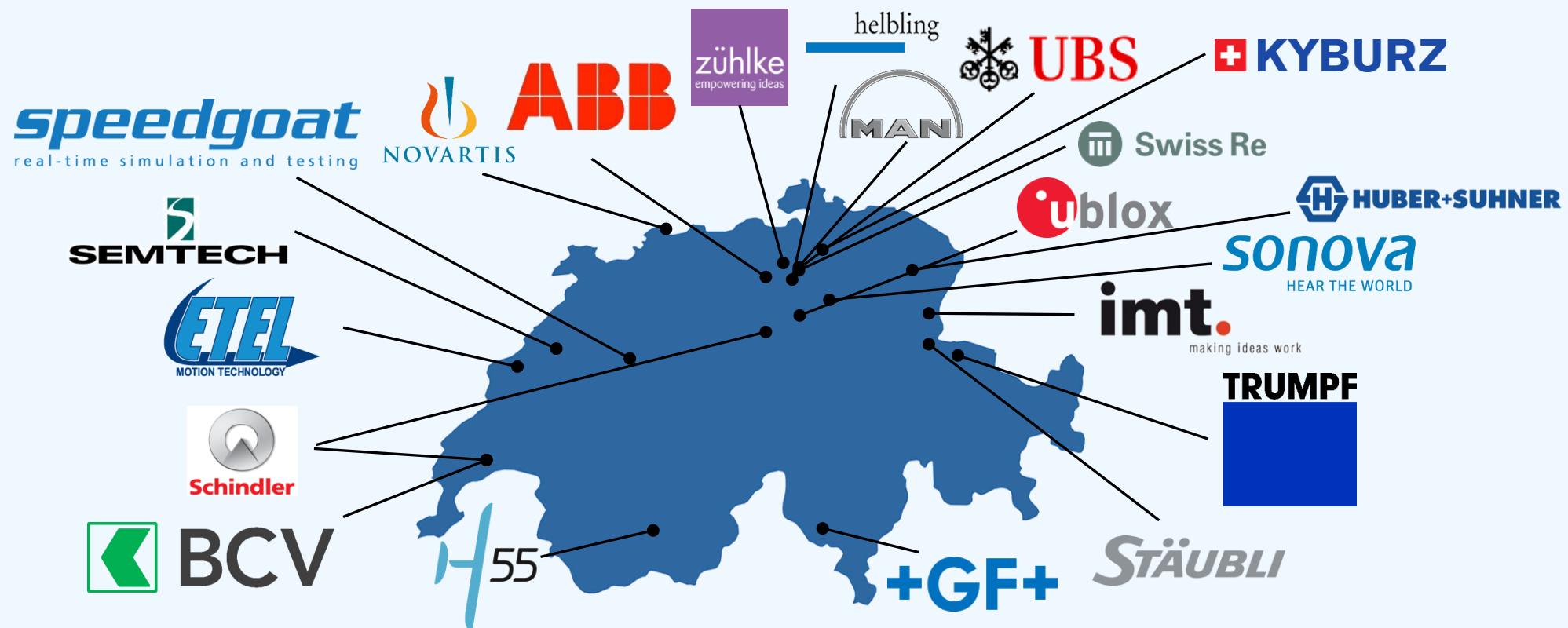


Semiconductors



Software and Internet

Commercial Usage in Switzerland



100,000+ business,
government, and
university sites



Top 10 auto
manufacturers¹



Top 10 aerospace
companies²



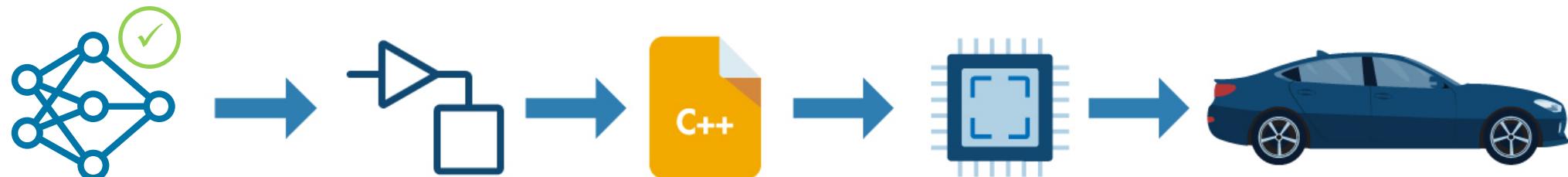
Three of the top five
internet companies

¹OICA: 2016 World Motor Vehicle Production

²PwC: Aerospace and Defense 2017 Year in Review

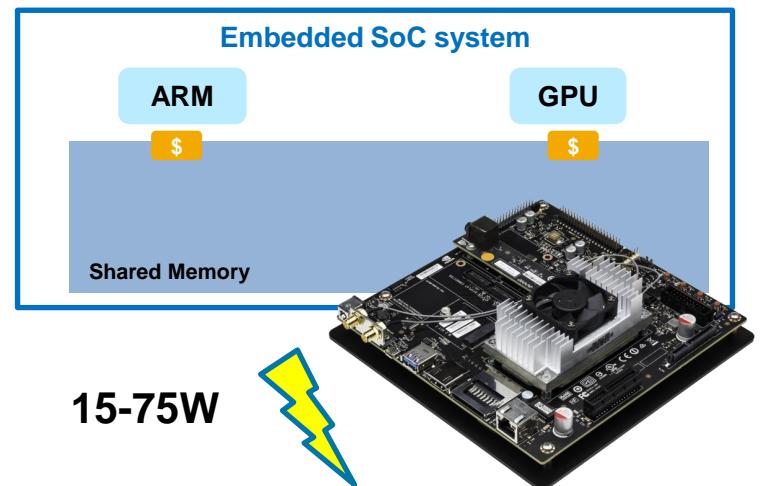
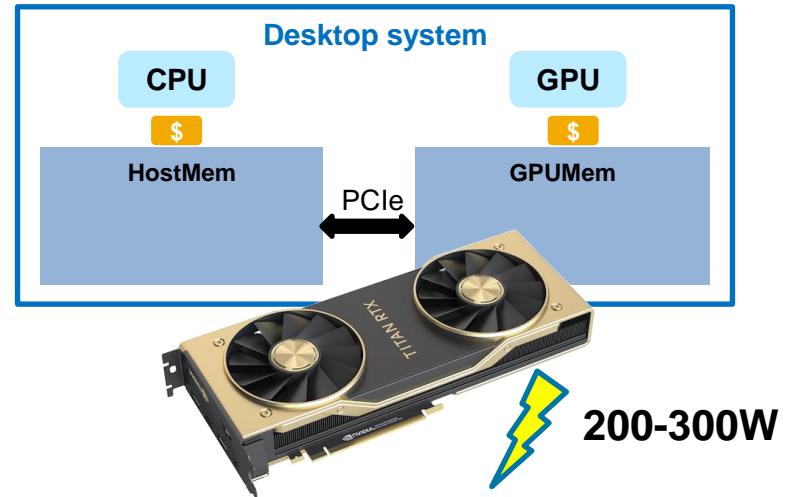
Integrating AI Algorithms into embedded Hardware

- Motivation:
 - AI algorithms can handle complex tasks that can hardly be solved using classical methods
 - Would like to use these algorithms on edge devices
- Issues:
 - How to integrate models into larger systems and deploy them to embedded hardware
 - Availability of computational resources (processor, memory)
 - Translation of code to hardware language (code generation)
 - AI models are sometimes hard to understand (black box, interpretability, robustness)
 - Models have to be verified and validated



Types of GPUs

- Desktop GPUs
 - Designed for performance (more memory and cores)
 - For interactive tasks
(e.g. simulations, video editing, 3D rendering)
- Embedded GPUs
 - Designed for integration into compact or power constrained environments
 - Optimized for specific tasks
(e.g. real-time processing and AI inference)
 - Often requires compiled code
(CUDA code needed to leverage benefits of NVIDIA GPUs)



Drass Develops Deep Learning System for Real-Time Object Detection in Maritime Environments

Challenge

Help ship operators monitor sea environments and detect objects, obstacles, and other ships

Solution

Create an object-detection deep learning model that can be deployed on ships and run in real time

Results

- Data labeling automated
- Development time reduced
- Flexible and reproducible framework established



First day of object detection tests with optronic system prototype.

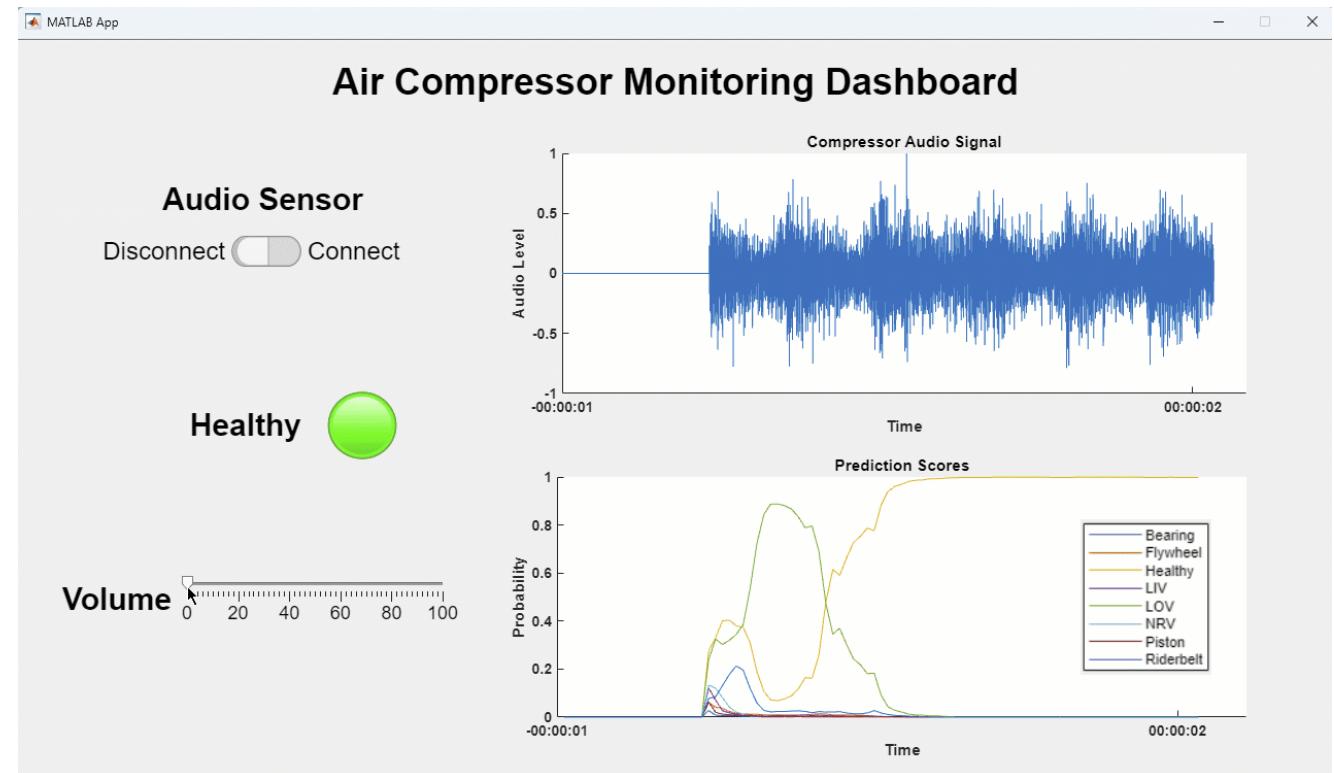
“From data annotation to choosing, training, testing, and fine-tuning our deep learning model, MATLAB had all the tools we needed—and GPU Coder enabled us to rapidly deploy to our NVIDIA GPUs even though we had limited GPU experience.”

- Valerio Imbriolo, Drass Group

Example: Classifying Sounds

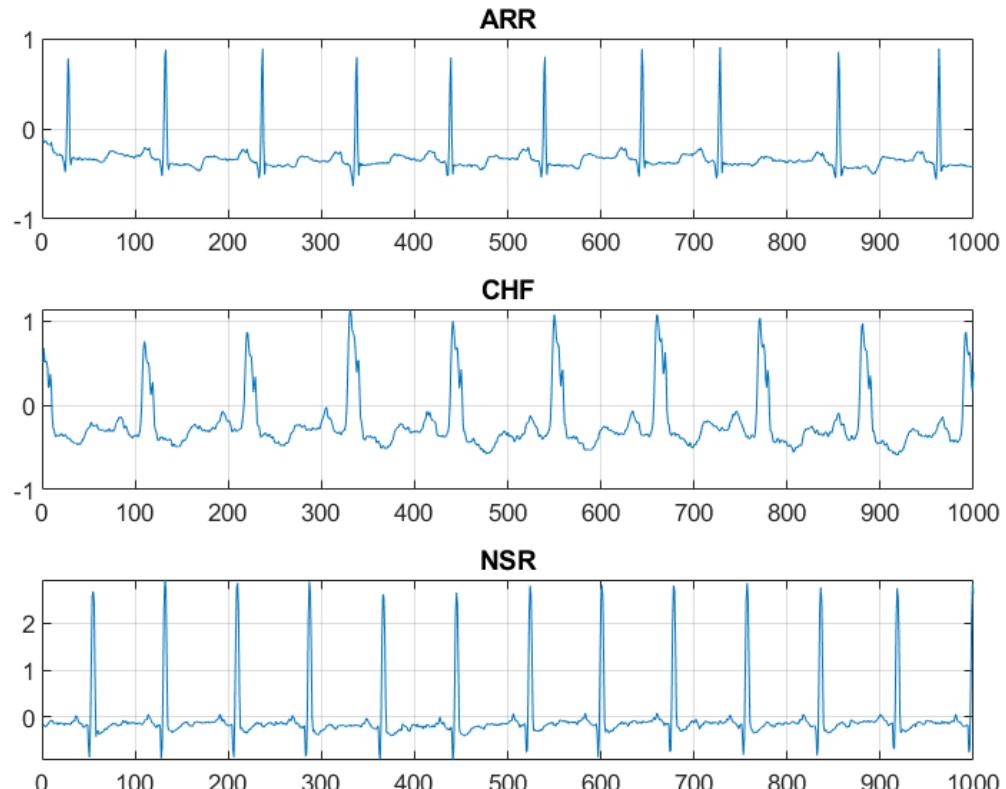
Build, test, and deploy a deep learning solution that can classify sequences in signal data.

- **Goal:** Detect faults in an air compressors based on their noise

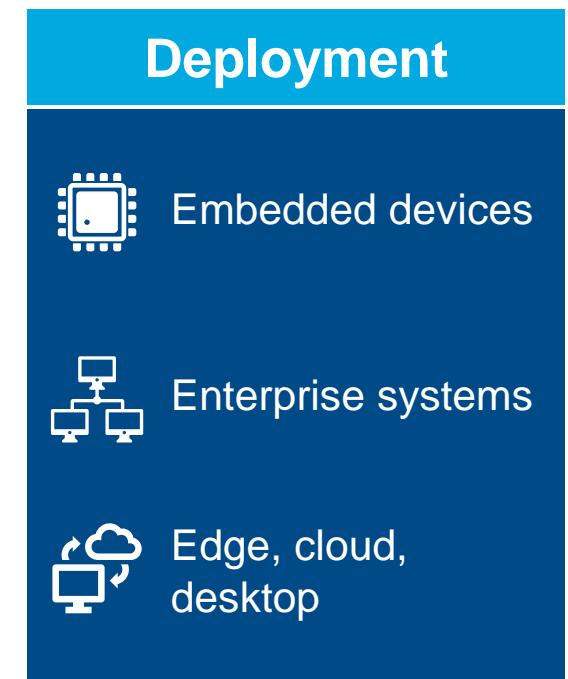
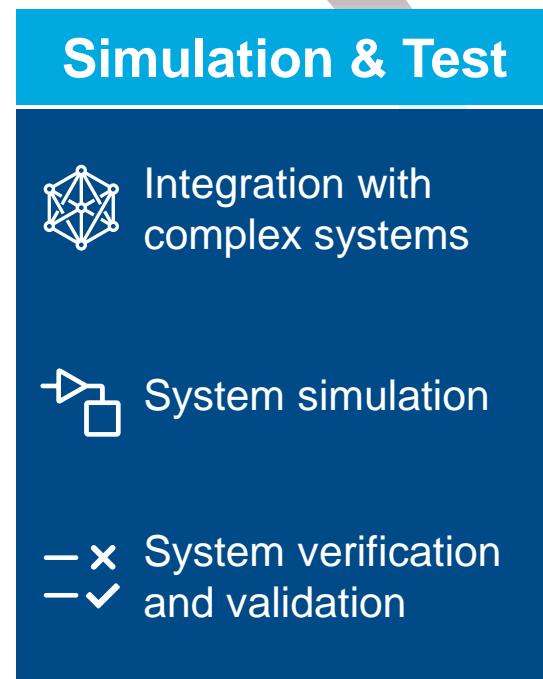
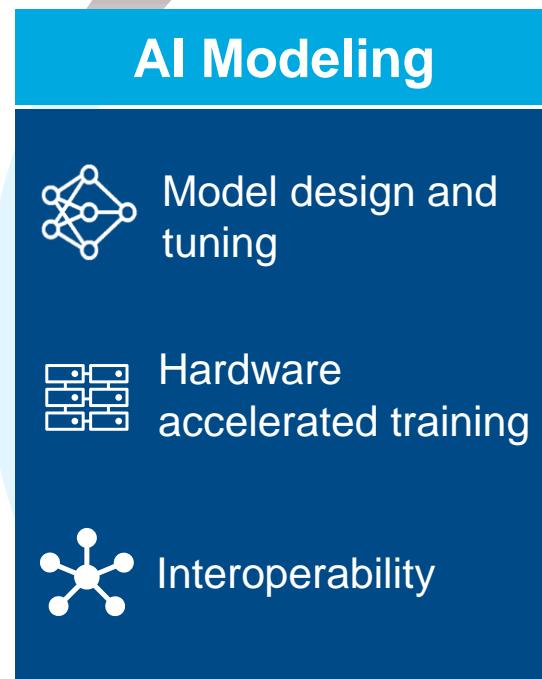
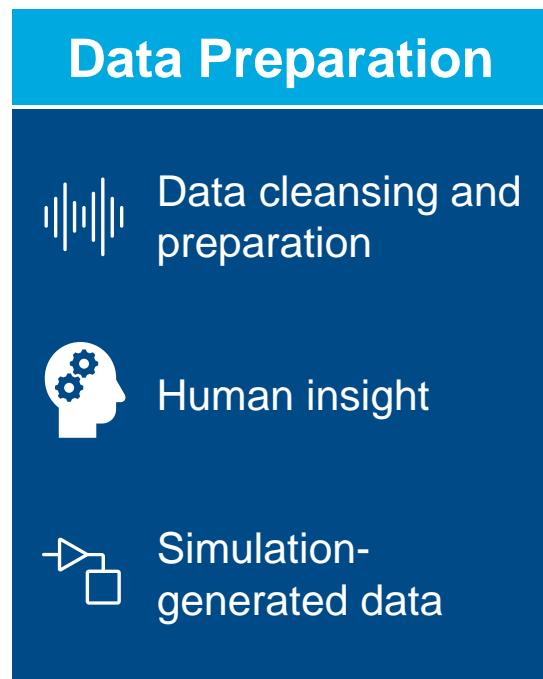


Today's use case: ECG Classification

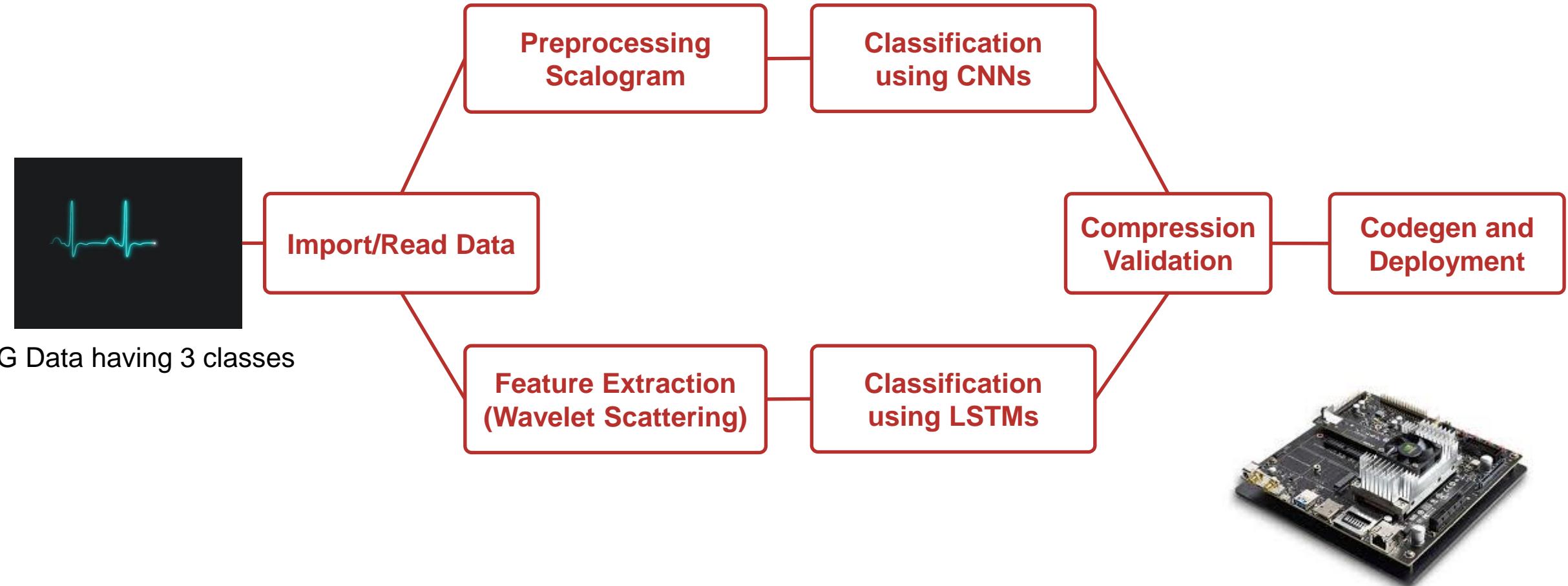
- Dataset contains 3 classes of heartbeats
 - cardiac arrhythmia (ARR, N=96)
 - congestive heart failure (CHF, N=30)
 - normal sinus rhythms (NSR, N=36)
- PhysioNet databases:
 - [MIT-BIH Arrhythmia Database](#)
 - [MIT-BIH Normal Sinus Rhythm Database](#)
 - [The BIDMC Congestive Heart Failure Database](#)
- References for the dataset
- Timeseries signals with 128Hz
- 162 recordings with 65535



AI-Driven System Design



In this workshop we will ...



Setup Check-In

Requirements:

- MathWorks Account
- MATLAB Online: use supported browser (Google Chrome or Microsoft Edge)

- Get the workshop materials: https://github.com/resjoehr/AMLD_2025
- Access the MATLAB Online license: <https://tinyurl.com/AMLD-2025>

The screenshot shows a GitHub README page for the "MathWorks Workshop at AMLD 2025: Developing and Deploying AI Algorithms on GPU Accelerated Hardware". The page includes a title, a section for "Access to the workshop materials", and a button labeled "Open in MATLAB Online". A red arrow points to this button, and the text "Click here" is overlaid in red.

README

MathWorks Workshop at AMLD 2025: Developing and Deploying AI Algorithms on GPU Accelerated Hardware

Access to the workshop materials

Click on the button below to download the materials.

Open in MATLAB Online

Click here

AI-Driven System Design

Data Preparation

 Data cleansing and preparation

 Human insight

 Simulation-generated data

AI Modeling

 Model design and tuning

 Hardware accelerated training

 Interoperability

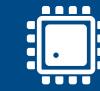
Simulation & Test

 Integration with complex systems

 System simulation

 System verification
✓ and validation

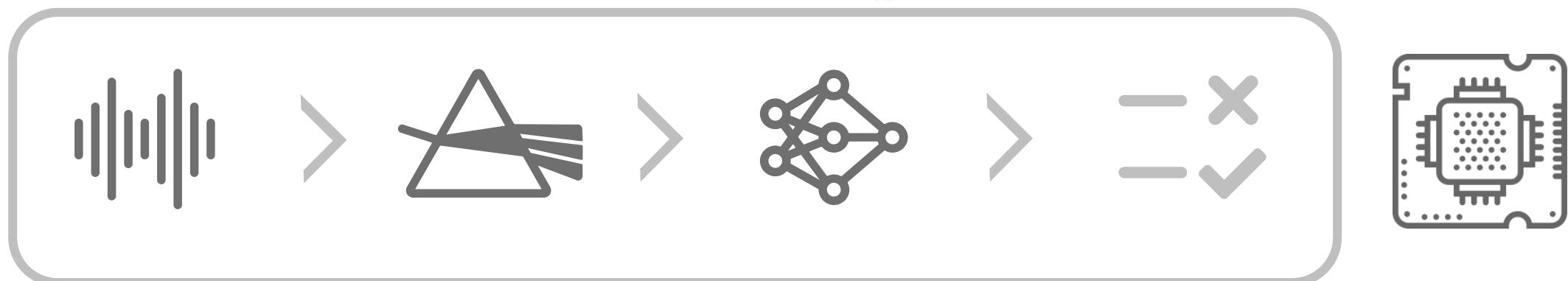
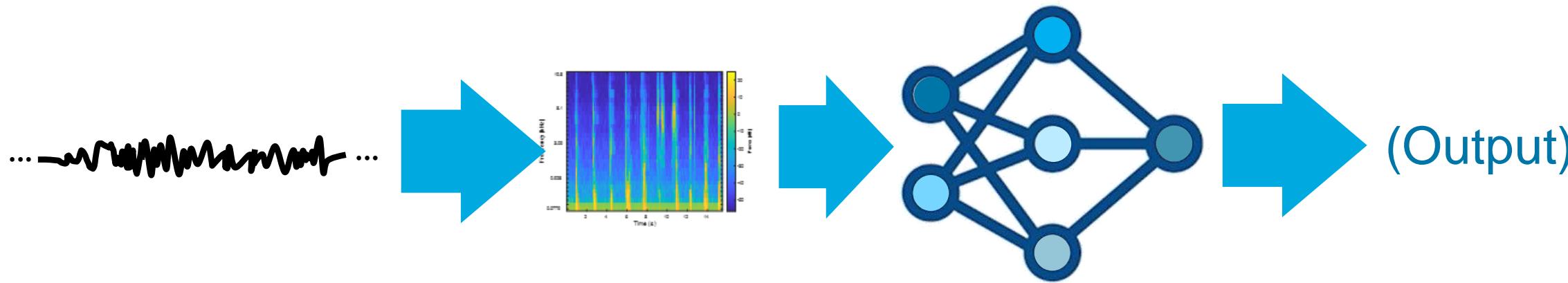
Deployment

 Embedded devices

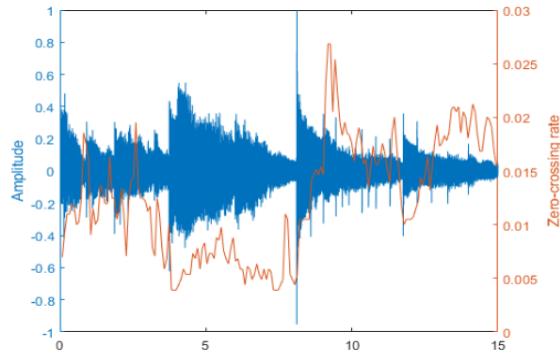
 Enterprise systems

 Edge, cloud, desktop

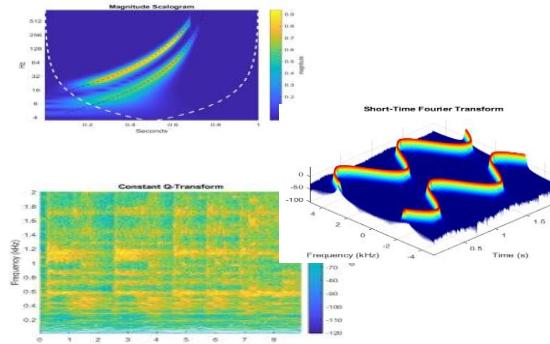
Deep networks most often don't learn directly from raw signals



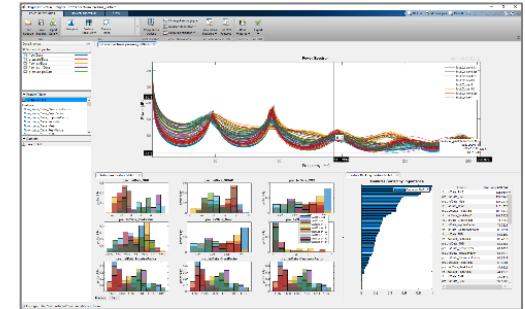
Transform raw data for useful modeling and analysis



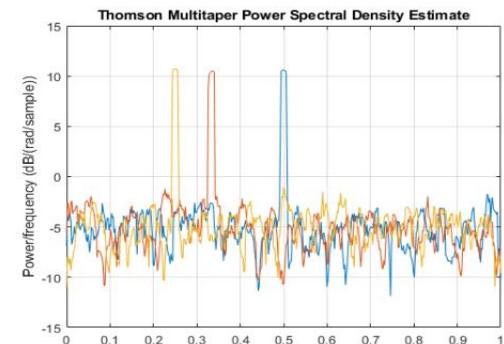
`signalTimeFeatureExtractor`



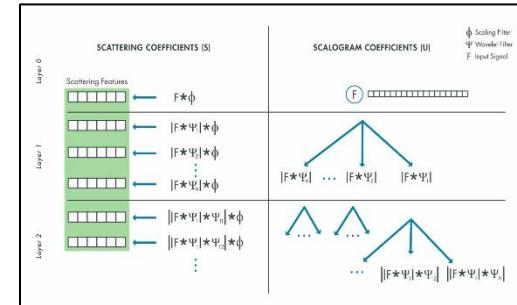
`signalFrequencyFeatureExtractor`



`diagnosticFeatureDesigner`

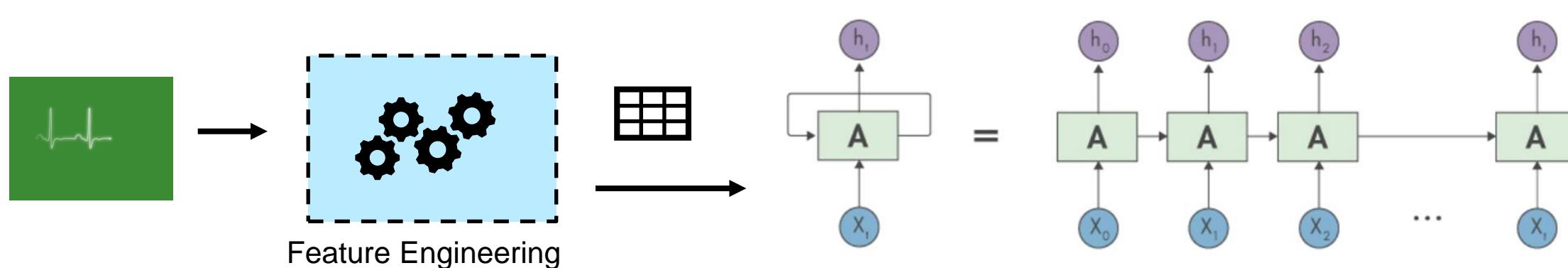
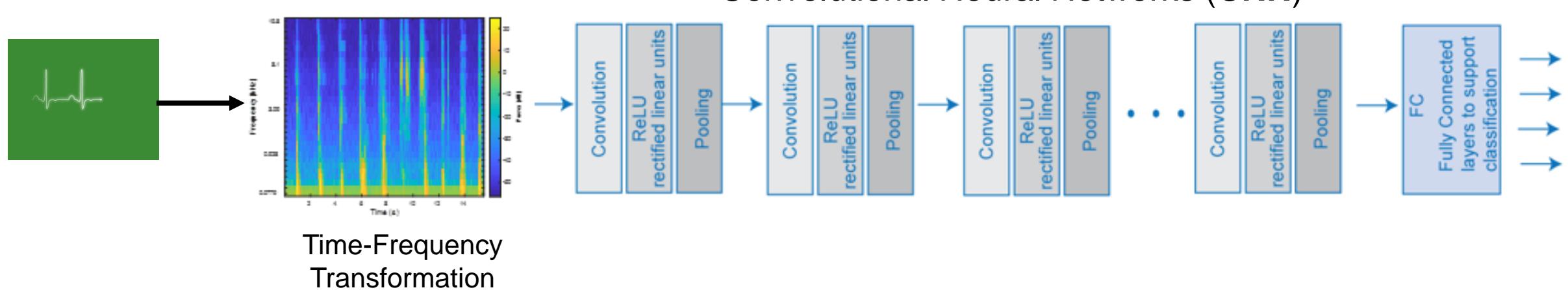


`signalTimeFrequencyFeatureExtractor`



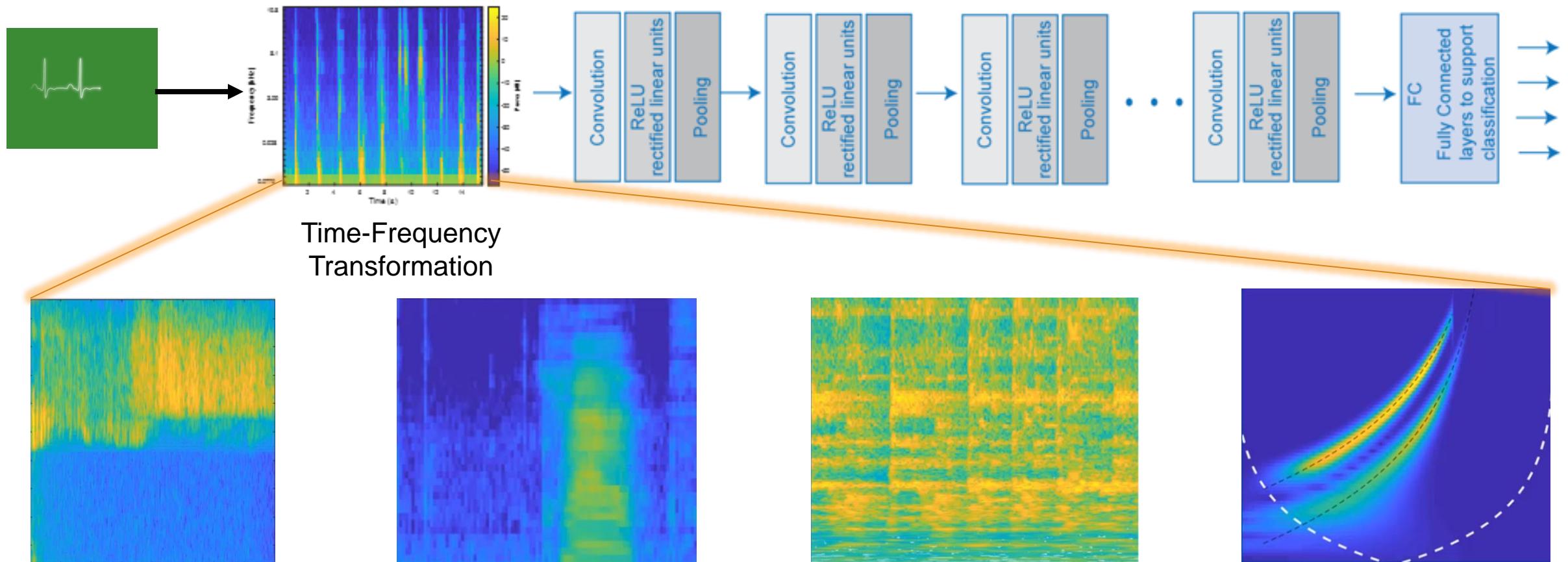
Wavelet based features

Common Network Architectures – Signal Processing



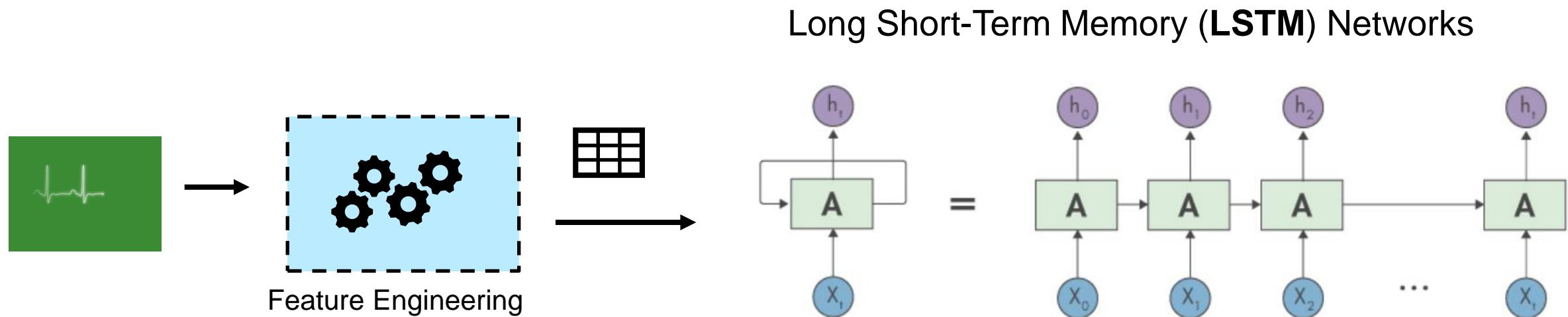
Common Network Architectures – Signal Processing

Convolutional Neural Networks (CNN)



mathworks.com/help/signal/ug/time-frequency-gallery.html

Common Network Architectures – Signal Processing



mathworks.com/discovery/lstm.html

Wavelet Scattering Basic Idea

- method to generate features from 1D and 2D input data with minimal configuration
- produces data representations that
 - minimize differences *within* a class
 - while preserving discriminability *across* classes
- works with a small amount of data

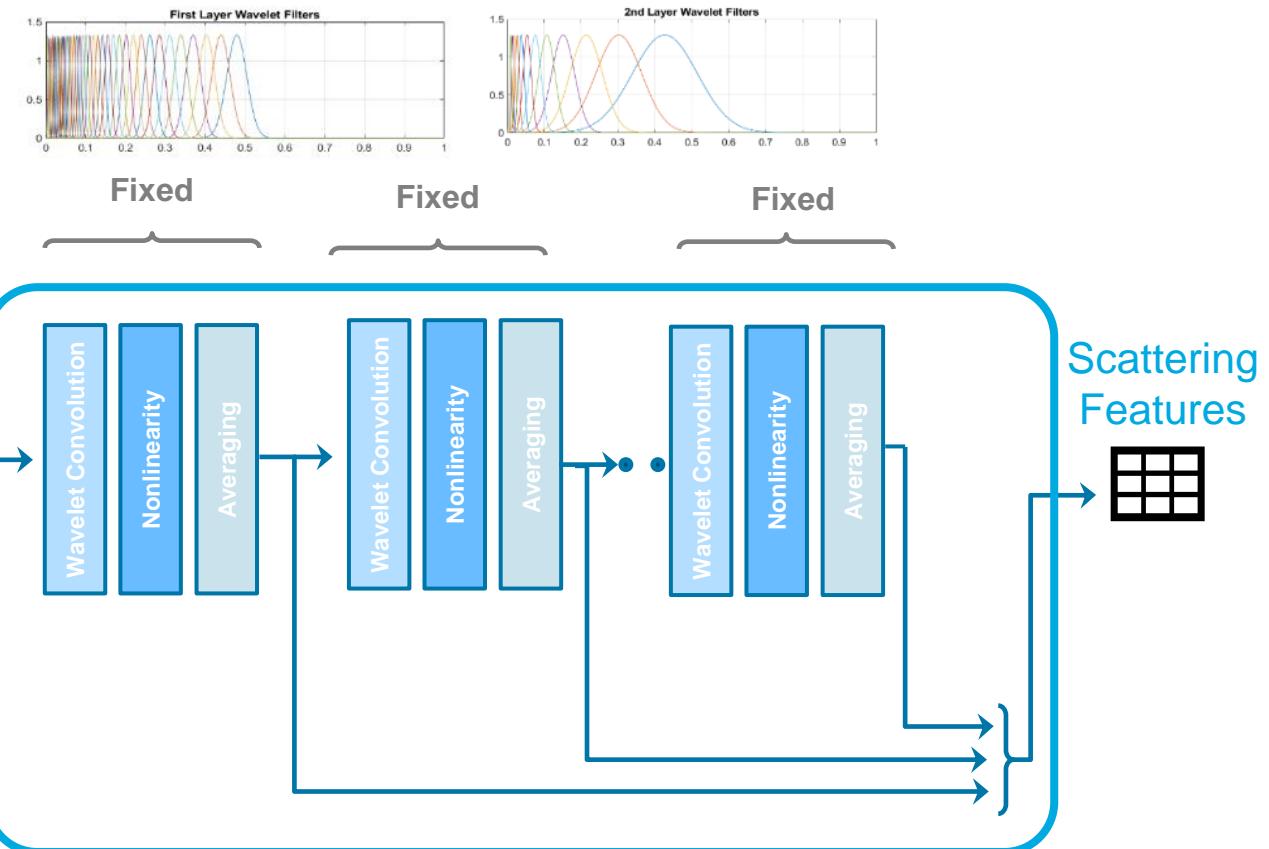


Additional Resources:

- [Wavelet scattering Tech talk](#) [4 min video]
- [Wavelet scattering for ECG](#) [doc example]
- [Blog about Wavelet scattering](#)

Deep Feature Extraction with Wavelets

- **waveletScattering**
 - Uses structure of Wavelet-based convolutional layers
 - State-of-the-art results:
 - ECG classification
 - Music genre classification
 - Spoken digit recognition
 - Acoustic scene recognition



Wavelet Scattering extracts features in a similar way to CNNs

CNN

- Step 1: Convolution
 - Learn convolution filters during training
- Step 2: Apply non-linearity
 - Use ReLu function
- Step 3: Apply averaging
 - Use pooling function

Wavelet Scattering

- Step 1: Convolution
 - Use wavelets as **fixed** convolution kernels
- Step 2: Apply non-linearity
 - Use **modulus function** (= calculate absolute value)
- Step 3: Apply averaging
 - Convolve with **scaling function** to apply time averaging

AI-Driven System Design

Data Preparation

 Data cleansing and preparation

 Human insight

 Simulation-generated data

AI Modeling

 Model design and tuning

 Hardware accelerated training

 Interoperability

Simulation & Test

 Integration with complex systems

 System simulation

 System verification
and validation

Deployment

 Embedded devices

 Enterprise systems

 Edge, cloud, desktop

Deep Learning and Machine Learning Models for Signals

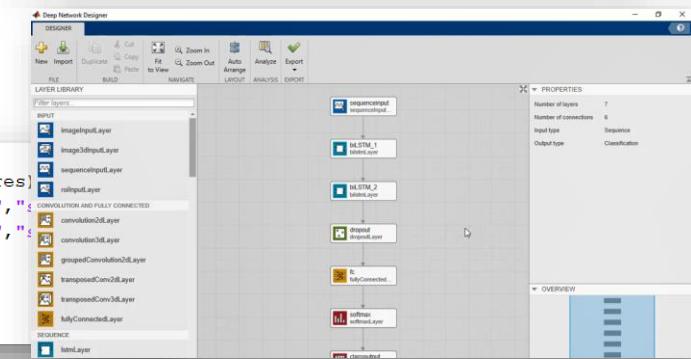
Three ways to get started in MATLAB

1. Start from a model available in MATLAB

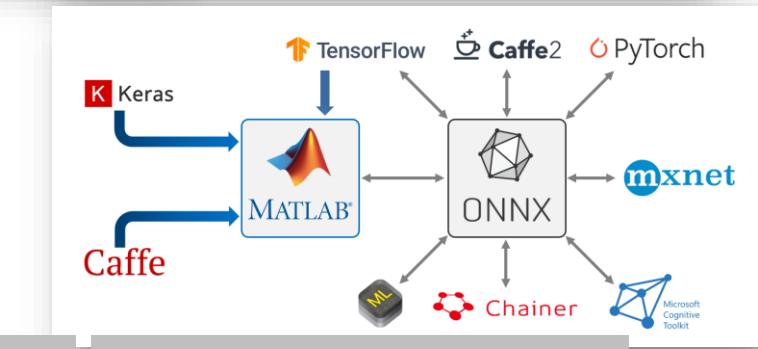
Pre-Trained Models for Audio, Speech, and Acoustics Shipping features with wide workflow coverage					
Model name	Pre-training data	Model / Network	Pre/post-process	Single-line	Application
YAMNet (2D CNN)	AudioSet corpus + ontology (YouTube audio, 16 kHz sample rate) 2M 10s human-labeled clips	yamnet	yamnetPreprocess yamnetGraph	classifySound	Sound type classification, segmentation, content tagging....
VGGish (2D CNN)	AudioSet (same as above)	vggish	vggishPreprocess	vggishFeatures	Audio Embeddings
OpenL3 (2D CNN)	AudioSet (musical + environment subsets + videos, 48 kHz sample rate)	openl3	openl3Preprocess	openl3Features	Audio Embeddings
CREPE (1D CNN)	6 different monophonic musical datasets (16 kHz sample rate) - RE	crepe	crepePreprocess crepePostprocess	pitchnn	Pitch estimation
L-Vectors (System of Machine Learning models)	LibriSpeech data set (approx. 1000-hour English speech sampled, 16 kHz sample rate)	lvectorSystem	trainExtractor, trainClassifier, enroll, verify	speakerRecognition	Speaker verification ++

2. Define your own from scratch

```
layers = [ ...  
    sequenceInputLayer(numFeatures),  
    biLSTM(150, "OutputMode", "last"),  
    biLSTM(150, "OutputMode", "last"),  
    fullyConnectedLayer(2),  
    softmaxLayer,  
    classificationLayer  
];
```



3. Import one developed using another framework



Available Models in MATLAB

The screenshot shows the MATLAB Deep Network Designer interface. At the top, there are four buttons: 'Blank Network', 'From Workspace', 'From PyTorch' (with a warning icon), and 'From TensorFlow'. Below this, there are sections for 'Image Networks (Pretrained)' and 'Sequence Networks'. The 'Image Networks (Pretrained)' section contains icons for SqueezeNet, GoogLeNet, ResNet-50, EfficientNet-b0, DarkNet-53, DarkNet-19, ShuffleNet, NasNet-Mobile, NasNet-Large, Xception, Places365-Goog..., MobileNet-v2, DenseNet-201, ResNet-18, and Inception-ResNe... (with a 'Show more' link). The 'Sequence Networks' section contains icons for Sequence-to-Label and Sequence-to-Seque... (with a 'Show more' link). The 'Audio Networks (Pretrained)' section contains icons for CREPE, OpenL3, VADNet, VGGish, and YAMNet.

Train in MATLAB's **Deep Learning Framework**

Data Preparation

AI Modeling

Simulation & Test

Deployment

Deep Learning and Machine Learning Models for Signals

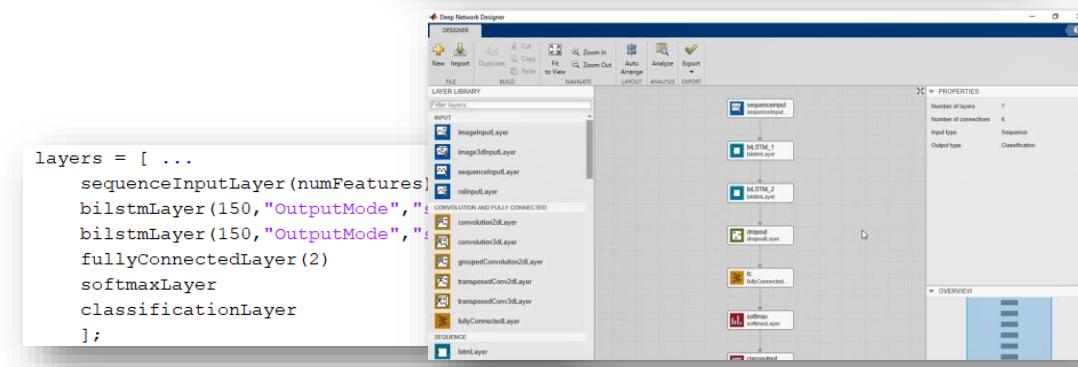
Three ways to get started in MATLAB

1. Start from a model available in MATLAB

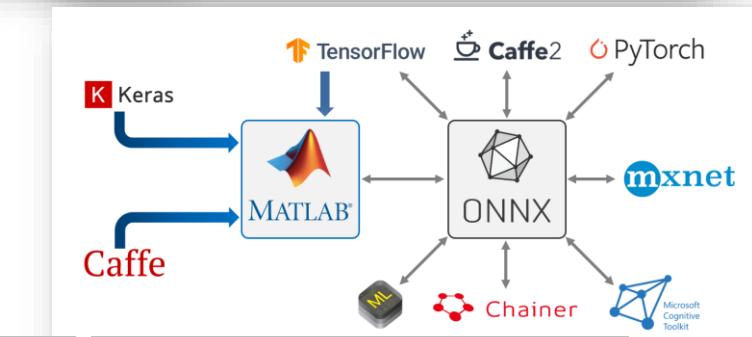
Pre-Trained Models for Audio, Speech, and Acoustics
Shipping features with wide workflow coverage

Model name	Pre-training data	Model / Network	Pre/post-process	Single-line	Application
YAMNet (2D CNN)	AudioSet corpus + ontology (YouTube audio, 16 kHz sample rate) 2M 10s human-labeled clips	yamnet	yamnetPreprocess yamnetGraph	classifySound	Sound type classification, segmentation, content tagging....
VGGish (2D CNN)	2M 10s human-labeled clips (same as above)	vggish	vggishPreprocess	vggishFeatures	Audio Embeddings
OpenL3 (2D CNN)	AudioSet (musical + environment subsets + videos, 48 kHz sample rate)	openl3	openl3Preprocess	openl3Features	Audio Embeddings
CREPE (1D CNN)	6 different monophonic musical datasets (16 kHz sample rate) - RE	crepe	crepePreprocess crepePostprocess	pitchnm	Pitch estimation
iVectors (System of Machine Learning models)	LibriSpeech data set (approx. 1000-hour English speech sampled, 16 MHz sample rate)	iVectorSystem	trainExtractor, trainClassifier, enroll, verify	speakerRecognition	Speaker verification ++

2. Define your own from scratch

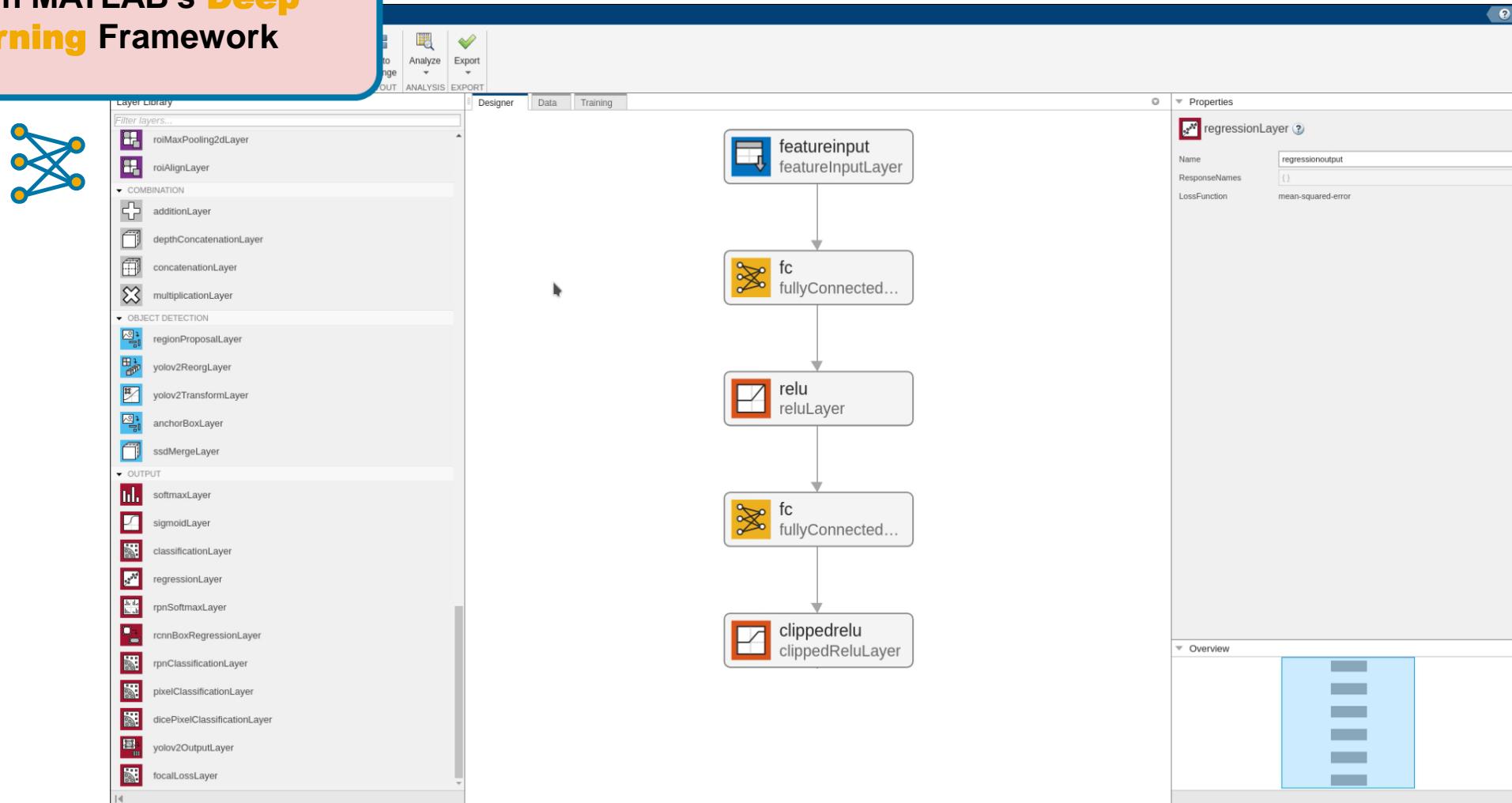


3. Import one developed using another framework



AI Modeling

Train in MATLAB's Deep Learning Framework



Data Preparation

AI Modeling

Simulation & Test

Deployment

MATLAB Demo

Ch1_ECG_Classification_CNN mlx

MATLAB Demo

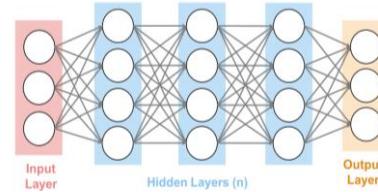
Ch2_ECG_Classification_LSTM_Wavelets mlx

Neural Network Architectures

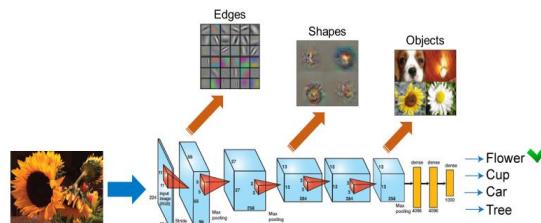
[MATLAB Deep Learning Models on Github](#)

Standard Framework (App workflow)

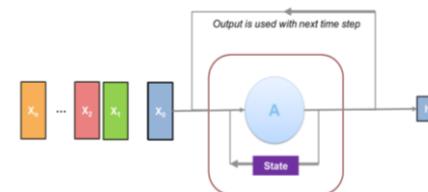
MLP (Multi-layer Perceptrons)



CNN (Convolutional NN)

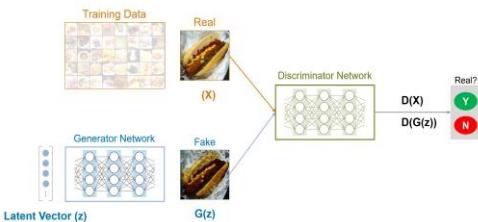


LSTM (Recurrent NN)

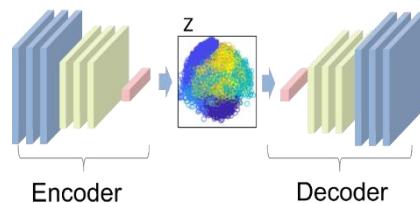


Advanced Framework (programmatic workflow)

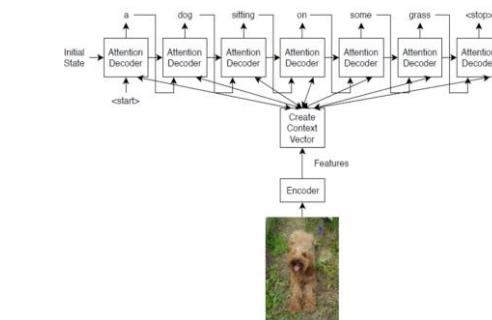
GAN (Generative Adversarial Networks)



VAE (Variational Autoencoder)

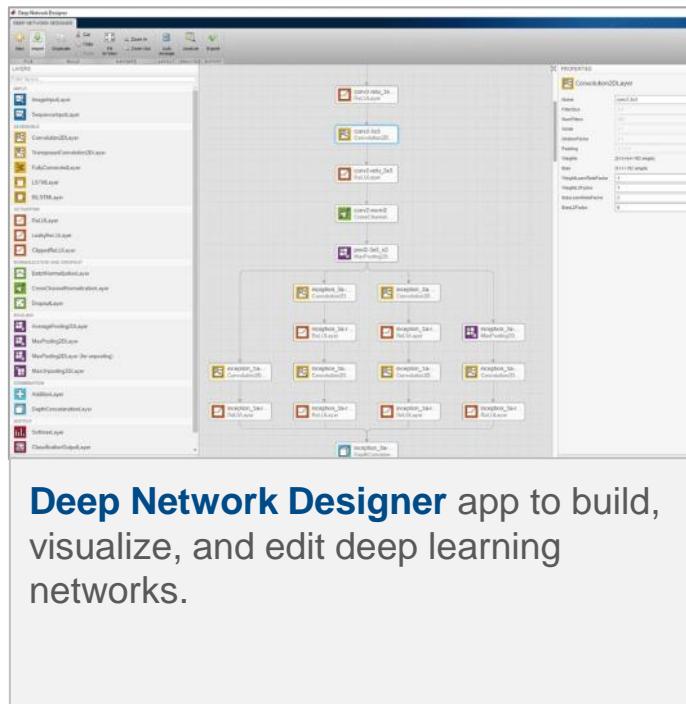


Attention Mechanisms

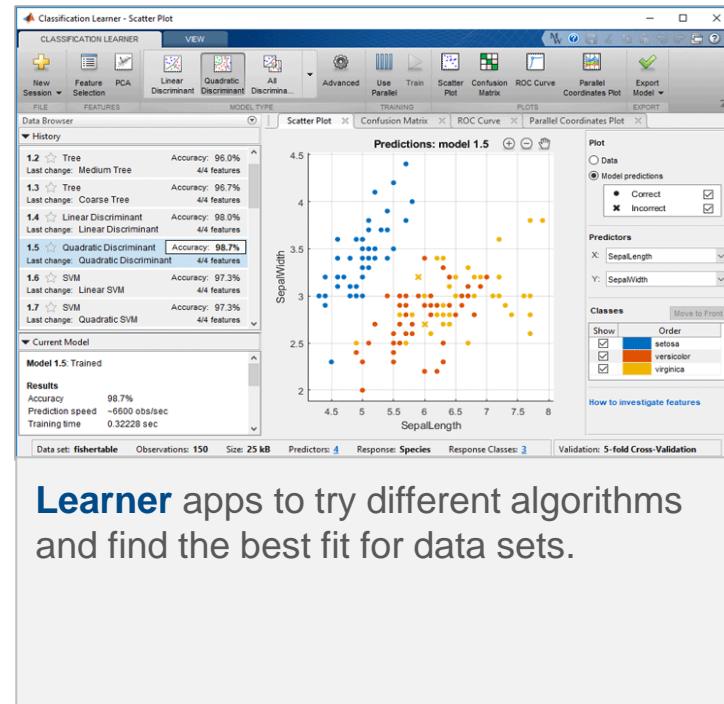


Increase productivity using apps for design and analysis

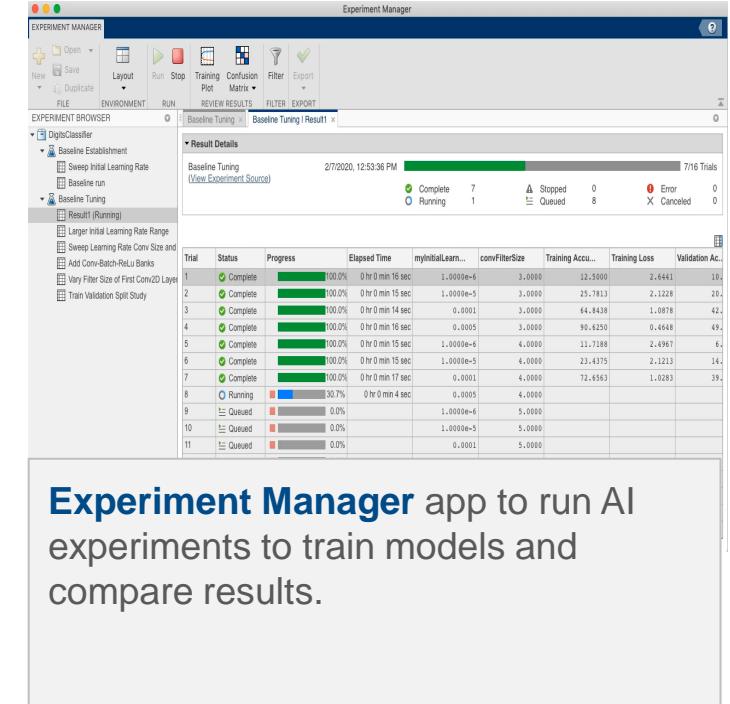
Use MATLAB Apps to design deep learning networks, explore a wide range of classifiers, train regression models, train an optical character recognition model, and more.



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



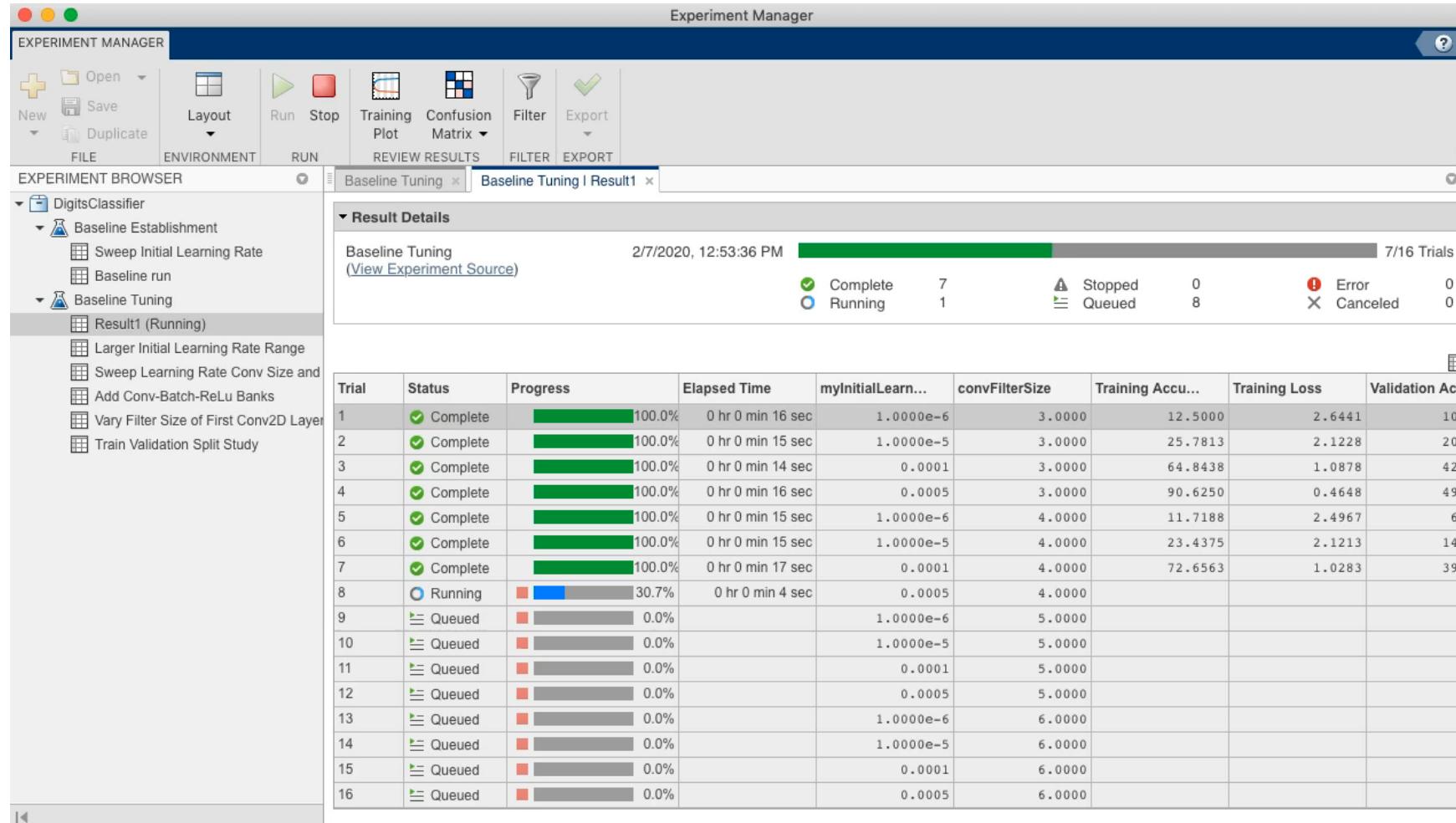
Learner apps to try different algorithms and find the best fit for data sets.



Experiment Manager app to run AI experiments to train models and compare results.

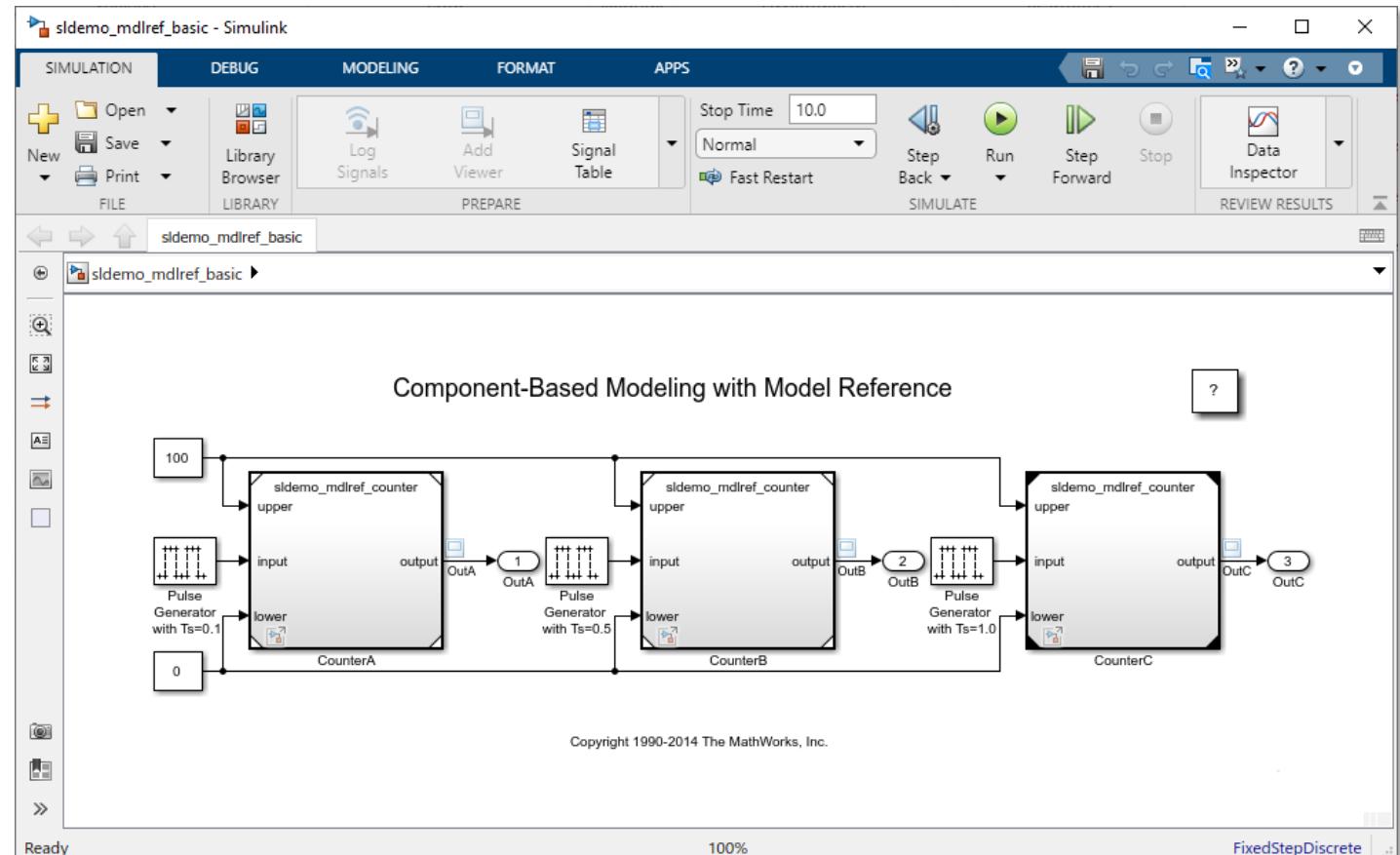
Find the Optimal Model Using Experiments

- Sweep through a range of hyperparameter values
- Test different deep network architectures and machine learning models



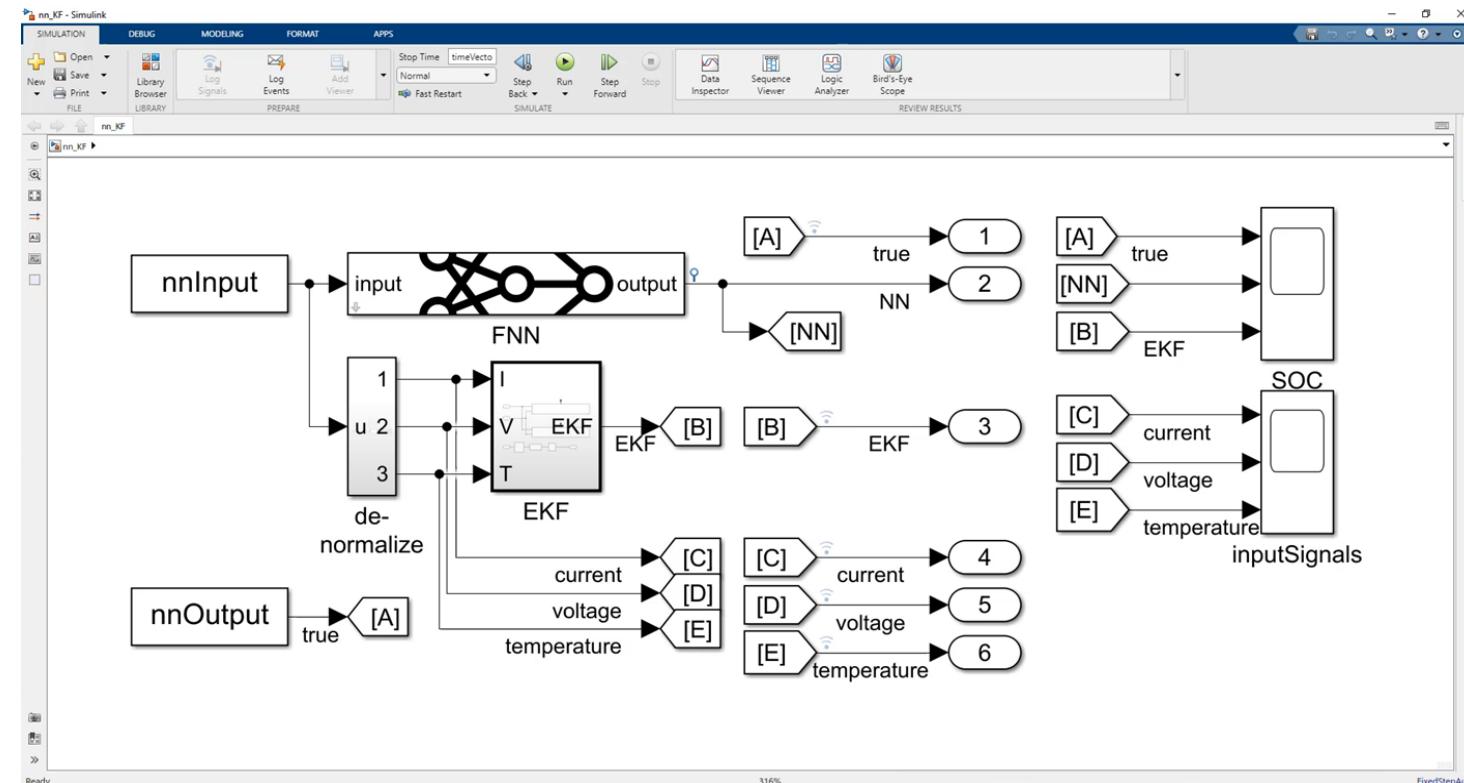
Intro to Simulink

- Design models using drag-and-drop blocks.
- Run simulations to test and validate models.
- Access to a vast library of blocks for various domains
- Generate C/C++ code for embedded systems



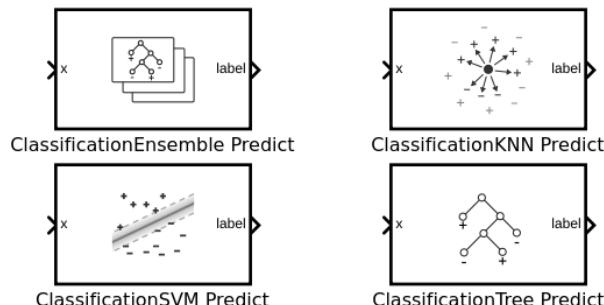
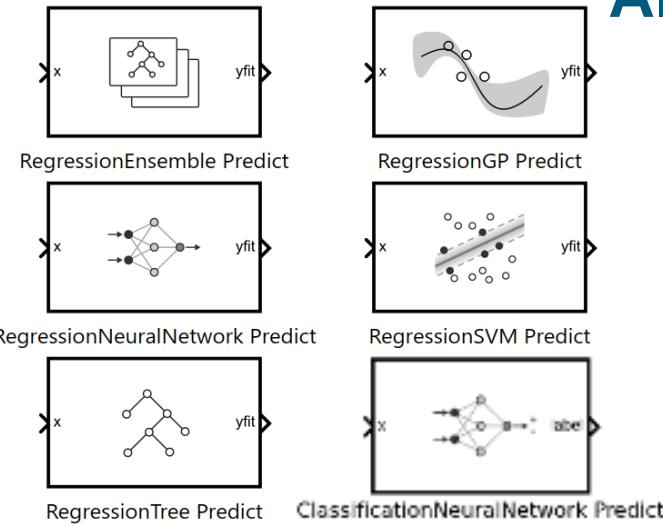
Integrate your AI model for system-level simulation and test

Integration of trained AI model into Simulink

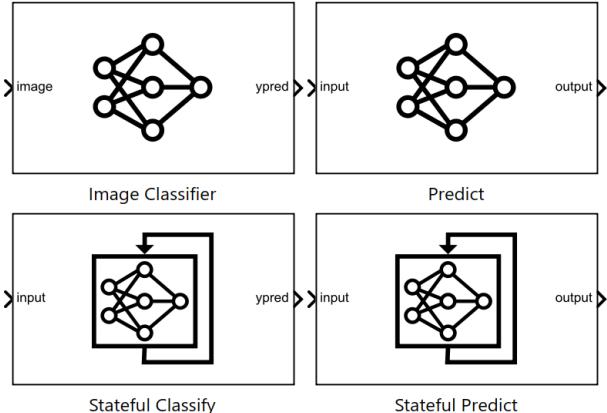


Simulink include more AI blocks for more applications

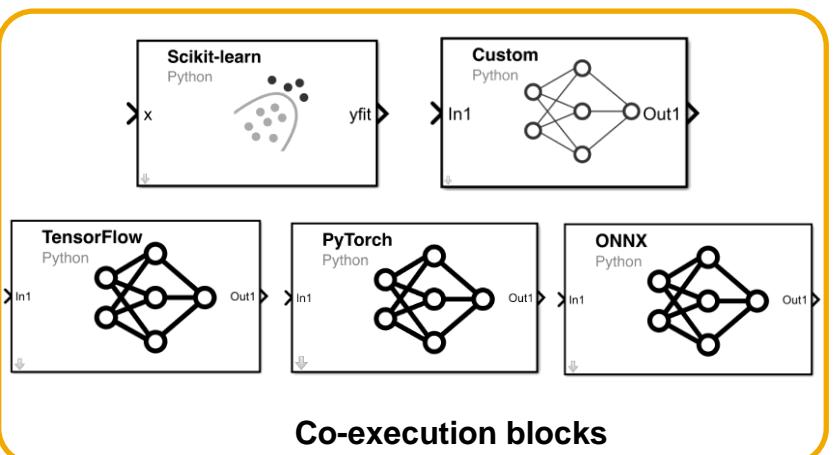
AI core



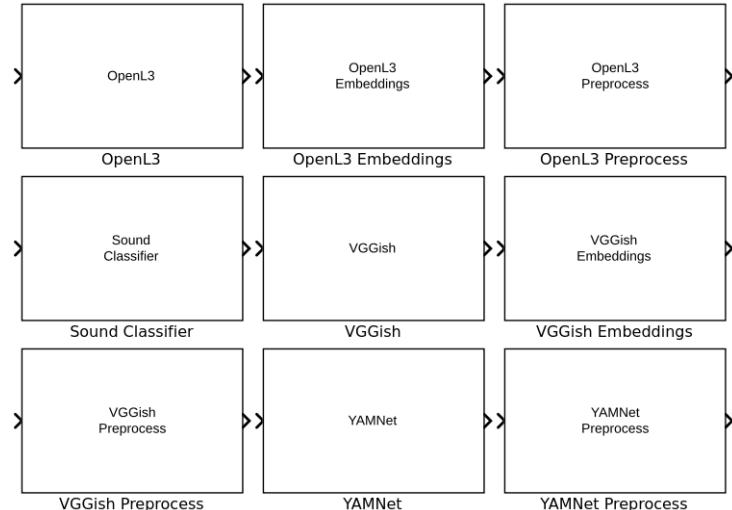
**Regression and classification blocks
(Statistics and Machine Learning Toolbox)**



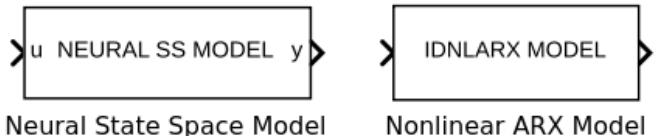
**Neural network classify and predict blocks
And the list of layer blocks
(Deep Learning Toolbox)**



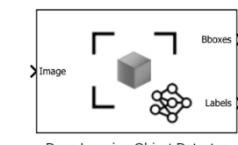
Specialized



Audio Toolbox



System Identification Toolbox



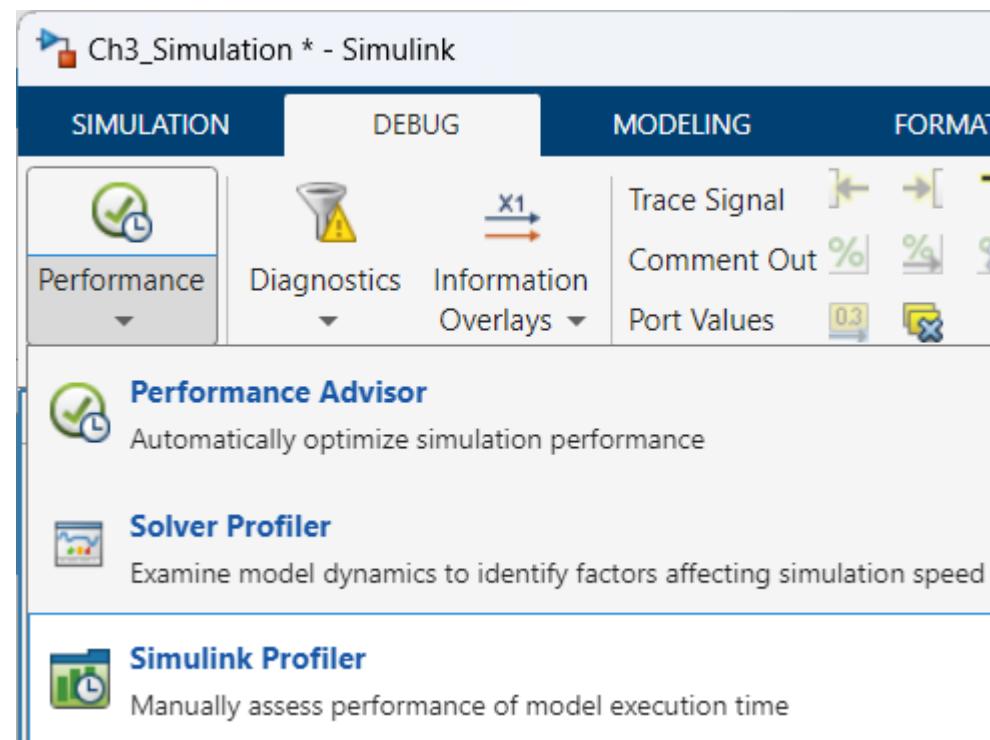
Computer Vision Toolbox

MATLAB Demo

[Ch3_FunctionalVerification_Simulation mlx](#)

Incorporate and simulate trained models in Simulink

Profiling in Simulink



MATLAB Exercise

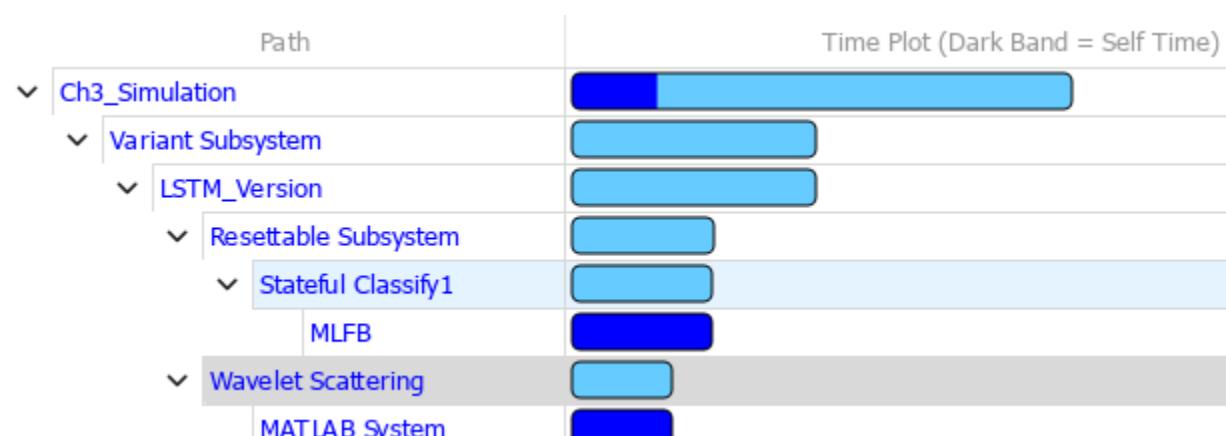
Ch3_Simulation.slx

Profile performance and compare models

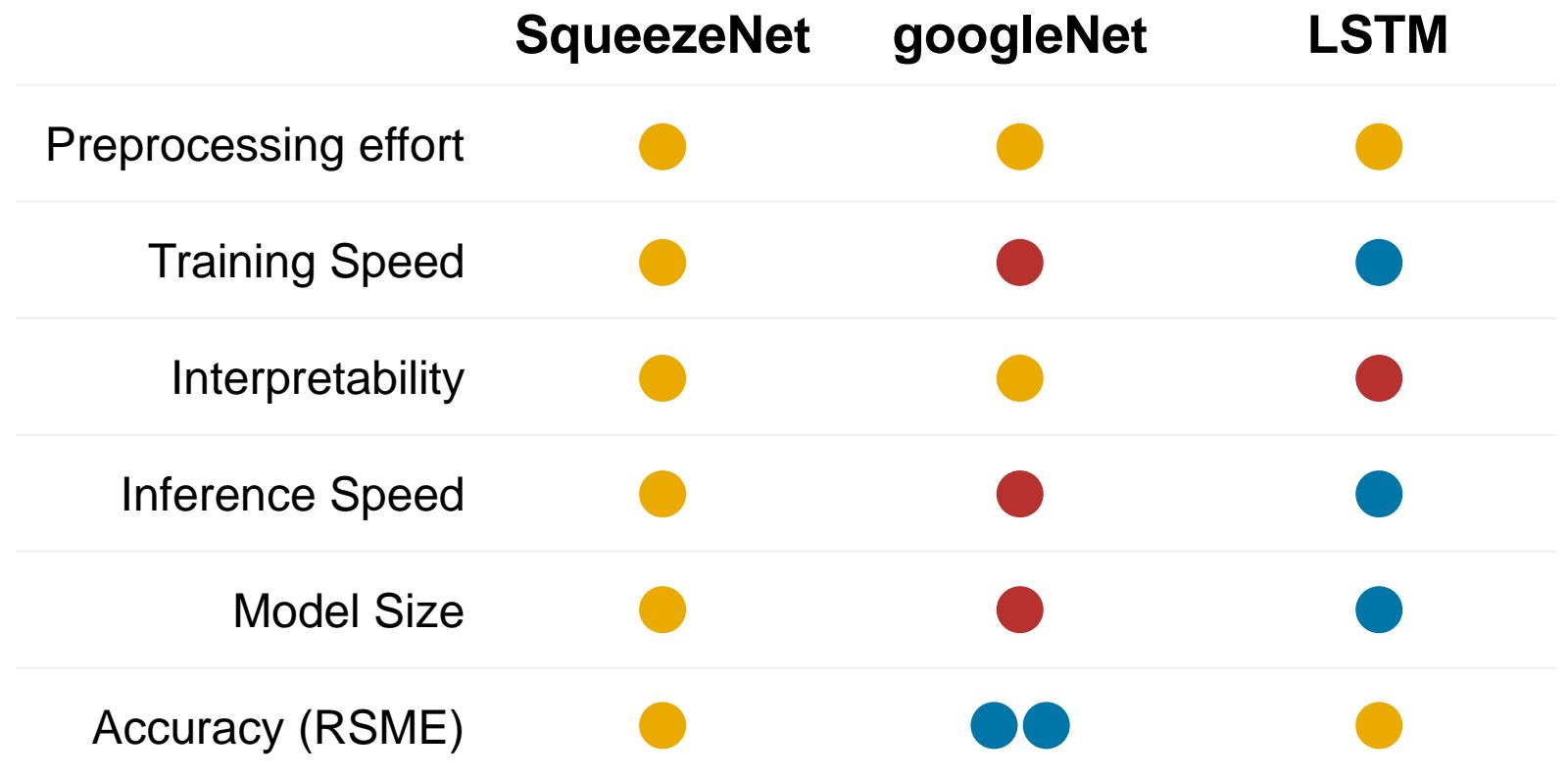
MATLAB Exercise

Ch3_Simulation.slx

Profile performance and compare models



Manage AI tradeoffs for your system



Results are specific to this ECG classification example

Much Better ● ● Better ● Okay ● Worse ●

Model compression can bridge the gap between AI modelling and embedded deployment

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

AI Modeling
 Model design and tuning
 Hardware accelerated training
 Interoperability

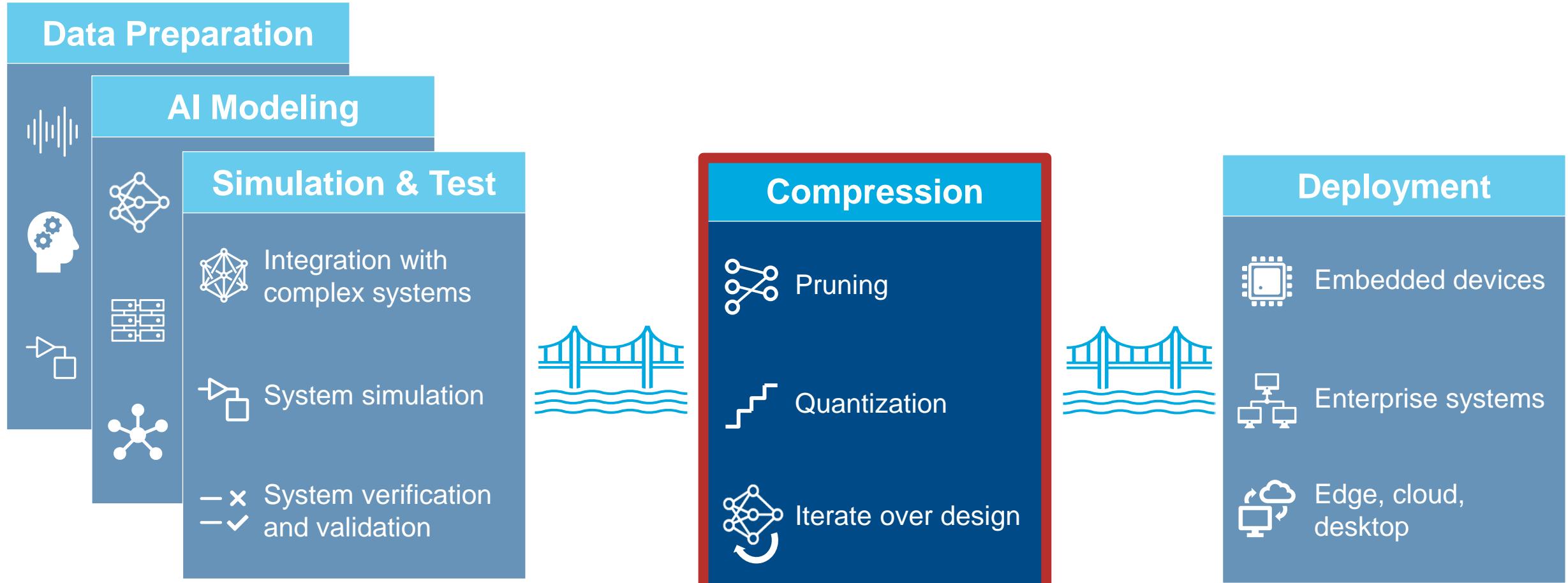
Simulation & Test
 Integration with complex systems
 System simulation
 System verification ✓ and validation

Deployment
 Embedded devices
 Enterprise systems
 Edge, cloud, desktop

Embedded Devices Have Limited Resources

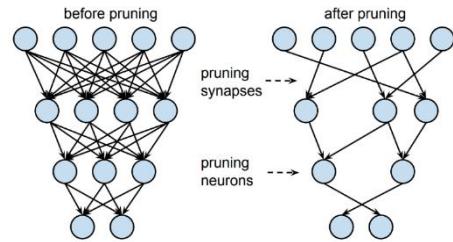
Memory	Performance	Power
32GB	TFLOPs	
8GB	GFLOPs	
1kB~1MB	MFLOPs	

Model compression can bridge the gap between AI modelling and embedded deployment

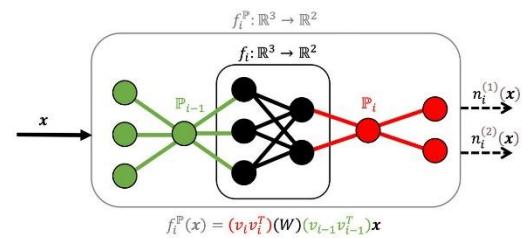


Model Compression

Structural Compression

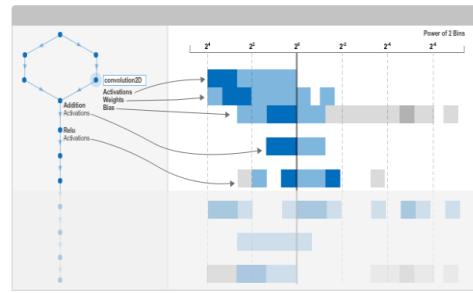


Pruning
convolutional neural
networks



Projection of deep
neural networks

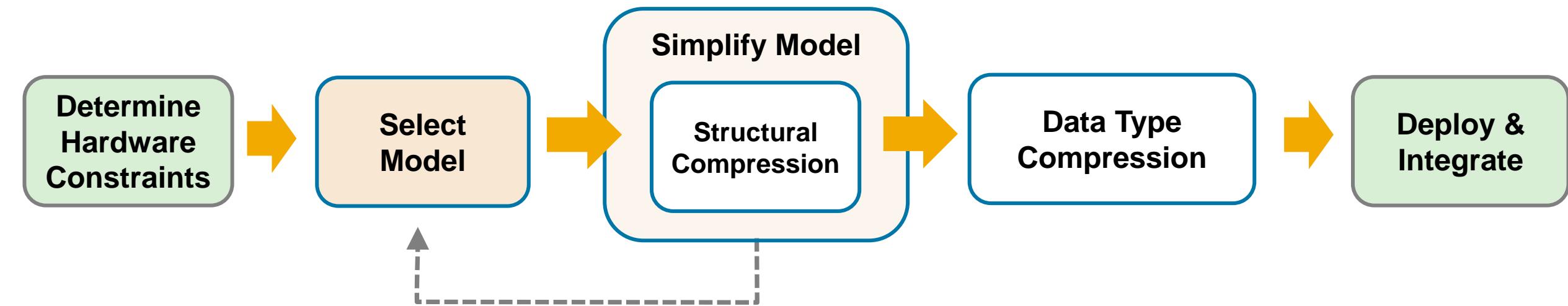
Datatype Compression



Quantization of network
weights to lower precision
datatypes (bfloat16, int8)

Quantization, Projection, and Pruning

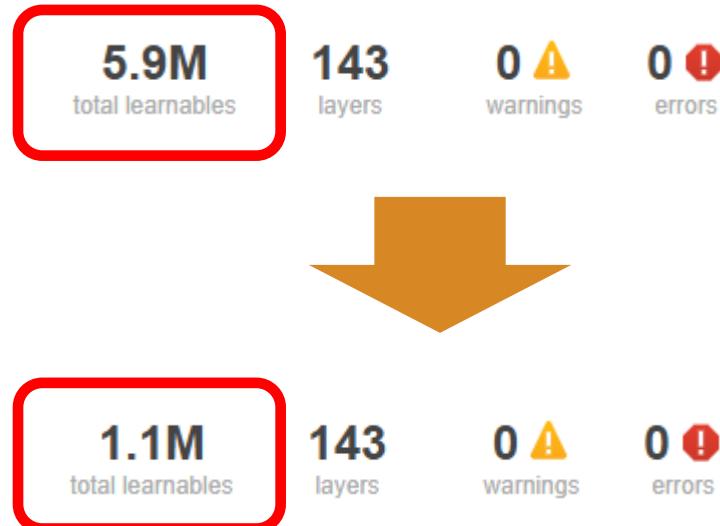
Iterative Workflow for Model Compression



MATLAB Exercise

Ch3_Compression_LSTM mlx

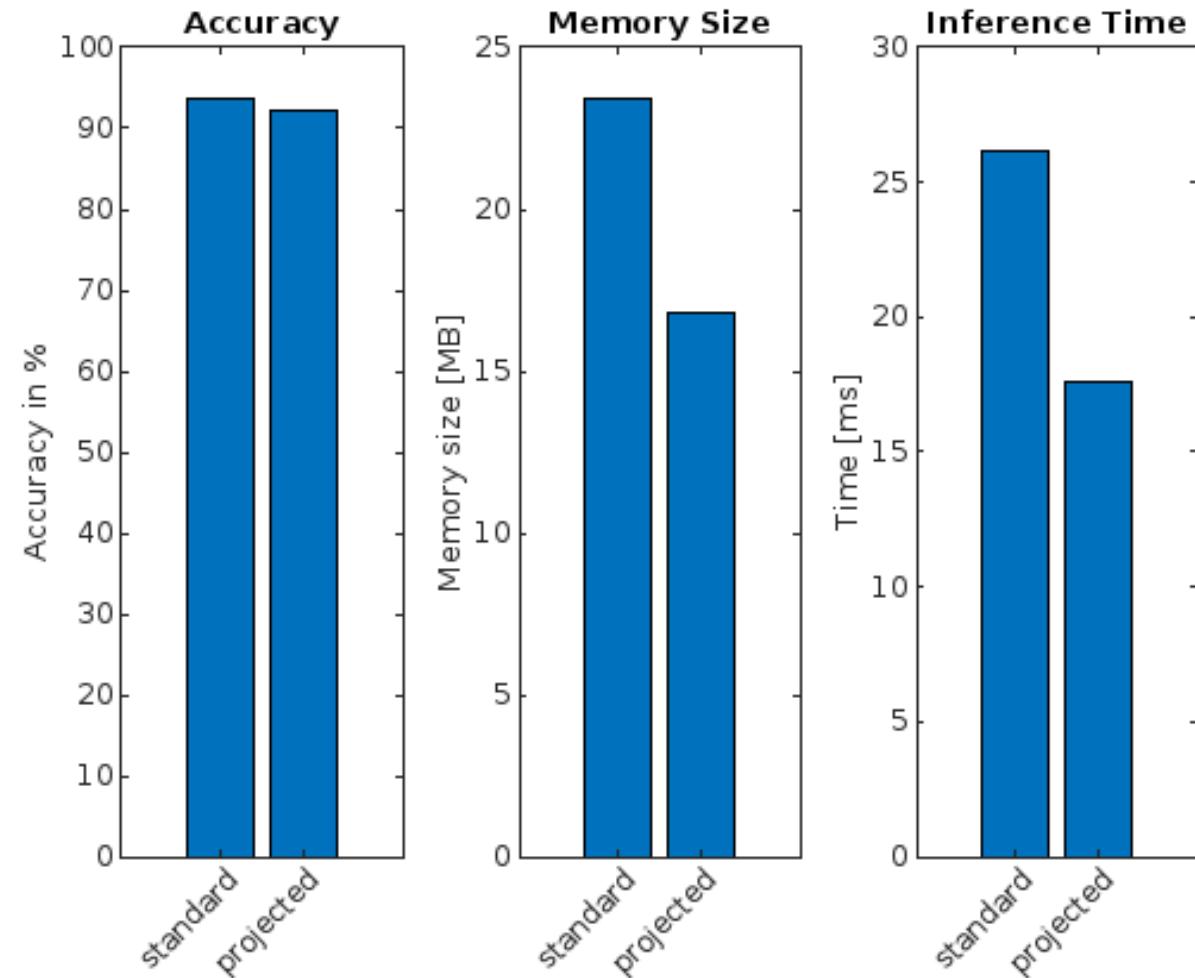
Results: Model Compression with Projection for the CNN



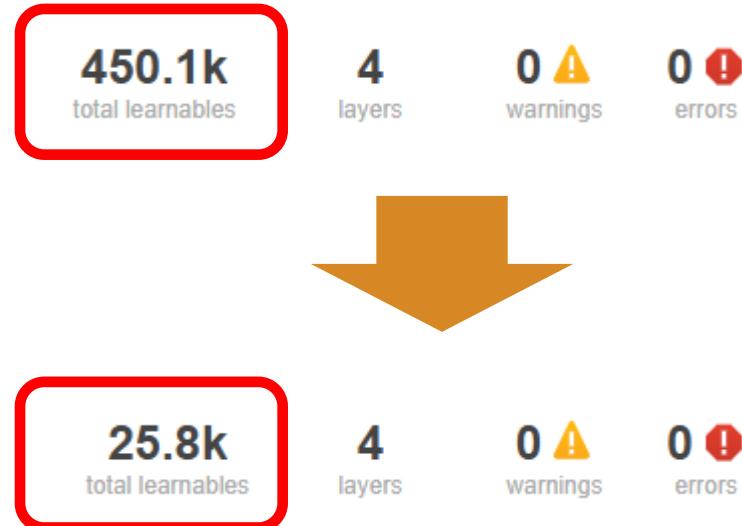
Number of total learnables decreased

Test setup

- Run on laptop CPU
- Averaged over 1000 predictions



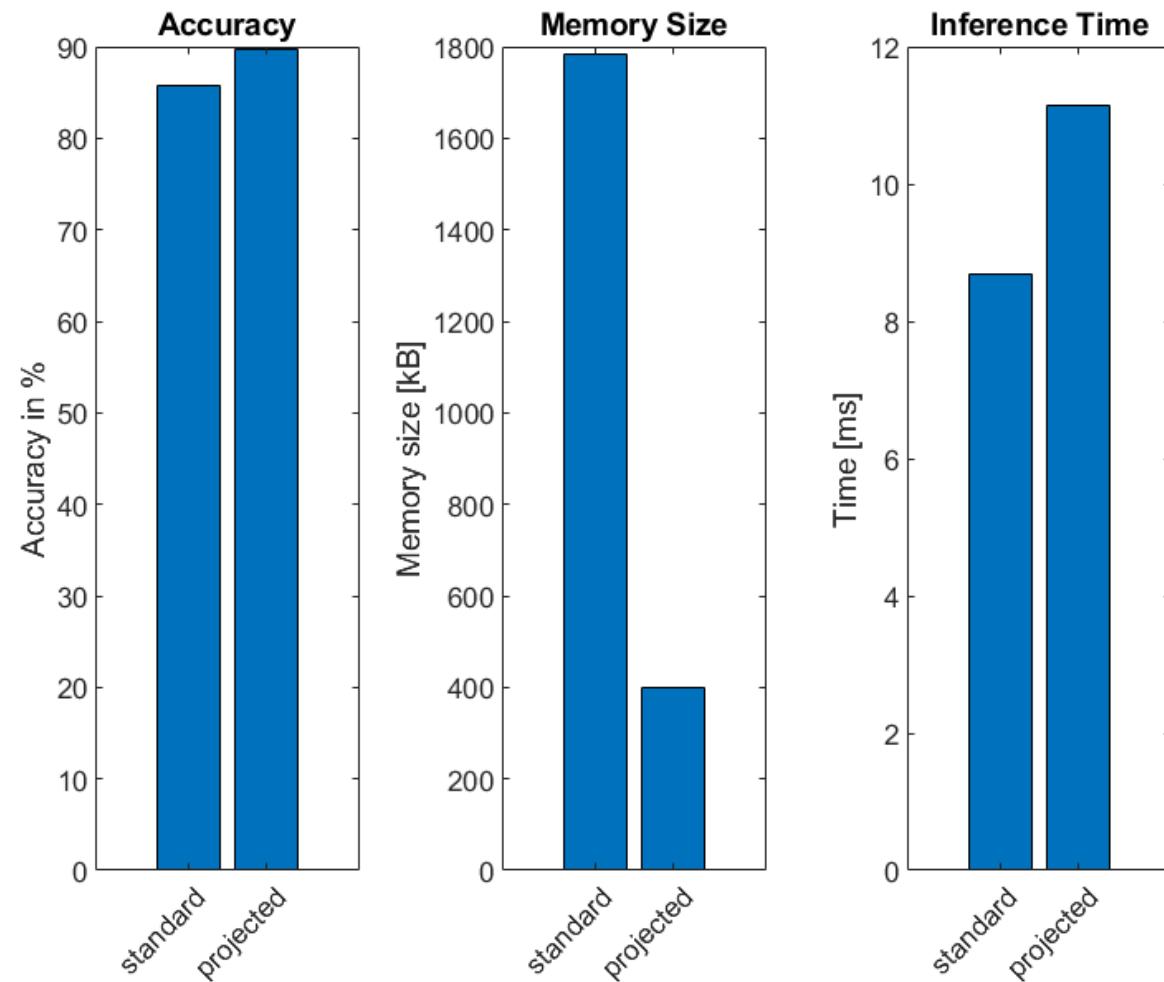
Results: Model Compression with Projection for the LSTM



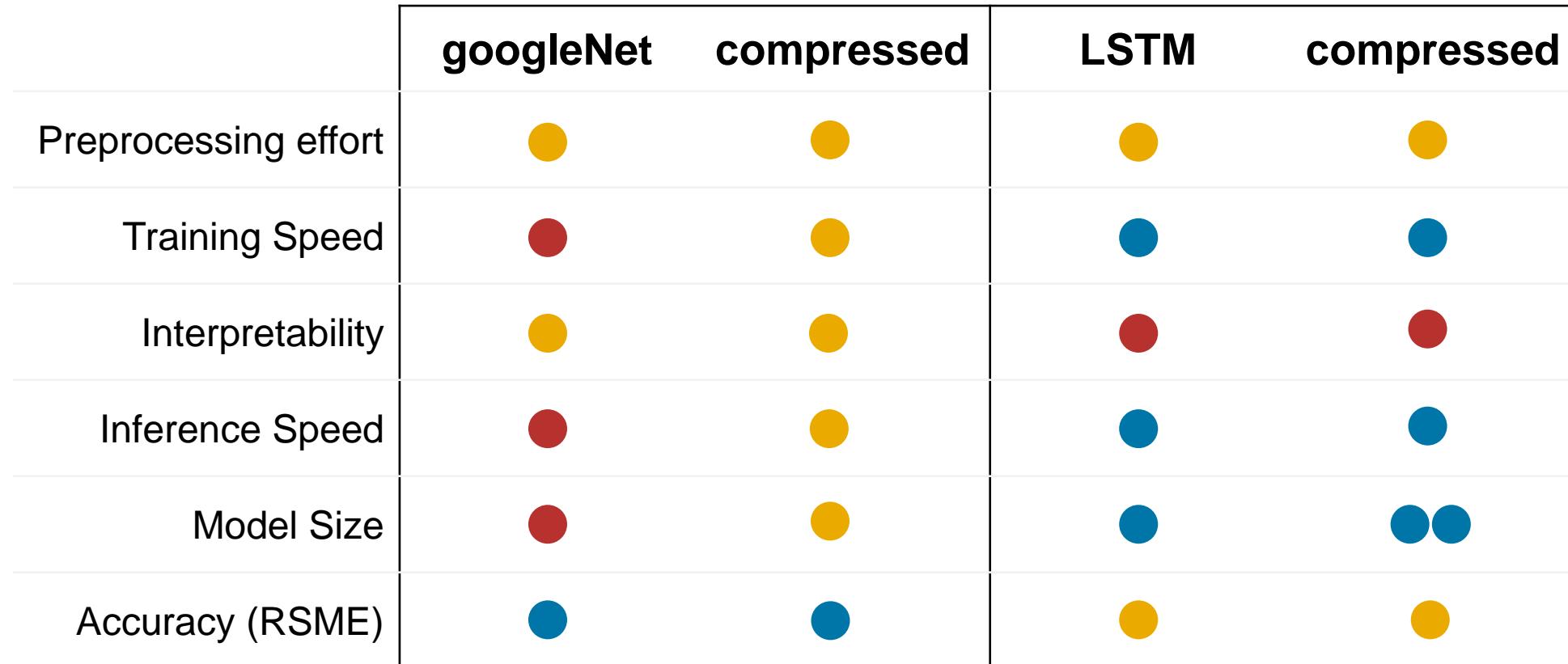
Number of total learnables decreased

Test setup

- 1000 predictions
- Averaged over 100 times



Manage AI tradeoffs for your system



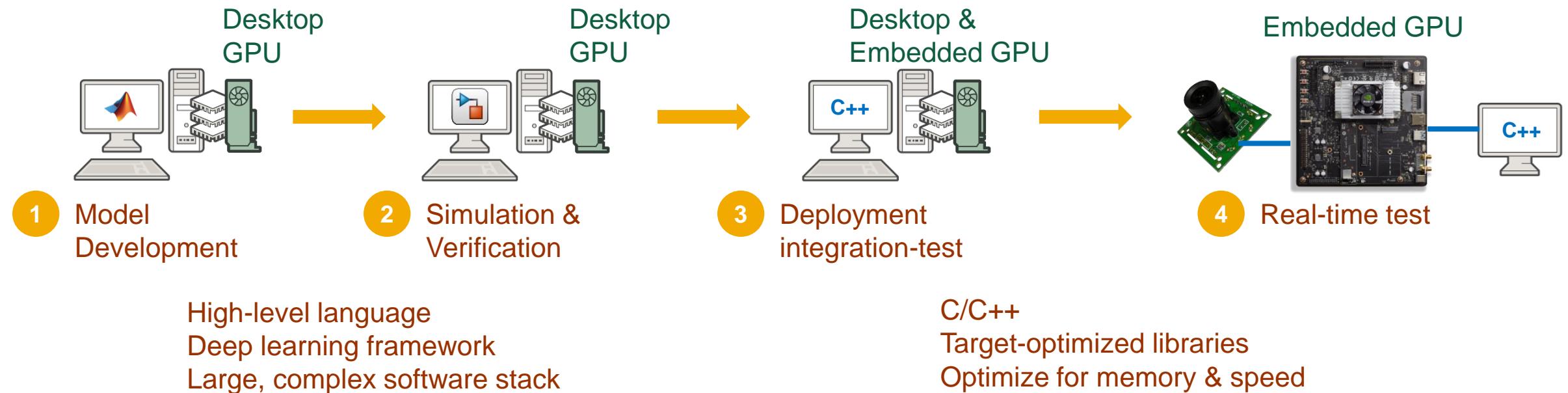
Results are specific to this ECG classification example

Much Better Better Okay Worse

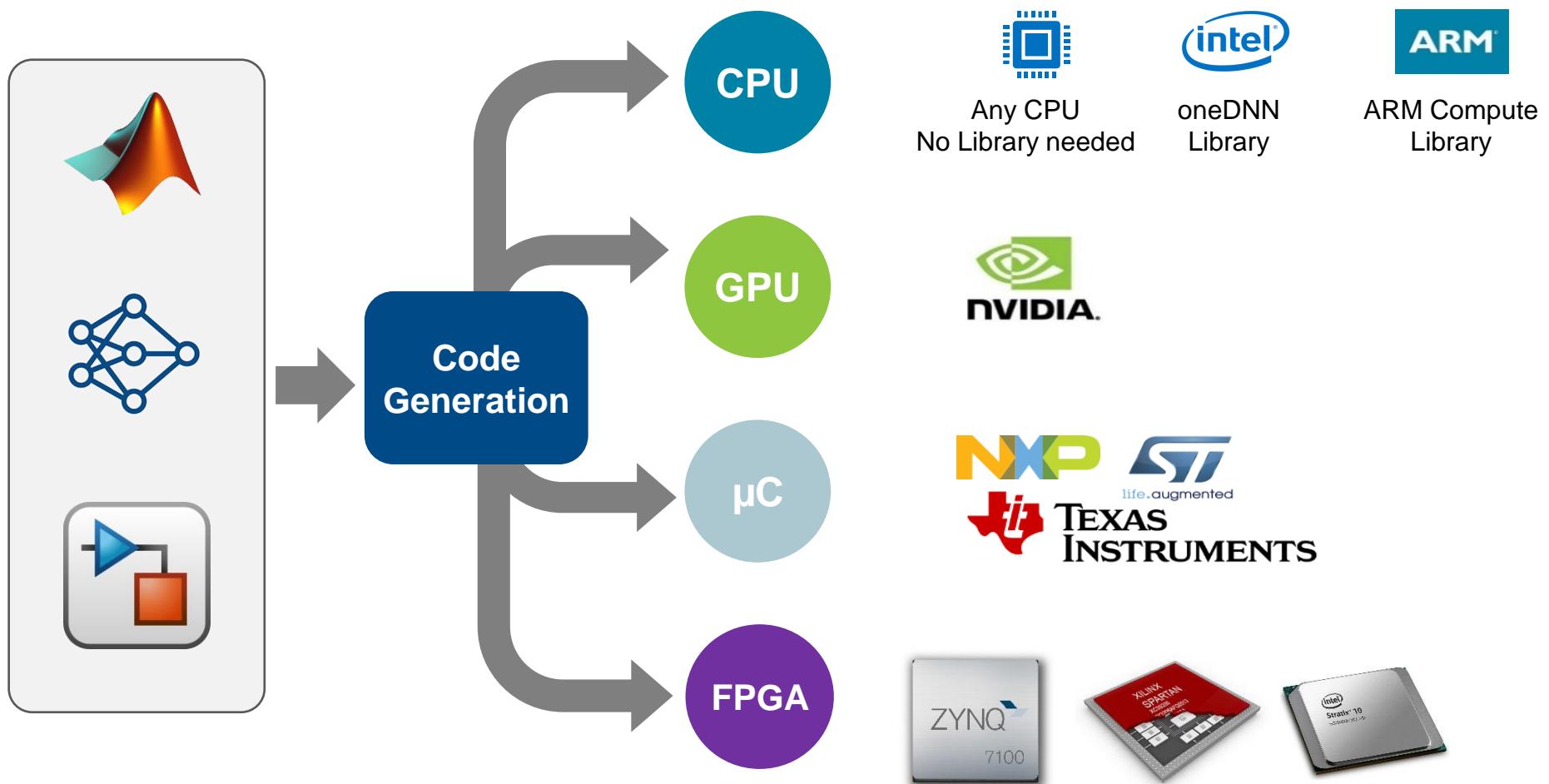
MATLAB & Simulink support hardware workflows

Data Preparation	AI Modeling	Simulation & Test	Deployment
 Data cleansing and preparation  Human insight  Simulation-generated data	 Model design and tuning  Hardware accelerated training  Interoperability	 Integration with complex systems  System simulation  System verification and validation	 Embedded devices  Enterprise systems  Edge, cloud, desktop

Algorithm Design to Embedded Deployment Workflow



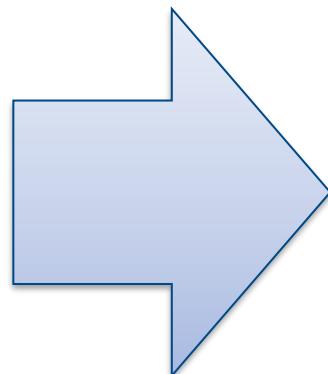
Deploy to target with zero coding errors



Why Use CUDA code generation ?

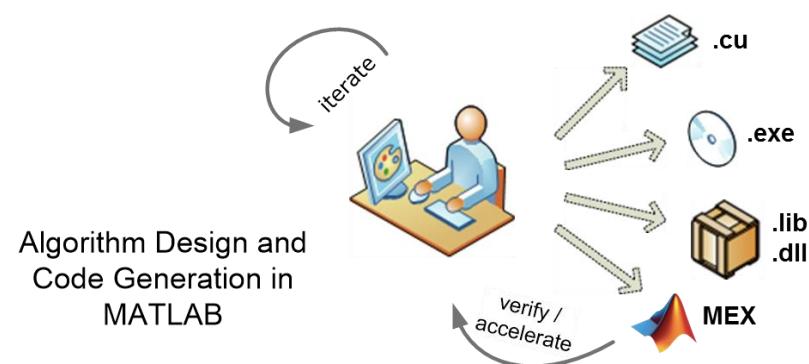
Pains: Hand code

- **Cannot code in CUDA**
- Time consuming
- Manual Coding Errors
- Multiple implementations
- Expensive

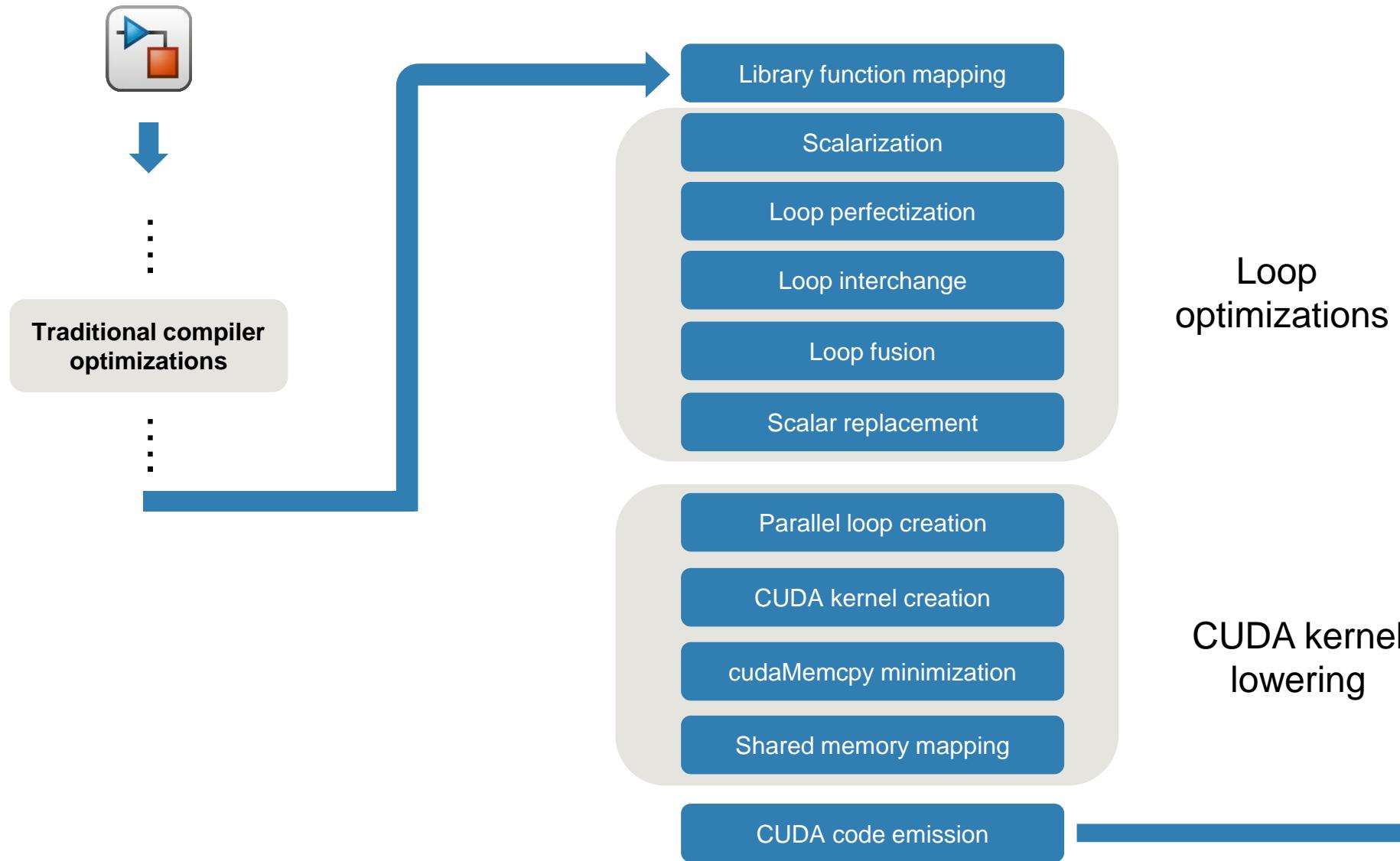


Solution: GPU Coder

- Automatically convert to CUDA
- Get to optimized CUDA faster
- Eliminate manual coding errors
- Maintain Single “Truth”
- Stay within MATLAB/Simulink at a higher level

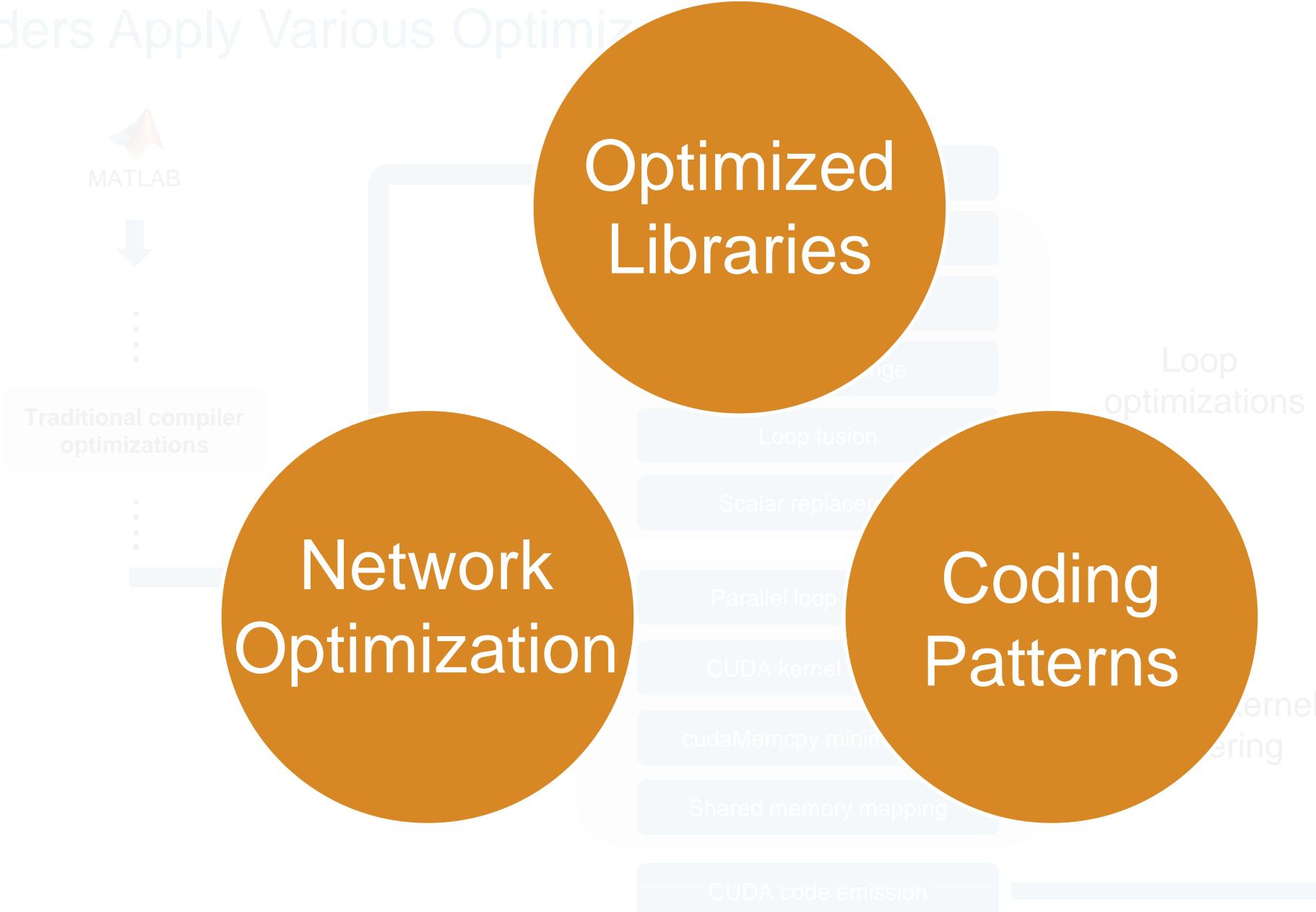


GPU Coder Apply Various Optimizations



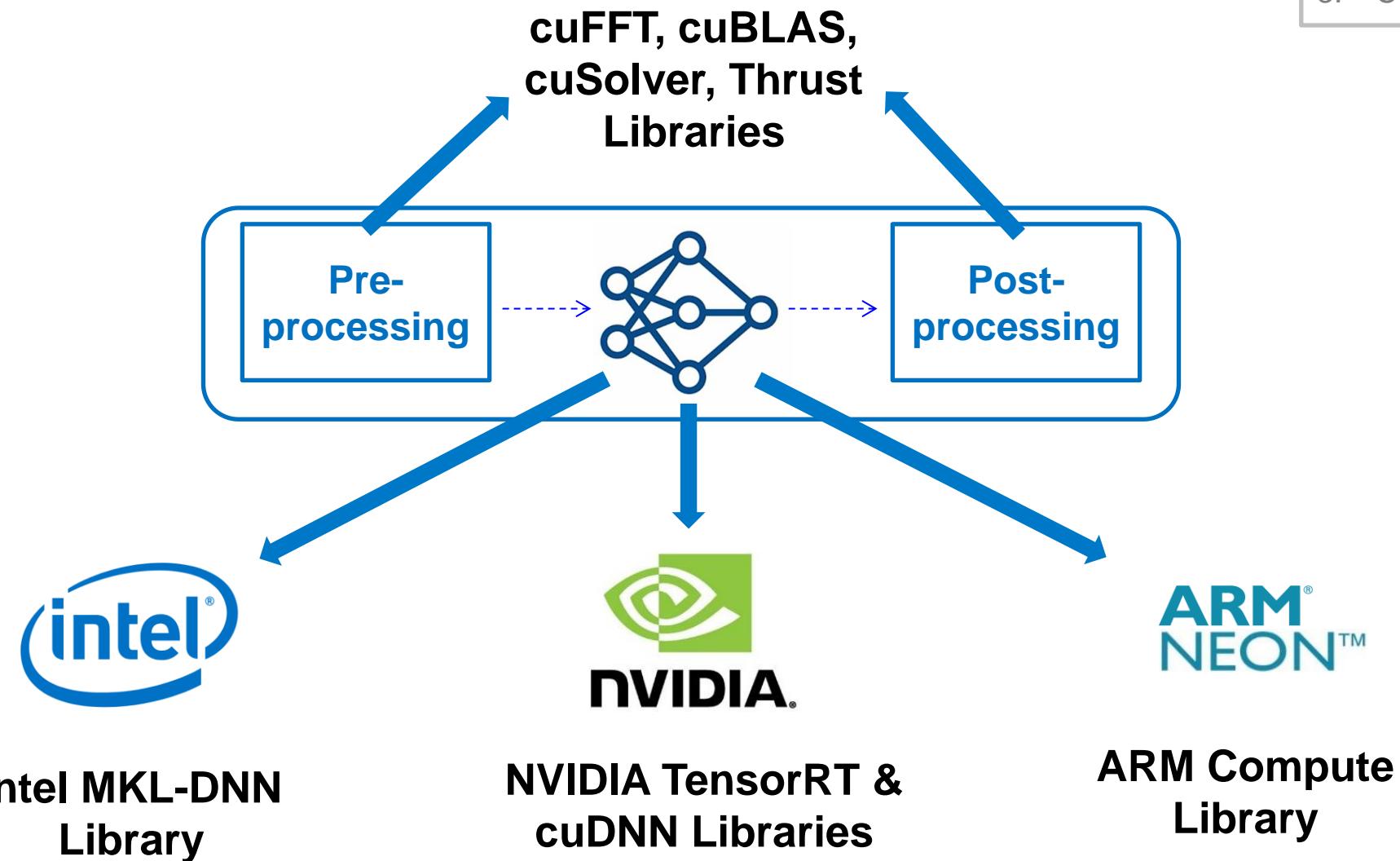
NVIDIA.
CUDA®
C/C++

Coders Apply Various Optimizations



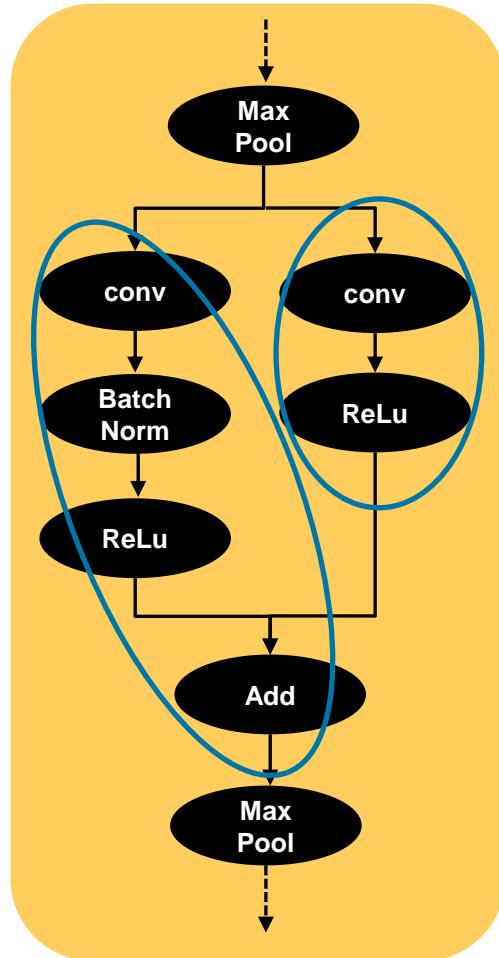
Generated Code Calls Optimized Libraries

- Performance
1. Optimized Libraries
 2. Network Optimizations
 3. Coding Patterns

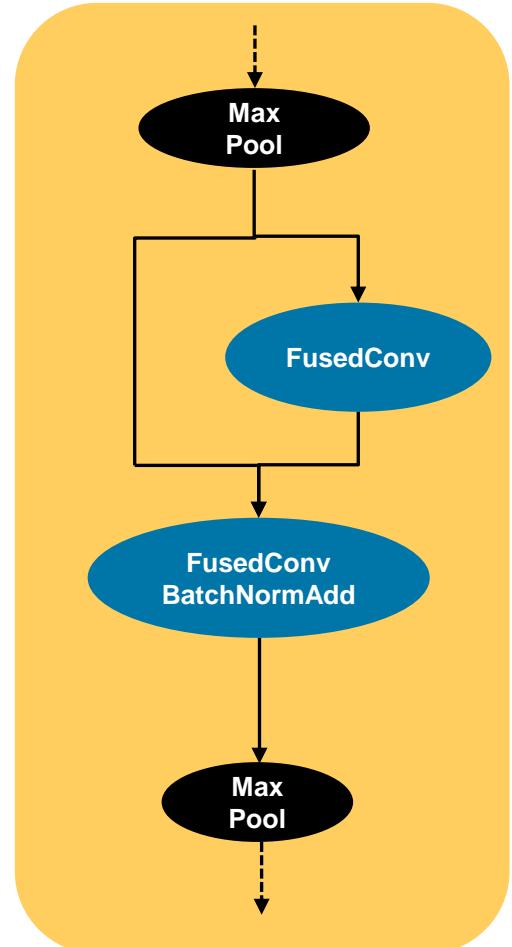


Deep Learning Network Optimization

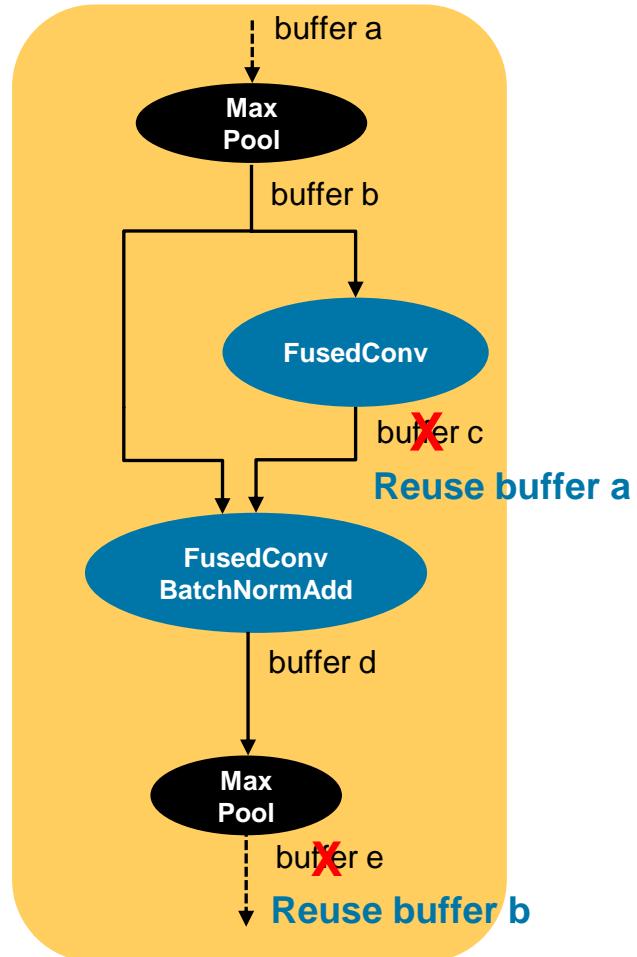
- Performance
1. Optimized Libraries
 - 2. Network Optimizations**
 3. Coding Patterns



Network



Layer fusion
Optimized computation

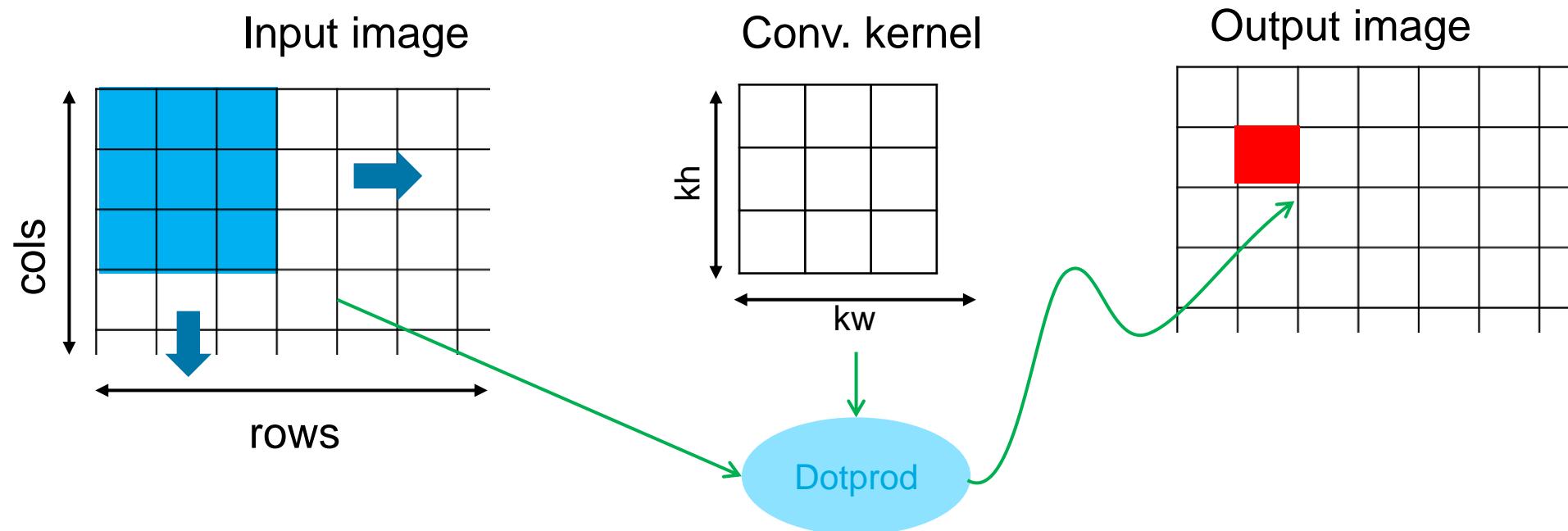


Buffer minimization
Optimized memory

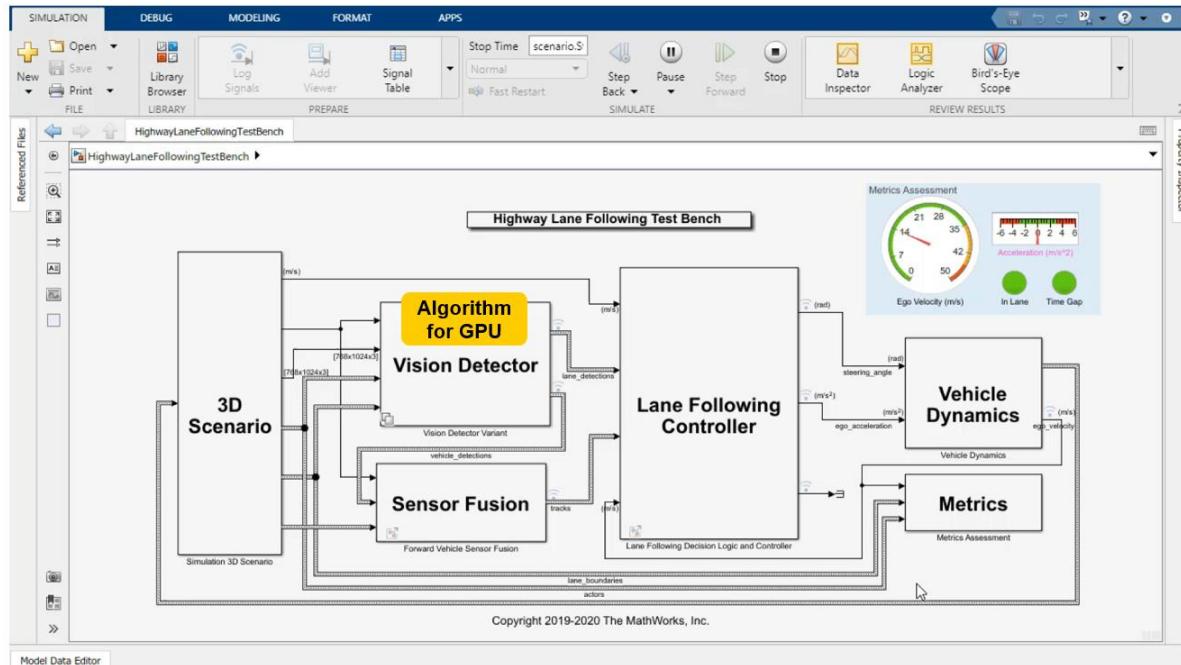
Coding Patterns: Stencil Kernels

- Performance
1. Optimized Libraries
 2. Network Optimizations
 - 3. Coding Patterns**

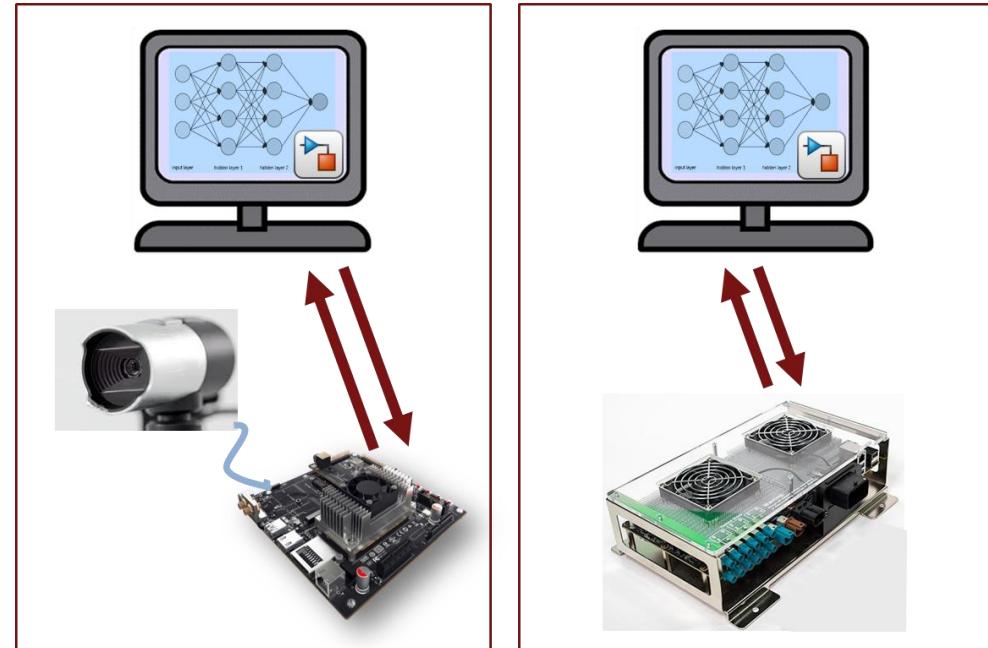
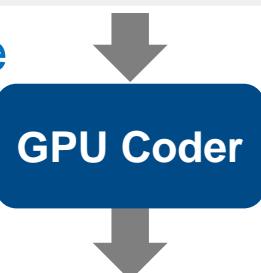
- Automatically applied for image processing functions (e.g. `imfilter`, `imerode`, `imdilate`, `conv2`, ...)
- Manually apply using `gpuCoder.stencilKernel()`



GPU Coder for Simulink Workflows



Deploy Standalone Application



Access Peripheral
from Simulink

Software-in-Loop,
Processor-in-Loop,
External Mode

Demo: Test algorithm on the Jetson

The screenshot illustrates the workflow for deploying a machine learning algorithm from MATLAB/Simulink to an NVIDIA Jetson target.

Simulink Model:

- Continuous Domain:** A subsystem block containing "Sensors" and "Sensors Analog" blocks. It receives a signal from a "SelectedSignal" source (labeled D1) and outputs "MeasuredSignal".
- Switch:** A switch block that routes the signal through a "SignalBuffer" block or directly to the "DigitalEcgSignal" output.
- PIL Block:** A block labeled "PIL" which processes the signal. It includes a "DigitalEcgSignal" block and a "final_image" block. The input to the PIL block is "D2 [1024x1]" and the output is "D2 [227x227x3]".
- ECG Output:** A subsystem block that takes the "final_image" output from the PIL block and displays it in a "Video Viewer".
- Diagnostic Viewer:** Shows the MATLAB command window output during deployment. Key logs include:
 - Connectivity configuration for "C:\MATLAB\Content\Di1_Artificial_Intelligence\M3_Workshops\AdaptedMaterial\AMLD_2025\Ch4_Deployment\Subsystem_ert_rtw": [NVIDIA_Jetson](#) ##
 - Preparing to start PIL block simulation: [CodeGen/Subsystem](#) ...
 - Skipping makefile generation and compilation because C:\MATLAB\Content\Di1_Artificial_Intelligence\M3_Workshops\AdaptedMaterial\AMLD_2025\Ch4_Deployment\Subsystem_ert_rtw\pil\Subsystem.elf is up to date
 - Starting application: 'Subsystem_ert_rtw\pil\Subsystem.elf'
 - Launching application Subsystem.elf...
 - Runtime log on Target:
 - [sudo] password for bern:
 - PIL execution terminated on target.

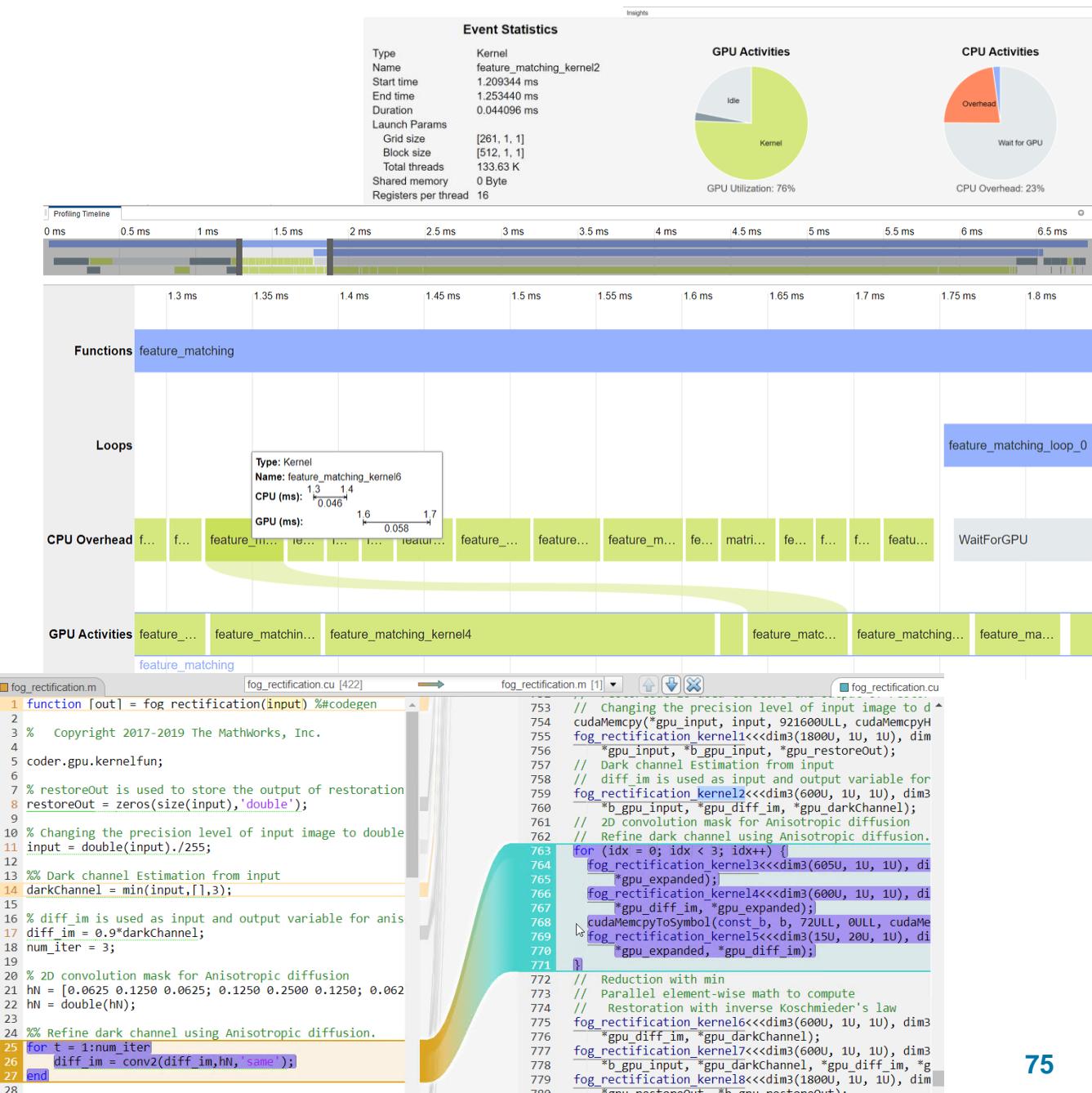
NVIDIA Jetson Hardware: An NVIDIA Jetson Dev Board is shown, featuring a central processor, memory, and various connectors. A black oval highlights the board, indicating where the deployed code is running.

ECG Output Window: Displays a 2D heatmap visualization of an ECG signal. The top status bar shows "Class:Rhythm (CNN): CHF" and "Confidence: 99.33%". The bottom status bar shows "Magnification: 100%" and "RGB: 227x227 T=511.000".

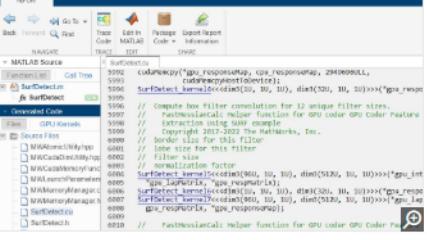
How to Optimize Further?

Profiling and Bidirectional Traceability Tools

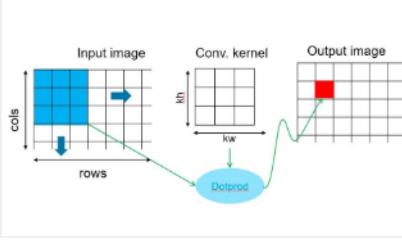
- Use the GPU Coder Performance Analyzer to profile the generated CUDA code
 - Identify bottlenecks & opportunities to optimize performance
- Use bidirectional traceability to map to/from CUDA code back to MATLAB code
 - Helps you to understand how CUDA kernels are created from your MATLAB algorithms



Outlook


Generate CUDA Code from MATLAB

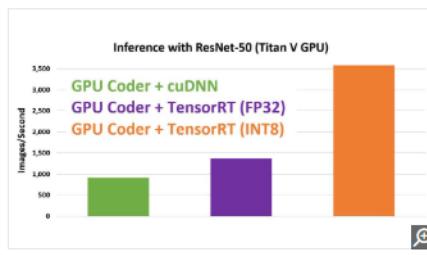
Compile and run CUDA code generated from MATLAB algorithms on popular NVIDIA GPU desktop RTX cards to data centers to embed Jetson and DRIVE platforms. Deploy the generated code royalty-free to your customers at no cost.


Use Design Patterns to Boost Performance

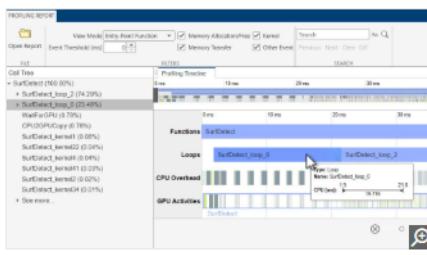
Design patterns, including stencil processing, reductions, are applied automatically when generating code to increase the performance of generated code. You can also manually invoke them using specific pragmas.


Generate Code for Deep Learning

Deploy a variety of predefined or customized deep learning networks to NVIDIA GPUs. Generate code for preprocessing and postprocessing along with your trained deep learning networks to deploy complete algorithms.


Optimize Generated Code

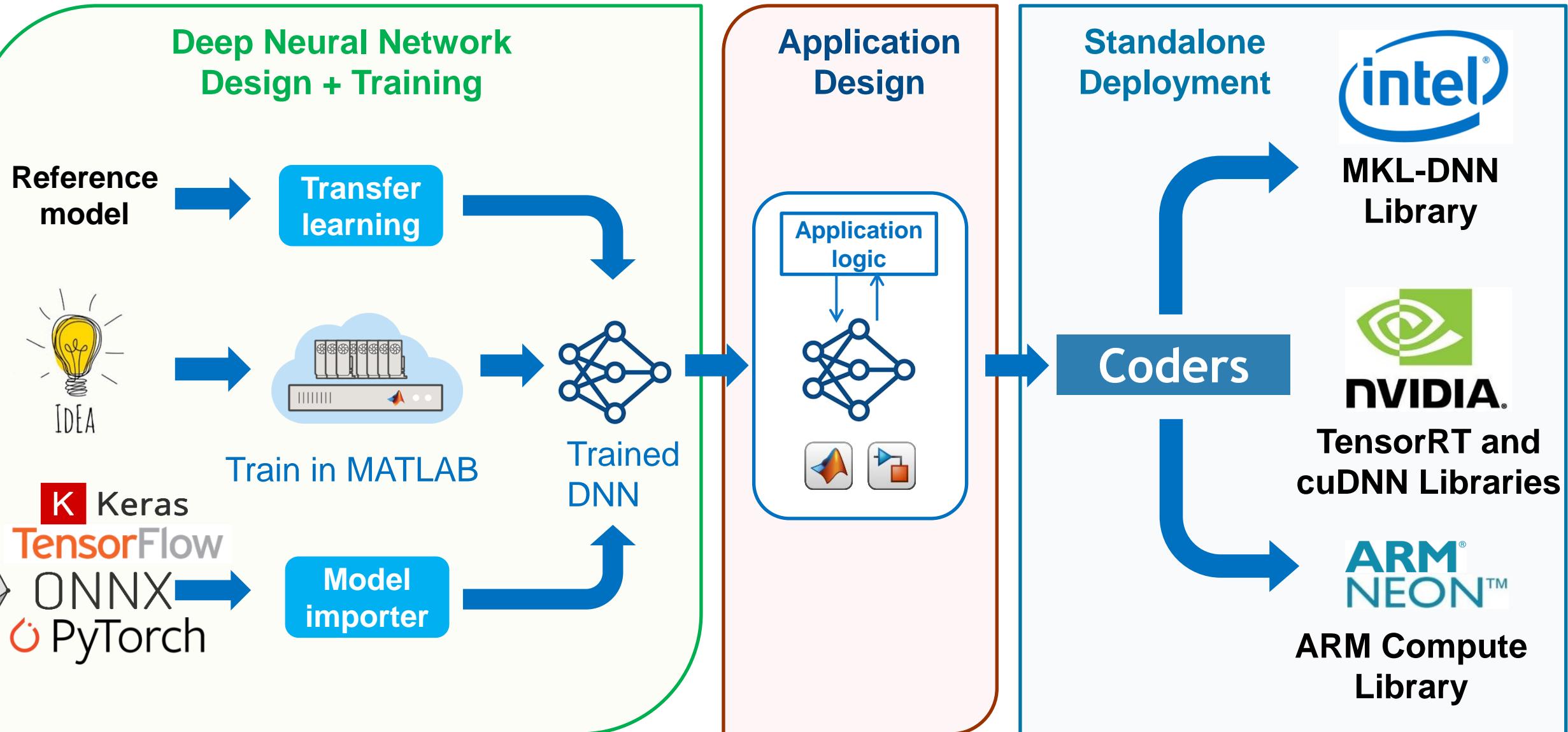
GPU Coder automatically applies optimizations including memory management, kernel fusion, and auto-tuning. Reduce memory footprint by generating INT8 or bfloat16 code. Further boost performance by integrating with TensorRT.


Profile and Analyze Generated Code

Use the GPU Coder Performance Analyzer to profile generated CUDA code and identify opportunities to further improve execution speed and memory footprint.

<https://www.mathworks.com/products/gpu-coder.html>

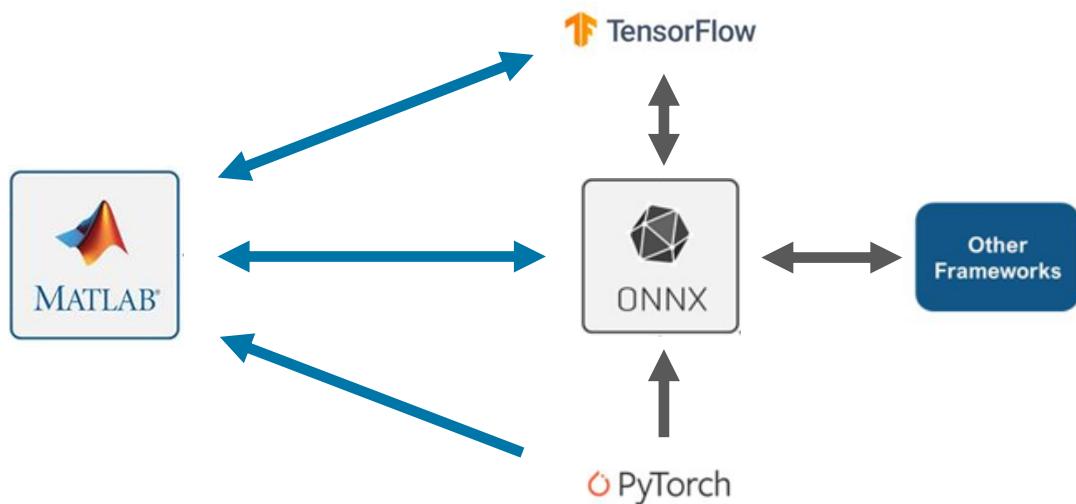
Deep Learning Workflow in MATLAB and Simulink



Access pretrained models

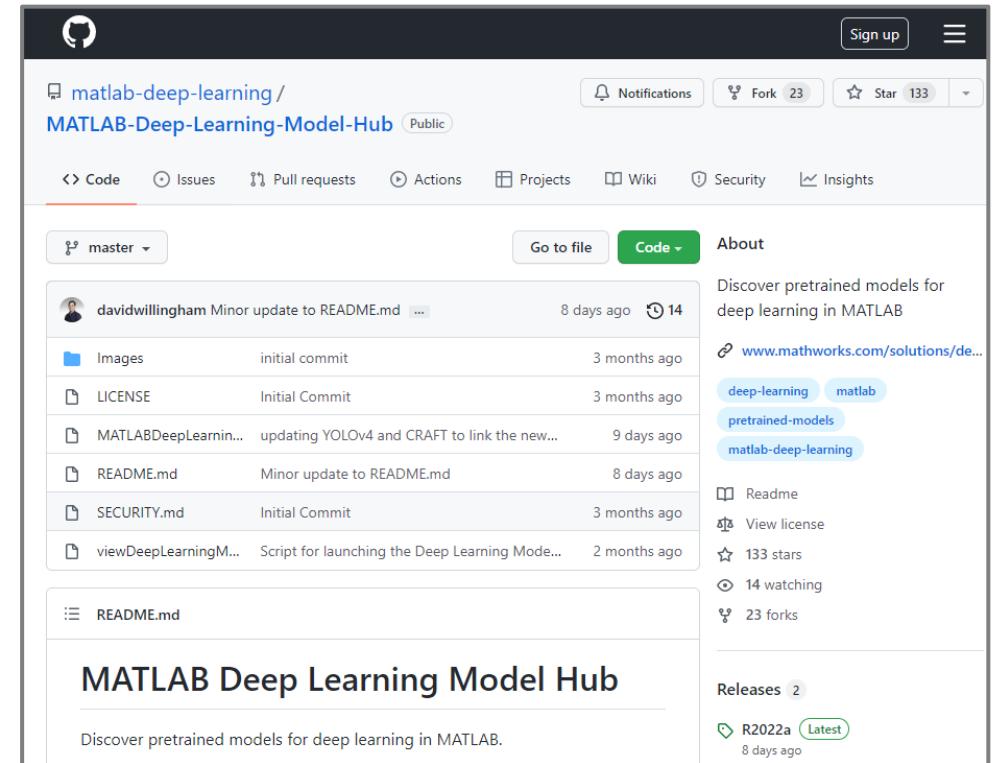
Take advantage of the knowledge provided by pretrained networks to learn new patterns in new data

Import it from other platforms



Find one directly in MATLAB

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



AI Modeling

Import model from
TensorFlow or PyTorch

The screenshot shows a MATLAB code editor window titled "Seminars\AI_MBD_VirtualSensors\Part 1 - AI Modeling\SOC_2_ImportFromTensorflow.mlx". The code is organized into sections:

- Import TensorFlow Network into MATLAB**

```
9 battery_SOC_net = importNetworkFromTensorFlow(fullfile(projectPath,"models","TF2.3","batterySOC_net"))

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

        Layers: [8x1 nnet.cnn.layer.Layer]
        Connections: [7x2 table]
        Learnables: [8x3 table]
        State: [0x3 table]
        InputNames: {'input_2'}
        OutputNames: {'dense_7'}
        Initialized: 1

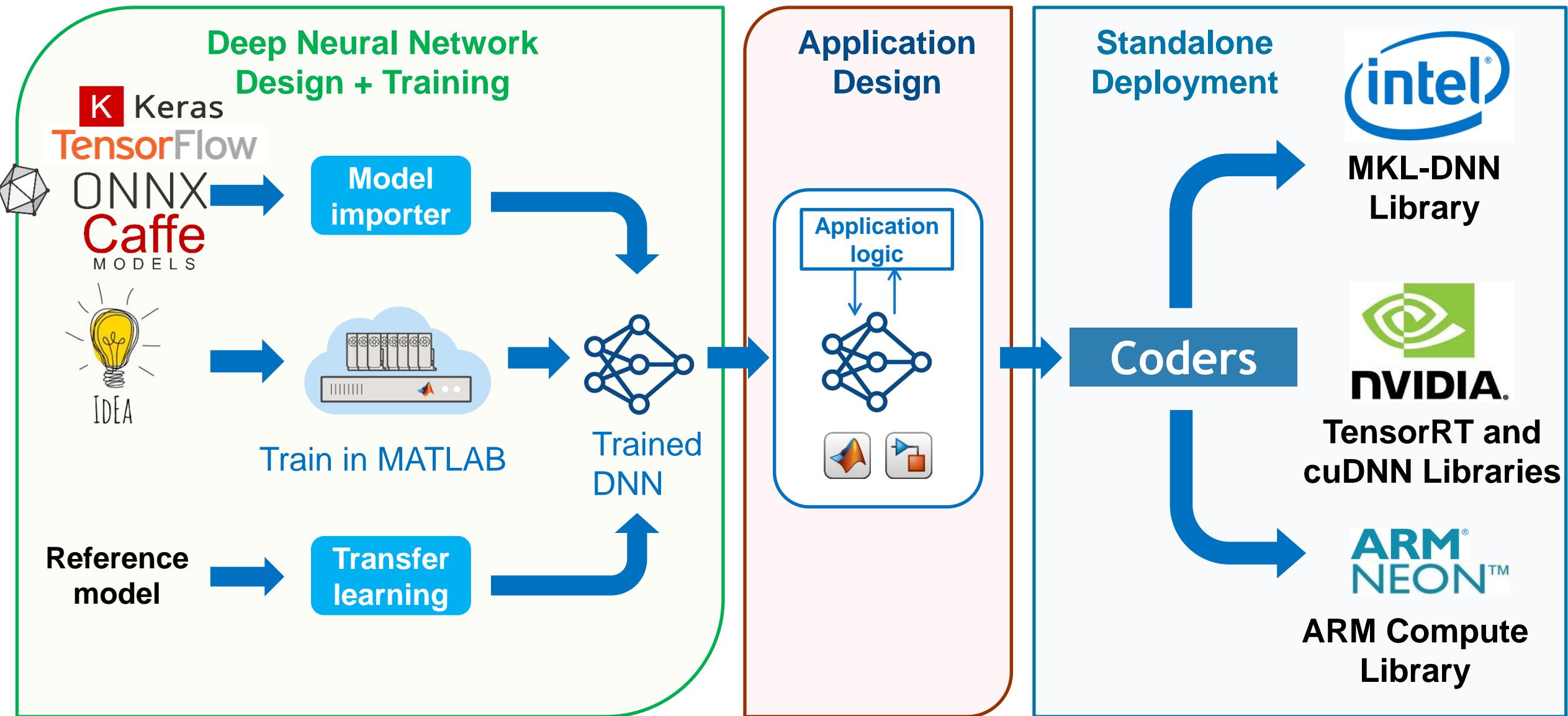
    View summary with summary.
10 save(fullfile(projectPath,"models","battery_SOC_net.mat"),"battery_SOC_net")
```
- Analyze the Imported Network Architecture**

```
11 analyzeNetwork(battery_SOC_net);
```
- Load Test Data**

```
12 % Load test data
13 load(fullfile(projectPath,"data","testData.mat"));
14
15 %% Evaluate for test case for negative 10deg C temperature.
16 X_Test_n10degC_Norm = zeros(size((Xinput{1,1}')));
17 for i=1:size(Xinput{1,1},2)
18     X_Test_n10degC_Norm(i,:) = Xinput{1,1}(:,i)';
19 end
20 Y_Test_n10degC_Norm = Y_Observed{1,1}';
```

The code editor has a toolbar at the top with buttons for "Output on Right", "Output Inline", "Hide Code", and "REVIEW". The status bar at the bottom shows "Zoom: 100%" and "UTF-8".

Deep Learning Workflow in MATLAB and Simulink



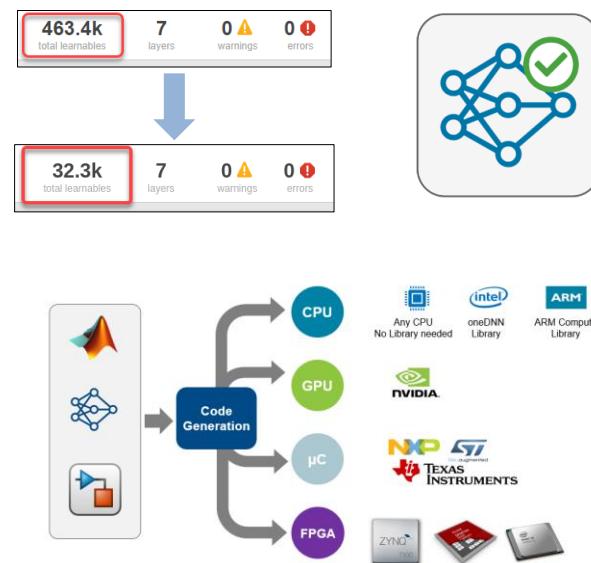
Key takeaways

- Easily design and develop AI models and integrate them in system level simulations
- Optimize neural network to fit hardware requirements
- Deployment AI applications on embedded systems using code generation
- Interoperability with open-source AI frameworks

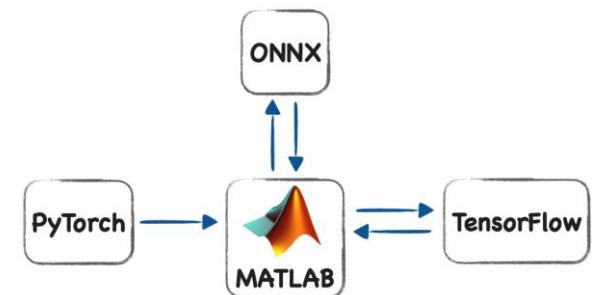
AI + Model-Based Design



Compression, Verification, Code Generation

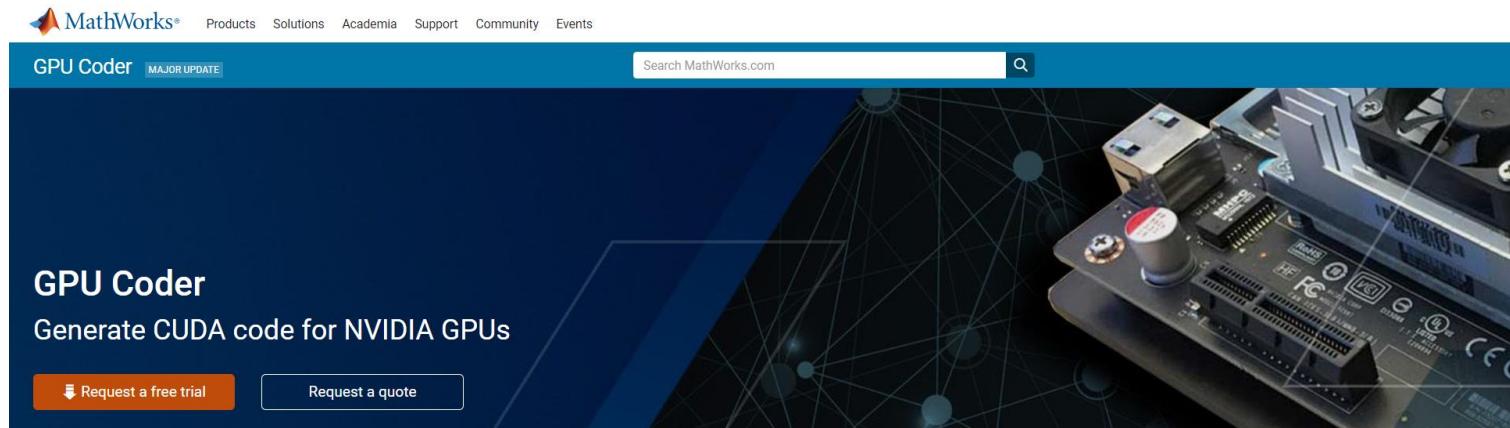


Interoperability



Next Steps

- To learn more, including playing with a trial license, visit:
 - www.mathworks.com/products/gpu-coder
- Read [white paper](#) on generating CUDA code from MATLAB
- Attend 4-hour hands-on workshop
 - Arrange with your account manager



Q&A