



Doğukan M. KILINÇ
Yapay Zeka ve Makine Öğrenmesi
Mühendisi

Pamukkale Üniversitesi

29.09.2025

MATLAB ile Yapay Zeka



Agenda

01 Introduction

- What is MATLAB?
- MATLAB for Artificial Intelligence

02 What is Machine Learning?

- Machine Learning Workflow
- Machine Learning vs Deep Learning
- Types of Machine Learning

03 Statistics and Machine Learning Introduction & Toolbox

- Classification Learner App
- Regression Learner App
- Clustering

04 What is Deep Learning?

- Deep Learning Workflow

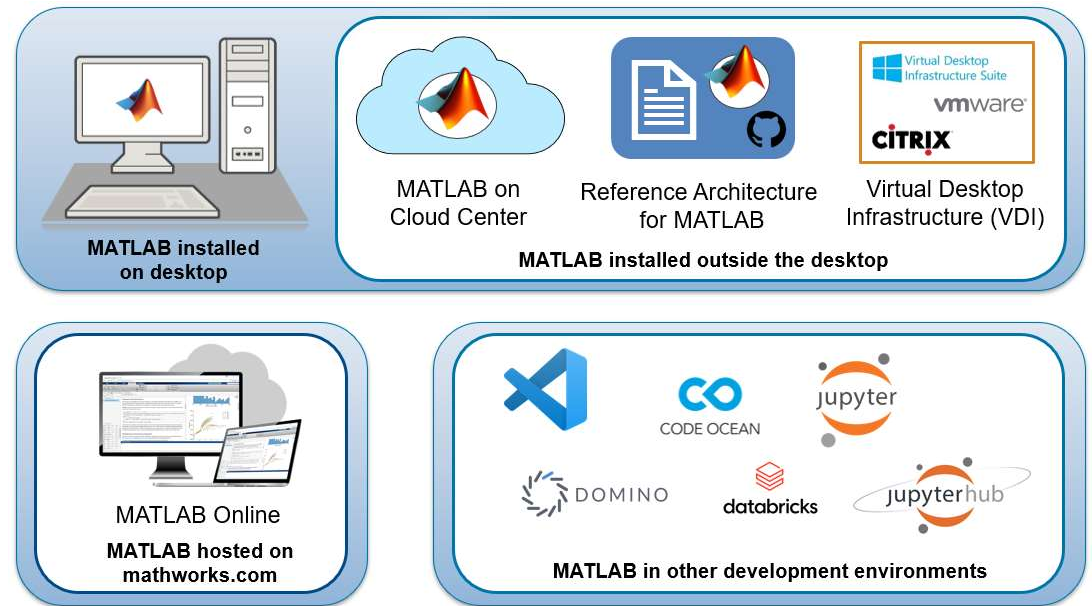
Example: Deployment Algorithm to Microprocessor

Example for PAÜ: Deep Learning for Signals

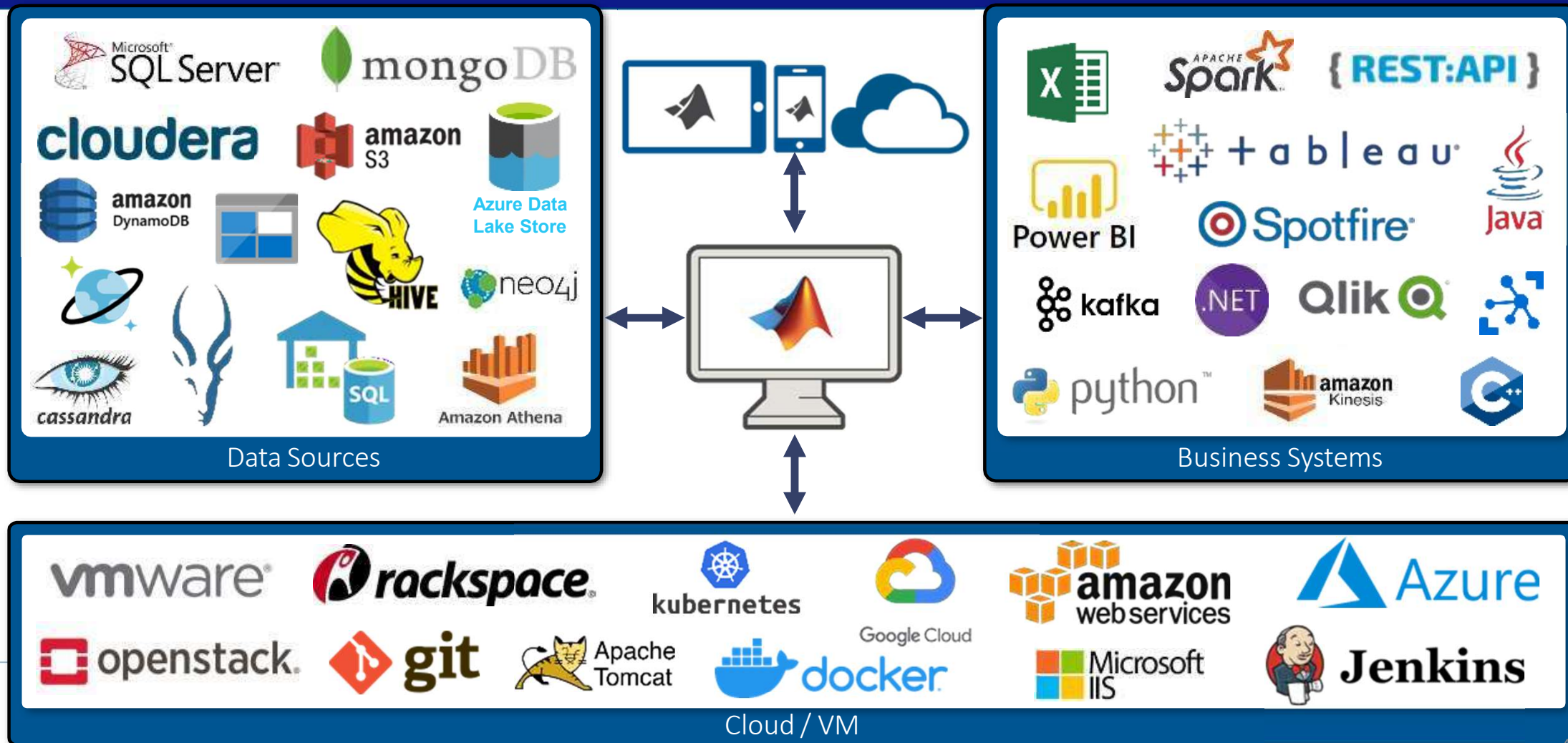
Example for PAÜ: Code Generation for ECG Data Project

What is MATLAB?

➤ Interactive Development Environment

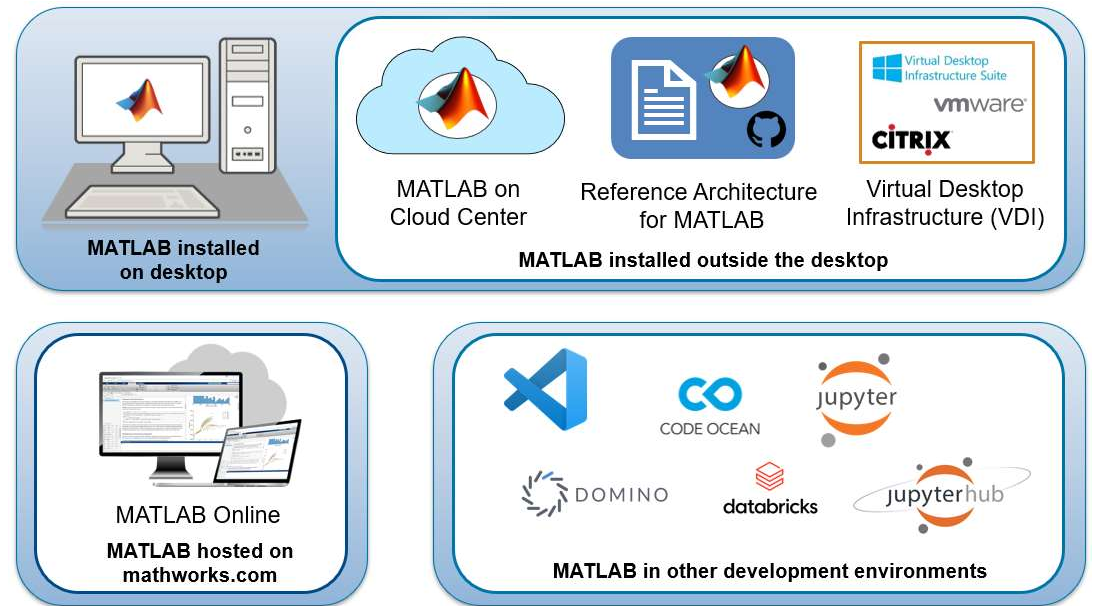


MATLAB and the Analytics Ecosystem



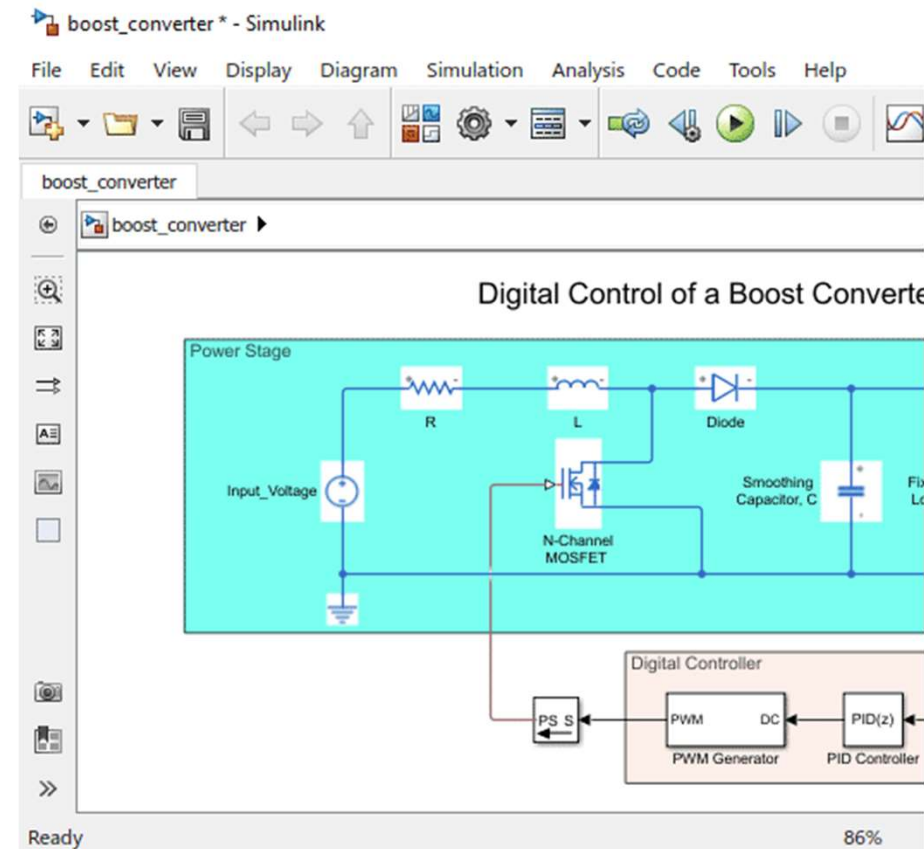
What is MATLAB?

- Interactive Development Environment
- Technical Computing Language
- Data Analysis and Visualization
- Algorithm Development and Application Development

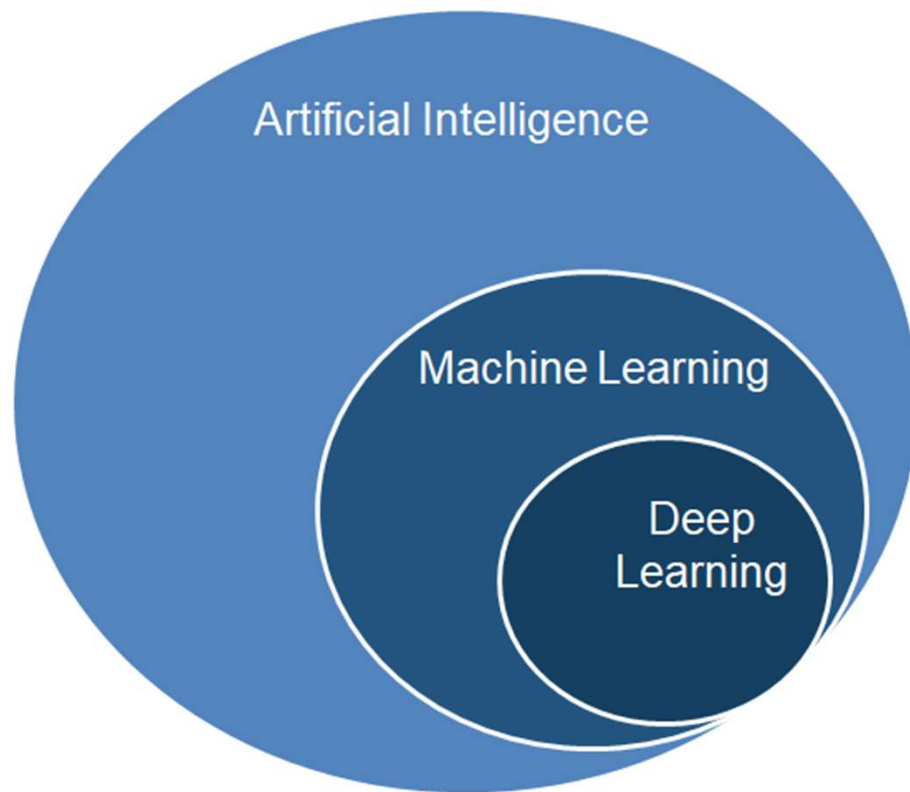


What is Simulink?

- Graphical environment
- Integration with MATLAB
- Multi-domain modelling
- Simulation and validation
- Model-Based Design



MATLAB for Artificial Intelligence



- Machine Learning
- Deep Learning
- Image Processing
- Reinforcement Learning
- Predictive Maintenance
- Data Science / Data Analytics
- Signal Processing
- ...and more

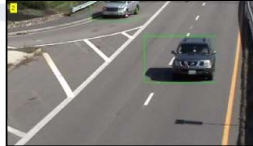


Machine Learning is Everywhere

Solution is too complex for hand written rules or equations



Speech
Recognition



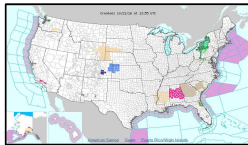
Object
Recognition



Engine Health
Monitoring

learn complex non-linear relationships

Solution needs to adapt with changing data



Weather
Forecasting



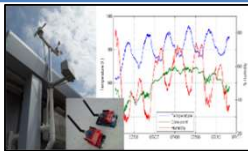
Energy Load
Forecasting



Stock Market
Prediction

update as more data becomes available

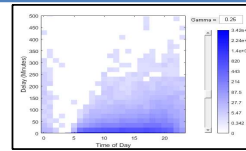
Solution needs to scale



IoT
Analytics



Taxi
Availability

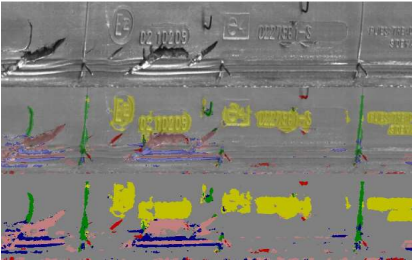


Airline Flight
Delays

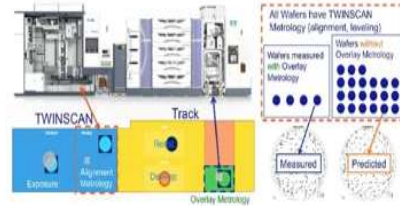
learn efficiently from very large data sets



Machine Learning is Everywhere



Tire Wear



Overlay metrology improvement



Telecom customer churn prediction



Forecasting & Risk Analysis



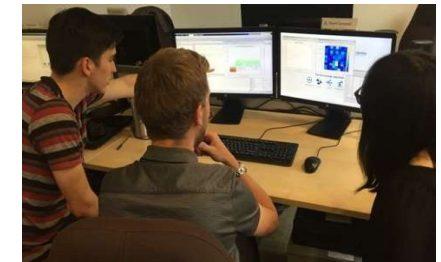
Detect Oversteer



Monitor Deployed Compressors using Digital Twin



Building energy use optimization



Portfolio Allocation



Bizi TAKİP EDİN!

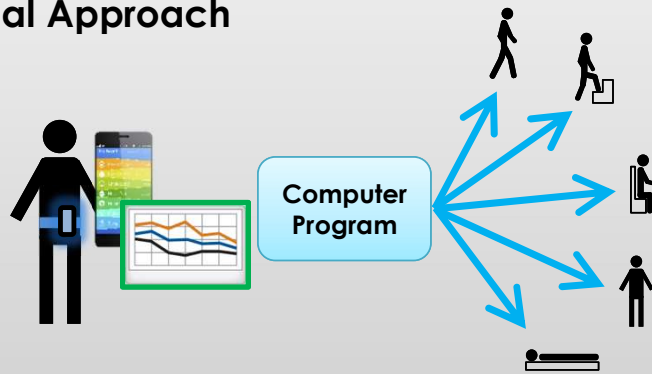


What is Machine Learning?

Machine learning uses **data** and produces a **program** to perform a **task**

Task: Human Activity Detection

Traditional Approach



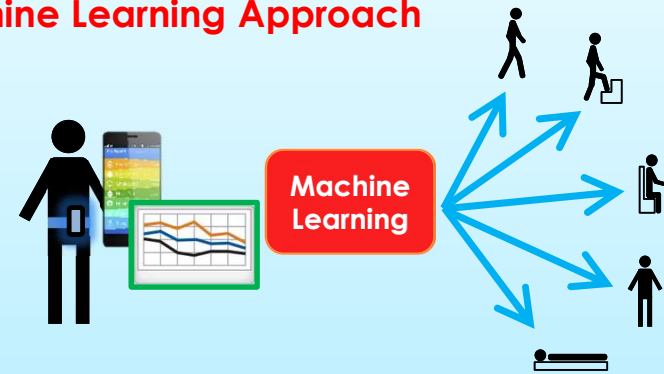
Hand Written Program

If $X_{acc} > 0.5$
then "SITTING"
If $Y_{acc} < 4$ and $Z_{acc} > 5$
then "STANDING"
...

Formula or Equation

$$Y_{activity} = \beta_1 X_{acc} + \beta_2 Y_{acc} + \beta_3 Z_{acc} + \dots$$

Machine Learning Approach

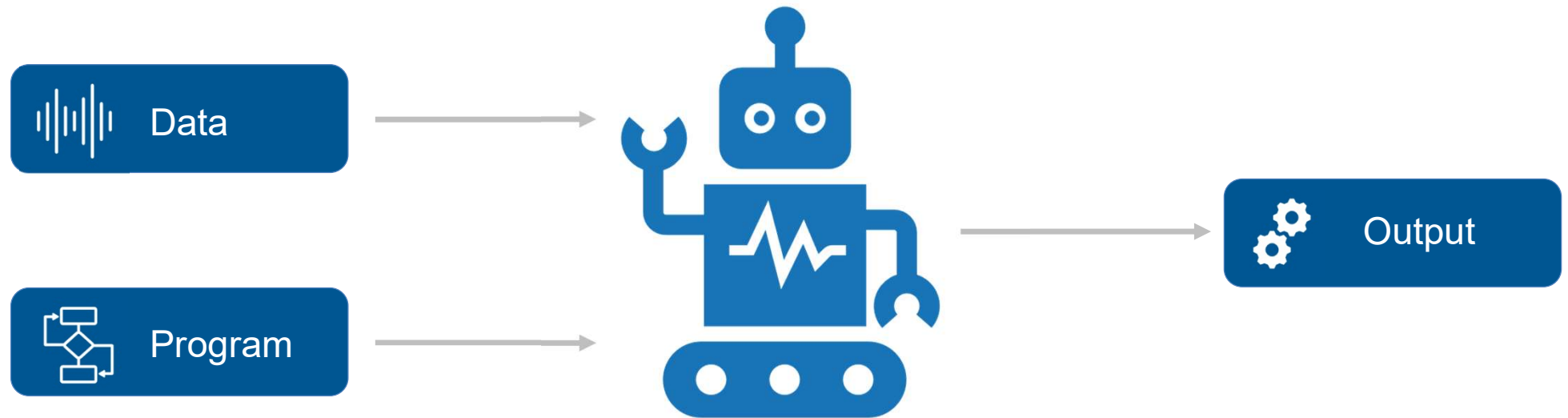


model: Inputs → Outputs

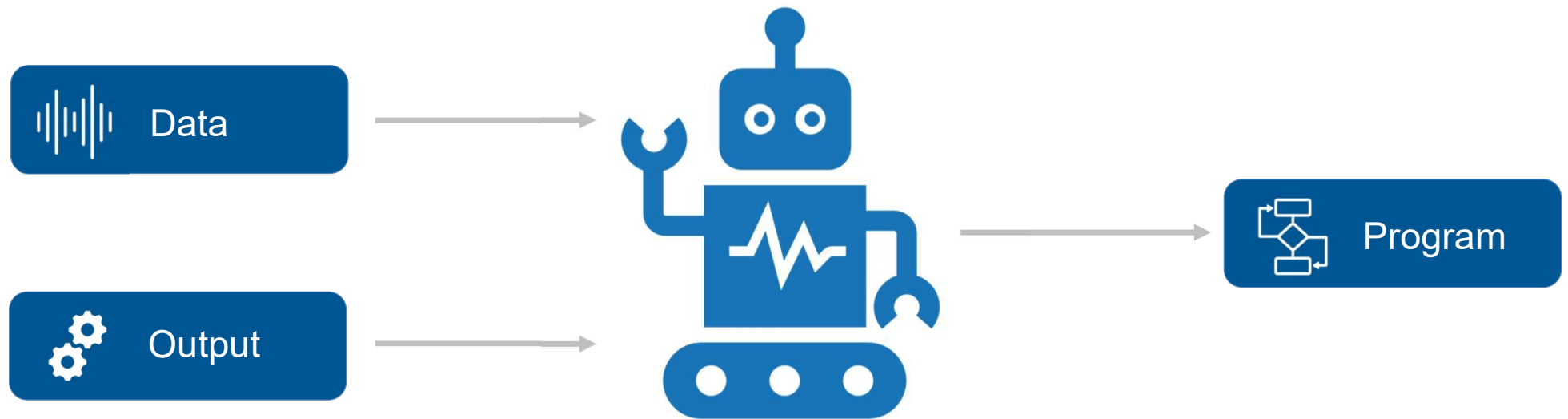
model = < Machine Learning Algorithm > (sensor_data, activity)



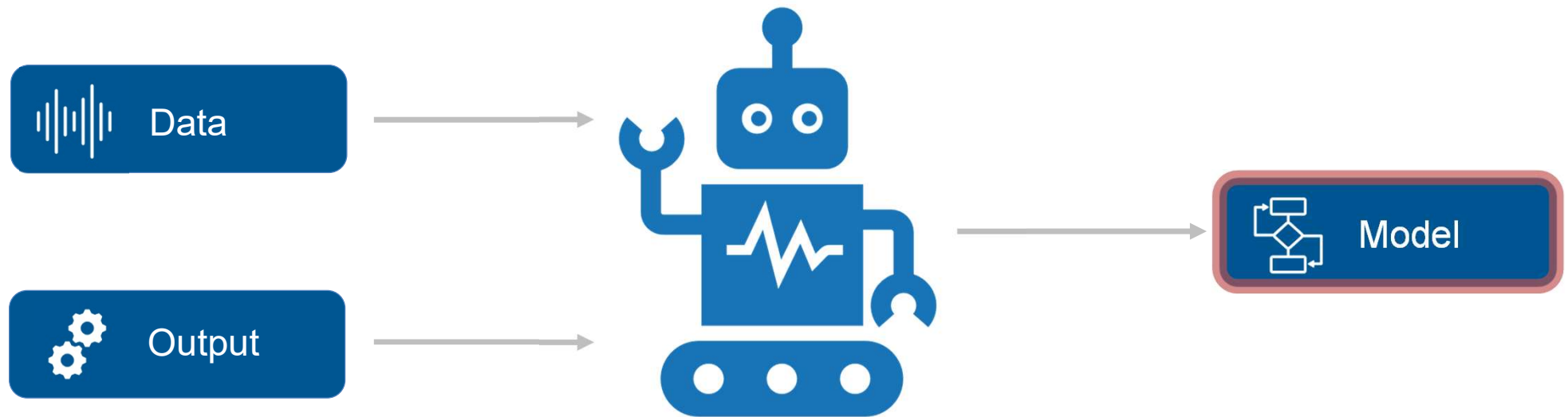
There are two ways to get a robot to do what you want



There are two ways to get a robot to do what you want



There are two ways to get a robot to do what you want



There are two ways to get a robot to do what you want

Machine Learning



Deep Learning



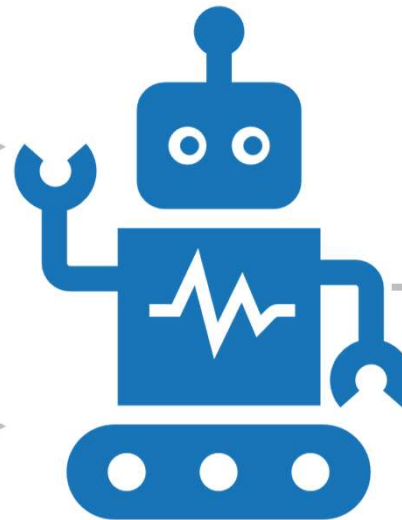
Reinforcement Learning



Data



Output



Artificial Intelligence



Model



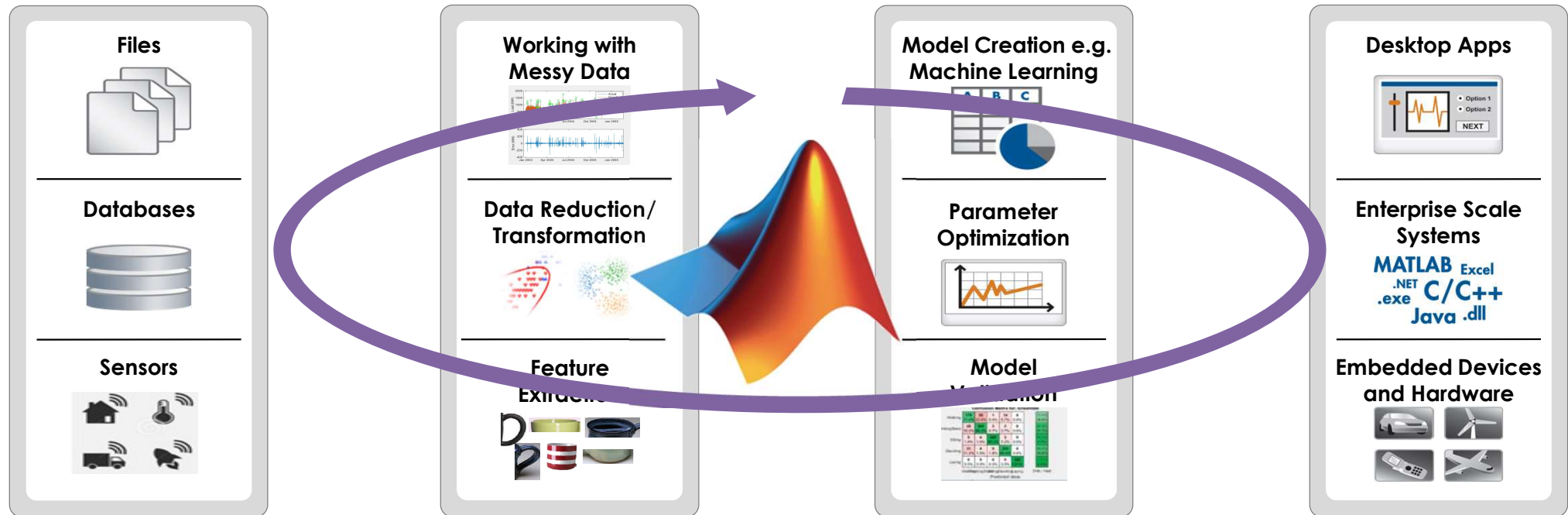
Machine Learning Workflow

Access and Explore
Data

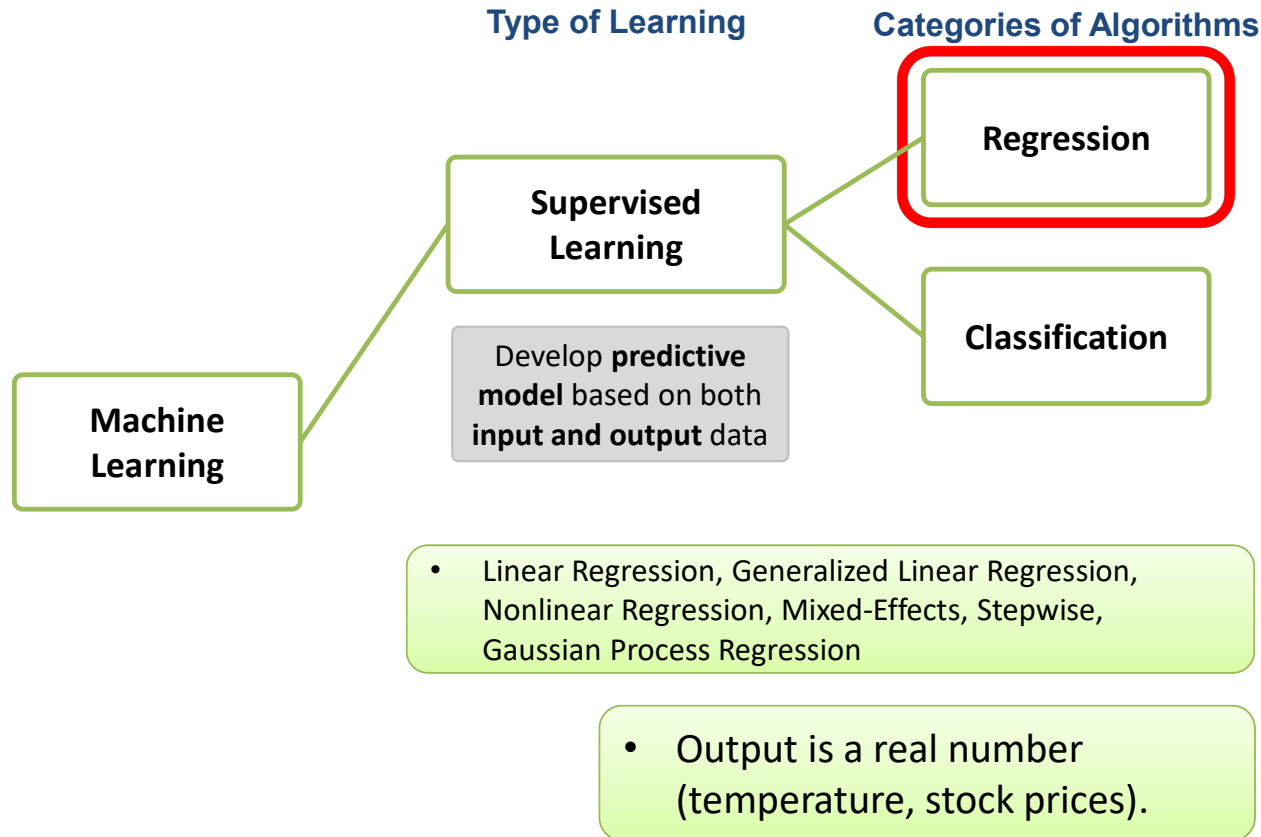
Preprocess Data

Develop Predictive
Models

Integrate Analytics with
Systems

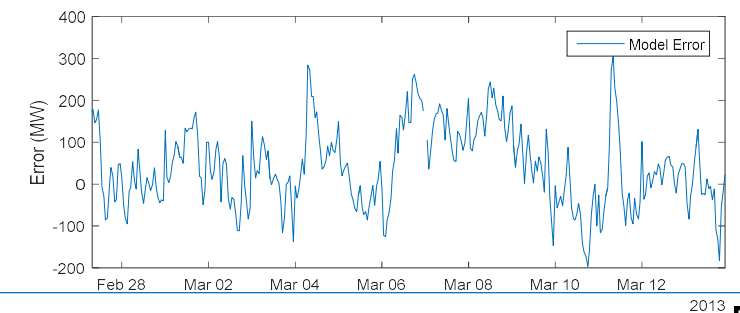
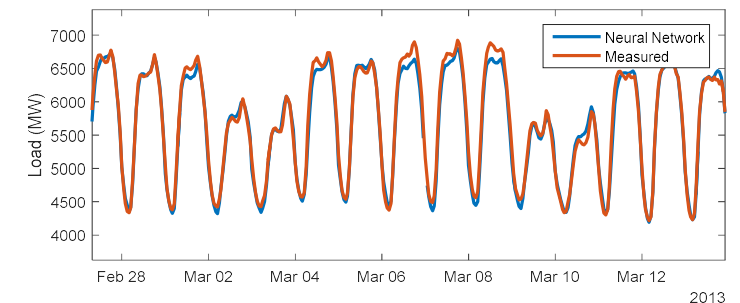


Types of Machine Learning

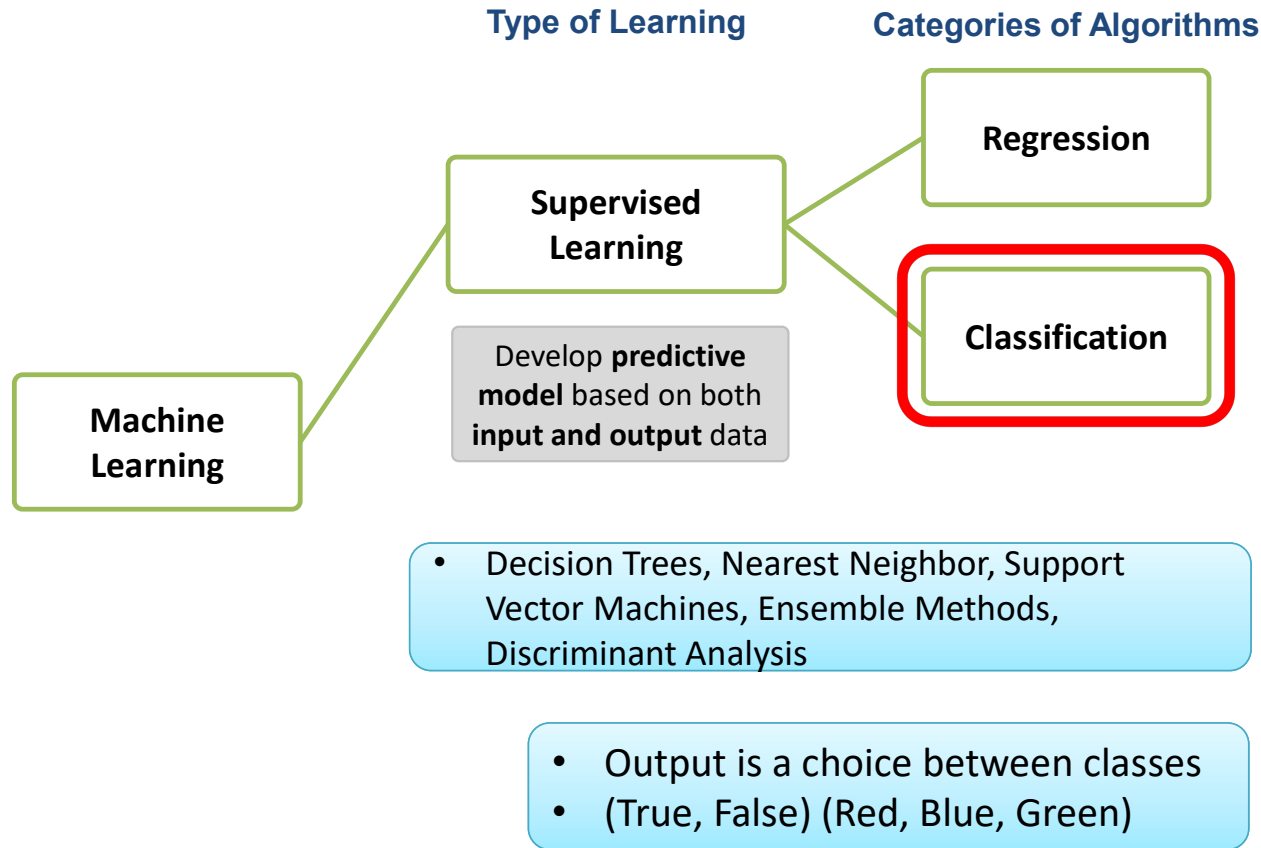


Objective:

Easy and accurate computation of day-ahead system load forecast





Types of Machine Learning



Objective:

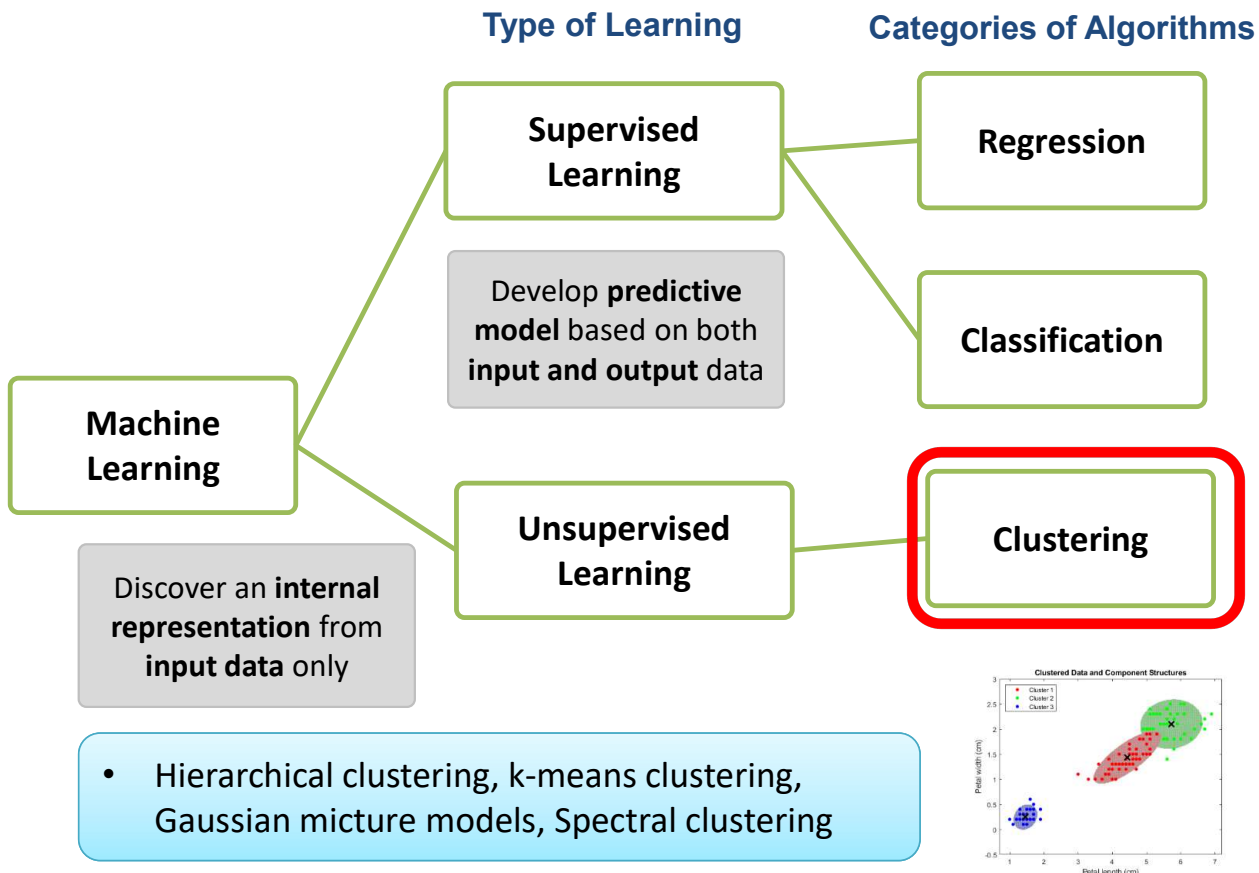
Train a classifier to classify human activity from sensor data

Data:

Inputs	3-axial Accelerometer 3-axial Gyroscope	
Outputs		



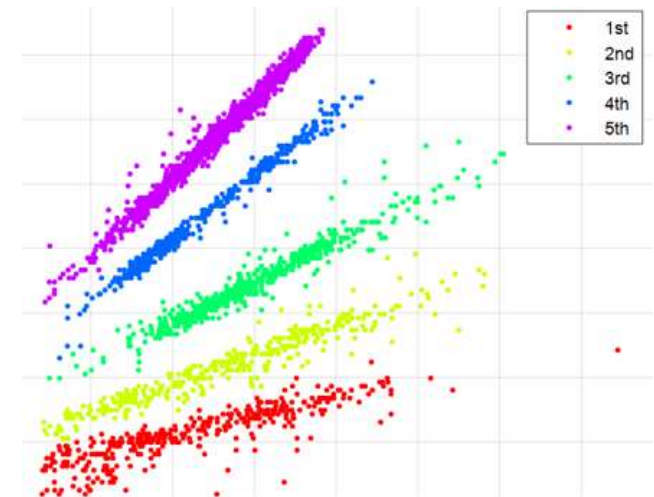
Types of Machine Learning



Objective:

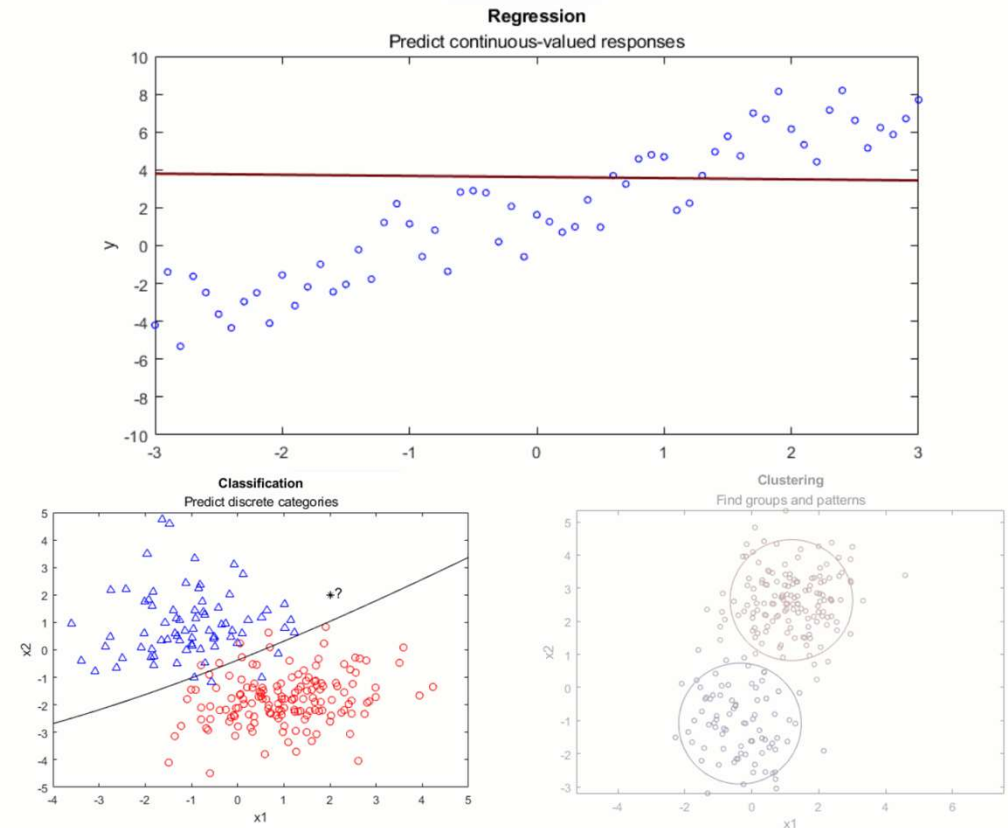
Given data for engine speed and vehicle speed, identify clusters

- Discover a good internal representation
- Learn a low dimensional representation



Statistic and Machine Learning Toolbox

- ✓ Provides functions and apps to describe, analyze, and model data. Use descriptive statistics, visualizations, and clustering for exploratory data analysis; fit probability distributions to data.
- ✓ Regression and classification algorithms let you draw inferences from data and build predictive models either interactively, using the Classification and Regression Learner apps, or programmatically, using AutoML.



Classification Learner App

The Classification Learner app interface is shown in three overlapping screenshots. The top screenshot shows the 'New Session' menu with 'From Workspace' highlighted. The middle screenshot shows the 'Data set' dialog box with 'meas' selected as the data set variable and 'species' as the response variable. The bottom screenshot shows the 'Validation' tab with a list of models and a confusion matrix for Model 2.1.

Data set dialog box:

Data Set Variable: **meas** (150x4 double)

Response: **species** (150x1 cell)

Predictors:

Name	Type	Range
column_1	double	4.3 - 7.9
column_2	double	2 - 4.4
column_3	double	1 - 6.9
column_4	double	0.1 - 2.5

Validation tab:

Sort by: Accuracy (Validation)

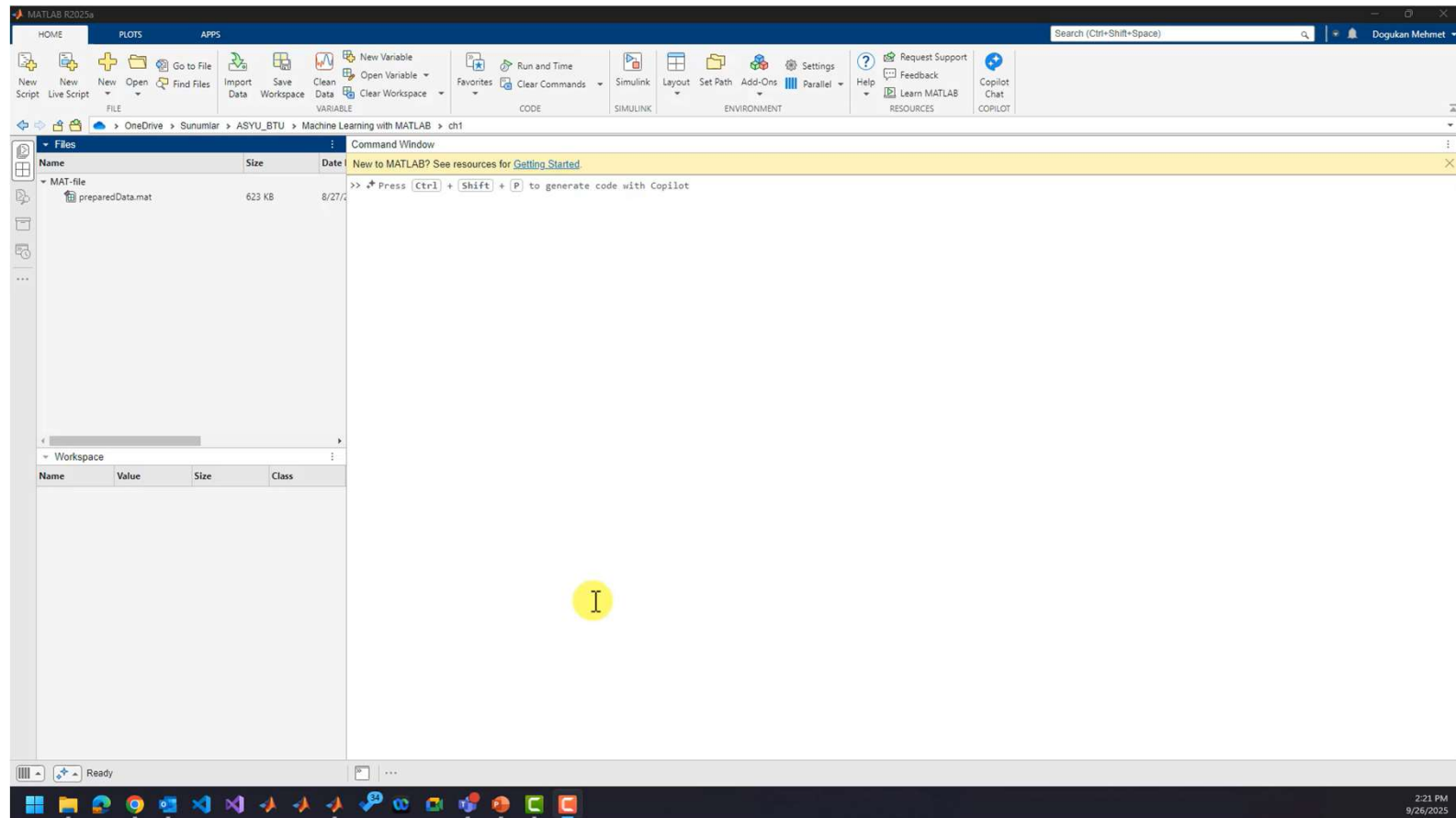
Model	Accuracy (Validation)	Features
2.8 SVM	97.8%	4/4 features
2.4 Linear D...	96.3%	4/4 features
2.5 Quadratic...	96.3%	4/4 features
2.7 Naive B...	96.3%	4/4 features
2.12 SVM	96.3%	4/4 features
2.13 SVM	96.3%	4/4 features
2.6 Naive B...	95.6%	4/4 features
2.10 SVM	95.6%	4/4 features
2.15 KNN	95.6%	4/4 features
2.19 KNN	95.6%	4/4 features
2.21 Ensam...	95.6%	4/4 features

Confusion Matrix for Model 2.1:

	setosa	versicolor	virginica
setosa	45		
versicolor		41	4
virginica		3	42

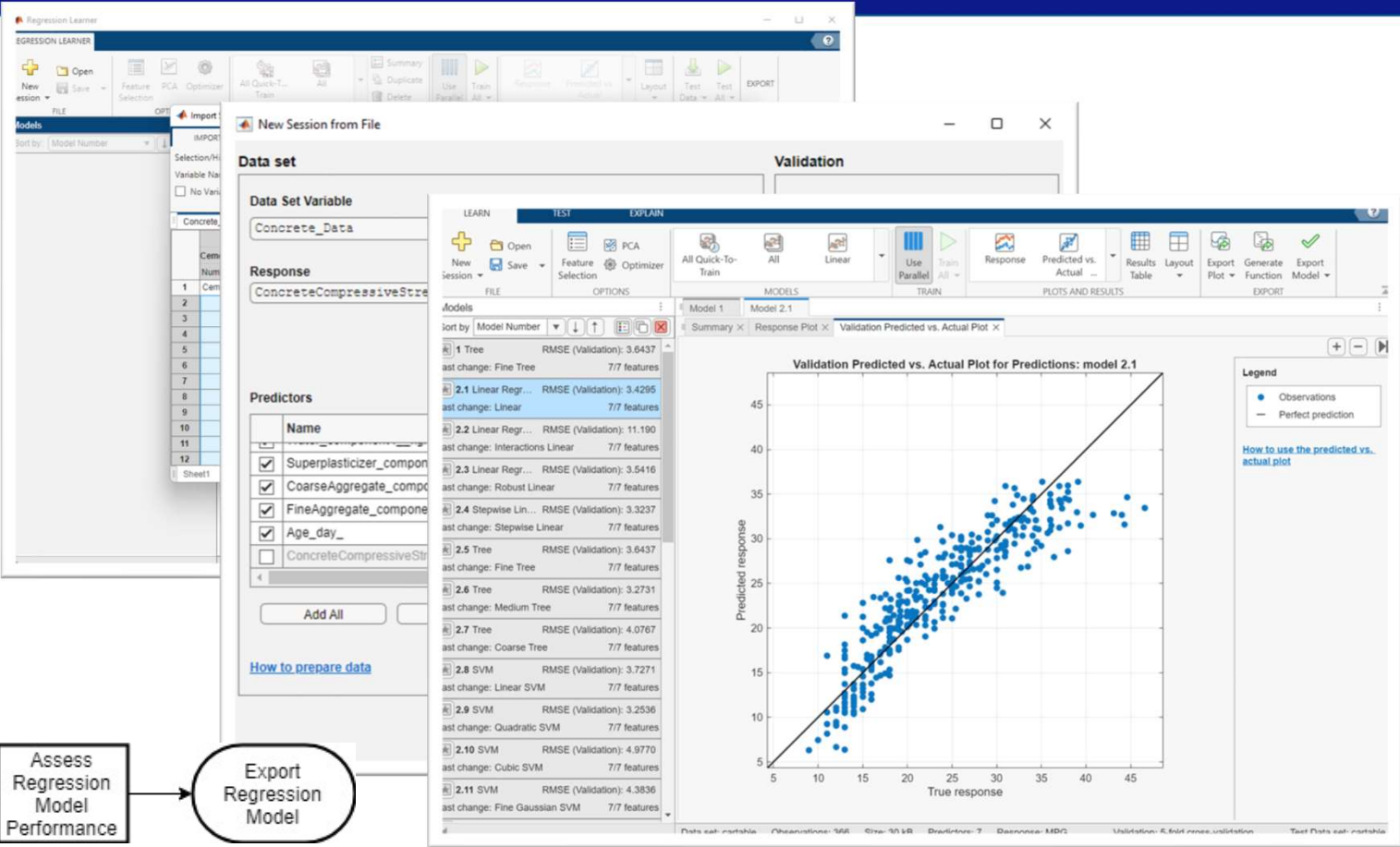
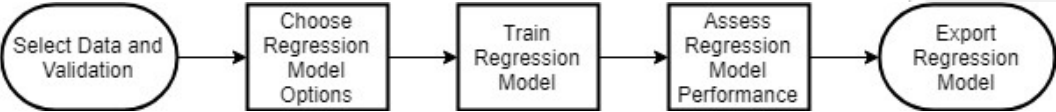


Classification Learner App

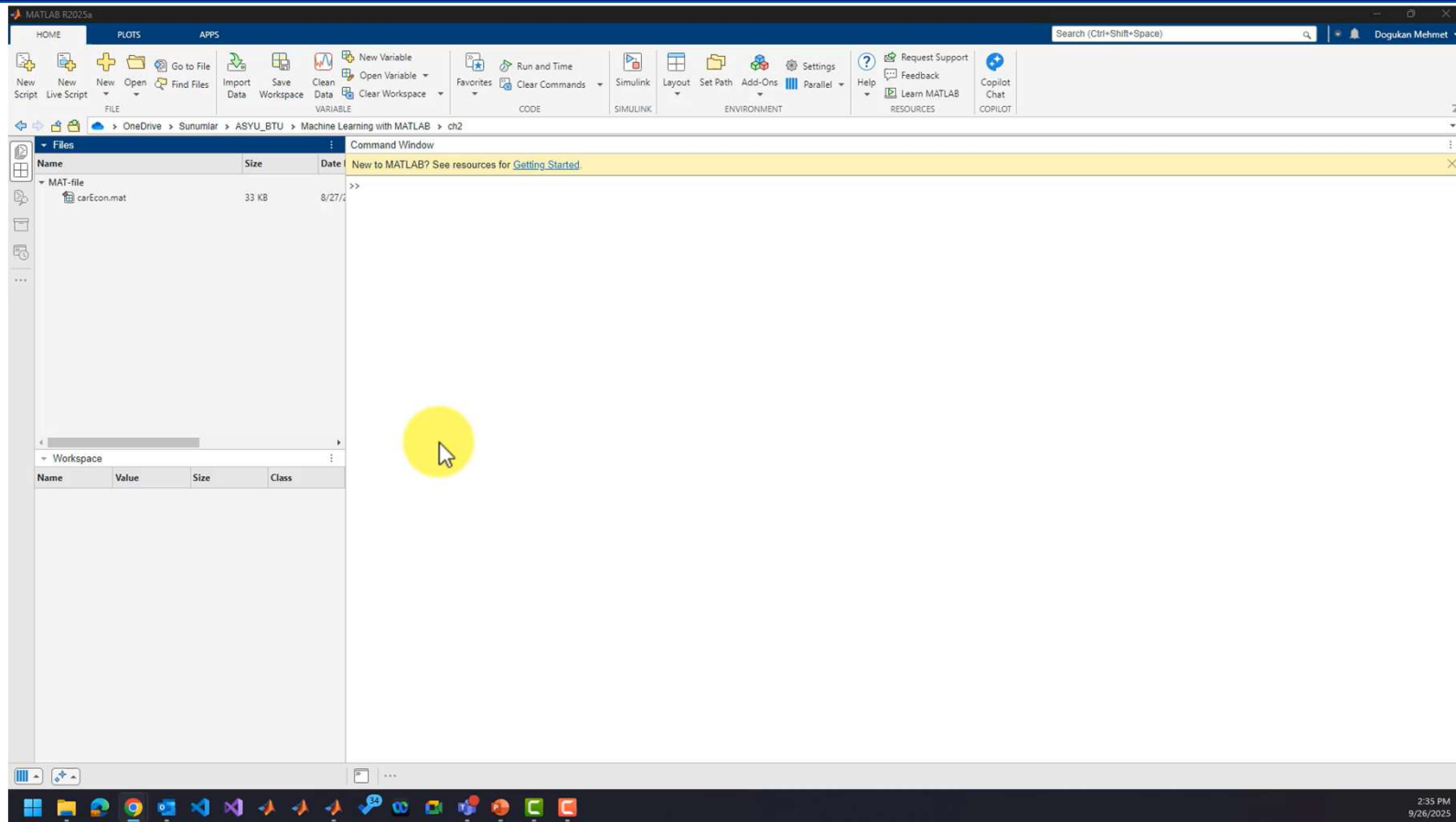


Regression Learner App

Regression models describe the relationship between a response (output) variable, and one or more predictor (input) variables.



Regression Learner App

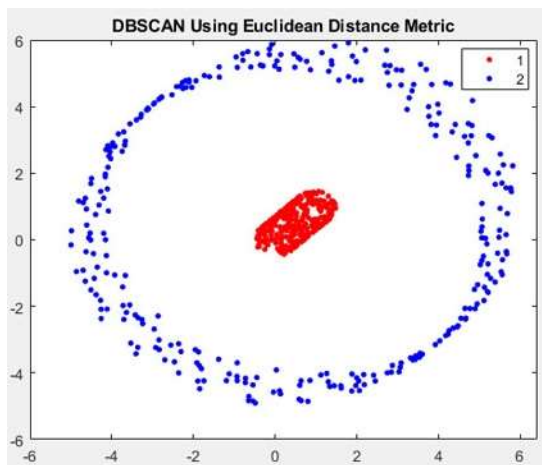


Unsupervised Learning

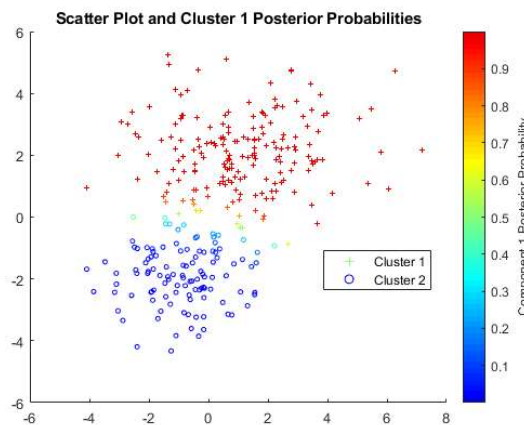
Clustering

Data Clustering / Unsupervised Learning

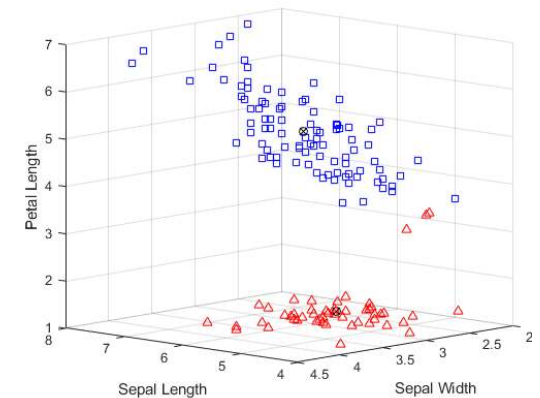
Discover patterns by grouping data using k-means, k-medoids, DBSCAN, hierarchical clustering, and Gaussian mixture and hidden Markov models.



Applying DBSCAN to two concentric groups.



Scatter Plot and Clusters

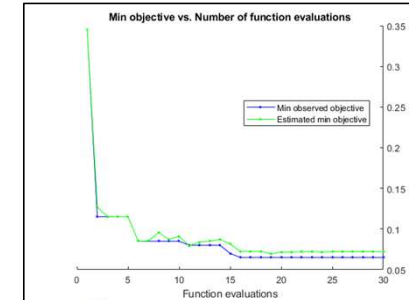
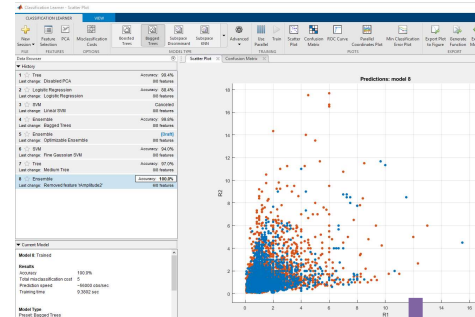


K-means Clustering Results



Machine Learning Wrap-up

Interactive Model Training



Automated Model Optimization

Import Data

Preprocess Data

Engineer Features

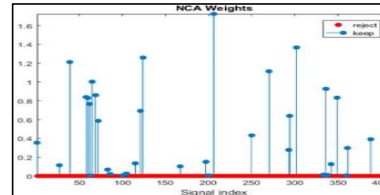
Build Model

Deploy & Integrate

Clean Messy Raw Data



Feature Selection



Automated Code Generation



Machine Learning Summary

[Machine Learning Onramp](#) (2 hr online introduction)

[Practical Data Science with MATLAB](#) (4 course Specialization)

Machine Learning with MATLAB:

- [Overview, Cheat sheet](#)
- [Machine Learning Intro](#) (Tech talks)
- [Machine Learning with MATLAB Introduction](#) (eBook)
- [Mastering Machine Learning](#) (eBook)
- [Applied Machine Learning](#) (Tech Talk videos)

Machine and Deep Learning

- [Deep vs. Machine Learning: Choosing the Best Approach](#) (eBook)
- [Deep learning Onramp](#) (2hr online introduction)

Five Interactive Apps for Machine Learning

No matter what type of problem you're trying to solve, MATLAB® is here to help. Discover apps to interactively model, fit, and label data for machine learning.

Classification Learner

Regression Learner

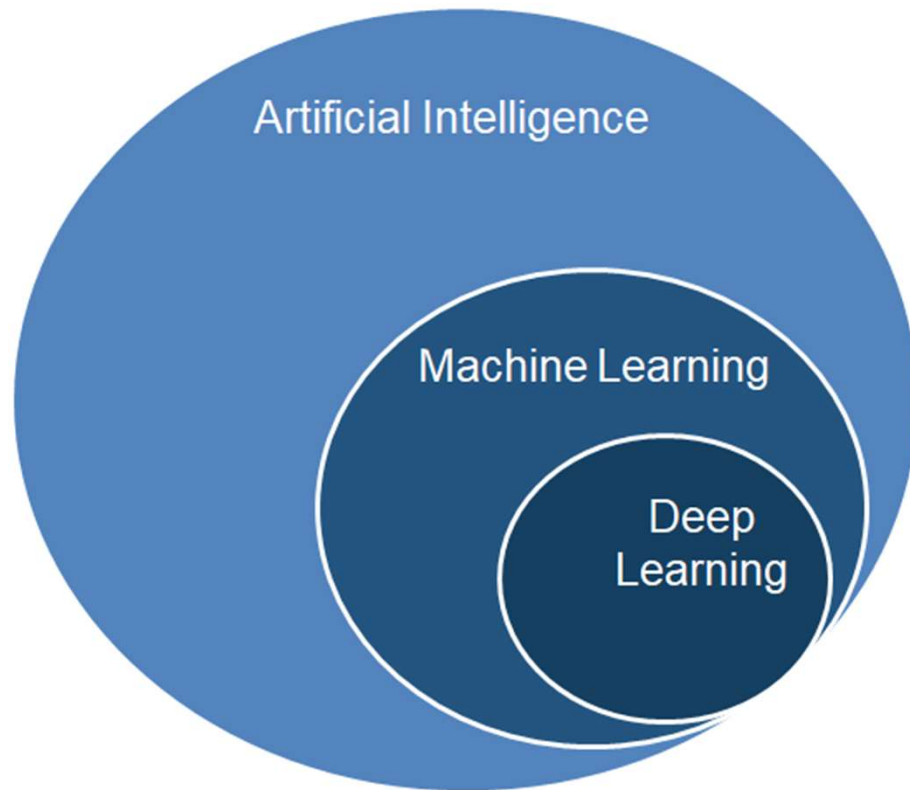
Curve Fitting

Image Labeler

Signal Labeler



MATLAB for Artificial Intelligence



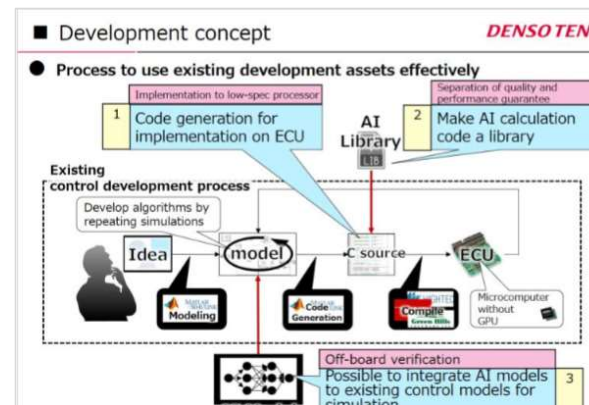
- Machine Learning
- Deep Learning
- Image Processing
- Reinforcement Learning
- Predictive Maintenance
- Data Science / Data Analytics
- Signal Processing
- ...and more



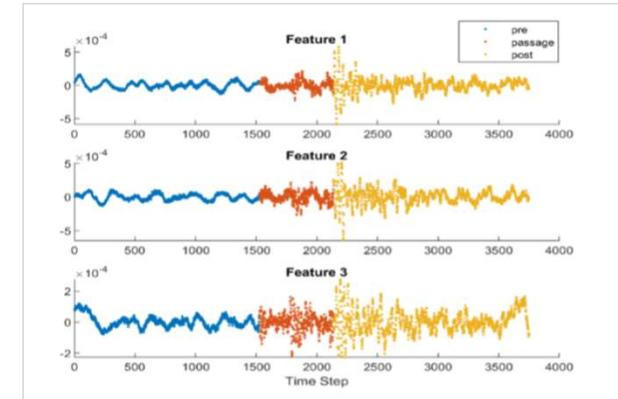
MATLAB Deep Learning used in Industry



**Automatic Defect
Detection**
Airbus



ECU Vehicle Control
Denso



Seismic Event Detection
Shell



What is Deep Learning?

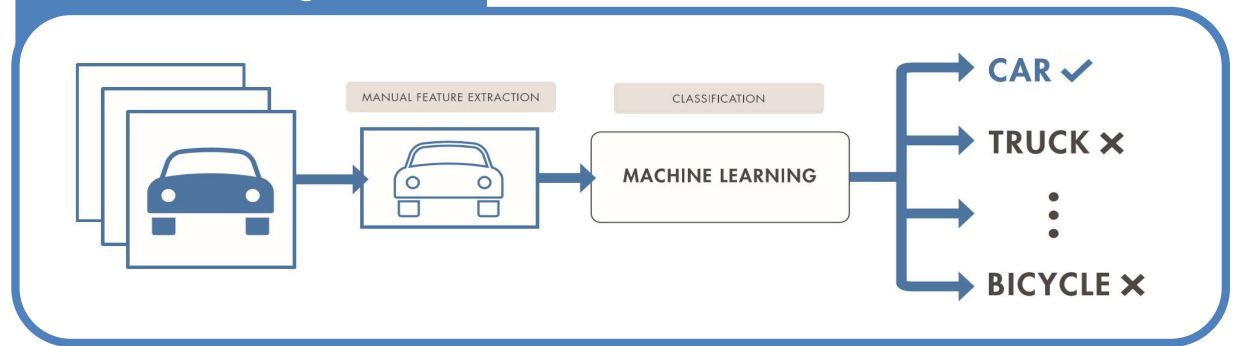
Machine Learning

Deep Learning

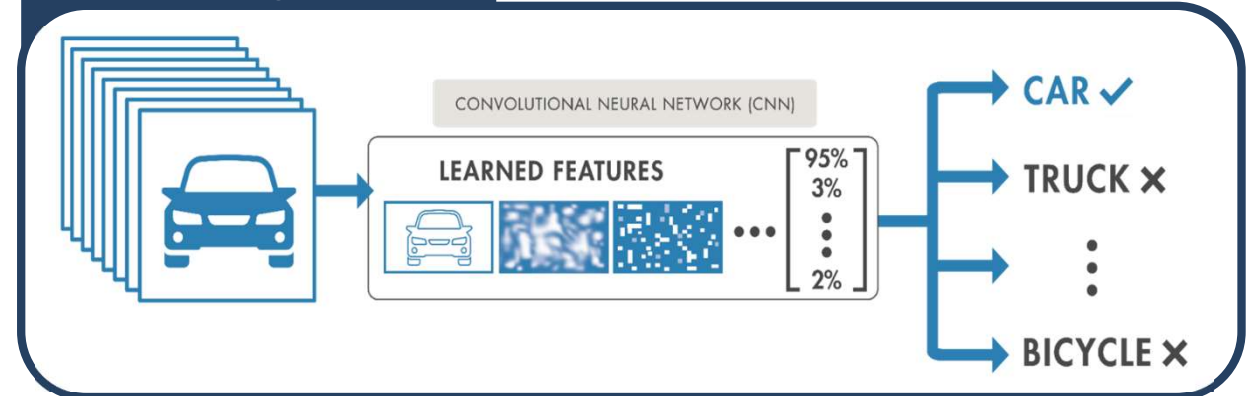
Neural Networks
with many Hidden
Layers

- Learns directly from data
- More Data = better model
- Computationally Intensive
- **Not interpretable**

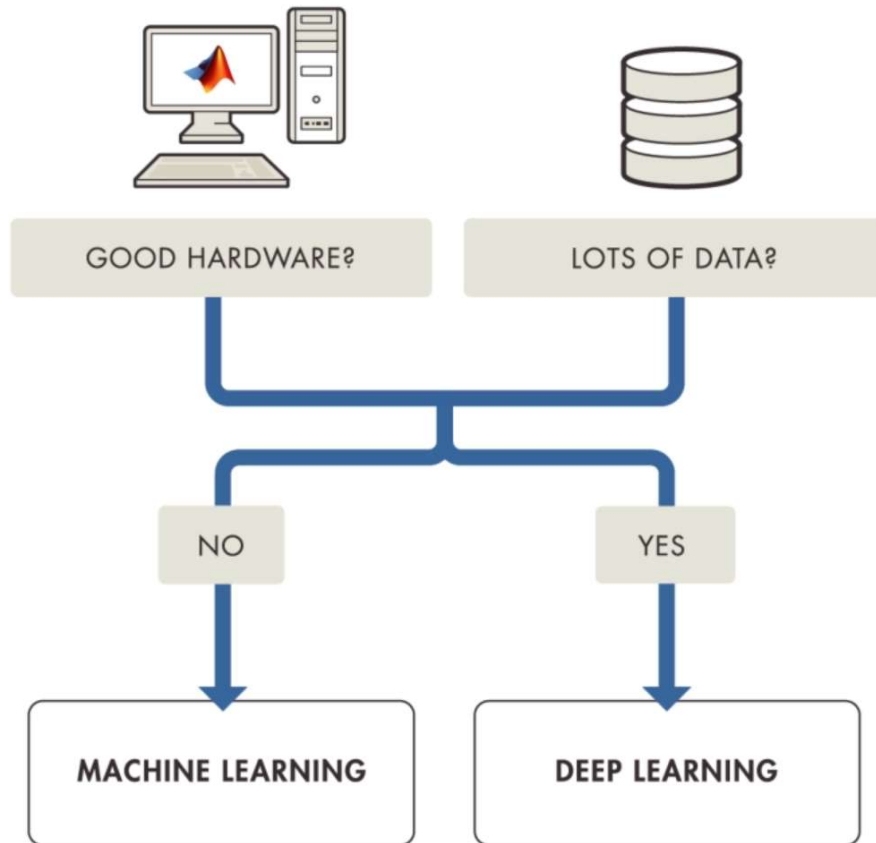
Machine Learning



Deep Learning



Machine Learning vs Deep Learning



	Machine Learning	Deep Learning
Training dataset	Small	Large
Choose your own features	Yes	No
# of classifiers available	Many	Few
Training time	Short	Long



Applications of deep learning for images and video



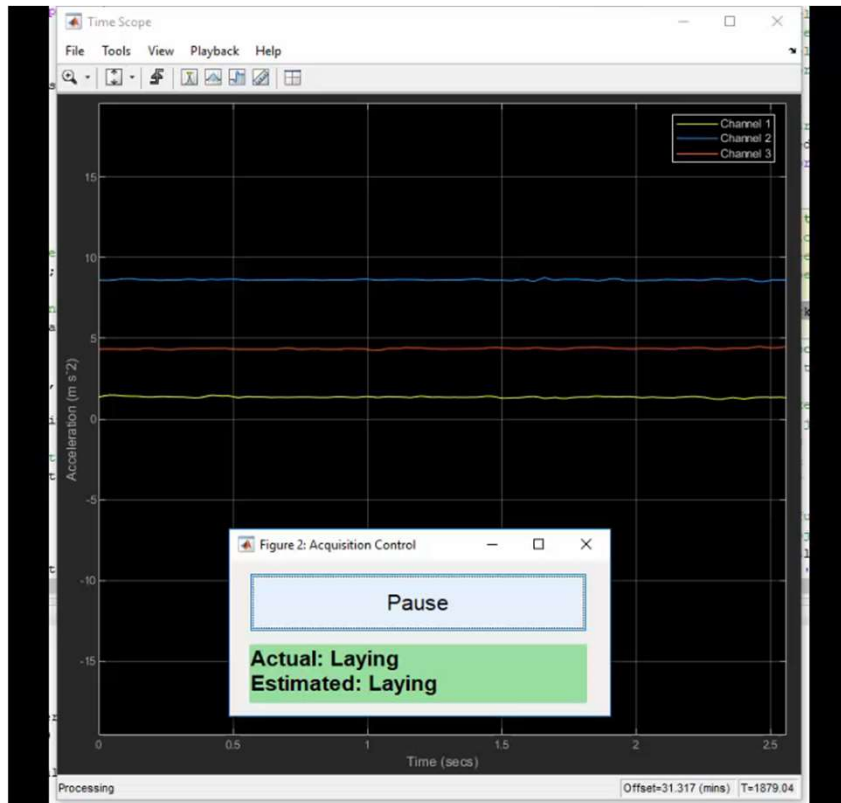
YOLO v2 (You Only Look Once)



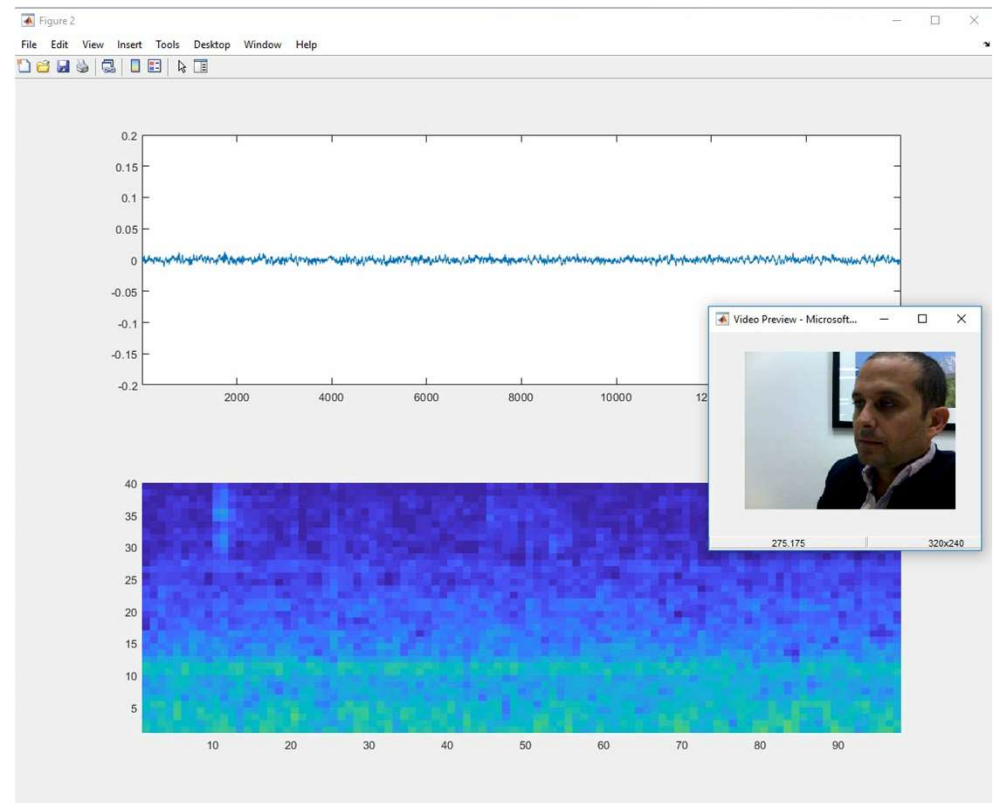
Semantic Segmentation using SegNet



Applications of deep learning for signal processing



Signal Classification using LSTMs



Speech Recognition using CNNs



Deep Learning Workflow

ACCESS AND EXPLORE
DATA

LABEL AND PREPROCESS
DATA

DEVELOP PREDICTIVE
MODELS

INTEGRATE MODELS WITH
SYSTEMS

Files



Databases



Sensors



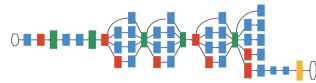
Data Augmentation/
Transformation



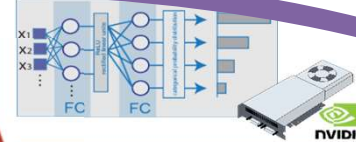
Labeling Automation



Import Reference
Models



Hardware-Accelerated
Training



Hyperparameter Tuning



Network Visualization



Desktop Apps



Enterprise Scale Systems

Java
MATLAB
C/C++
Python

Embedded Devices and
Hardware



Import Data for Deep Learning Networks

ACCESS AND EXPLORE
DATA

Files



Databases

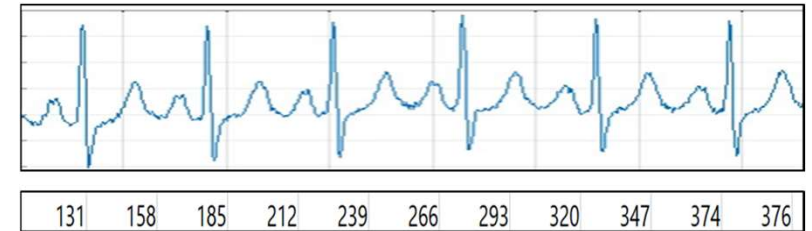


Sensors



Images are a numeric matrix

199	206	208	201	188	178	165	164	180
202	205	202	188	176	169	178	186	183
203	206	189	178	181	183	182	154	87
203	192	184	186	177	167	153	181	192
191	182	176	166	153	141	136	180	227
166	165	154	154	138	137	169	170	211
158	150	145	183	144	156	158	154	179
143	51	98	144	129	130	143	178	123
107	50	33	95	152	173	192	159	87
104	100	84	120	132	172	131	64	94
119	101	97	81	90	109	87	106	111
127	122	110	97	108	120	133	131	134
111	117	108	119	131	143	146	141	156
126	122	113	119	139	142	155	161	151
129	126	130	111	103	130	149	149	156
138	128	136	144	136	129	134	122	145
154	133	134	141	168	150	126	127	151



Signals are numeric vectors

The Bird Flies = [0 13 5 6]

The Leaf Is Brown = [13 3 11 2]

Text is processed as numeric vectors



Deep Learning requires a lot of data

ACCESS AND EXPLORE
DATA

Files



Databases

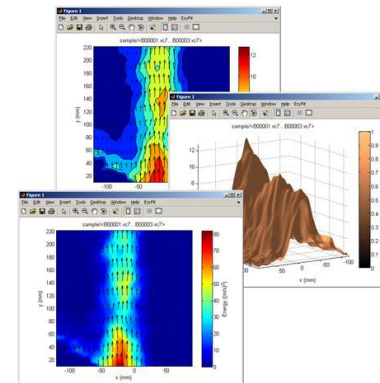
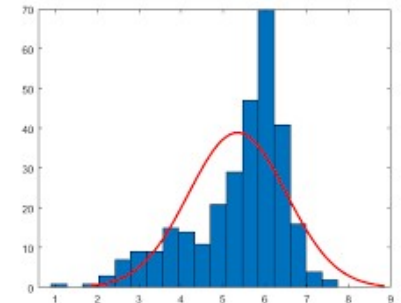
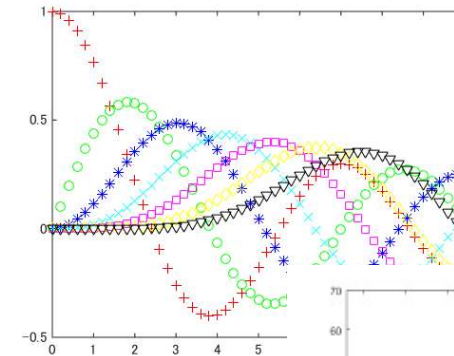


Sensors



load
niftiread
imread
fopen
importdata
readtable

BIG DATA
MATLAB
ANALYTICS
STORAGE
TECHNOLOGIES



Automated Labeling Apps

LABEL AND PREPROCESS DATA

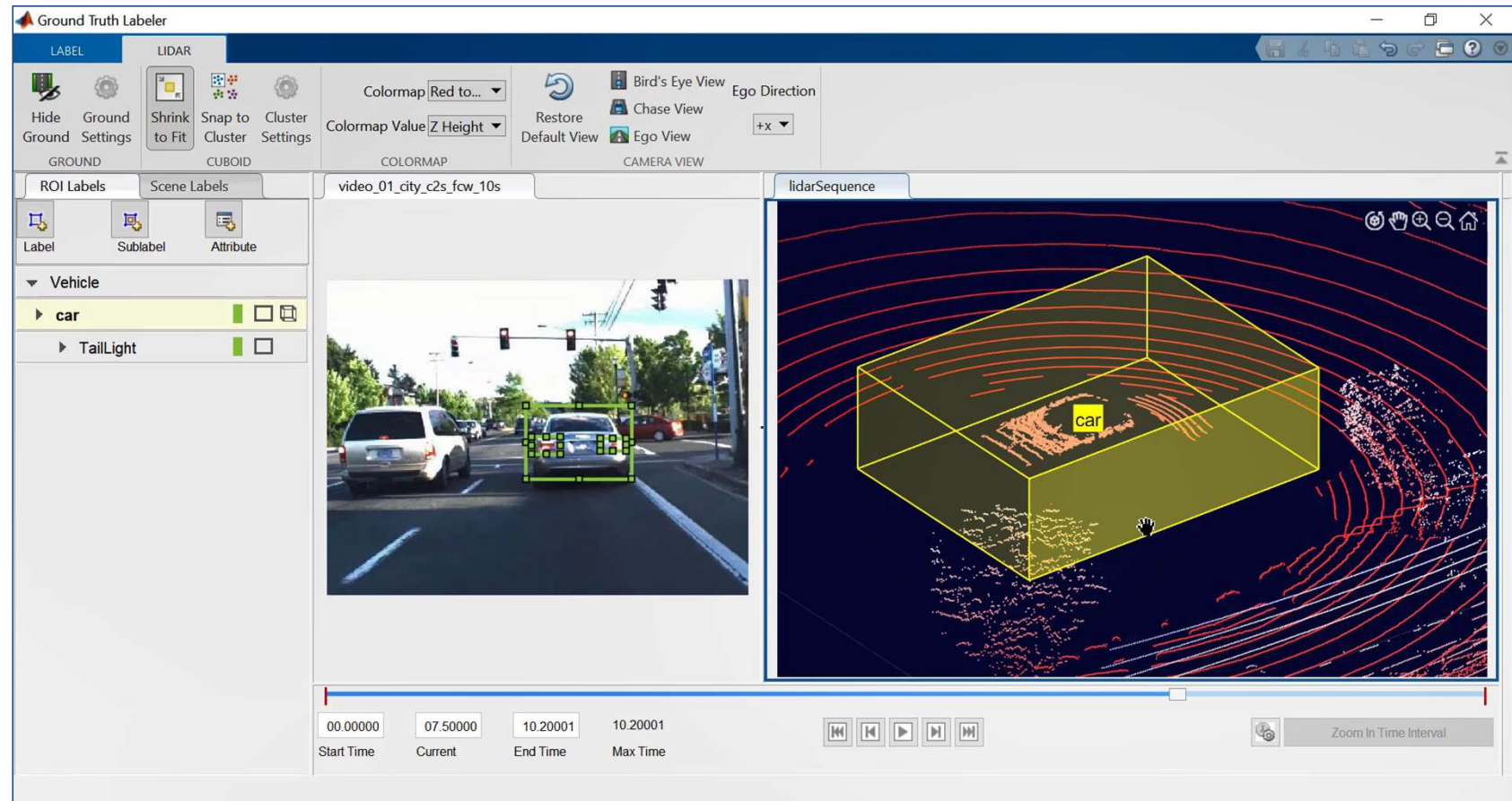
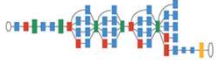
Data Augmentation/ Transformation



Labeling Automation



Import Reference Models



Synthetic Data Generation

LABEL AND PREPROCESS DATA

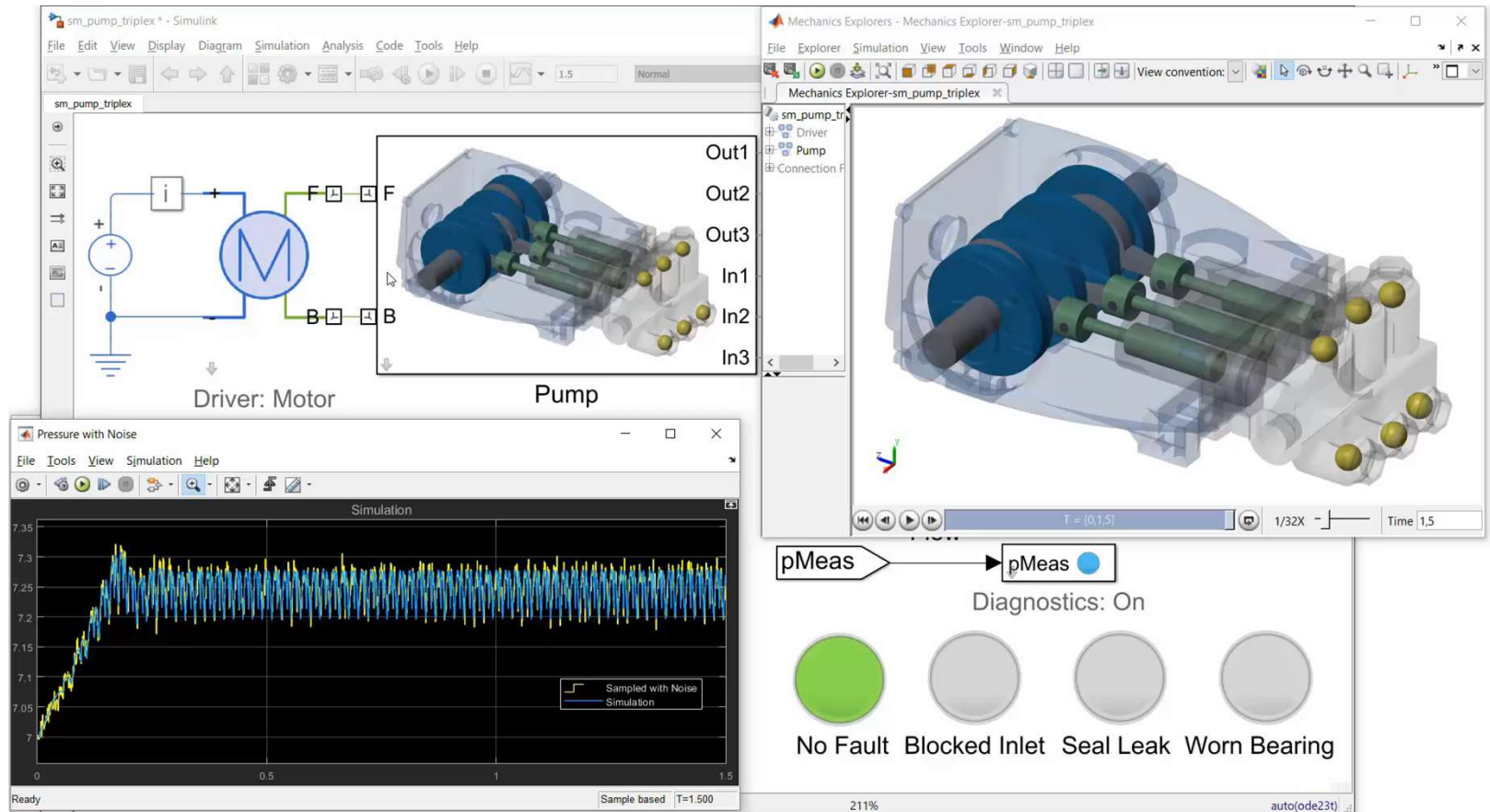
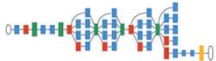
Data Augmentation/ Transformation



Labeling Automation



Import Reference Models



Automated Labeling Apps Save You Weeks to Months

LABEL AND PREPROCESS DATA

Data Augmentation/ Transformation



Labeling Automation



Import Reference Models

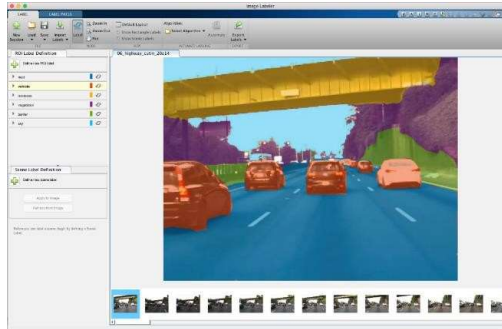


Image Labeler app to label images manually and semi-automatically.

AUTOMOTIVE



Ground Truth
Labeler

SIGNAL PROCESSING AND COMMUNICATIONS



Audio Labeler



Signal Labeler

IMAGE PROCESSING AND COMPUTER VISION



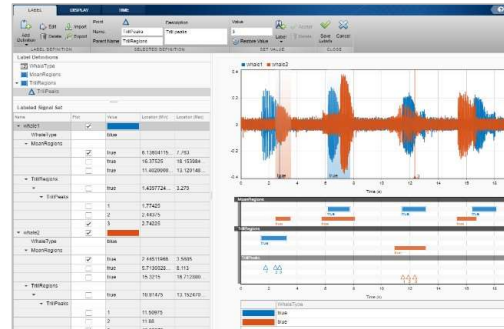
Image Labeler



Lidar Labeler



Video Labeler



Signal Labeler app to label signals manually and semi-automatically.

AUTOMOTIVE



Ground Truth
Labeler

SIGNAL PROCESSING AND COMMUNICATIONS



Audio Labeler



Signal Labeler

IMAGE PROCESSING AND COMPUTER VISION



Image Labeler



Lidar Labeler



Video Labeler



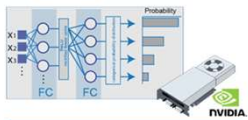
Diagnostic Feature Designer to extract time and frequency domain features from signals.



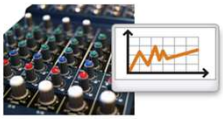
There are 2 training approaches for Deep Learning with Images

DEVELOP PREDICTIVE MODELS

Hardware-Accelerated Training



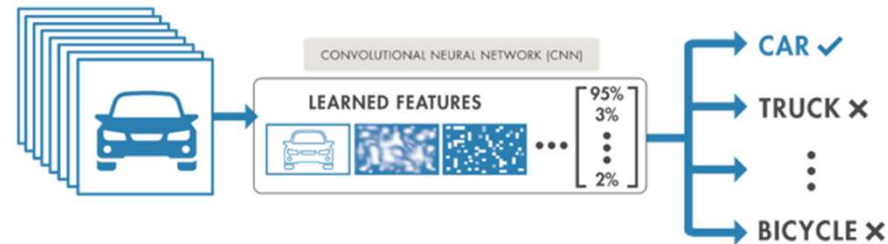
Hyperparameter Tuning



Network Visualization



Train a deep neural network from scratch



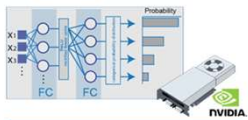
Use a pretrained model



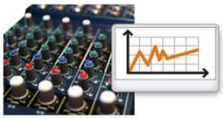
Designing a network from scratch requires some knowledge

DEVELOP PREDICTIVE MODELS

Hardware-Accelerated Training



Hyperparameter Tuning



Network Visualization



Feature Extraction - Images

- 2D and 3D convolution
- Transposed convolution (...)

Activation Functions

- ReLU
- Tanh (...)

Sequence Data

Signal, Text, Numeric

- LSTM
- BiLSTM
- Word Embedding (...)

Normalization

- Dropout
- Batch normalization
- (...)

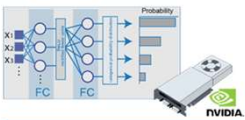
Research papers and [doc examples](#) can provide guidelines for creating architecture.



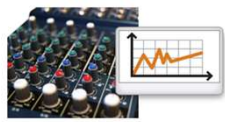
Start with pre-built models and existing examples

DEVELOP PREDICTIVE
MODELS

Hardware-Accelerated
Training



Hyperparameter Tuning



Network Visualization



Use pre-built models

Image classification models

AlexNet, GoogLeNet, VGG, SqueezeNet,
ShuffleNet, ResNet, DenseNet, Inception...

Reference examples

Object detection

Vehicles, pedestrians, faces...

Semantic segmentation

Roadway detection, land cover classification,
tumor detection...

Signal and speech processing

Denoising, music genre recognition, keyword
spotting, radar waveform classification...

...and more...

Transfer Learning

Modify and reuse pre-built models

[Get started with Transfer Learning](#)



Don't Reinvent



Perfect It

AlexNet

VGG-16

VGG-19

GoogLeNet

ResNet-18

ResNet-101

ResNet-50

Inception-v3

DenseNet-201

Xception

SqueezeNet

MobileNet-v2

ShuffleNet

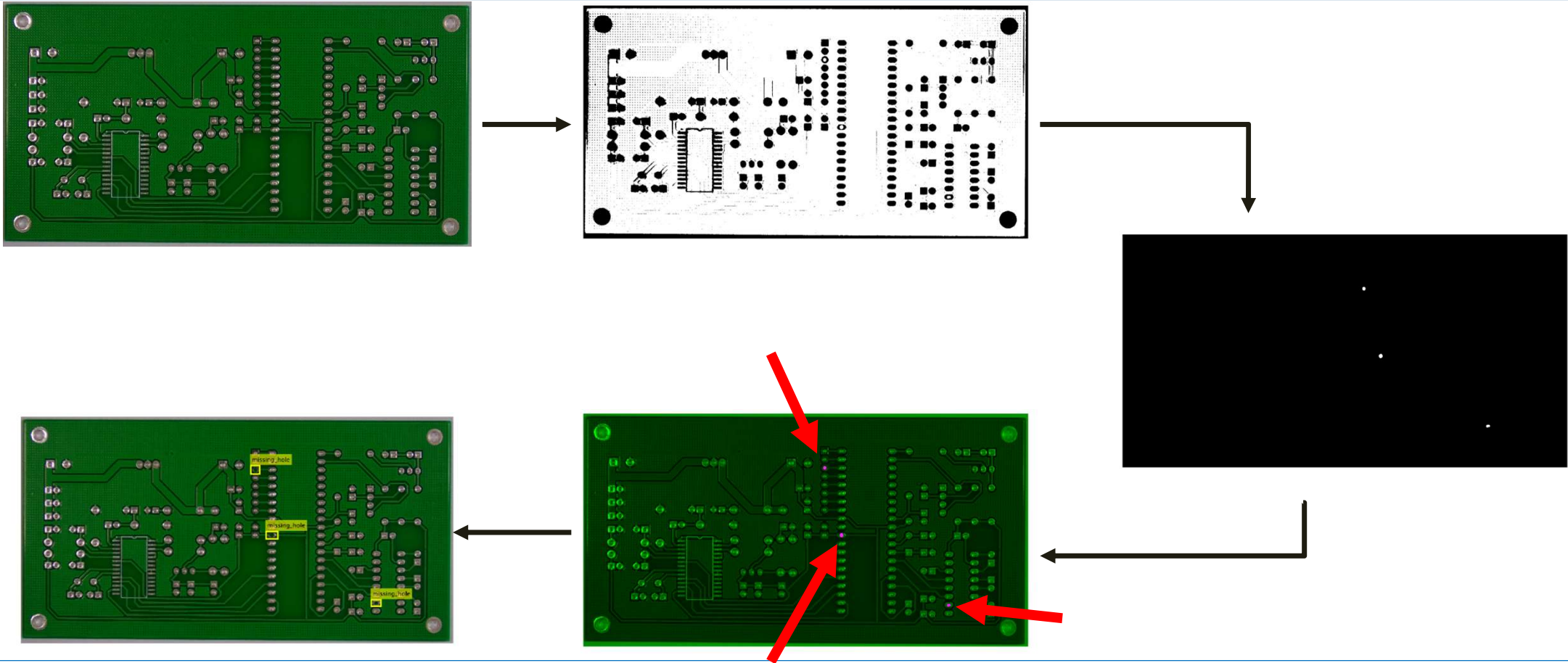
*Get started
with these
Models*

*Effective for object
detection and semantic
segmentation workflows*

*Lightweight and
computationally
efficient*



Data Preparation Demo



Data Preparation Demo – Pre-recorded Video

Live Editor - C:\Users\kvenulap\Documents\Large\GitLab_Internal\IP_Core\pcb-defection-detection-using-yolox\Part01_DataPreparation.mlx

PCB Defect Detection
Part 1: Data Preparation
Copyright © 2024 MathWorks, Inc.

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

This example shows how to detect, localize, and classify defects in printed circuit boards (PCBs) using a YOLOX object detector.
PCBs contain individual electronic devices and their connections. Defects in PCBs can result in poor performance or product failures. By detecting defects in PCBs, production lines can remove faulty PCBs and ensure that electronic devices are of high quality.

Configuration
Make sure we run this as a project.

```
1 try
2   prj = currentProject;
3 catch
4   open("PCBDefectDetection.prj");
5   OpenPart1;
6   prj = currentProject;
7 end
```

Download PCB Defect Data Set
This example uses the PCB defect data set [1][2]. The data set contains 1,386 images of PCB elements with synthesized defects. The data has six types of defect: missing hole, mouse bite, open circuit, short, spur, and spurious copper. Each image contains multiple defects of the same category in different locations. The data set contains bounding box and coordinate information for every defect in every image. The size of the data set is 1.87 GB.

Missing Hole
Mouse Bite
Short
Spur

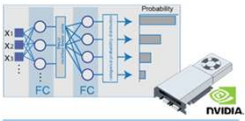
Zoom: 100% | UTF-8 | LF | script



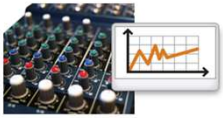
AI modeling Apps automate training, tuning, visualization...

DEVELOP PREDICTIVE MODELS

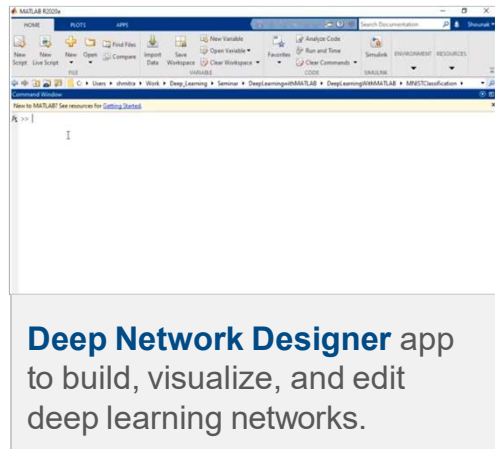
Hardware-Accelerated Training



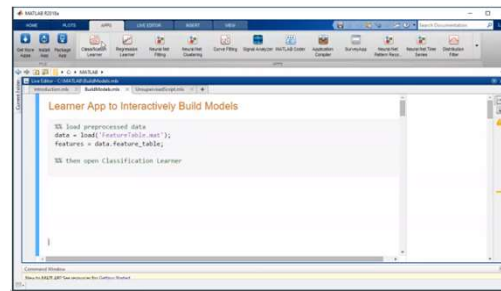
Hyperparameter Tuning



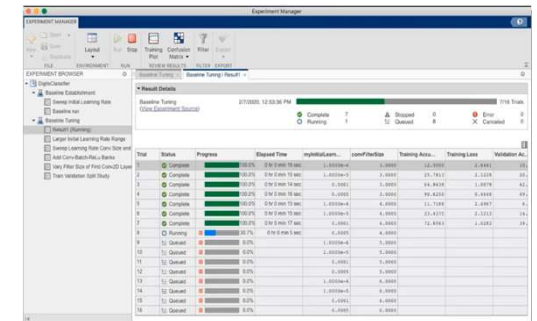
Network Visualization



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



Learner app to try different classifiers and find the best fit for data sets.



Experiment Manager app to run deep learning experiments to train networks and compare results.

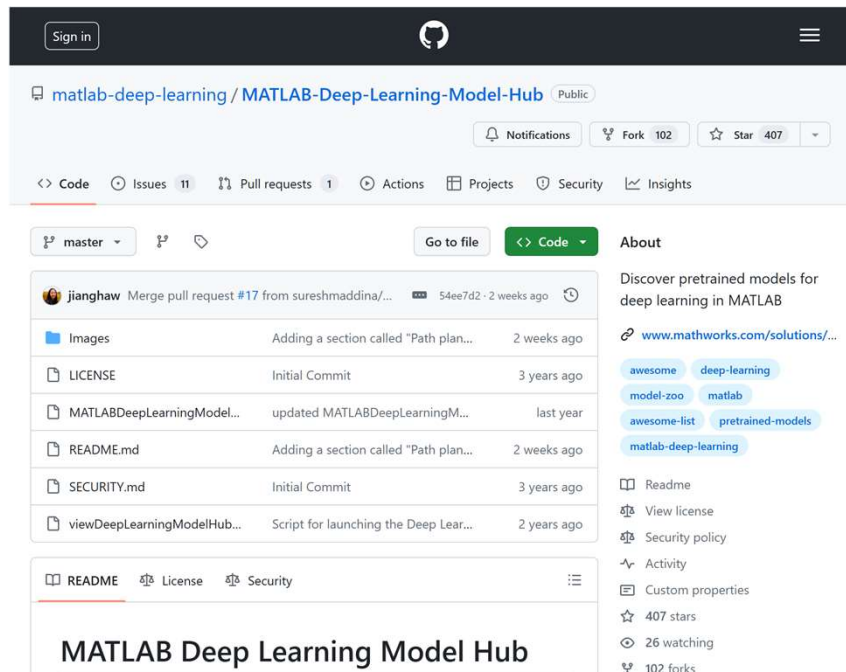


Access pretrained models

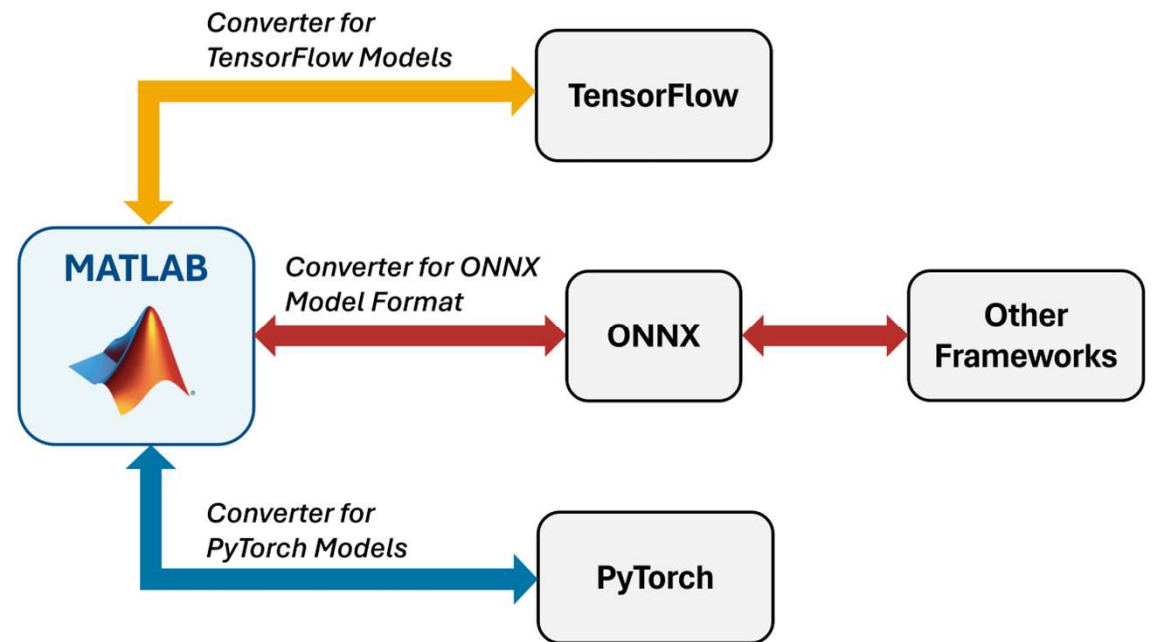
Take advantage of the knowledge provided by pretrained networks to learn new patterns in new data

Find one directly in MATLAB

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



Import it from other platforms



Modeling Demo

Detect Defects on Printed Circuit Boards Using YOLOX Network

The screenshot shows the MathWorks Help Center page for the example "Detect Defects on Printed Circuit Boards Using YOLOX Network". The page is structured with a left sidebar for navigation, a main content area with text and code blocks, and a right sidebar for additional resources.

Navigation Sidebar:

- CONTENTS
 - Documentation Home
 - AI and Statistics
 - Deep Learning Toolbox
 - Applications
 - Image Processing and Computer Vision
 - Computer Vision
 - Detect Defects on Printed Circuit Boards Using YOLOX Network**
 - ON THIS PAGE
 - Download Pretrained YOLOX Detector
 - Download PCB Defect Data Set
 - Perform Object Detection
 - Prepare Data for Training
 - Define YOLOX Object Detector Network Architecture
 - Specify Training Options
 - Train Detector
 - Evaluate Detector
 - References
 - See Also
 - Related Topics

Main Content Area:

Detect Defects on Printed Circuit Boards Using YOLOX Network

This example shows how to detect, localize, and classify defects in printed circuit boards (PCBs) using a YOLOX object detector. PCBs contain individual electronic devices and their connections. Defects in PCBs can result in poor performance or product failures. By detecting defects in PCBs, production lines can remove faulty PCBs and ensure that electronic devices are of high quality.

Download Pretrained YOLOX Detector

By default, this example downloads a pretrained version of the YOLOX object detector [1] using the `downloadTrainedNetwork` helper function. The helper function is attached to this example as a supporting file. You can use the pretrained network to run the entire example without waiting for training to complete.

```
trainedPCBDefectDetectorNet_url = "https://ssd.mathworks.com/supportfiles/" + ...  
    "vision/data/trainedPCBDefectDetectorYOLOX.zip";  
downloadTrainedNetwork(trainedPCBDefectDetectorNet_url,pwd);  
load("trainedPCBDefectDetectorYOLOX.mat");
```

Download PCB Defect Data Set

This example uses the PCB defect data set [2] [3]. The data set contains 1,386 images of PCB elements with synthesized defects. The data has six types of defect: missing hole, mouse bite, open circuit, short, spur, and spurious copper. Each image contains multiple defects of the same category in different locations. The data set contains bounding box and coordinate information for every defect in every image. The size of the data set is 1.87 GB.

Specify `dataDir` as the location of the data set. Download the data set using the `downloadPCBDefectData` helper function. This function is attached to the example as a supporting file.

```
dataDir = fullfile(tempdir,"PCBDefects");  
downloadPCBDefectData(dataDir)
```

Perform Object Detection

Read a sample image from the dataset.

```
sampleImage = imread(fullfile(dataDir,"PCB-DATASET-master","images", ...  
    "Missing_hole","01_missing_hole_01.jpg"));
```

Warning: Division by zero when processing CompressedBitsPerPixel. The value has been set to NaN.

Predict the bounding boxes, labels, and class-specific confidence scores for each bounding box by using the `detect` (Computer Vision Toolbox) function.

```
[bboxes,scores,labels] = detect(detector,sampleImage);
```

Display the results.

```
imshow(sampleImage)  
showShape("rectangle".bboxes,Label=labels);
```

Right Sidebar:

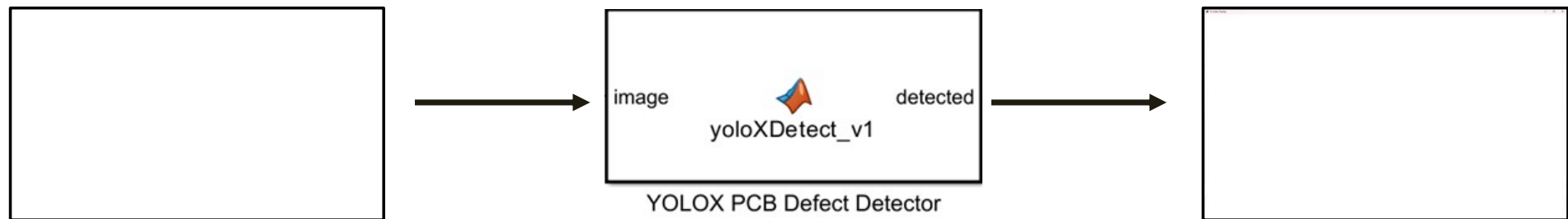
This example uses:

- Computer Vision Toolbox
- Deep Learning Toolbox
- Computer Vision Toolbox Automated Visual Inspection Library
- Image Processing Toolbox

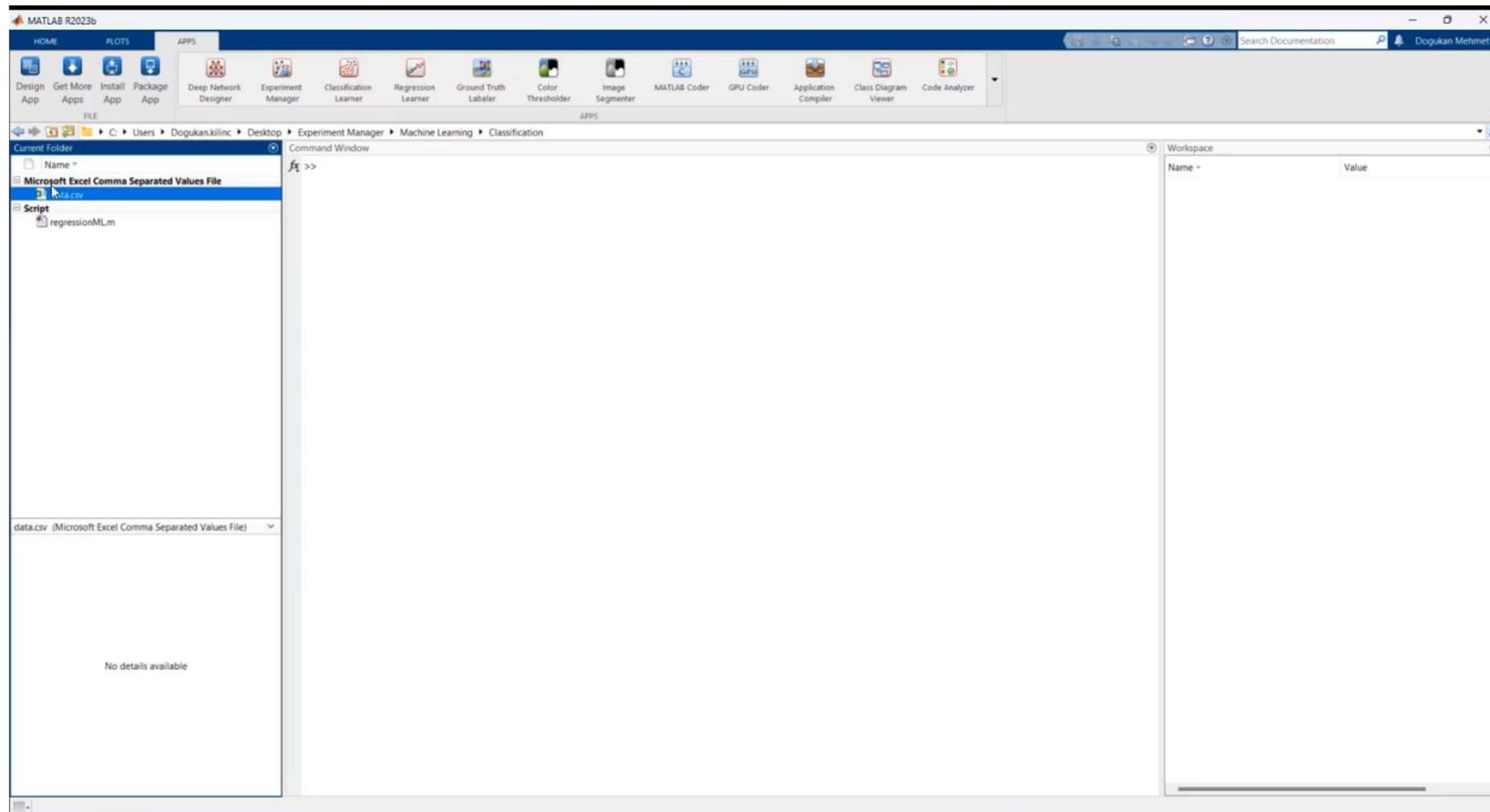
[Copy Command](#)



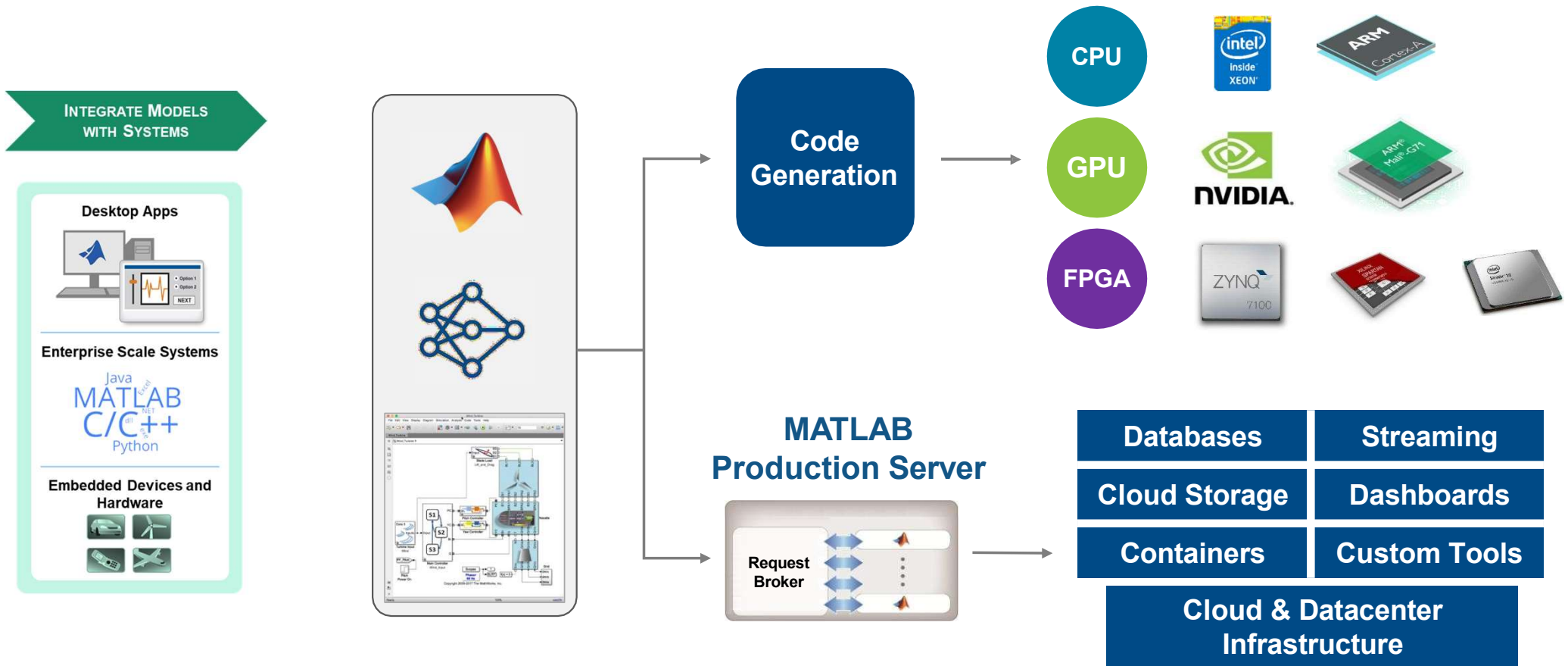
Simulation & Test Demo



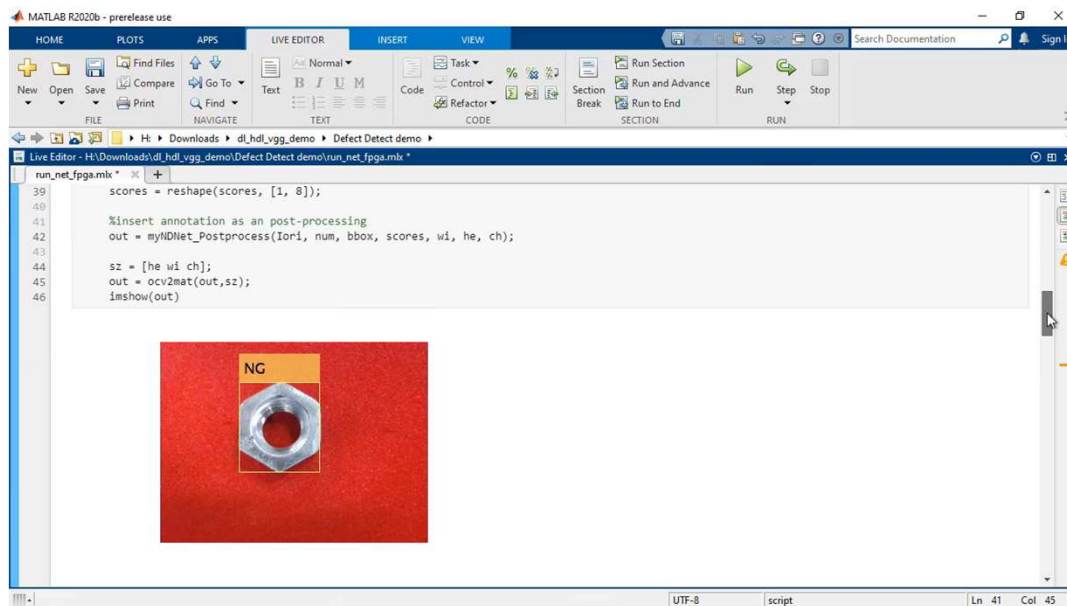
MATLAB Experiment Manager after Deep Learning



Deploy to Any Processor with Best-in-class Performance



Example: Deployment Algorithm to Microprocessor



Deploy defect detection algorithms from MATLAB
to ZCU102 board from Xilinx

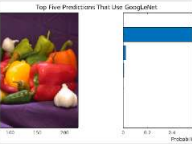


Deploy defect detection algorithms from MATLAB
to Jetson AGX Xavier




Deep Learning Summary

Deep Learning Code Generation — Examples




Code Generation for Deep Learning Networks

Perform code generation for an image classification application that uses deep learning. It uses the codegen command to generate a



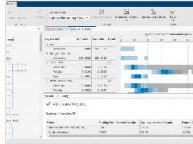
Code Generation for Semantic Segmentation Network

Code generation for an image segmentation application that uses deep learning. It uses the codegen command to generate a MEX



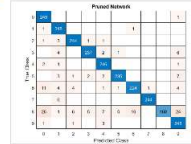
Lane Detection Optimized with GPU Coder

Develop a deep learning lane detection application that runs on NVIDIA® GPUs.



Quantize Residual Network Trained for Image Classification and Gener...

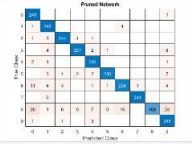
Quantize the learnable parameters in the convolution layers of a deep learning neural network that has residual connections and has been



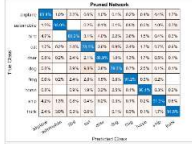
Parameter Pruning and Quantization of Image Classification Network

Use parameter pruning and quantization to reduce network size.


Quantization and Pruning



Parameter Pruning and Quantization of Image Classification Network

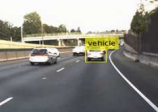


Prune Image Classification Network Using Taylor Scores



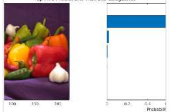
Prune Filters in a Deep Network Using Taylor

GPU Code Generation from MATLAB Applications



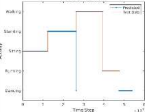
Code Generation for Object Detection Using YOLO v3 Deep Learning

Generate CUDA® MEX for a you only look once (YOLO) v3 object detector. YOLO v3 improves upon YOLO v2 by adding detection at multiple scales.




Code Generation for Deep Learning Networks

Perform code generation for an image classification application that uses deep learning. It uses the codegen command to generate a




Code Generation for a Sequence-to-Sequence LSTM Network

Demonstrates how to generate CUDA® code for a long short-term memory (LSTM) network. The example generates a MEX




Lane Detection Optimized with GPU Coder

Develop a deep learning lane detection application that runs on NVIDIA® GPUs.



Deep Learning Prediction with NVIDIA TensorRT Library

Generate code for a deep learning application by using the NVIDIA® TensorRT™ library. This example uses the codegen command to




Traffic Sign Detection and Recognition

Generate CUDA® MEX code for a traffic sign detection and recognition application that uses deep learning. Traffic sign detection and


Deep Learning Code Generation from MATLAB Applications

CPU Code Generation from MATLAB Applications




Code Generation for Deep Learning on ARM Targets

Generate and deploy code for prediction on an ARM®-based device without using a hardware support package.



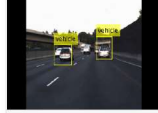
Deep Learning Prediction with ARM Compute Using codegen

Use codegen to generate code for a Logo classification application that uses deep learning on ARM® processors. The logo classification



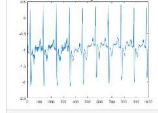
Classification with ResNet-50

Generate and deploy code for prediction on an ARM®-based device without using a hardware support package.



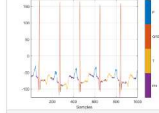
Generate C++ Code for Object Detection Using YOLO v2 and Intel MKL-DNN

Generate C++ code for the YOLO v2 Object detection network on an Intel® processor. The generated code uses the Intel Math Kernel




Deploy Signal Classifier Using Wavelets and Deep Learning on Raspberry Pi

The workflow to classify human electrocardiogram (ECG) signals using the Continuous Wavelet Transform (CWT) and a deep



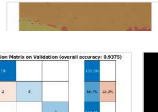
Deploy Signal Segmentation Deep Network on Raspberry Pi

Generate a MEX function and a standalone executable to perform waveform segmentation on a Raspberry Pi.




Classification with MobileNetV2

Generate and deploy code for prediction on an ARM®-based device without using a hardware support package.



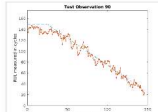
Code Generation for LSTM Network on Raspberry Pi

Generate code for a pretrained long short-term memory network to predict Remaining Useful Life (RUL) of a machine.




Code Generation for LSTM Network That Uses Intel MKL-DNN

Generate code for a pretrained LSTM network that makes predictions for each step of an input timeseries.



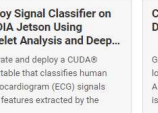
Cross Compile Deep Learning Code for ARM Neon Targets

Generate library or executable code on host computer for deployment on ARM hardware target.



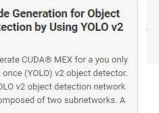
Deep Learning Prediction on ARM Mali GPU

Use the cnncoder function to generate code for an image classification application that uses deep learning on ARM® Mali GPUs.




Deploy Signal Classifier on NVIDIA Jetson Using Wavelet Analysis and Deep...

Generate and deploy a CUDA® executable that classifies human electrocardiogram (ECG) signals using features extracted by the




Code Generation for Object Detection by Using YOLO v2

Generate CUDA® MEX for a you only look once (YOLO) v2 object detector. A YOLO v2 object detection network is composed of two subnetworks. A




Lane Detection Optimized with GPU Coder

Develop a deep learning lane detection application that runs on NVIDIA® GPUs.




Logo Recognition Network

Code generation for a logo classification application that uses deep learning. It uses a pretrained network called LogoNet and



Code Generation for Denoising Deep Neural Network

Generate CUDA® MEX from MATLAB® code and denoise grayscale images by using the denoising convolutional neural

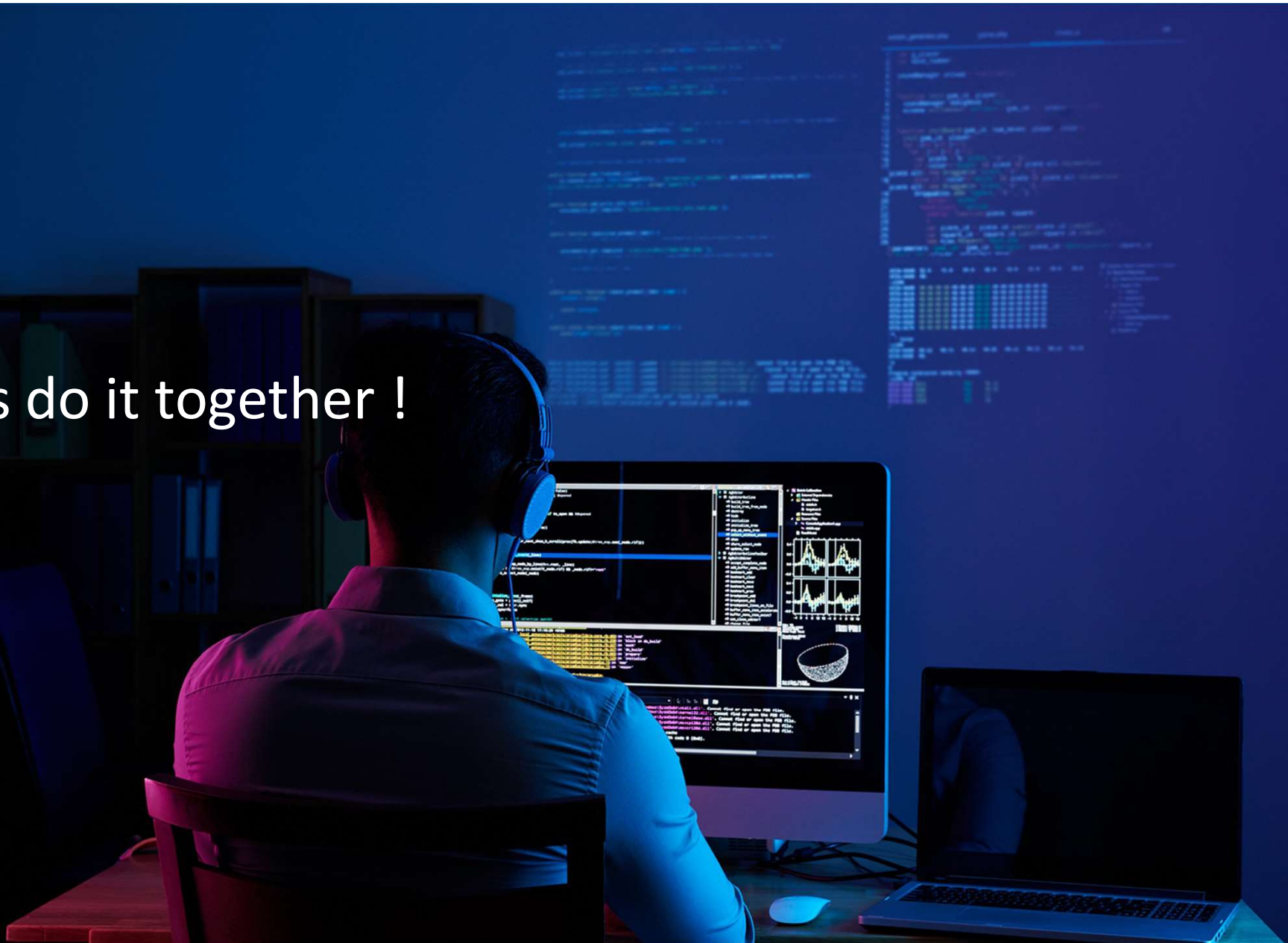
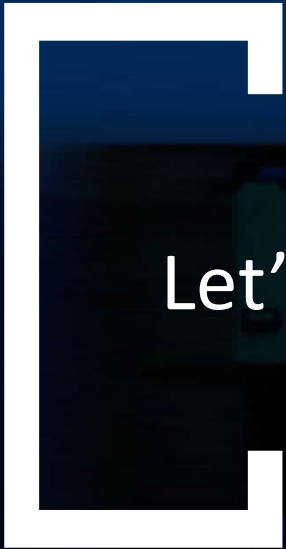


Code Generation for Semantic Segmentation Network

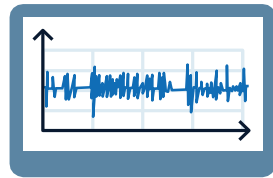
Code generation for an image segmentation application that uses deep learning. It uses the codegen command to generate a MEX



Let's do it together !



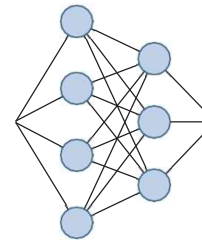
Signal Acquisition



Collect
Signals



Time-
Frequency
Image



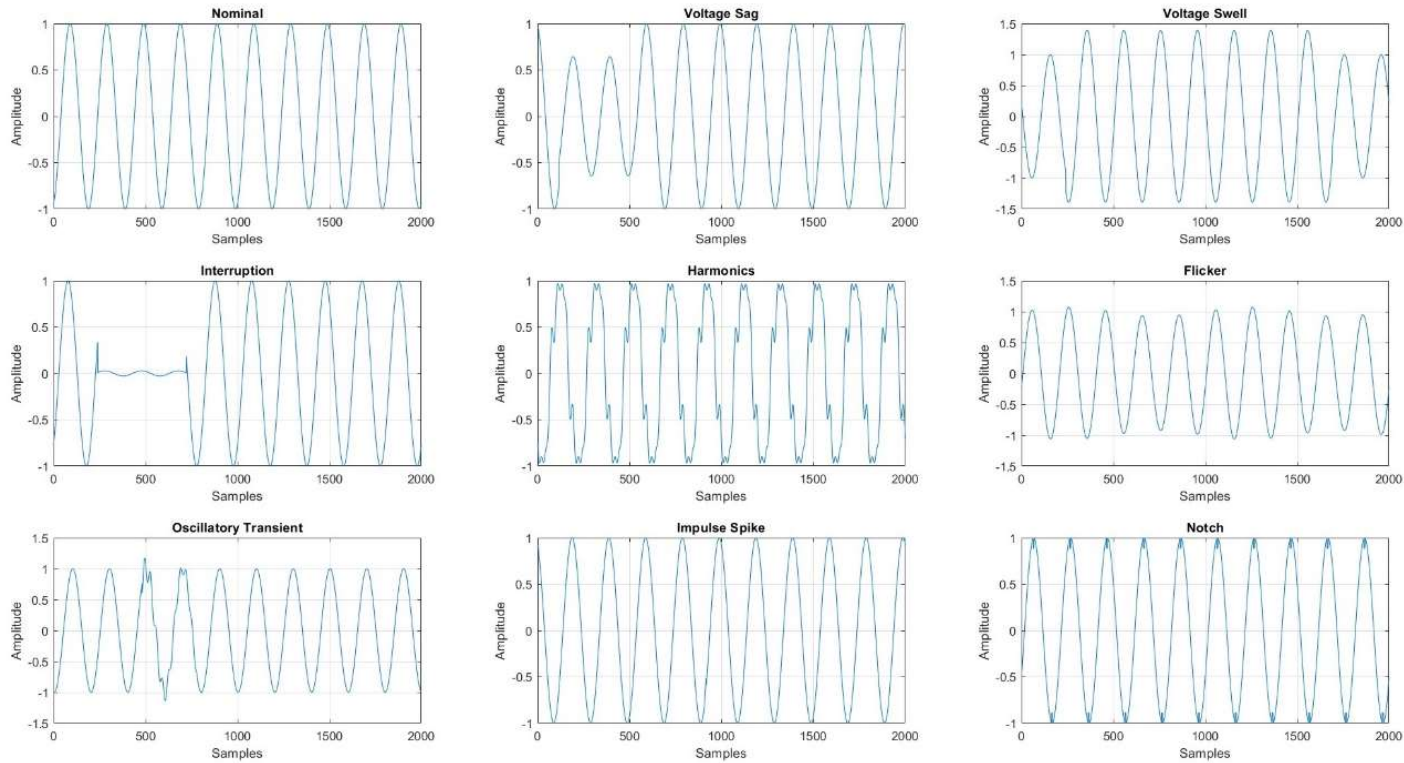
Pretrained
Network



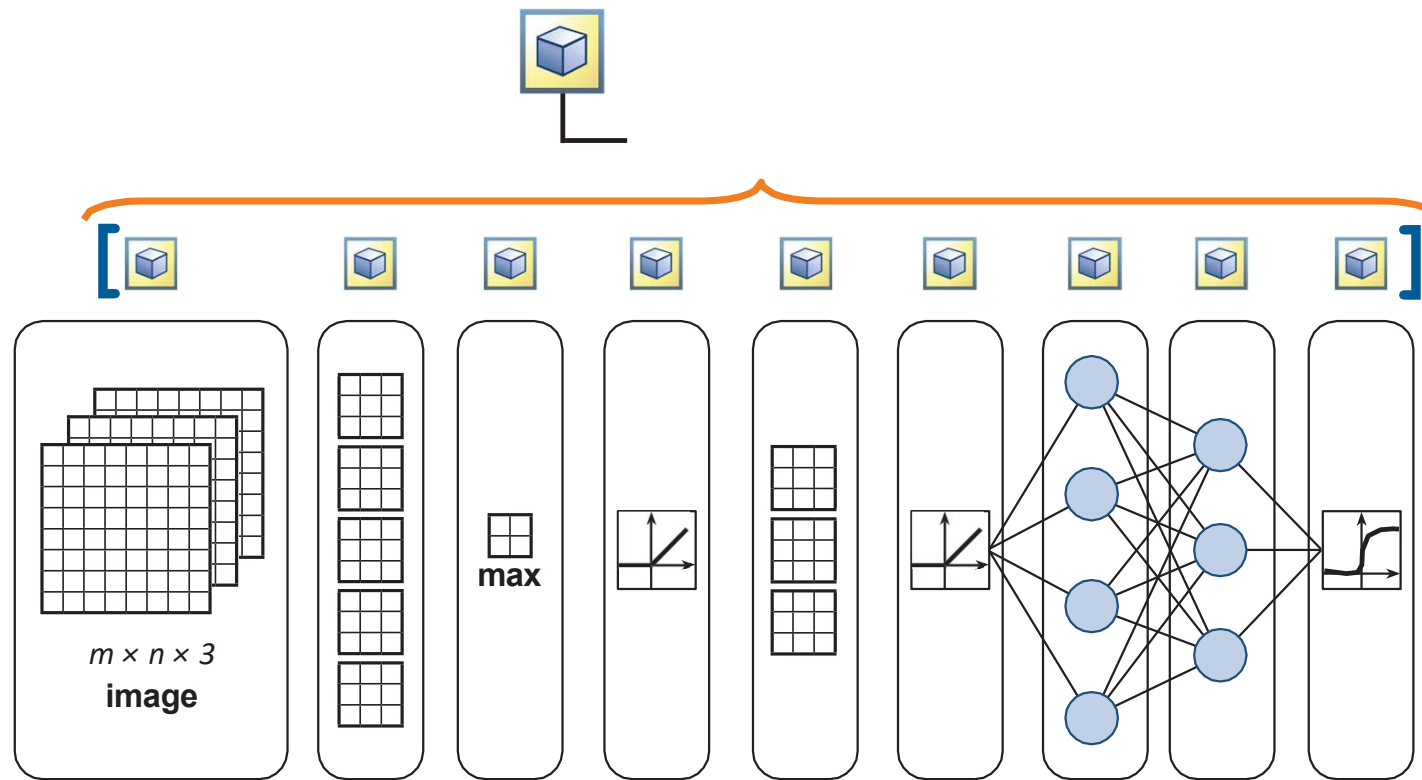
User
ID



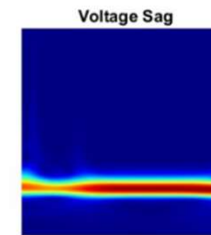
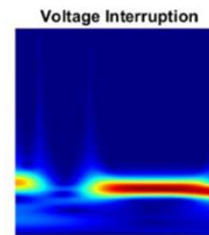
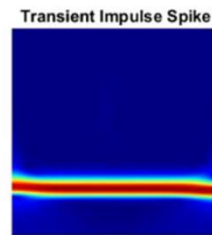
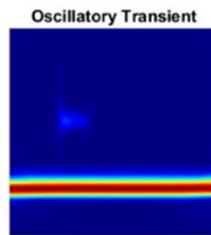
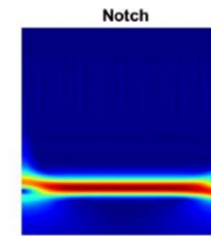
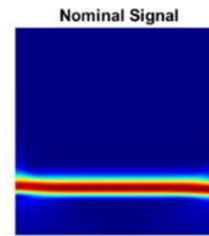
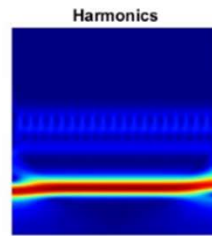
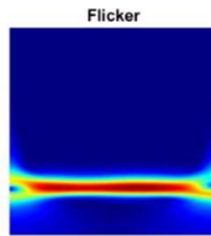
Preprocessing Signals for Classification



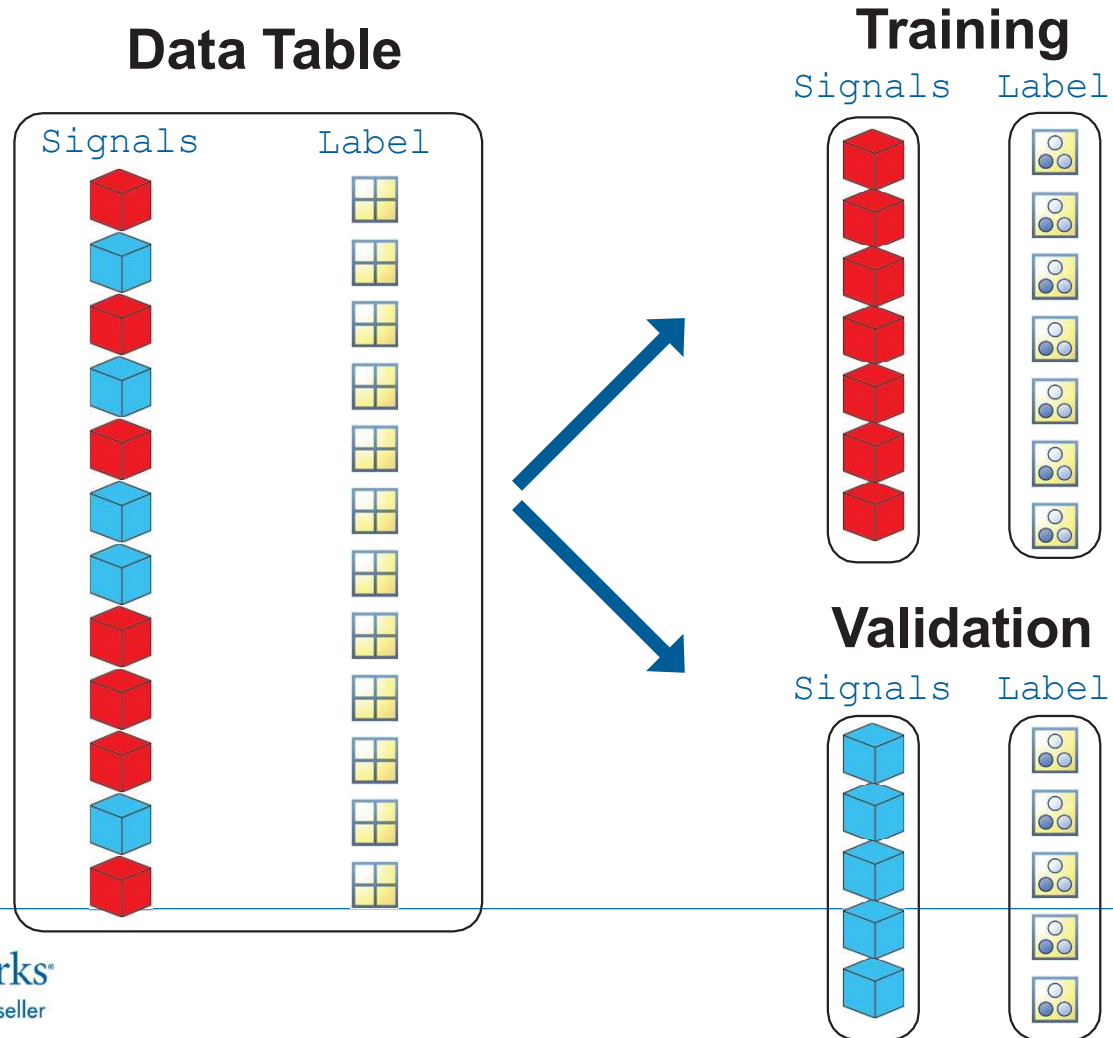
Pretrained Convolutional Neural Networks



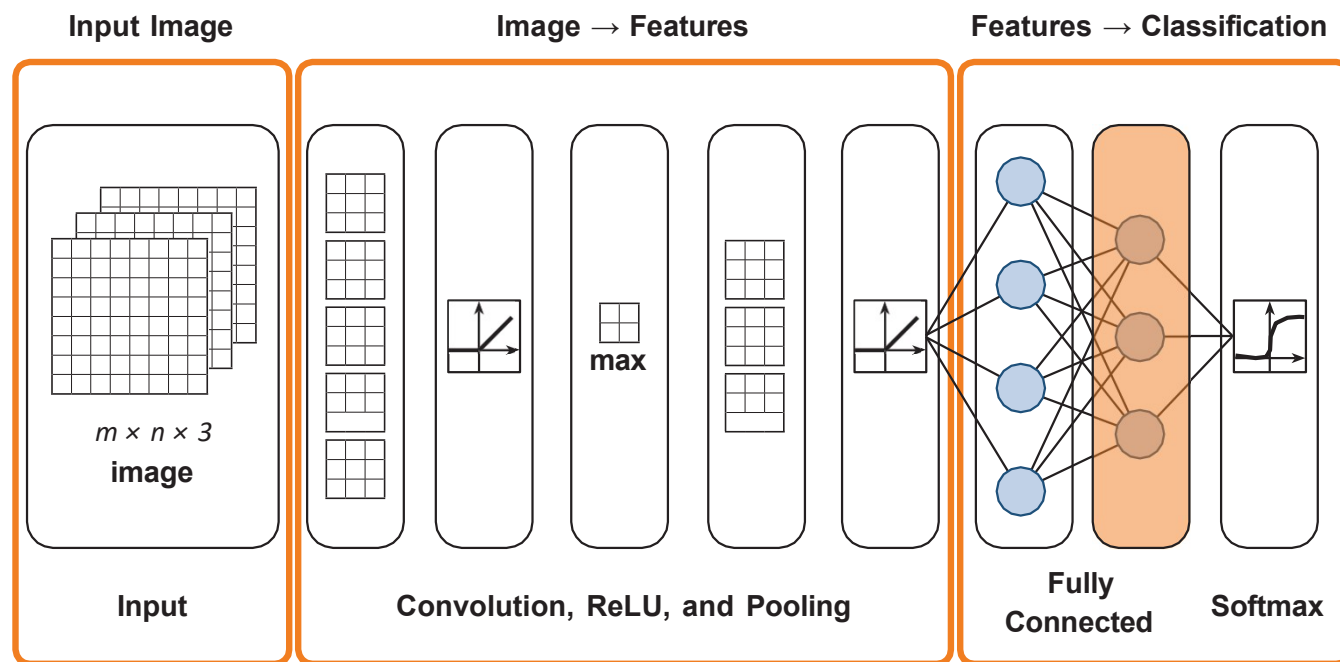
Converting Signals to Images



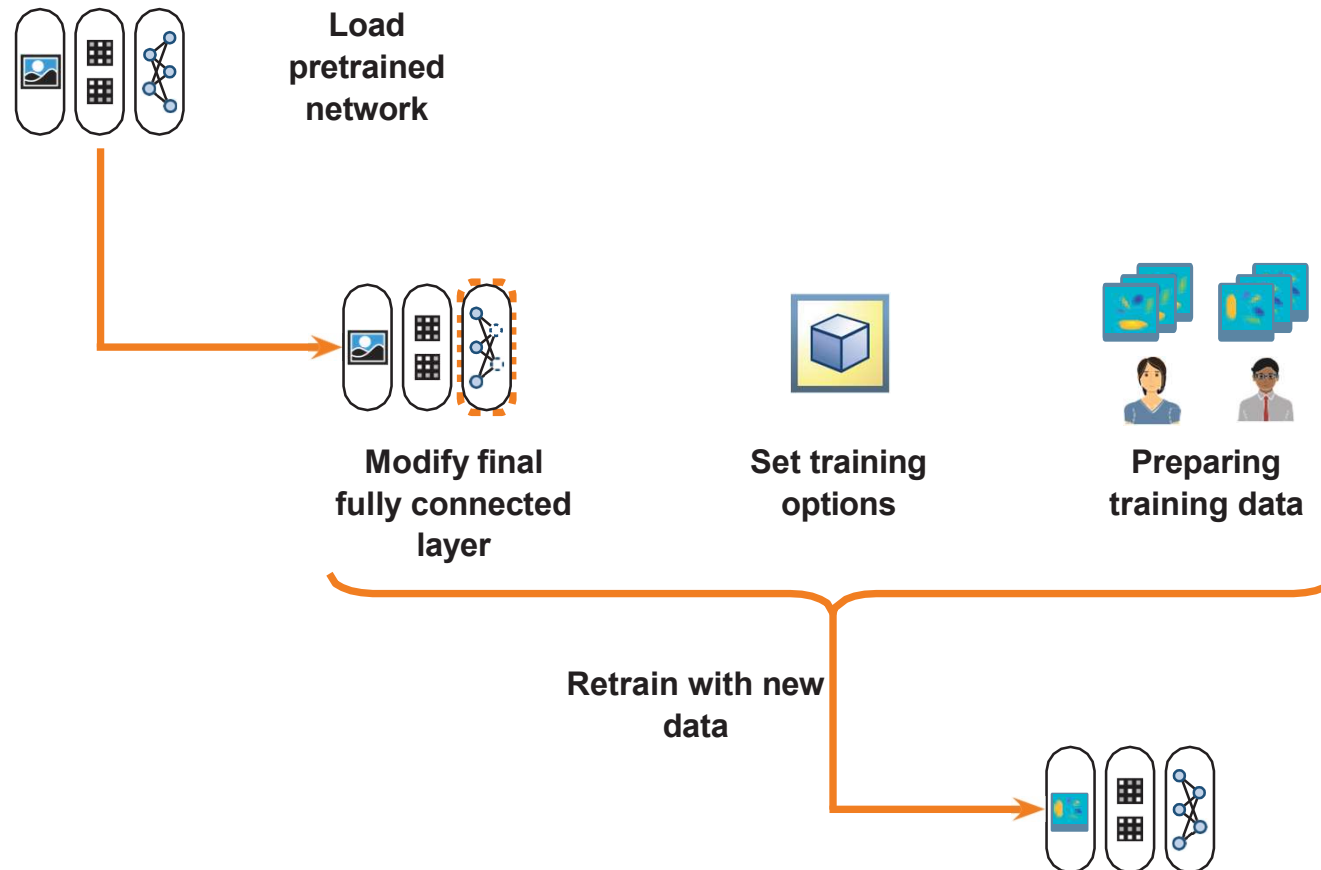
Extracting Training and Validation Data



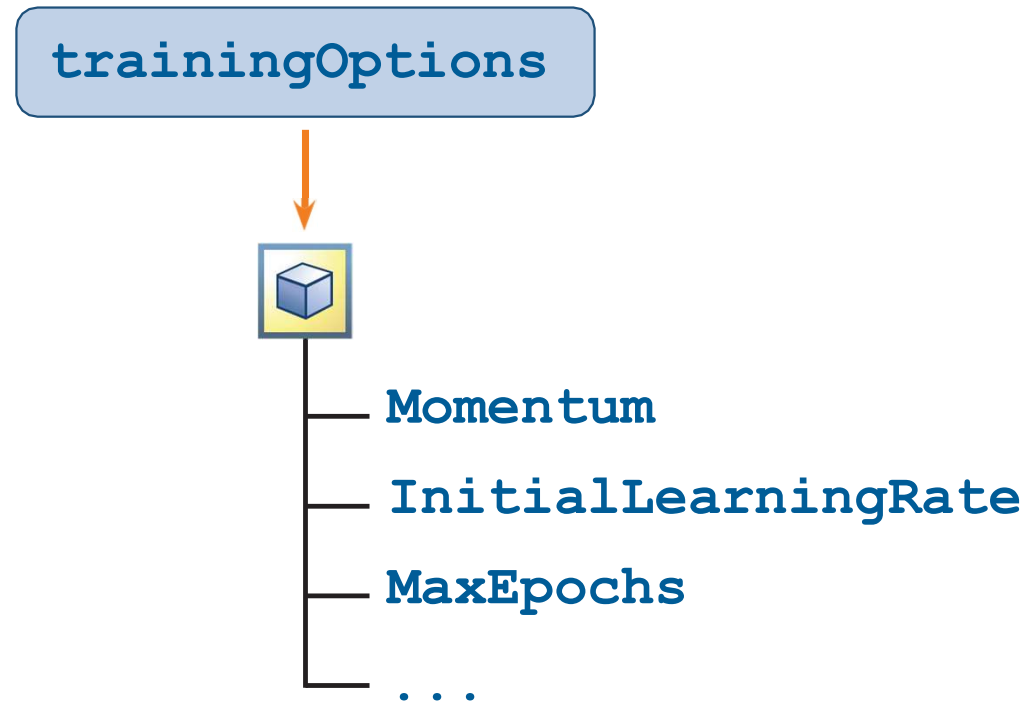
Modifying a Pretrained Network



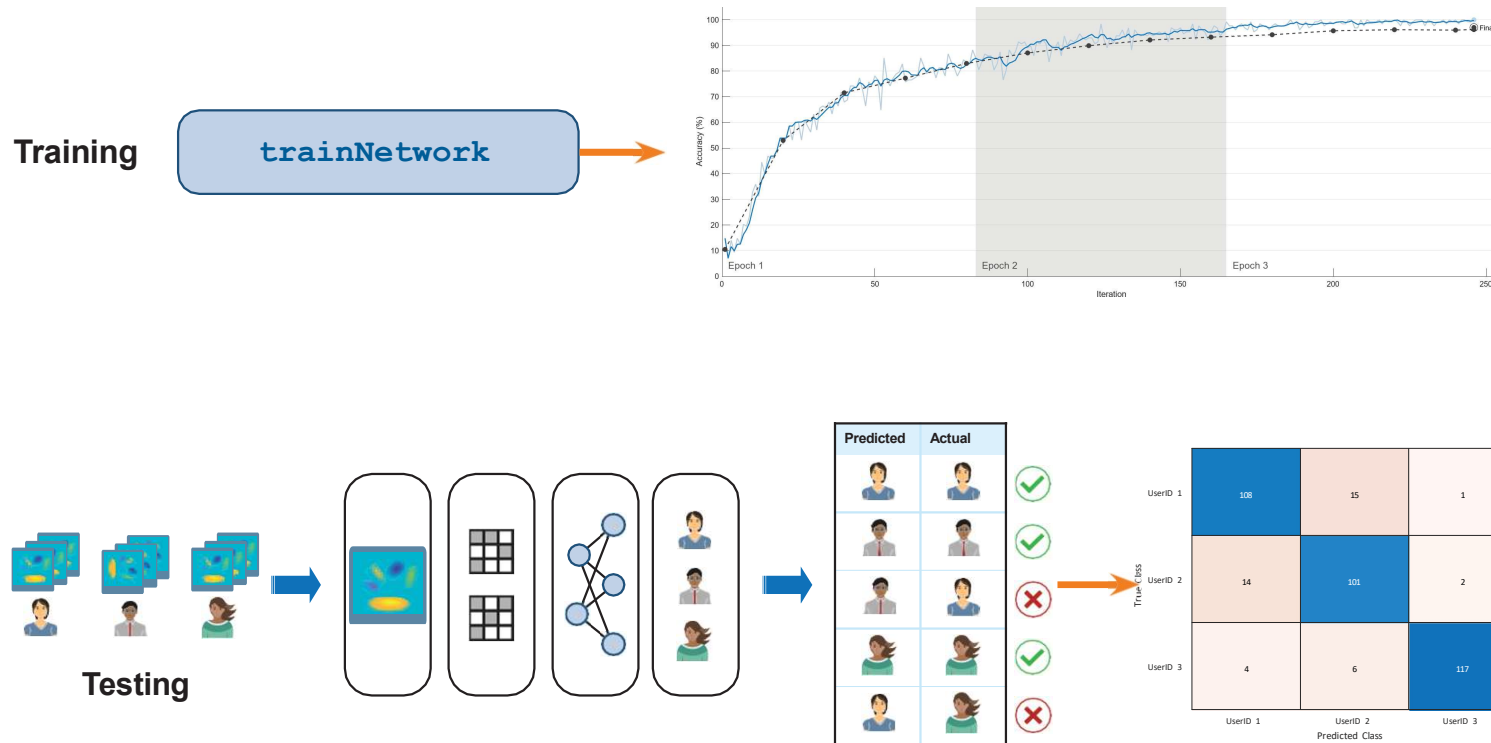
Transfer Learning



Setting Training Options



Training and Evaluating the Network



Deep Learning for PQD Signals (← Click with **CTRL + Left Click**)

The image shows the MATLAB R2025a interface. The main window displays a script titled "CSV Dosyalarını Okuma" (Reading CSV Files). The script reads several CSV files from the SEED - Power Quality Disturbance Dataset and processes them using the `rows2vars` function.

CSV Dosyalarını Okuma

SEED - Power Quality Disturbance Dataset klasöründeki her bir bozulma türü ayrı CSV dosyası olarak saklanıyor.

`readtable`: CSV dosyalarını tablo (table) formatında okur.

```
1 Flicker = readtable("SEED - Power Quality Disturbance Dataset\Flicker.csv");
2 Harmonics = readtable("SEED - Power Quality Disturbance Dataset\Harmonics.csv");
3 Interruption = readtable("SEED - Power Quality Disturbance Dataset\Interruption.csv");
4 Notch = readtable("SEED - Power Quality Disturbance Dataset\Notch.csv");
5 Sag = readtable("SEED - Power Quality Disturbance Dataset\Sag.csv");
6 Swell = readtable("SEED - Power Quality Disturbance Dataset\Swell.csv");
7 Transient = readtable("SEED - Power Quality Disturbance Dataset\Transient.csv");
```

Satırları Değişkenlere Dönüştürme (rows2vars)

`rows2vars`: Tablo satırlarını sütuna dönüştürür.

Yani, her satır bir değişken haline gelir.

Daha sonra ilk sütun (RowNames) kaldırılıp sadece sayısal değerler tutulur.

```
8 Flicker_T = rows2vars(Flicker);
9 Flicker_T = Flicker_T(:, 2:301);
10
11 Harmonics_T = rows2vars(Harmonics);
12 Harmonics_T = Harmonics_T(:, 2:301);
13
```

Workspace

Name	Value	Size	Class
Flicker_T	100×300 table	100×300	table
Harmonics_T	100×300 table	100×300	table
Interruption_T	100×300 table	100×300	table
Notch_T	100×300 table	100×300	table
Sag_T	100×300 table	100×300	table
Swell_T	100×300 table	100×300	table
Transient_T	100×300 table	100×300	table

Command Window

New to MATLAB? See resources for [Getting Started](#).

>>



Code Generation for ECG Data Project

The screenshot shows the MATLAB Help Center page for the tutorial "Classify ECG Signals in Simulink Using Deep Learning". The page is titled "Classify ECG Signals in Simulink Using Deep Learning" and includes a description of the example, which uses wavelet transforms and a deep learning network to classify ECG signals. The page also features a block diagram showing the algorithmic workflow: "Apply Continuous Wavelet Transform to get wavelet coefficients from ECG signal." → "Obtain scalogram from wavelet coefficients and convert to an image of size 227x227x3." → "Predict the label and confidence by feeding the scalogram in a pre-trained SqueezeNet CNN." The page also includes a code block for opening the Simulink model: `open_system('ecg_dl_cwt\DL');`.

Classify ECG Signals in Simulink Using Deep Learning

This example shows how to use wavelet transforms and a deep learning network within a Simulink (R) model to classify ECG signals. This example uses the pretrained convolutional neural network from the *Classify Time Series Using Wavelet Analysis and Deep Learning* example of the Wavelet Toolbox™ to classify ECG signals based on images from the CWT of the time series data. For information on training, see *Classify Time Series Using Wavelet Analysis and Deep Learning* (Wavelet Toolbox).

ECG Data Description

This example uses ECG data from *PhysioNet* database. It contains data from three groups of people:

1. Persons with cardiac arrhythmia (ARR)
2. Persons with congestive heart failure (CHF)
3. Persons with normal sinus rhythms (NSR)

It includes 96 recordings from persons with ARR, 30 recordings from persons with CHF, and 36 recordings from persons with NSR. The `ecg_signals` MAT-file contains the test ECG data in time series format. The image classifier in this example distinguishes between ARR, CHF, and NSR.

Algorithmic Workflow

The block diagram for the algorithmic workflow of the Simulink model is shown.

```
graph LR; A[Apply Continuous Wavelet Transform to get wavelet coefficients from ECG signal.] --> B[Obtain scalogram from wavelet coefficients and convert to an image of size 227x227x3.]; B --> C[Predict the label and confidence by feeding the scalogram in a pre-trained SqueezeNet CNN.];
```

ECG Deep Learning Simulink Model

The Simulink model for classifying the ECG signals is shown. When the model runs, the Video Viewer block displays the classified ECG signal.

```
open_system('ecg_dl_cwt\DL');
```





TEŞEKKÜRLER



dogukan.kilinc@figes.com.tr