

# MATLAB EXPO

산업용 어플리케이션을 위한 딥러닝 기반 머신비전  
솔루션

송완빈, MathWorks



# What is Automated Optical Inspection?

*“ Automated optical inspection is the **image-based** or **visual inspection** of manufacturing parts where a camera scans the device under test for both **failures** and **quality defects**”*

## Automated Defect Detection

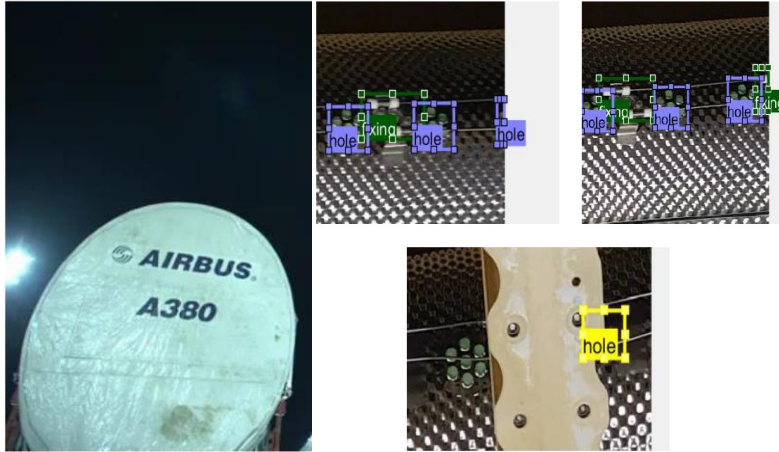
Machine Vision

Visual Inspection

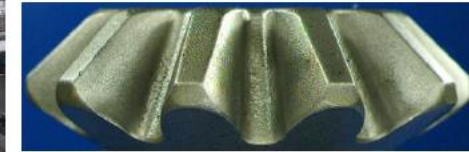
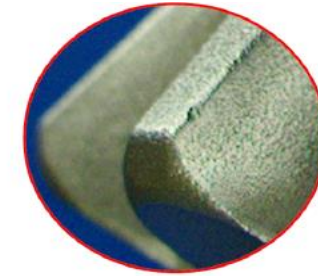
Automated Inspection

# Customer References

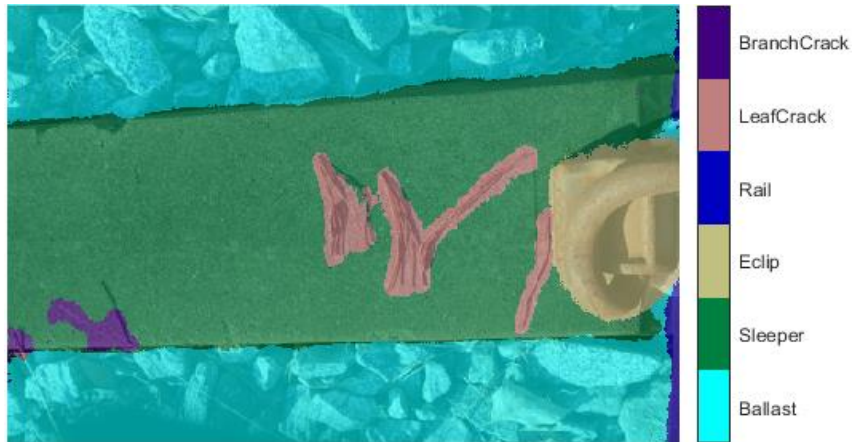
**AIRBUS**



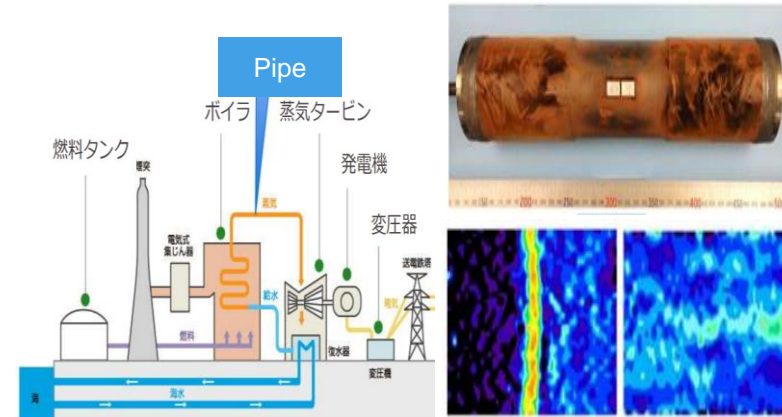
Automatic Defect Detection



Visual Inspection of Automotive Parts

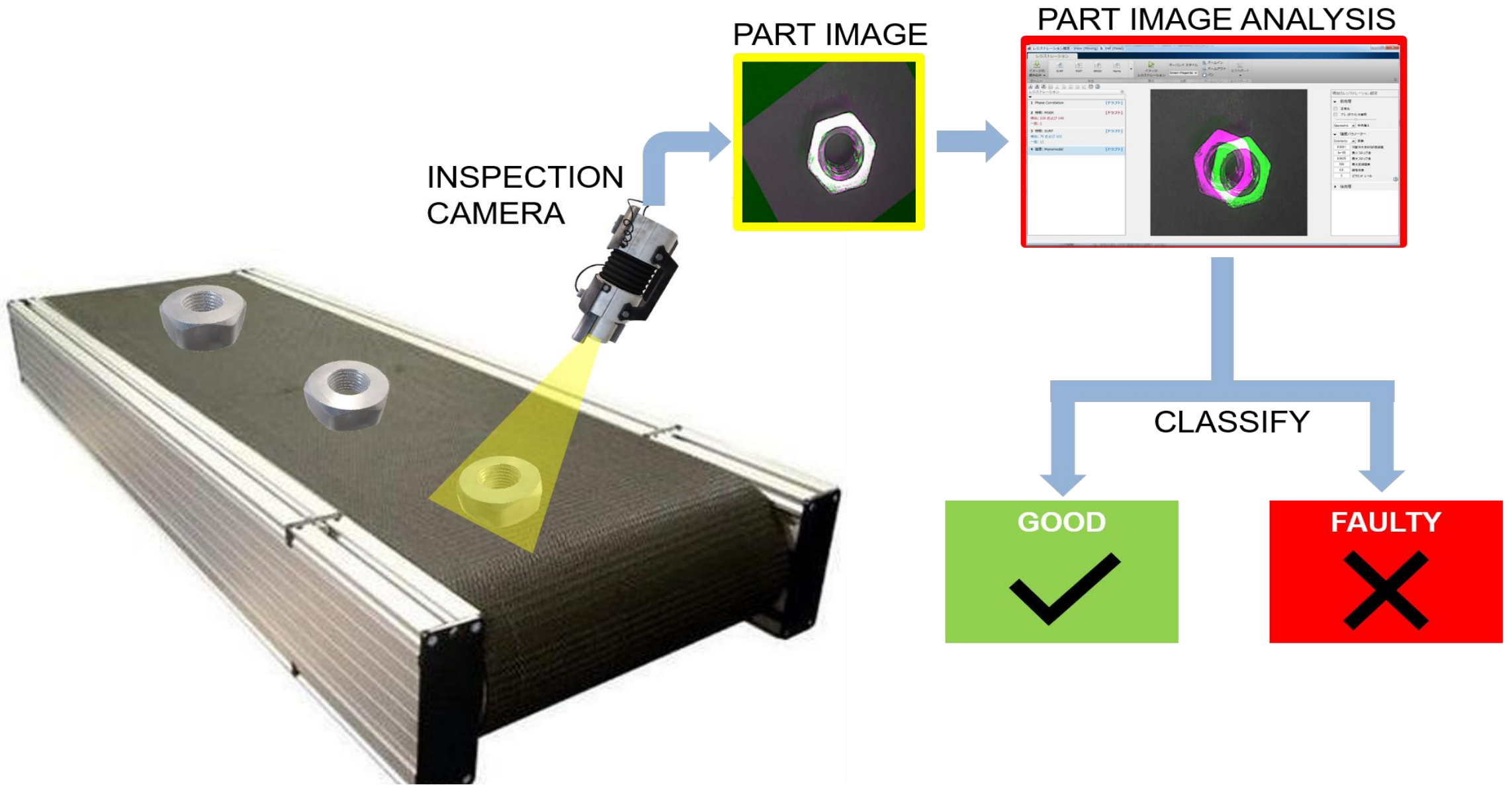


Defect Detection in Railway Components

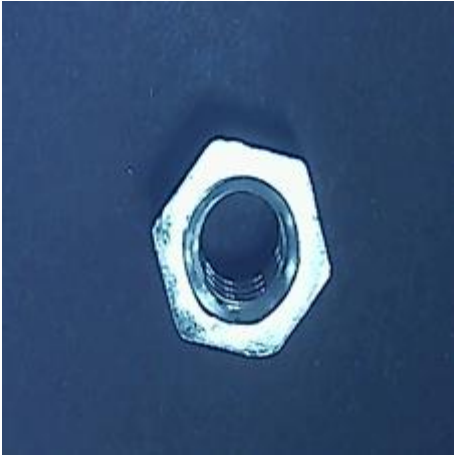


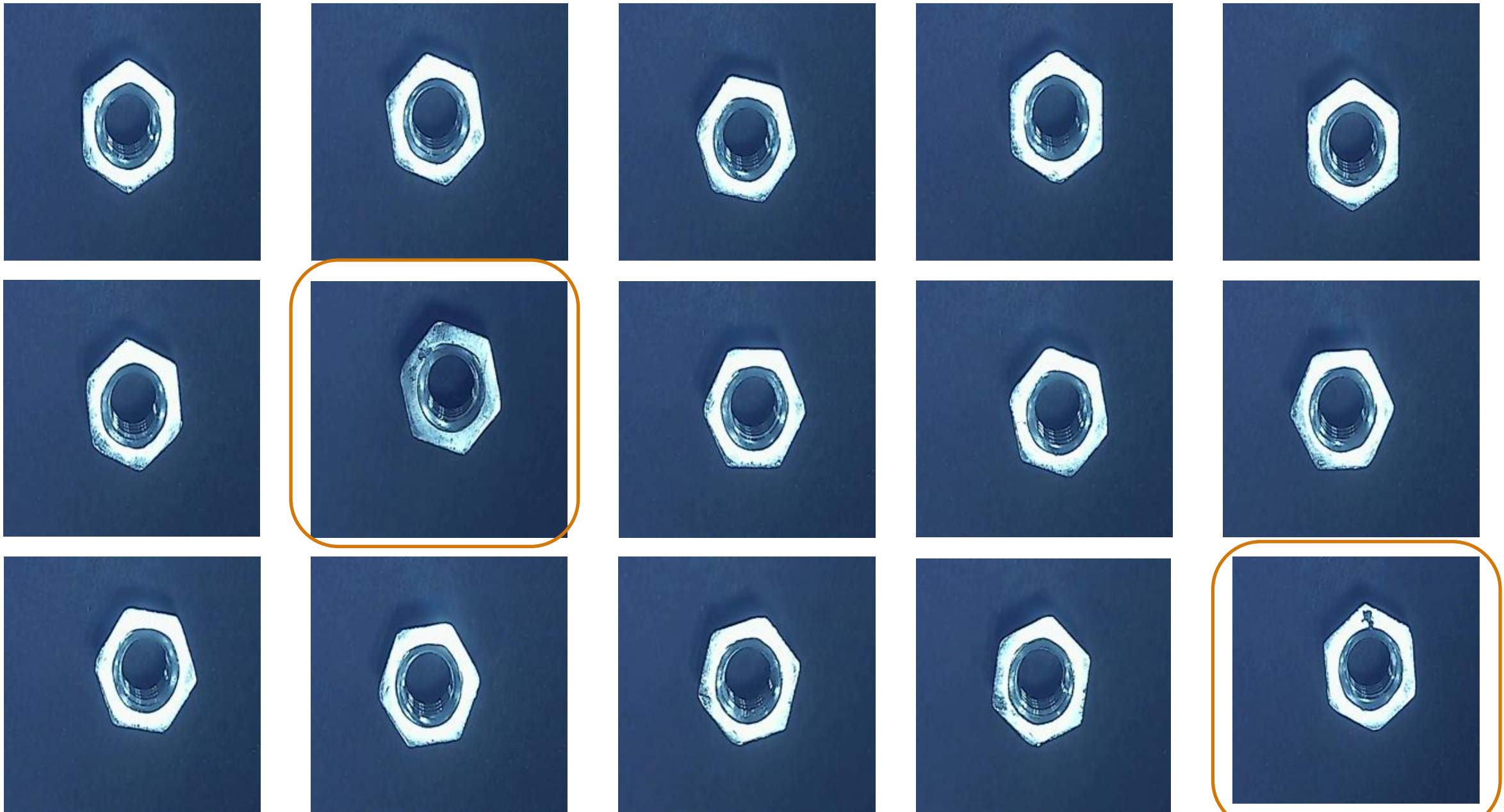
Assess Pipe Weld Damage at Power Plants

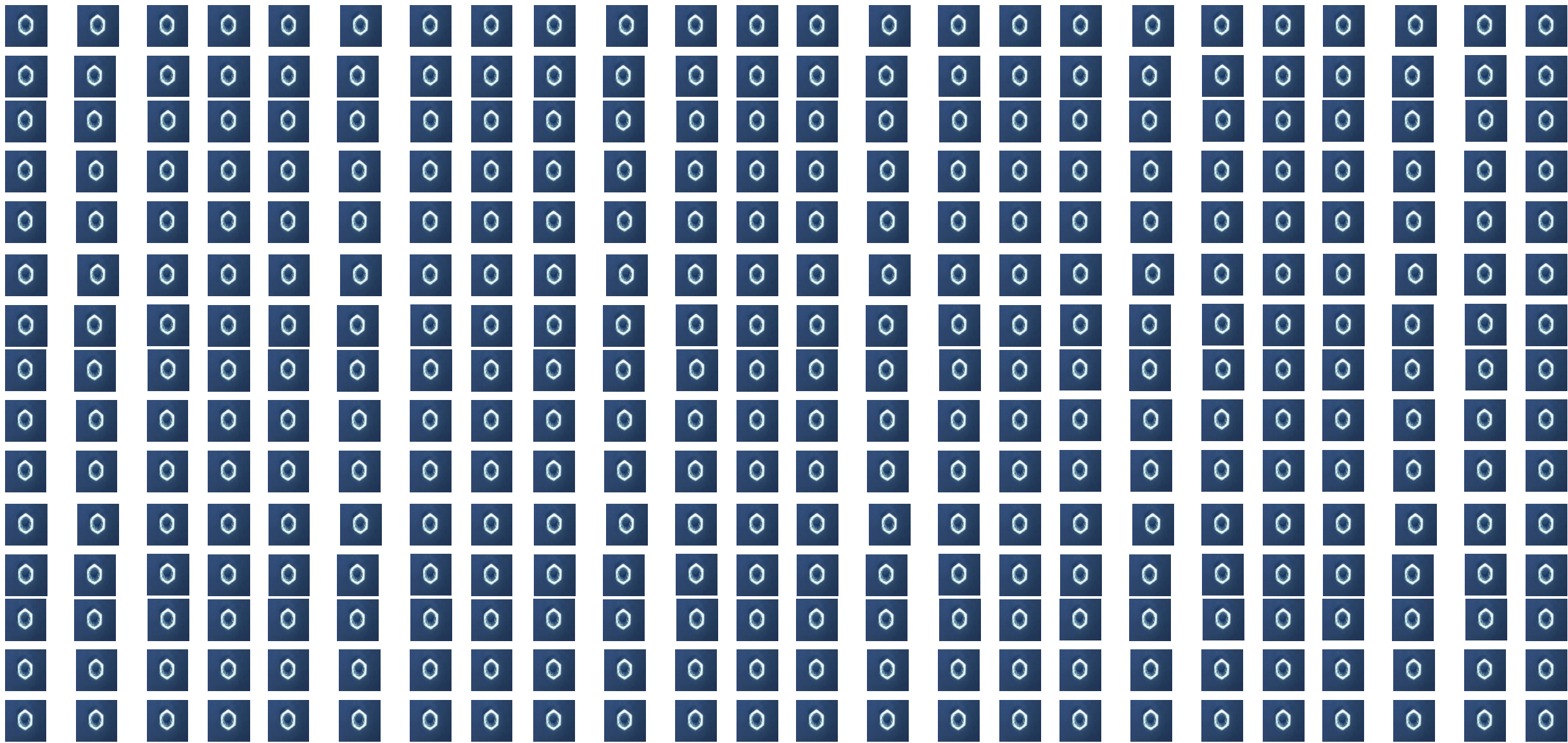


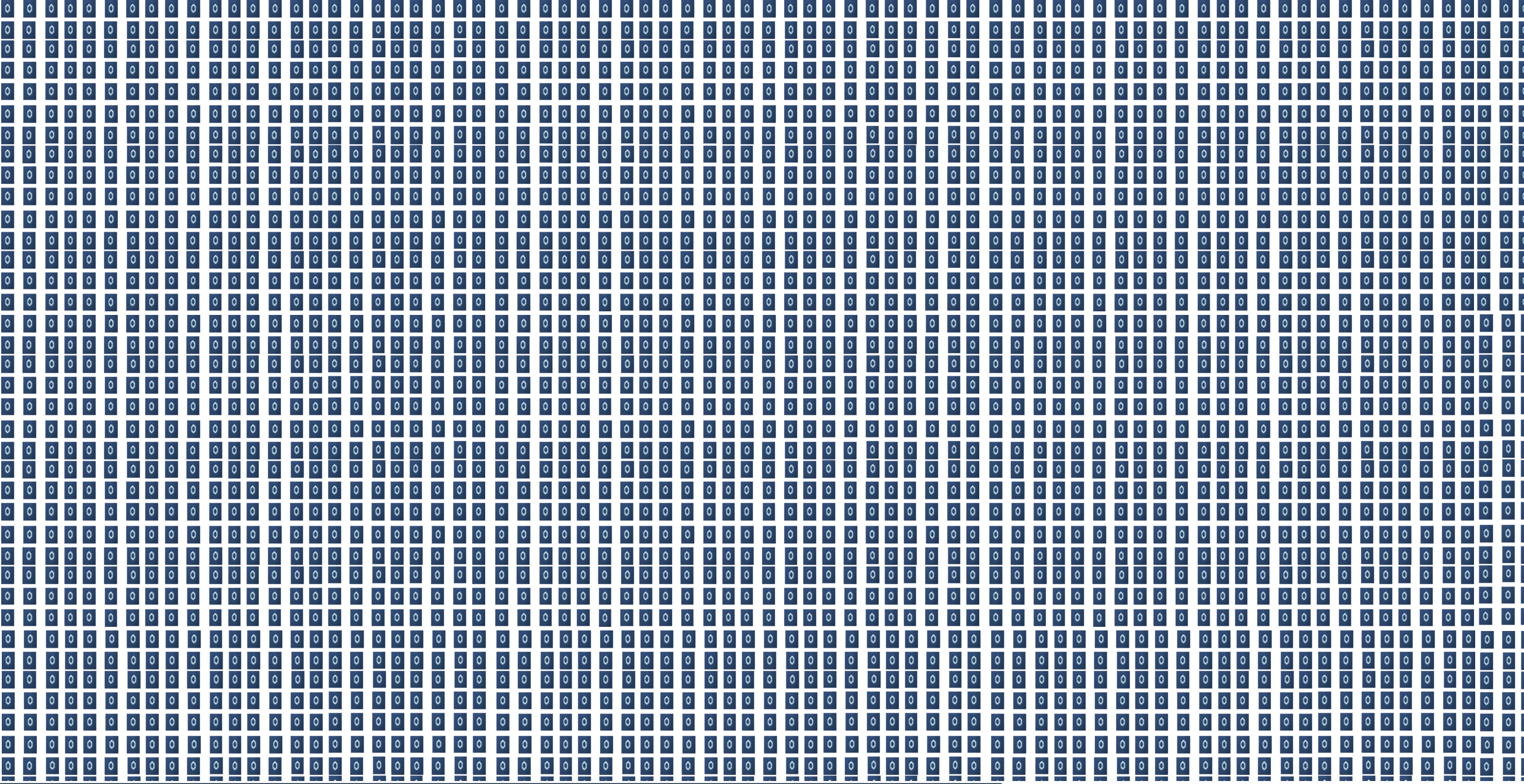


# Can you find the defective hex nut?





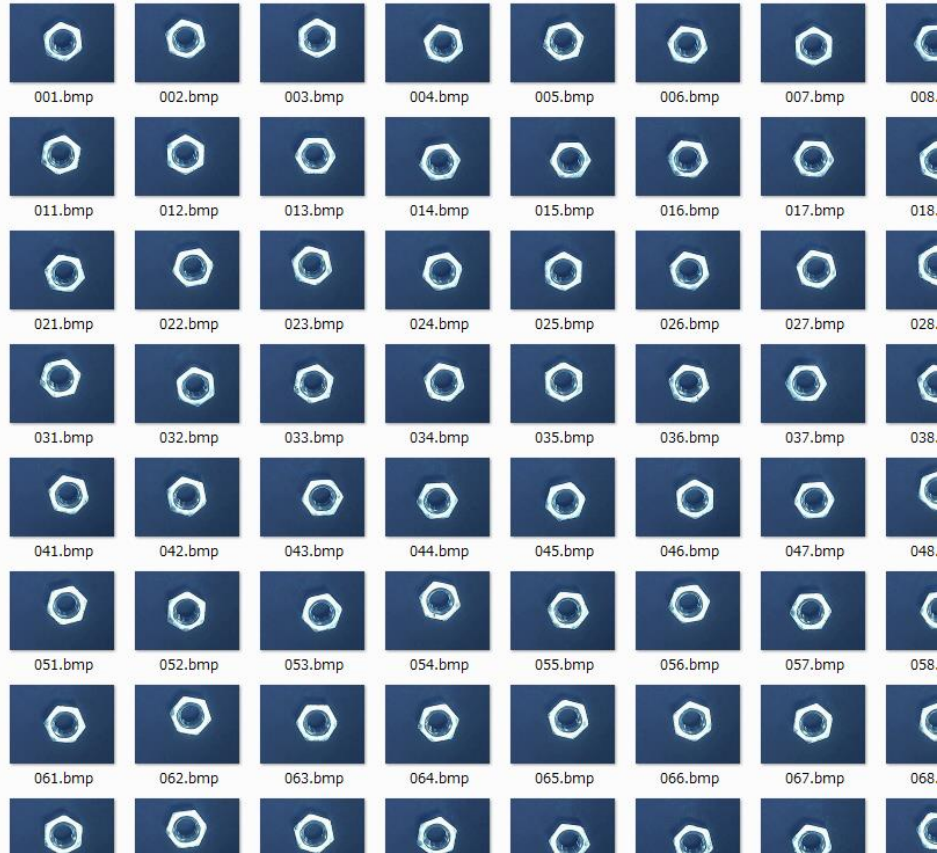




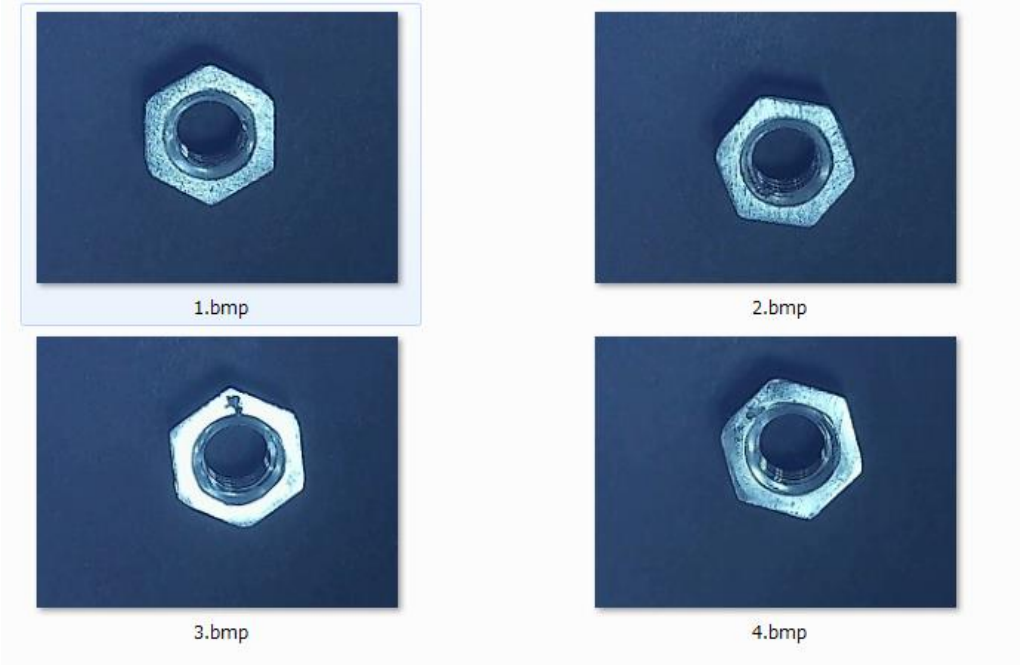


# Finding Defective Hex Nuts

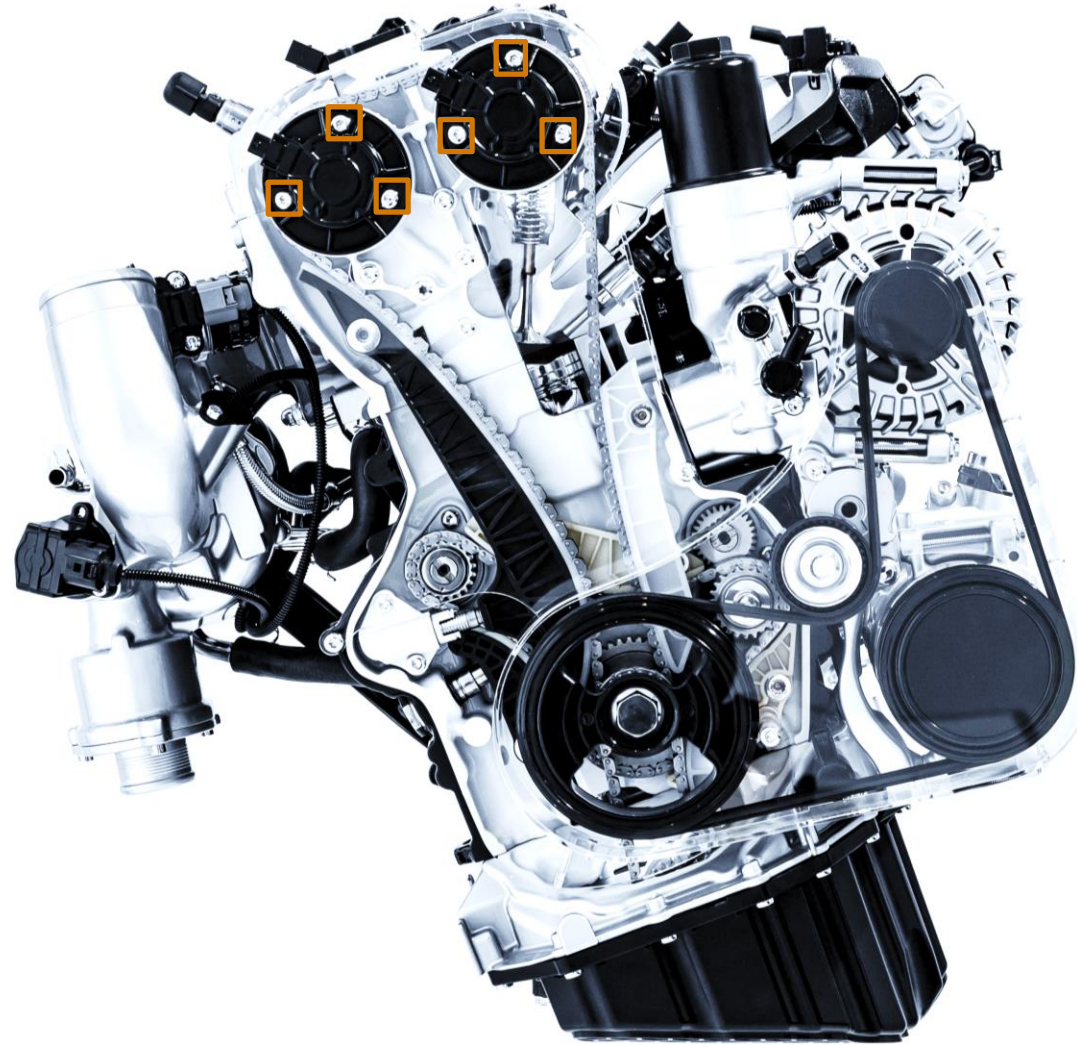
## Good



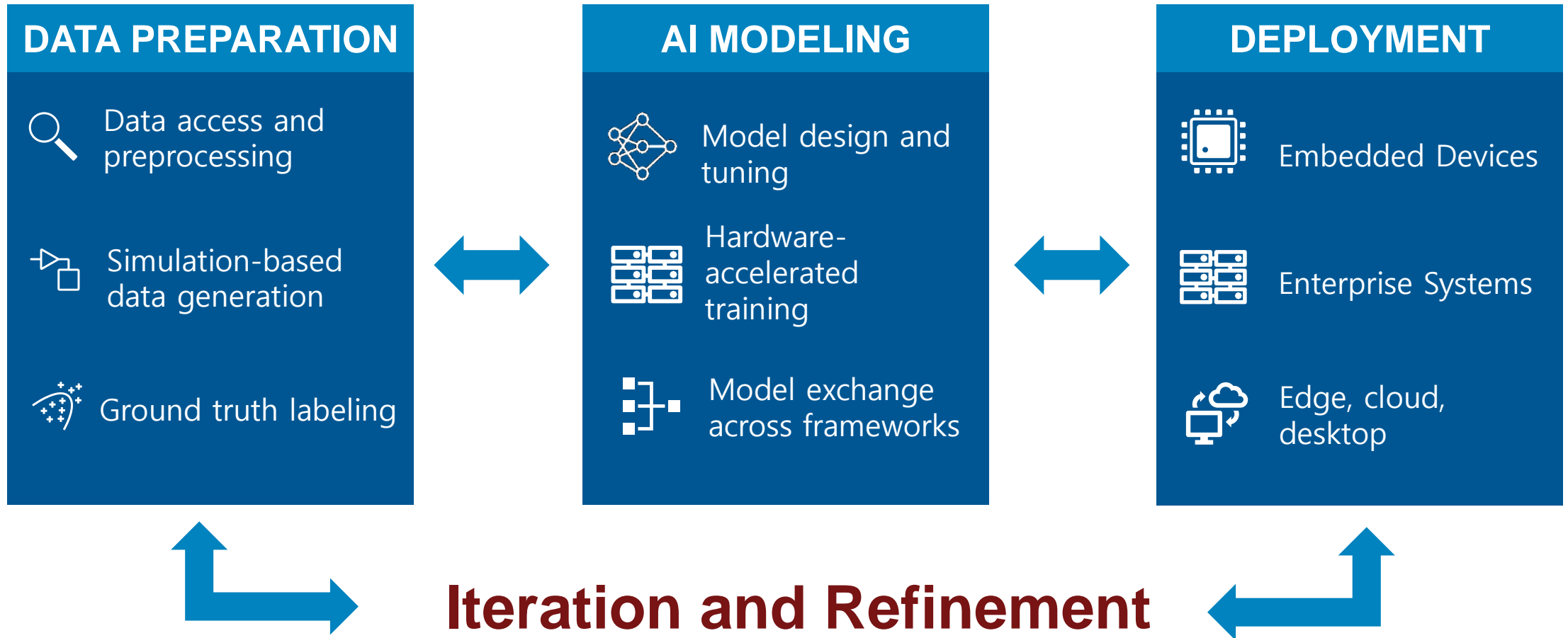
## Defective



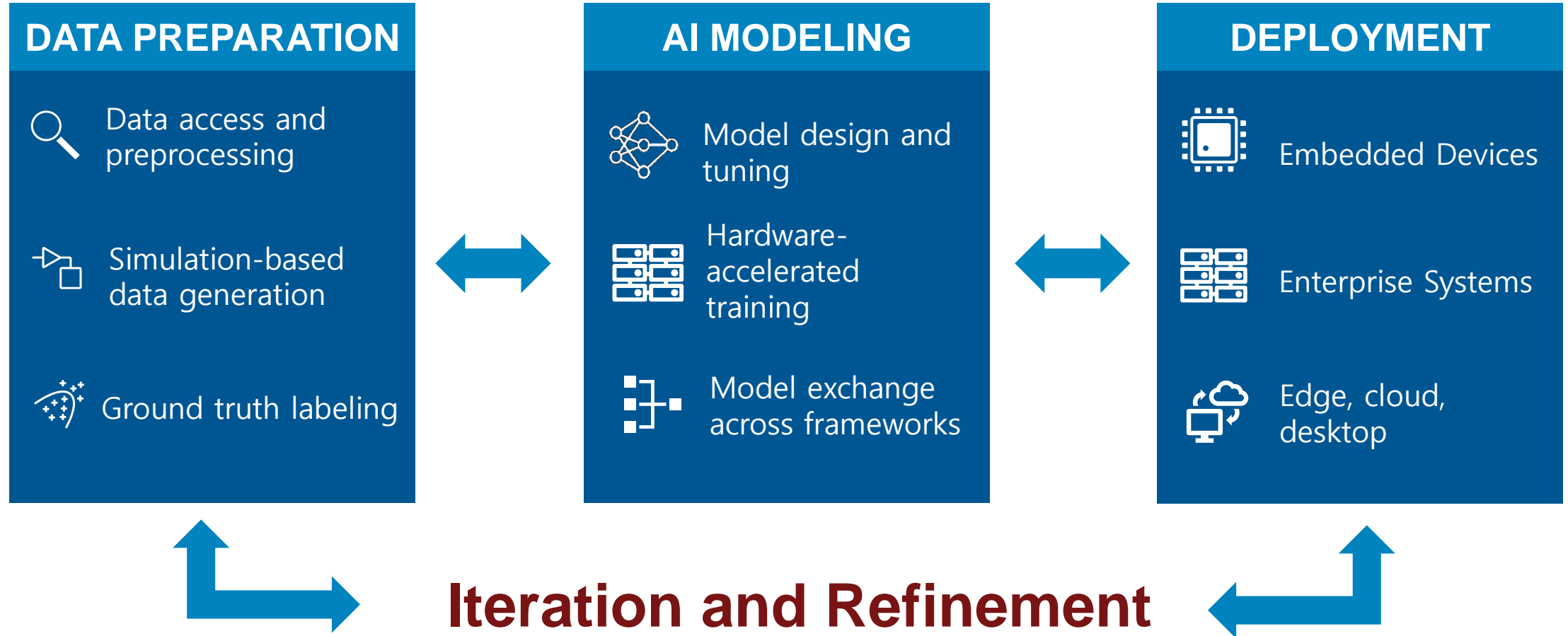
# Detecting Parts



# Defect Detection Workflow



# Defect Detection Workflow



# Data Access and Preprocessing – Common Challenges

How do I access large data that might not fit in memory?

How do I preprocess data and get the right features?

How do I label my data faster?

What if I have an imbalanced dataset or don't have enough data?

# Data Access and Preprocessing – Common Challenges

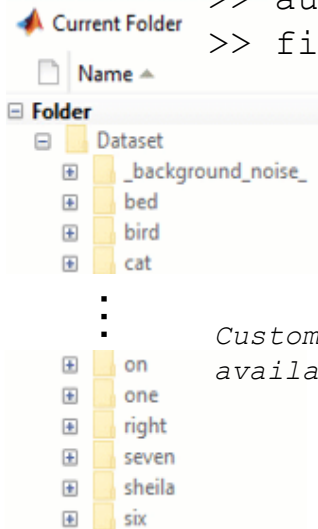
How do I access large data that might not fit in memory?

# How do I load and access large amounts of data?

## Datstores

Loads image/signal data into memory as and when needed

```
>> imageDatastore  
>> audioDatastore  
>> fileDatastore
```

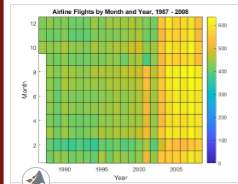


*Custom Datstores also available*

## Tall Arrays

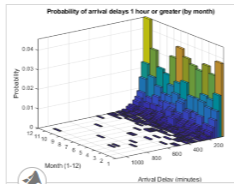
Work with out-of-memory numeric data

- Train deep neural networks for numeric arrays



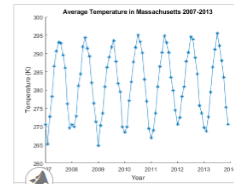
Analyze Big Data in MATLAB Using Tall Arrays

Use tall arrays to work with big data in MATLAB®. You can use tall arrays to perform a variety of calculations on different types of



Histograms of Tall Arrays

Use histogram and histogram2 to analyze and visualize data contained in a tall array.

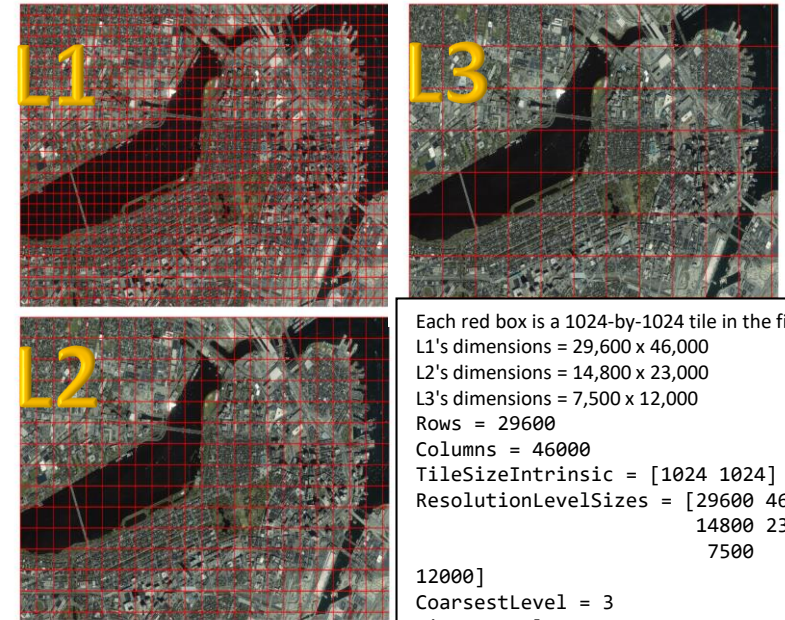


Process Big Data in the Cloud

Access a large data set in the cloud and process it in a cloud cluster using MATLAB capabilities for big data.

## BigImage

Work with very large, tiled and multi-resolution images



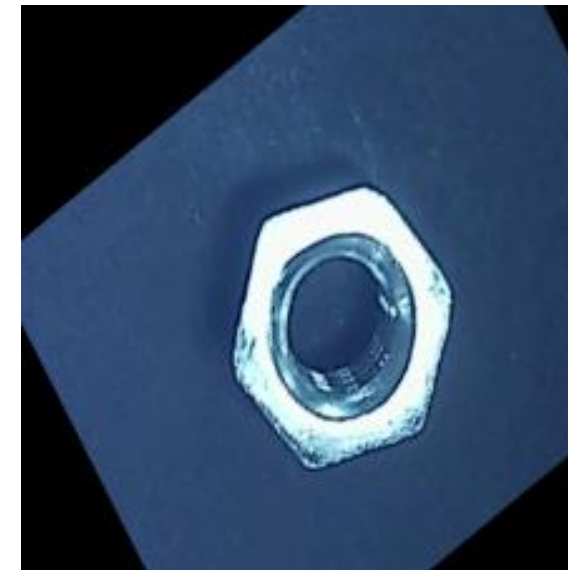
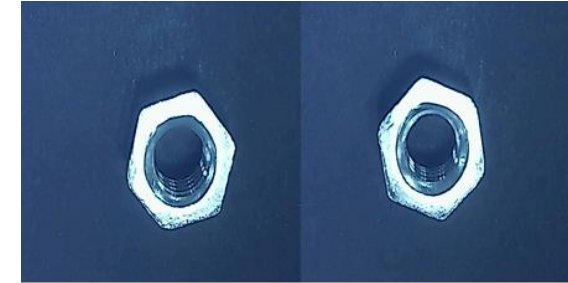
Each red box is a 1024-by-1024 tile in the file.  
L1's dimensions = 29,600 x 46,000  
L2's dimensions = 14,800 x 23,000  
L3's dimensions = 7,500 x 12,000  
Rows = 29600  
Columns = 46000  
TileSizeIntrinsic = [1024 1024]  
ResolutionLevelSizes = [29600 46000  
14800 23000  
7500  
12000]  
CoarsestLevel = 3  
FinestLevel = 1  
PixelSpacings = [1 1; 2 2; 3.947  
3.833]

How do I preprocess data  
and get the right features?

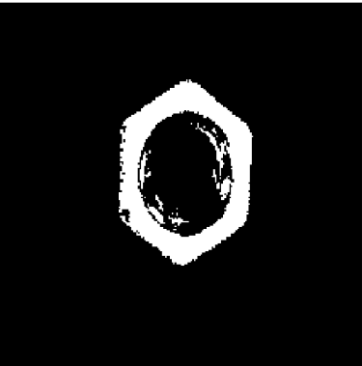
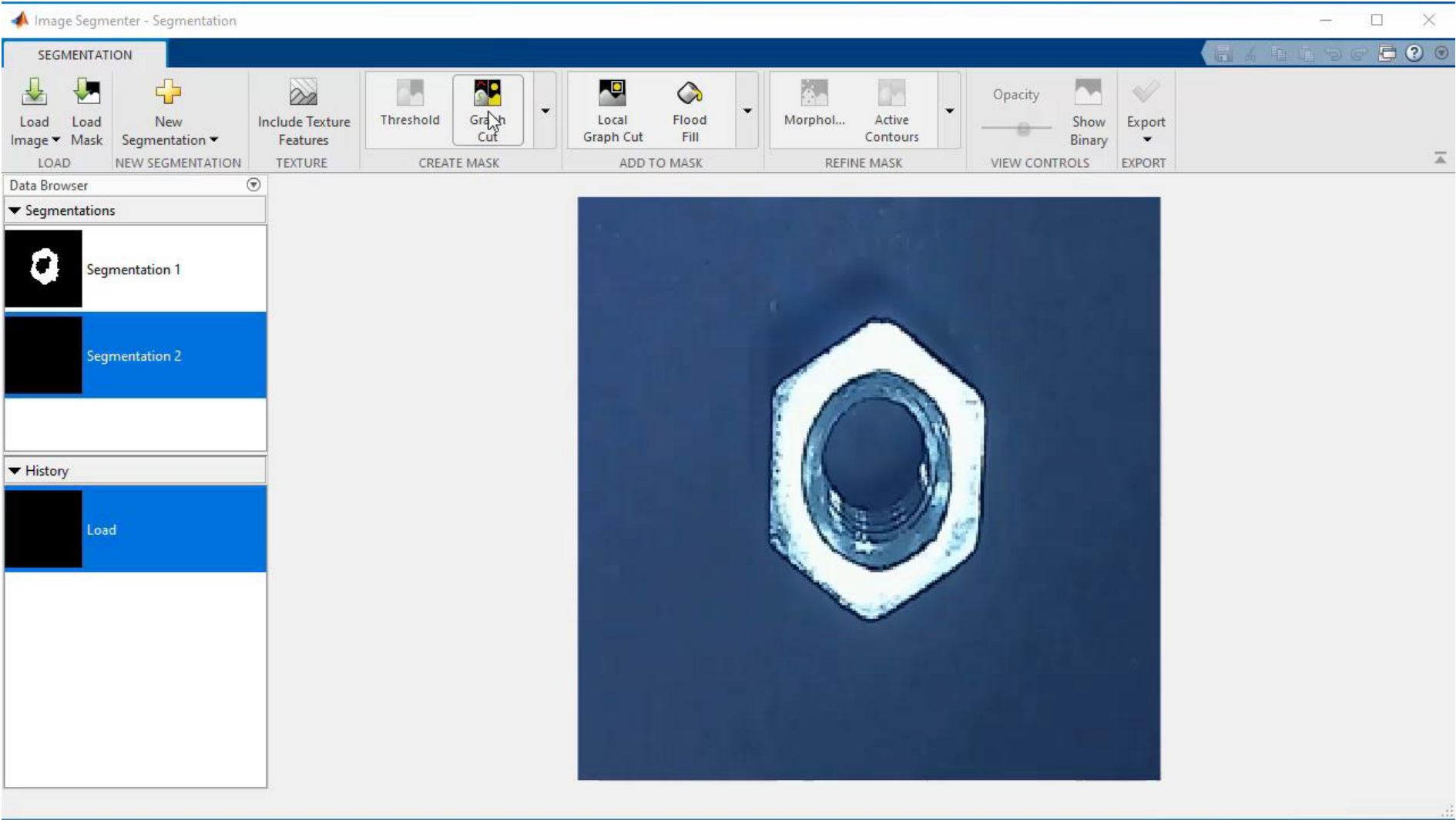


# Pre-processing Data – Registration Estimator App

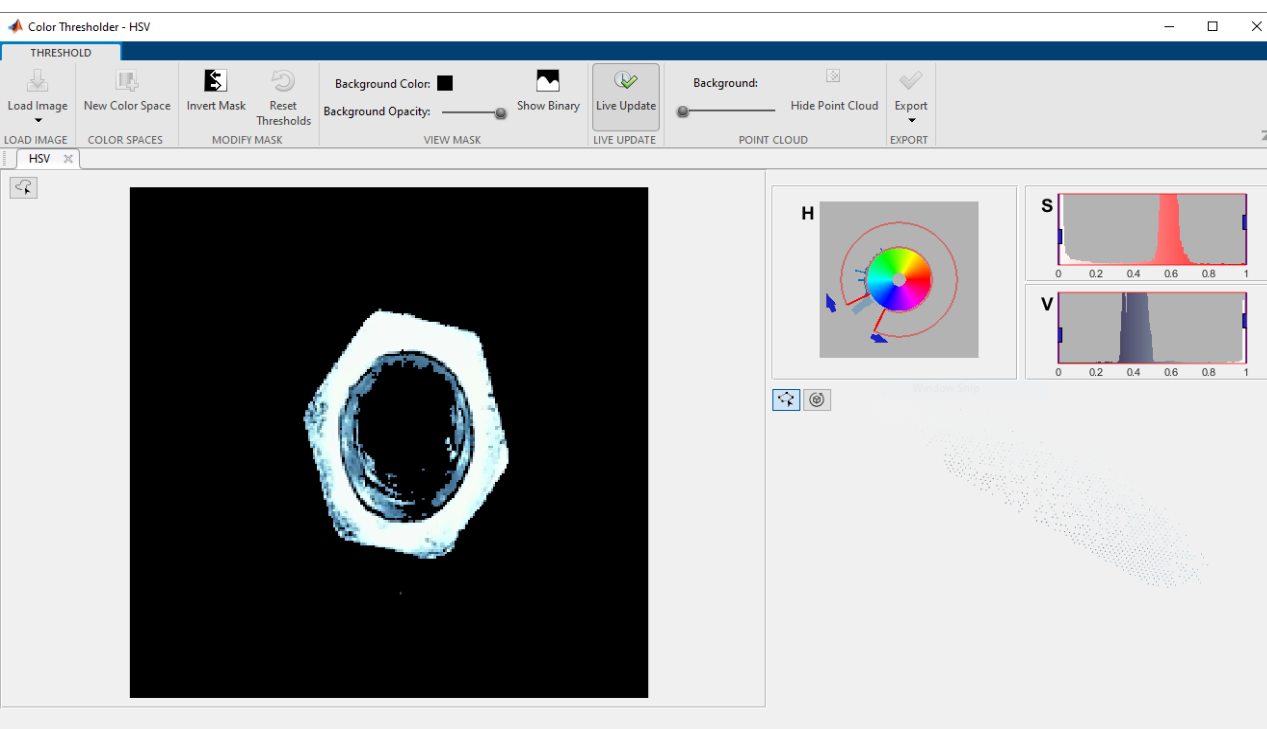
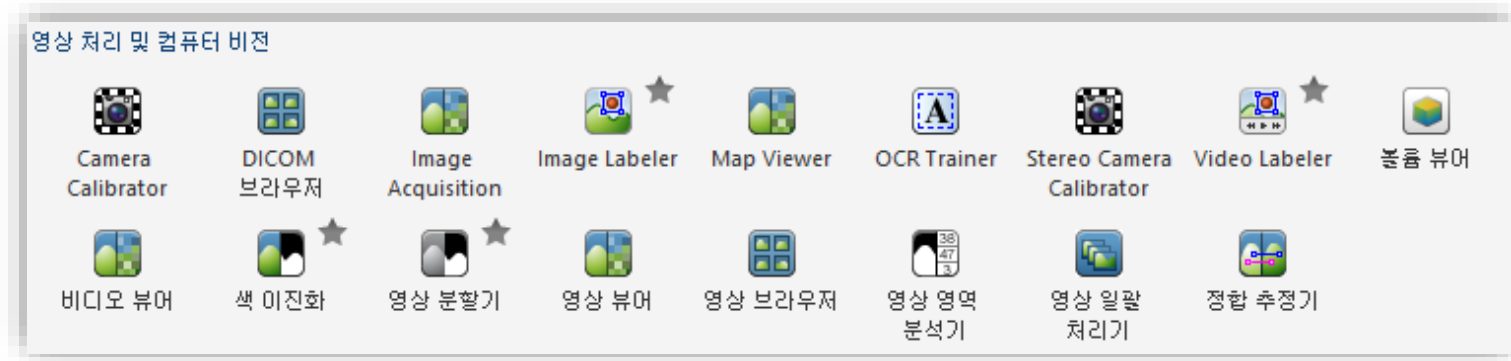
The screenshot displays the MATLAB Registration Estimator app interface. The title bar reads "Registration Estimator - Imov (Moving Image) & Iref (Fixed Image)". The main window is titled "REGISTRATION" and contains a toolbar with icons for "Load Images", "SURF", "FAST", "BRISK", and "Harris". Below the toolbar are buttons for "Register Images", "Overlay Style" (set to "Green-Magenta"), and "Export". The "Registrations" panel on the left lists three methods: "1 Phase Correlation [DRAFT]", "2 Feature: MSER [DRAFT] Detected: 104 and 11 Matched: 5", and "3 Feature: SURF [DRAFT] Detected: 79 and 101 Matched: 12". The central image shows two overlapping images of a nut, one in green and one in magenta, with yellow lines connecting detected features. The "Current Registration Settings" panel on the right shows "Projective" as the transformation type, "Number of Detected Features" and "Quality of Matched Features" sliders, a checked "Has Rotation" box, and a "Post-processing" section.



# Pre-processing Data – Image Segmenter App



# Preprocessing Data - Apps



Color Thresholder

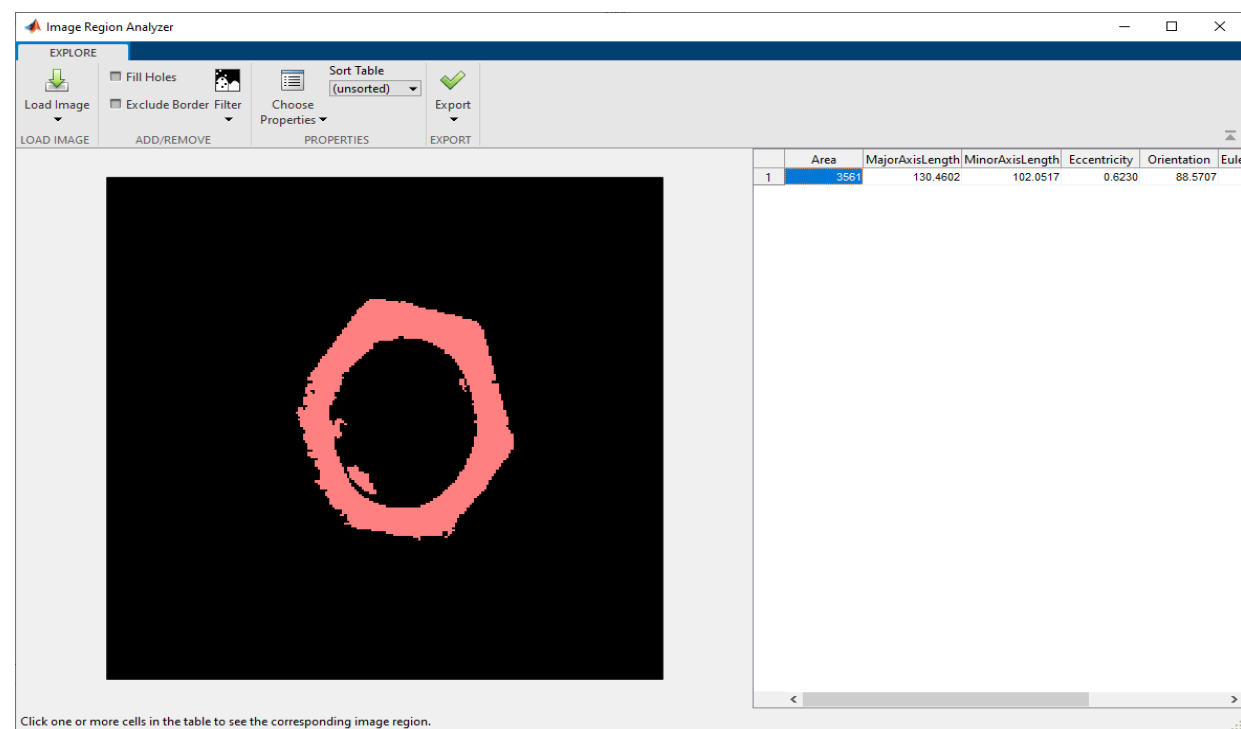
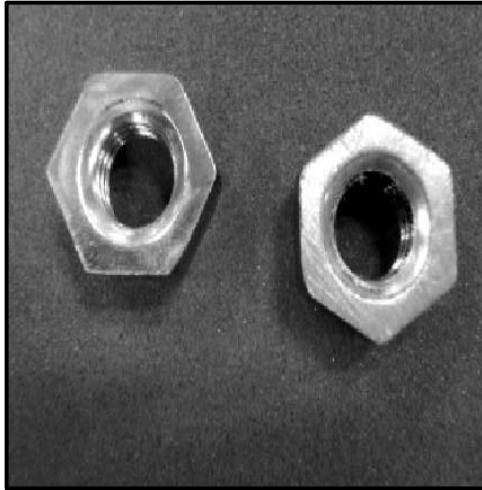


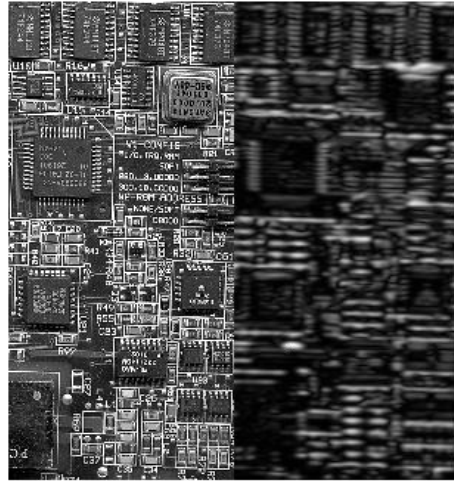
Image Region Analyzer

# Pre-processing Data – Built-in Algorithms

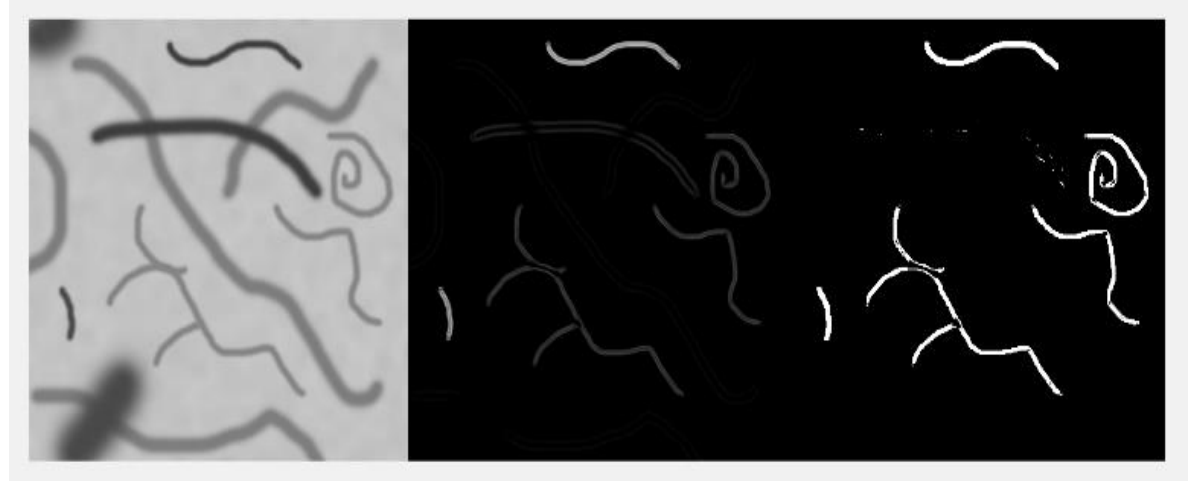
imadjust



imgaborfilt



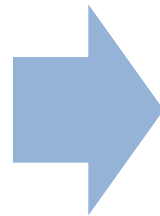
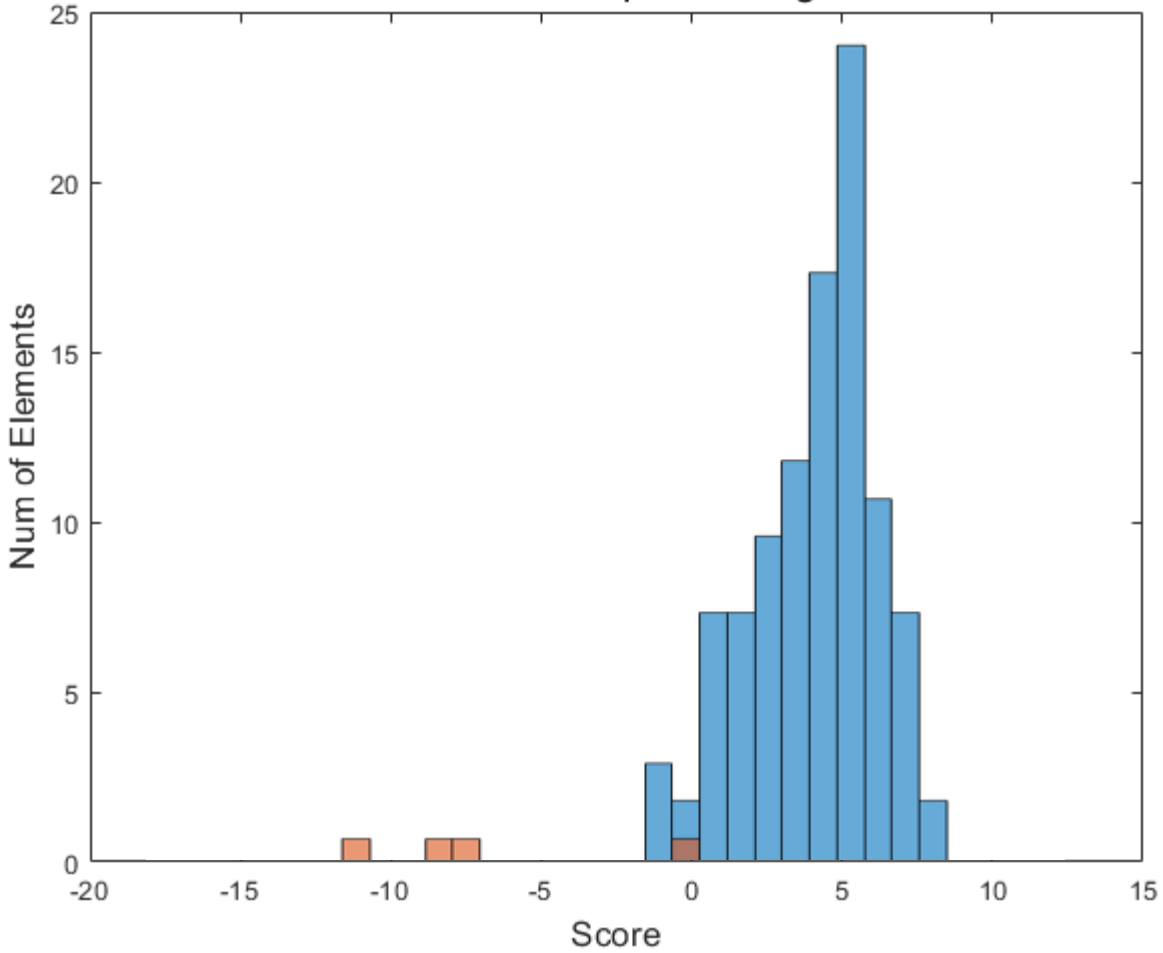
fibermetric



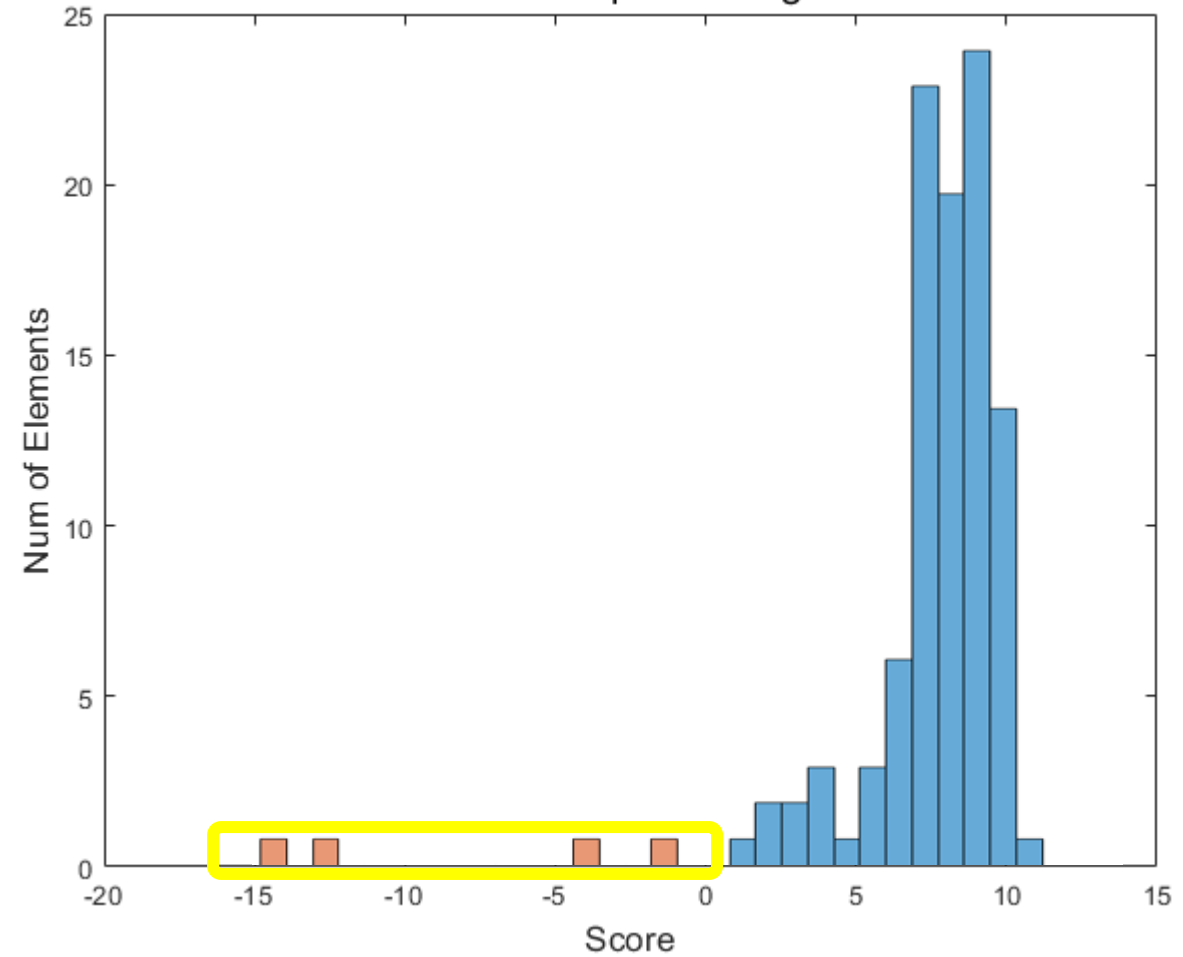
**And Many More!**

# Defect detection using AlexNet: Results with preprocessing

Without Preprocessing



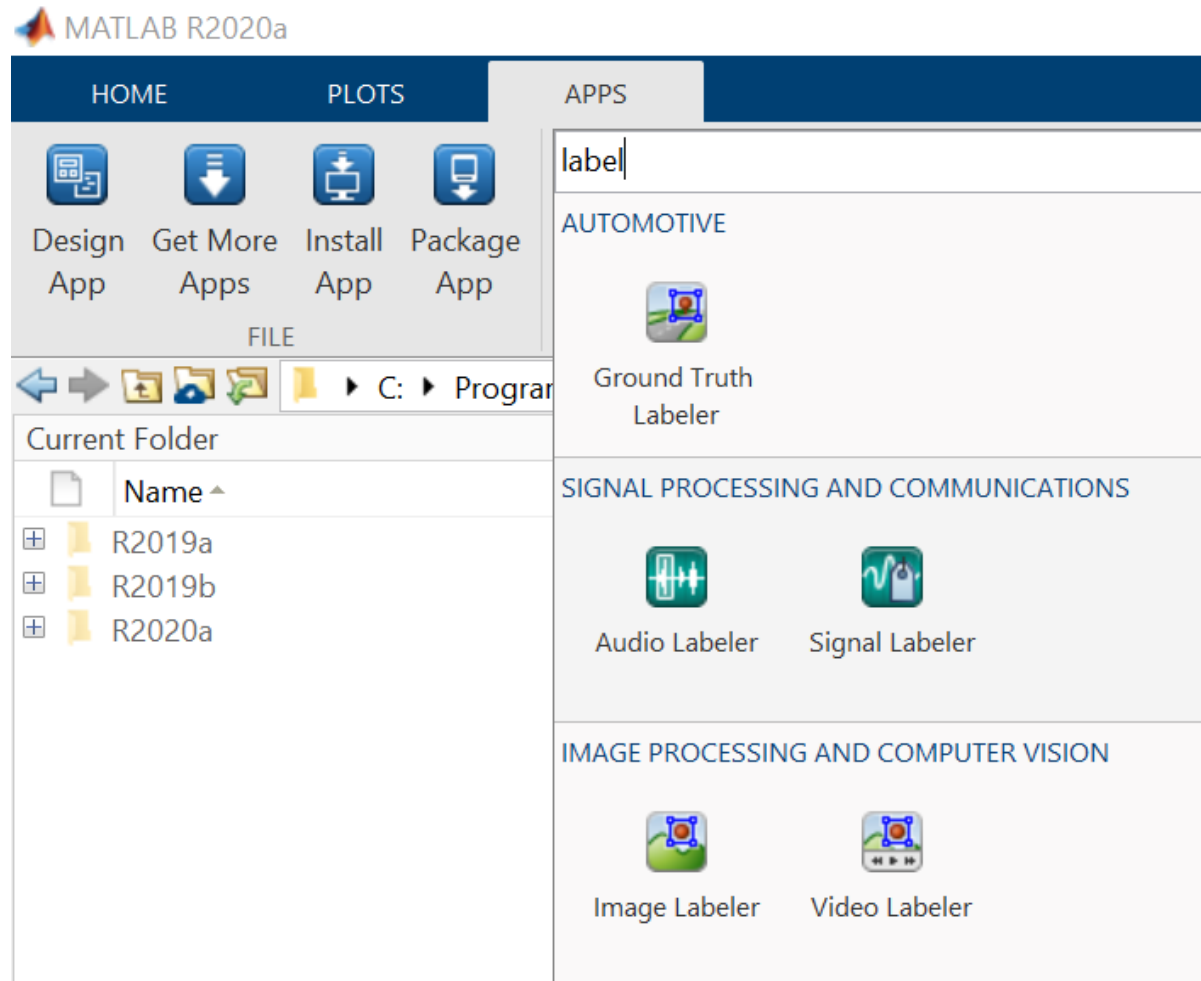
With Preprocessing



# Data Access and Preprocessing – Common Challenges

How do I label my data faster?

# Data Preprocessing - Labeling

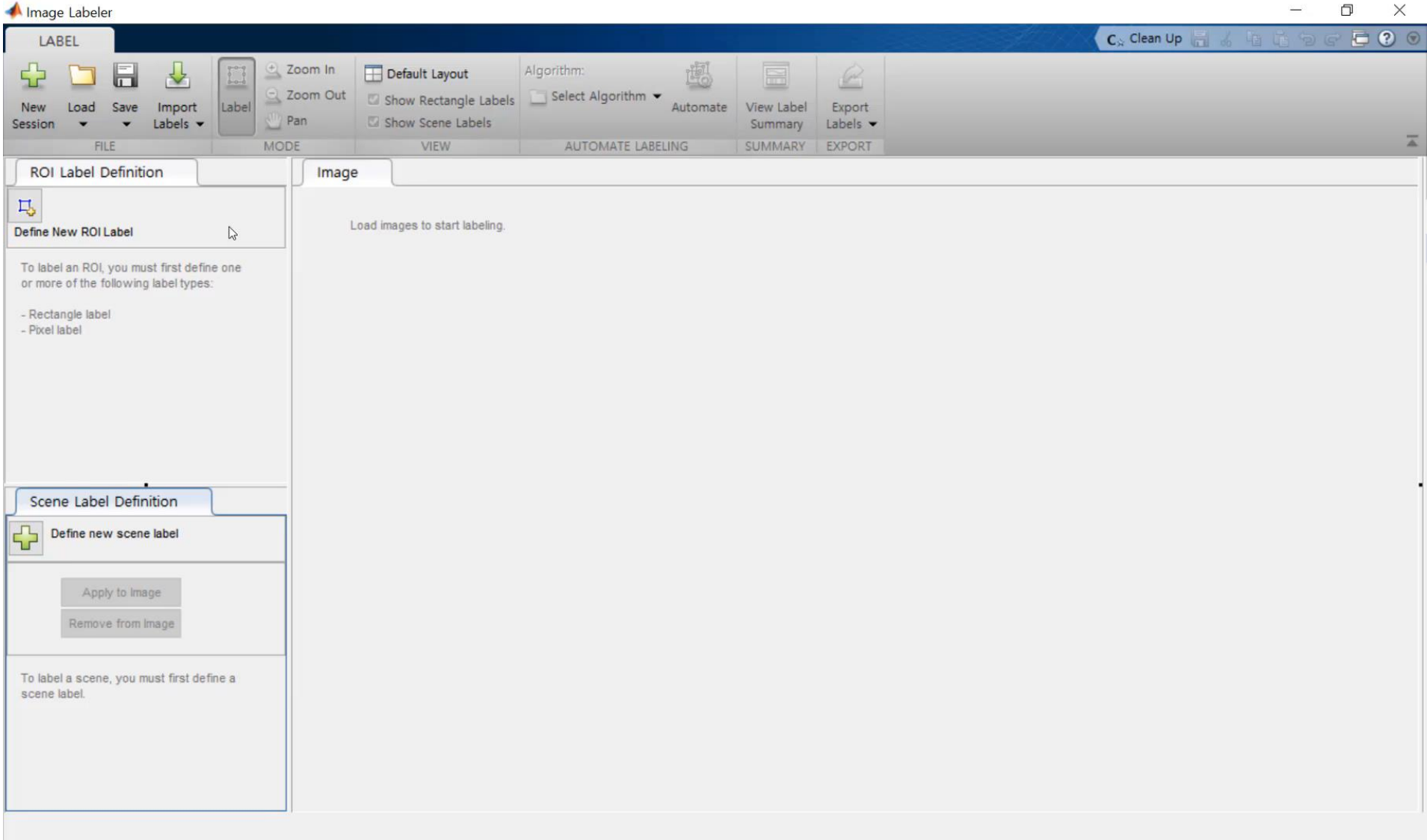


# Image Labeler

Image Labeler

Video Labeler

Big-Image Labeler



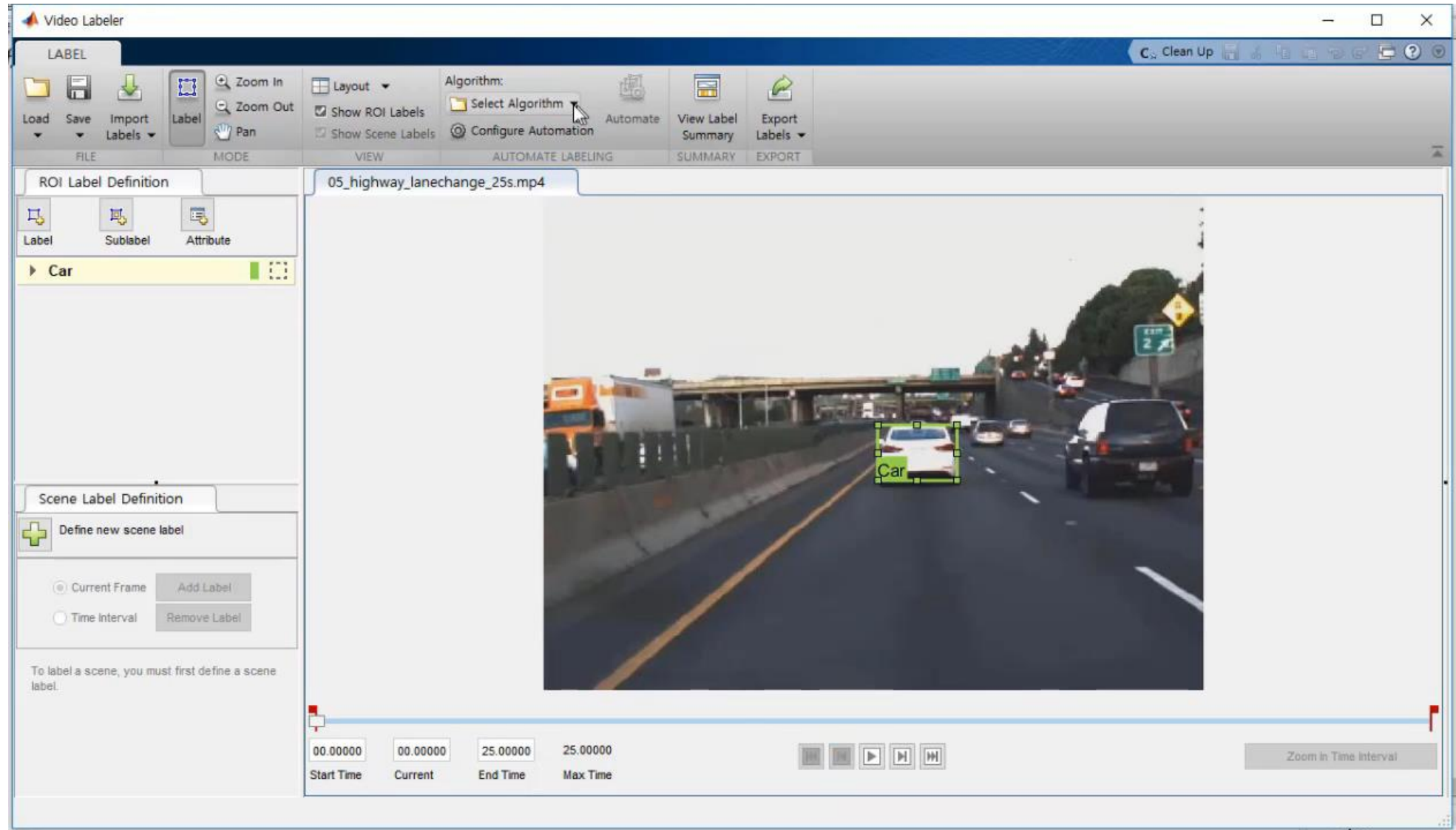


# Video Labeler

Image  
Labeler

Video  
Labeler

Big-Image  
Labeler



# Big Image Labeler

The screenshot displays the Big Image Labeler software interface. At the top, the title bar reads "Big Image Labeler" and the menu bar includes "FILE", "CONVERSIONS", "EXISTING LABELS", and "OPTIONS". Below the menu bar is a toolbar with various icons for navigation and editing. The main workspace is divided into two panels. The left panel, titled "Overview Region: [42497 48641 1536 1024]", shows a circular overview of a large image with a small purple box indicating the current subimage's location. Below this panel is a "Tooling/Options" section with several sub-sections: "DISPLAY/EXPORT Annotations", "Define/Extract Training Chips", "Recall Session", "LABEL VISIBILITY" (with checkboxes for "Overview" and "Subimage", and a checked "Auto-Save"), "Labeling Mode" (with radio buttons for "Manually Label", "Predict/Verify", and "Freehand", and a selected "Rectangle"), and "ROI Type" (with radio buttons for "Freehand" and "Rectangle", and a selected "Rectangle"). The right panel, titled "Subimage from: Scan004\_cellspot\_pyramid", shows a magnified view of a cell spot image. It features a pink outline around the cell spot and three green labels with the text "InitialAutoSegmentation". Below the subimage is a "Segmentation/Filter Settings" section with "Sensitivity" (0 to 1, set to 0.866667), "Minimum Size" (0 to 1e6, set to 5000), "Border Simplification" (0 to 1, set to 0), and "Segmentation Option" (with radio buttons for "Threshold", "Imextended", "Gray Connectivity", and "Custom", and a selected "Threshold"). To the right of the subimage is a table with columns "LABEL" and "ID". The table contains four rows, all labeled "InitialAutoSegmentation" with IDs 1, 2, 3, and 4. Below the table are buttons for "Select By Label", "GoTo (first)", "Delete Selected", "Rename Selected", and "Validate". At the bottom left of the interface, the pixel info is displayed as "Pixel info: (43690.68, 49491.69) [244 244 245]".

Image  
Labeler

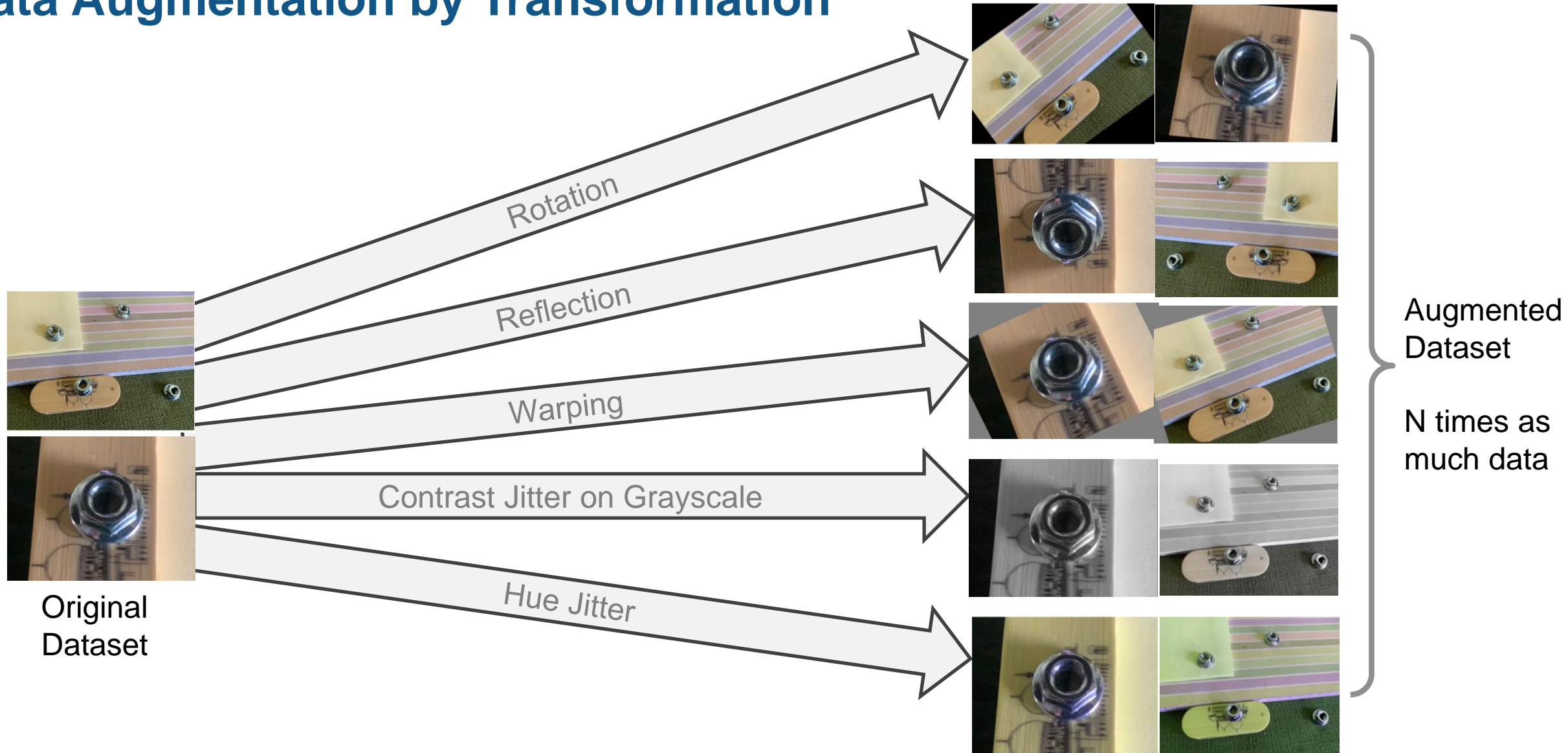
Video  
Labeler

Big-Image  
Labeler

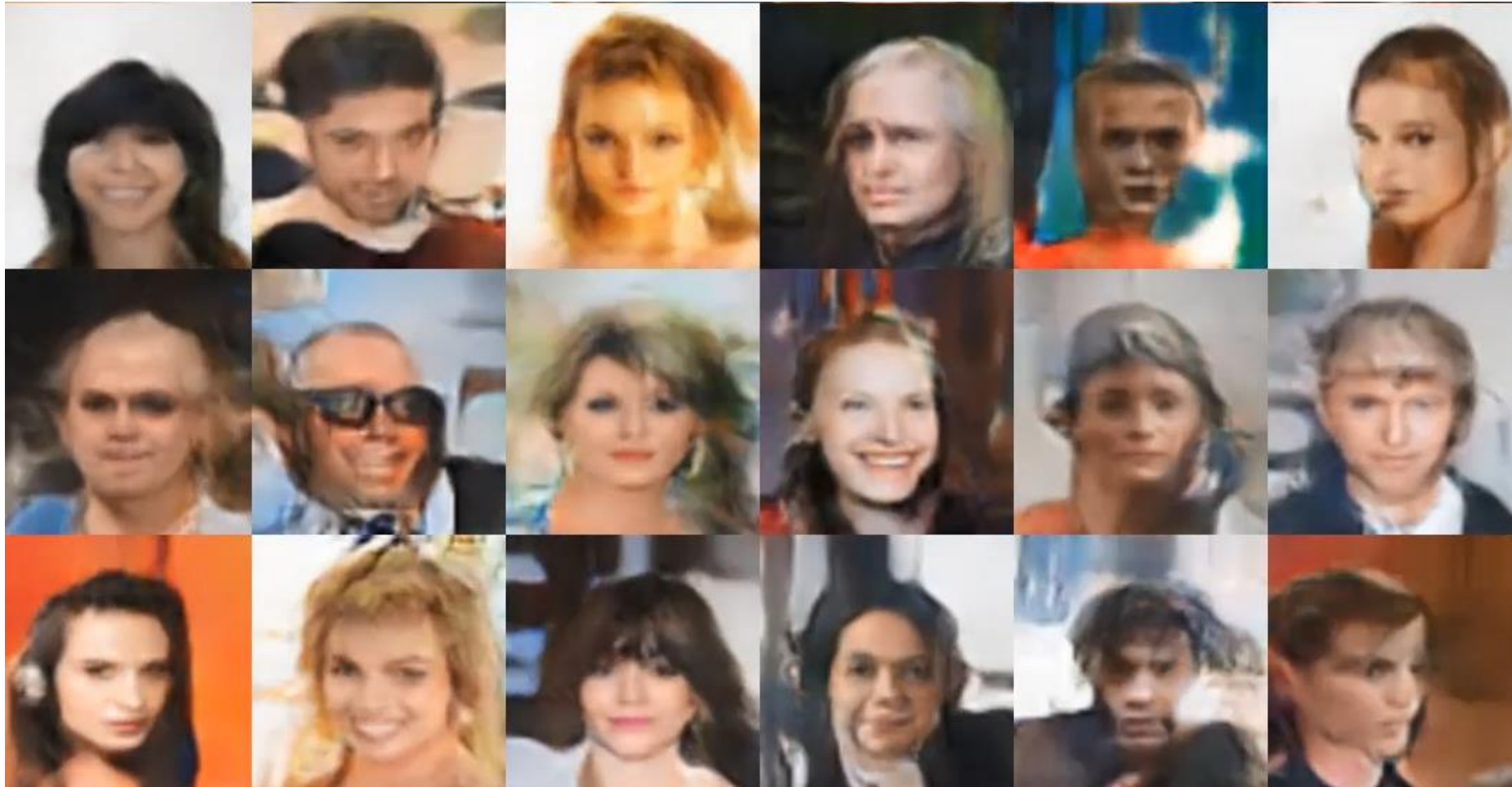
# Data Access and Preprocessing – Common Challenges

What if I have an imbalanced dataset or don't have enough data?

# Data Augmentation by Transformation

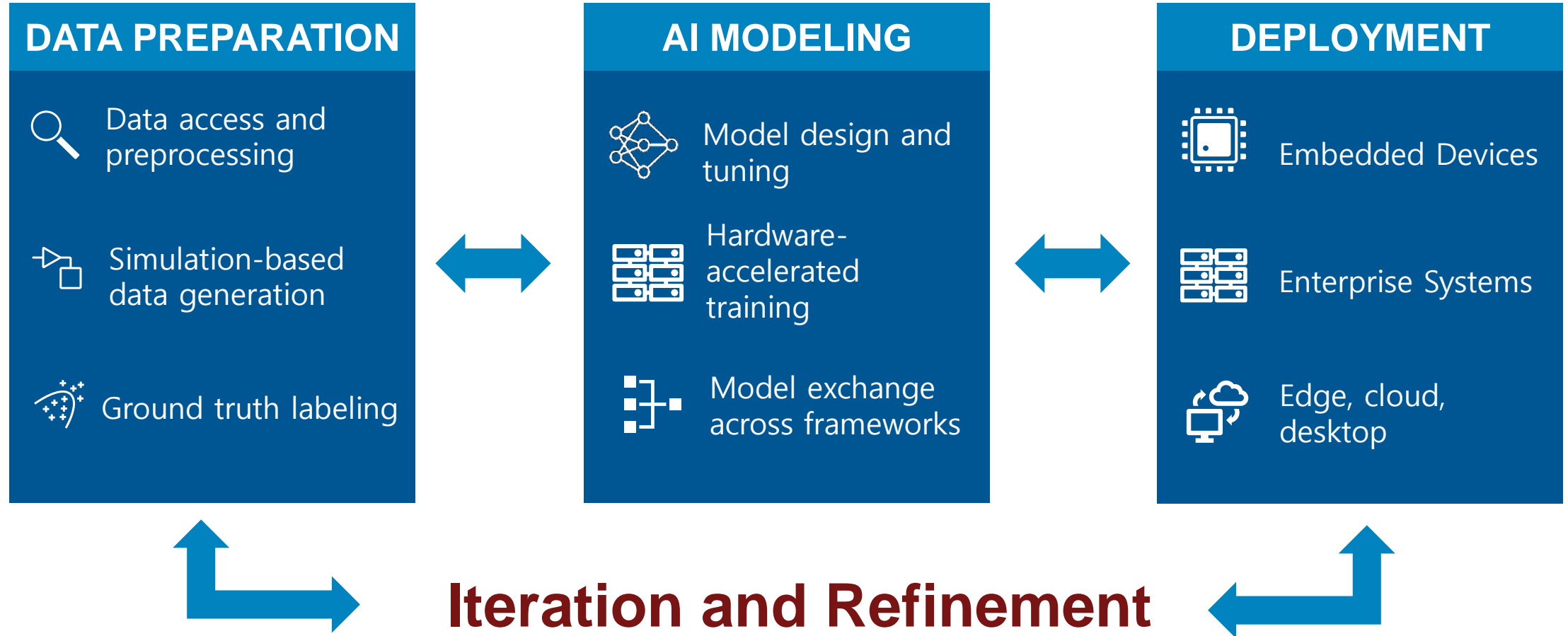


# Data Augmentation : Generative Adversarial Networks (GANs)



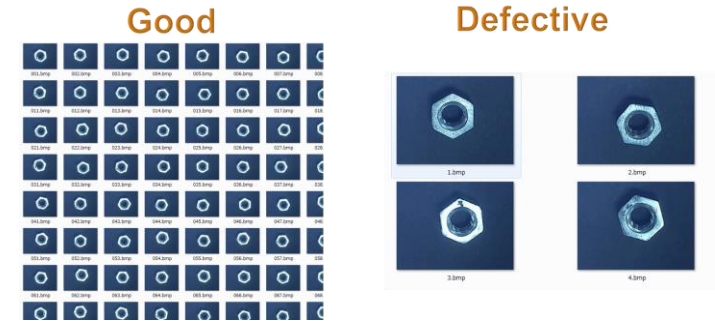
Generative Adversarial Networks

# Defect Detection Workflow

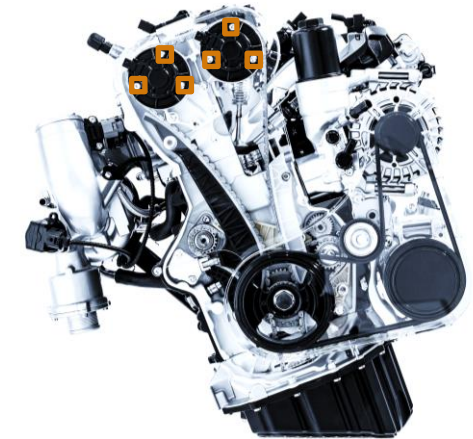


# Deep Learning for Defect Detection

Deep learning for  
Classification



Deep Learning for Object  
Detection

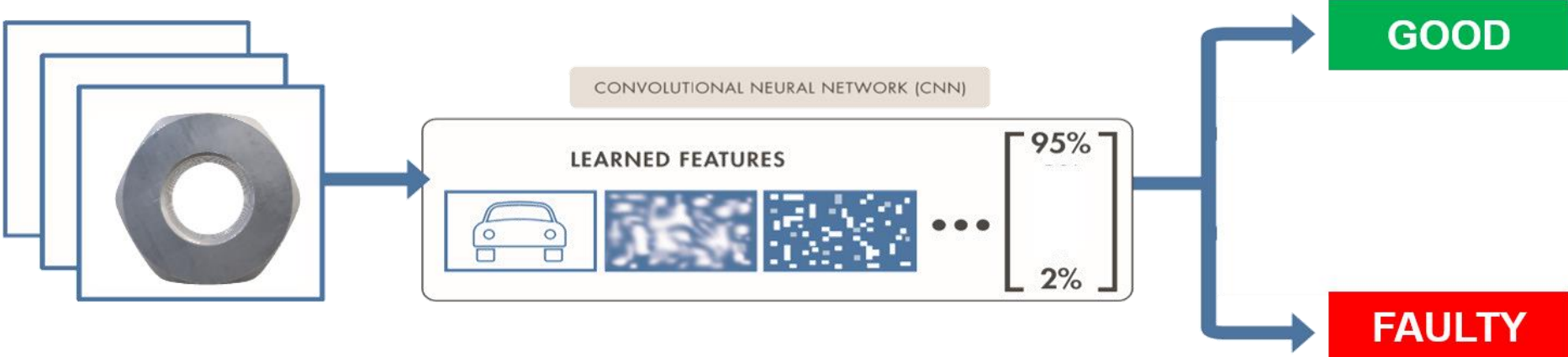


## Deep learning for Classification

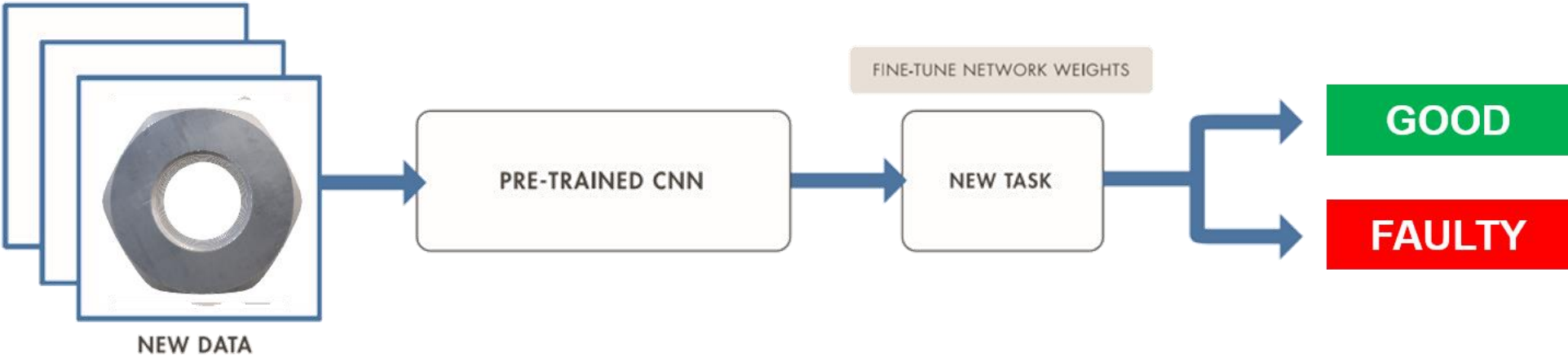


# Two Approaches for Deep Learning

## 1. Train a deep neural network from scratch



## 2. Fine-tune a pre-trained model (transfer learning)

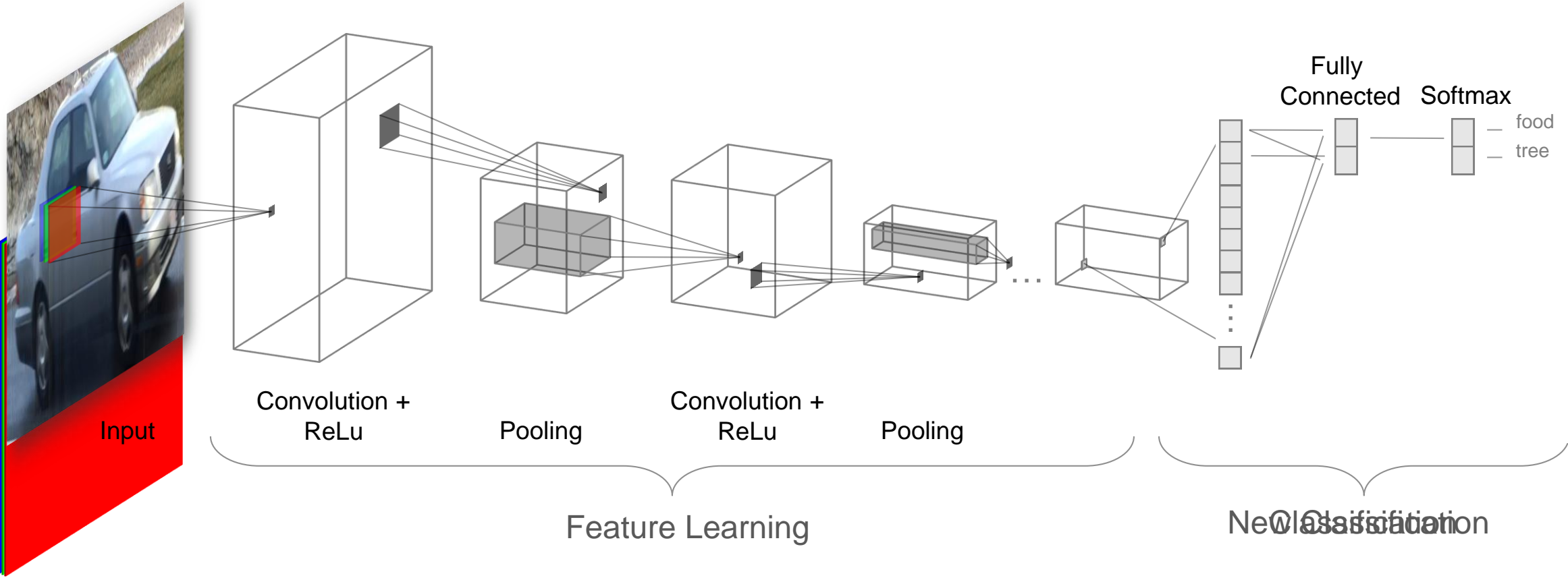


# Train a Deep Neural Network from Scratch

The screenshot displays the MATLAB Deep Network Designer interface. The main workspace shows a vertical sequence of layers: 'sequence sequenceInput...', 'lstm lstmLayer', 'fc fullyConnected...', 'softmax softmaxLayer', and 'classoutput classificationLa...'. The 'fc' layer is highlighted in blue. On the left, the 'LAYER LIBRARY' is visible, with 'OBJECT DETECTION' and 'OUTPUT' categories. The 'OUTPUT' category includes 'softmaxLayer', 'classificationLayer', 'regressionLayer', 'rpnSoftmaxLayer', 'rcnnBoxRegressionLayer', 'rpnClassificationLayer', 'pixelClassificationLayer', 'dicePixelClassificationLayer', and 'yolov2OutputLayer'. On the right, the 'PROPERTIES' panel for the selected 'fullyConnectedLayer' shows settings: Name (fc), InputSize (auto), OutputSize (10), Weights ([]), Bias ([]), WeightLearnRateFactor (1), WeightL2Factor (1), BiasLearnRateFactor (1), BiasL2Factor (0), WeightsInitializer (glorot), and BiasInitializer (zeros). An 'OVERVIEW' section at the bottom right shows a simplified diagram of the network structure.

# Two approaches for Deep learning

## Approach 2. Fine-tune a pre-trained model (Transfer learning)



# Fine-tune a Pre-trained Model (Transfer Learning)

Deep Network Designer

MATLAB<sup>®</sup> Deep Network Designer

Getting Started | Compare Pretrained Networks | Transfer Learning

SqueezeNet

GoogLeNet

ResNet-50

DarkNet-53

DarkNet-19

ShuffleNet

NasNet-Mobile

NasNet-Large

Xception

Places365-Goog...

MobileNet-v2

DenseNet-201

ResNet-18

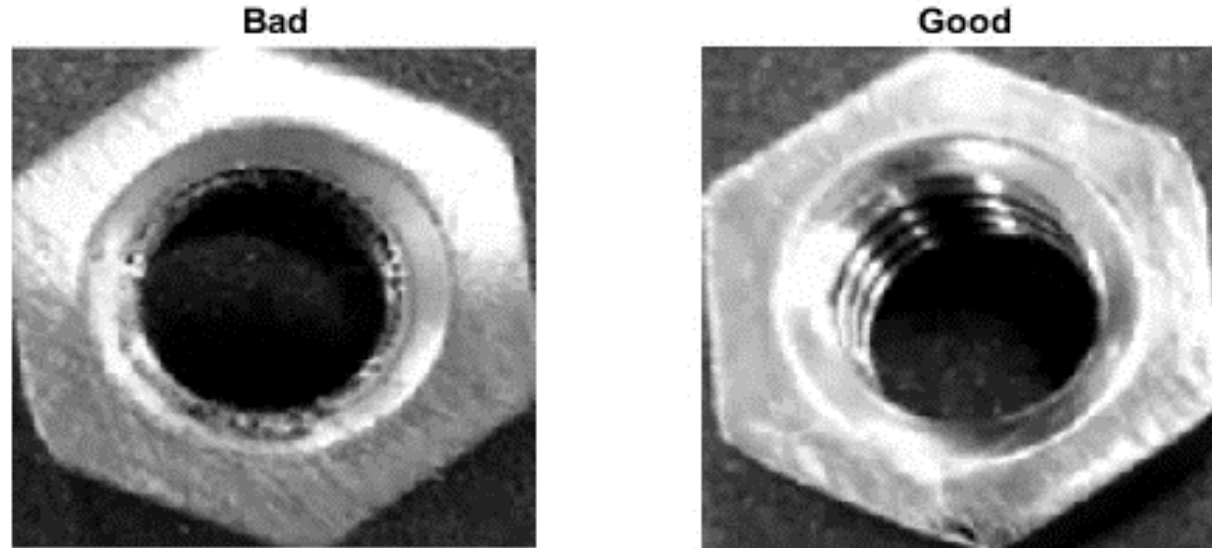
Inception-ResNe...

# Experiment Manager

The screenshot displays the MATLAB Experiment Manager interface. The top toolbar includes buttons for New, Save, Duplicate, Layout, Run, Stop, Training Plot, Confusion Matrix, Filter, and Export. The left sidebar shows a tree view of the experiment structure, including 'DigitsClassifier', 'Baseline Establishment', and 'Baseline Tuning'. The main area shows 'Result Details' for 'Baseline Tuning' on 2/7/2020 at 12:53:36 PM, with 7/16 trials completed. A summary table indicates 7 Complete, 1 Running, 0 Stopped, 8 Queued, 0 Error, and 0 Canceled trials. Below this is a detailed table of trial results.

Trial	Status	Progress	Elapsed Time	myInitialLearn...	convFilterSize	Training Accu...	Training Loss	Validation Ac..
1	Complete	100.0%	0 hr 0 min 16 sec	1.0000e-6	3.0000	12.5000	2.6441	10.
2	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-5	3.0000	25.7813	2.1228	20.
3	Complete	100.0%	0 hr 0 min 14 sec	0.0001	3.0000	64.8438	1.0878	42.
4	Complete	100.0%	0 hr 0 min 16 sec	0.0005	3.0000	90.6250	0.4648	49.
5	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-6	4.0000	11.7188	2.4967	6.
6	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-5	4.0000	23.4375	2.1213	14.
7	Complete	100.0%	0 hr 0 min 17 sec	0.0001	4.0000	72.6563	1.0283	39.
8	Running	30.7%	0 hr 0 min 4 sec	0.0005	4.0000			
9	Queued	0.0%		1.0000e-6	5.0000			
10	Queued	0.0%		1.0000e-5	5.0000			
11	Queued	0.0%		0.0001	5.0000			
12	Queued	0.0%		0.0005	5.0000			
13	Queued	0.0%		1.0000e-6	6.0000			
14	Queued	0.0%		1.0000e-5	6.0000			
15	Queued	0.0%		0.0001	6.0000			
16	Queued	0.0%		0.0005	6.0000			

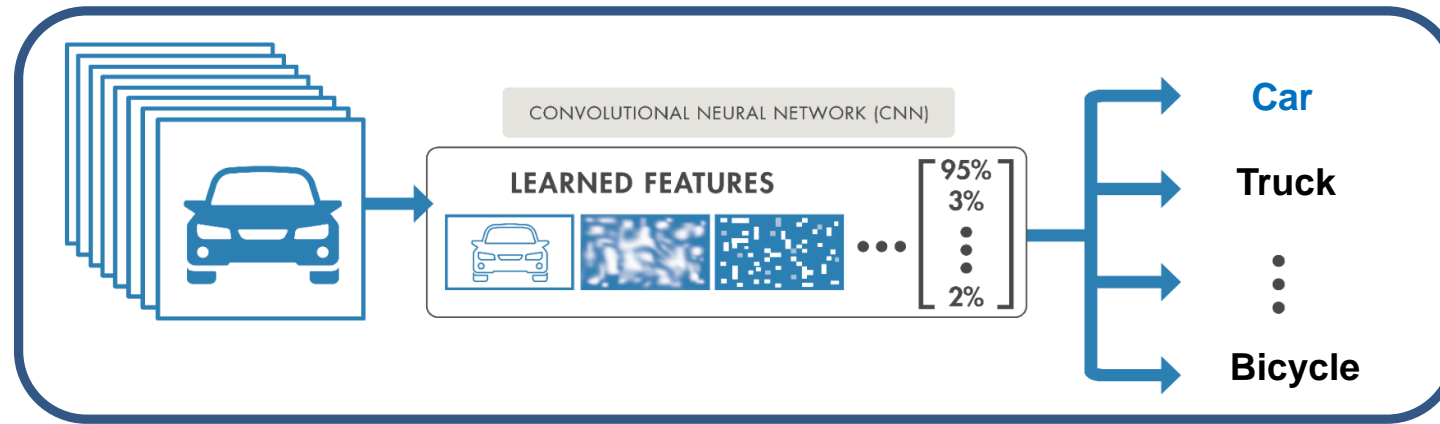
# Classification with Trained MobileNetV2



**Why the defect nuts are classified as 'Bad'?**

# Challenges with Deep Learning Models

- Deep Learning model is a black box model



- Is it possible to classify an unknown image correctly?
- Why the model misclassify for certain images?

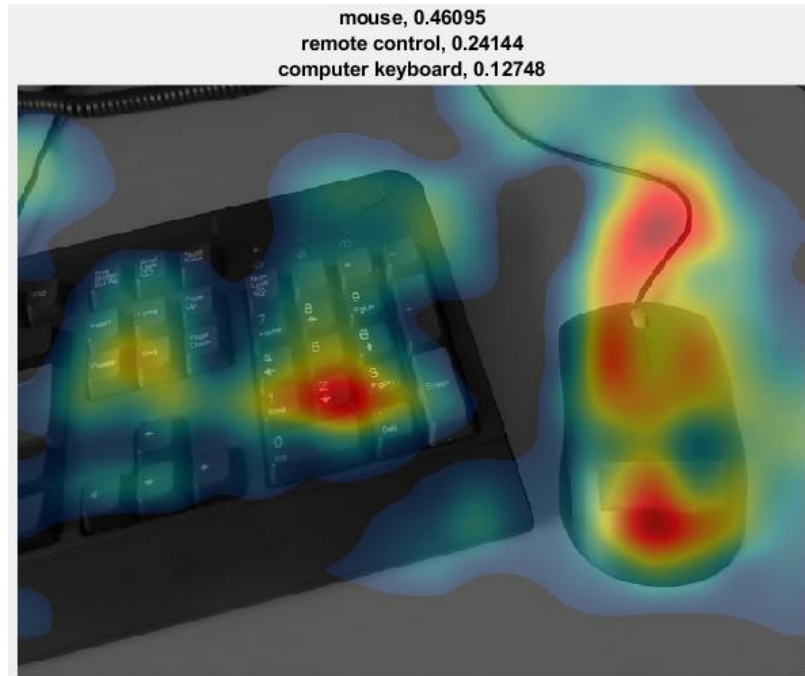
*Explainable AI  
is required*



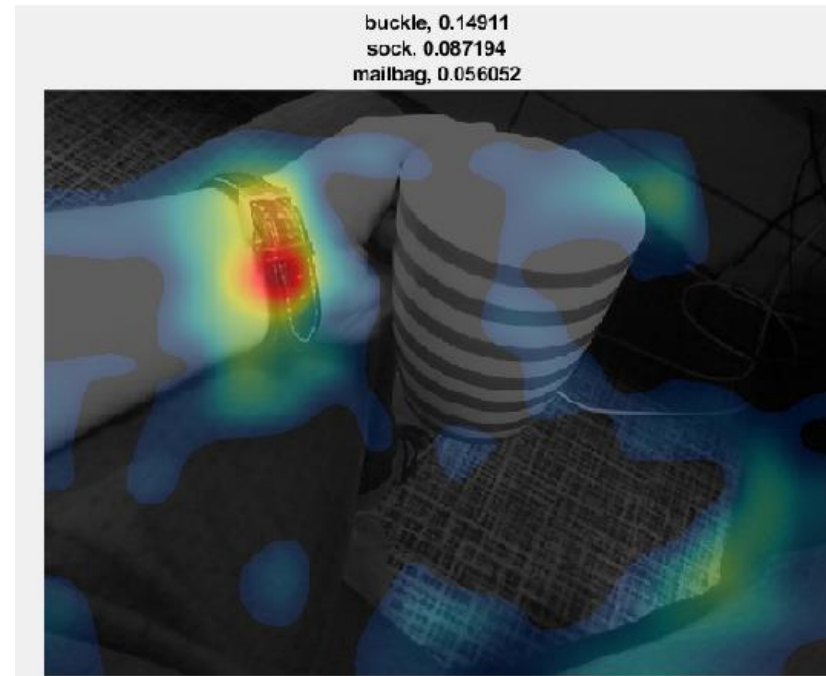
- [Class Activation Mapping \(CAM\)](#)
- [Grad-CAM](#)

# Class Activation Mapping to Investigate Network Predictions

Attribution Reveals the Why Behind Deep Learning Decisions [Full code available](#)



Classified as “keyboard” due to the presence of the mouse



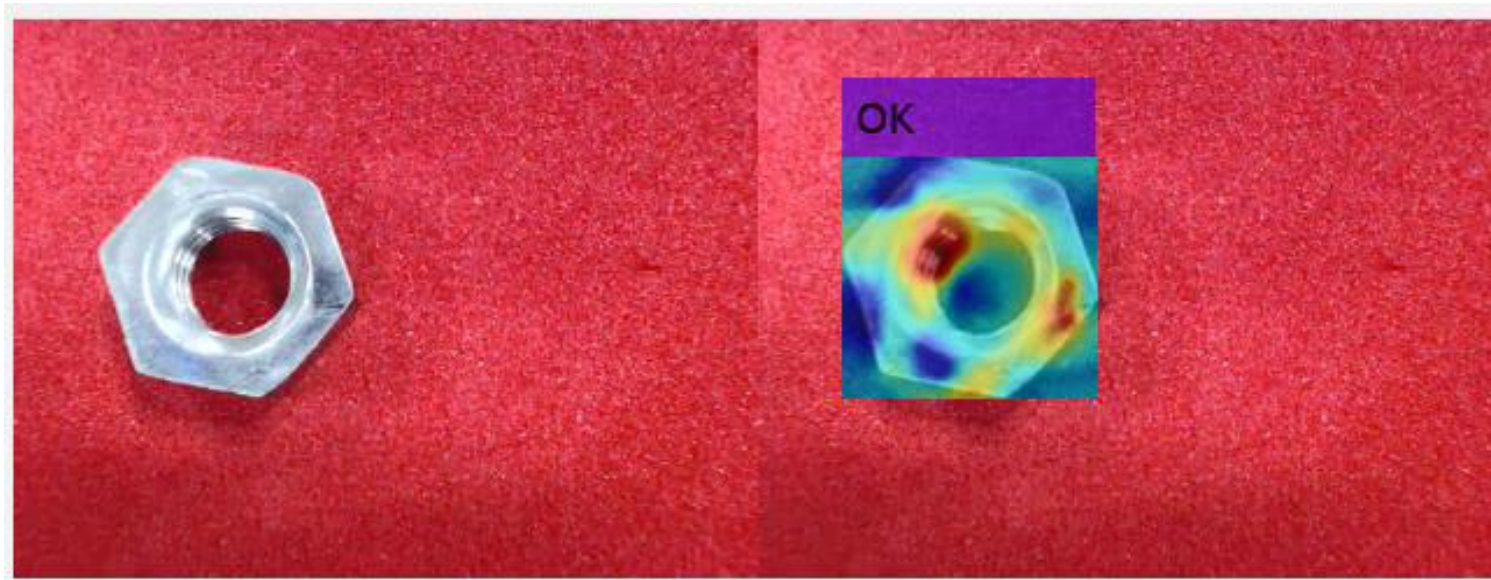
Incorrectly classified “coffee mug” as “buckle” due to the watch





# Visualization of Features with CAM

Captured Image

Classification and CAM

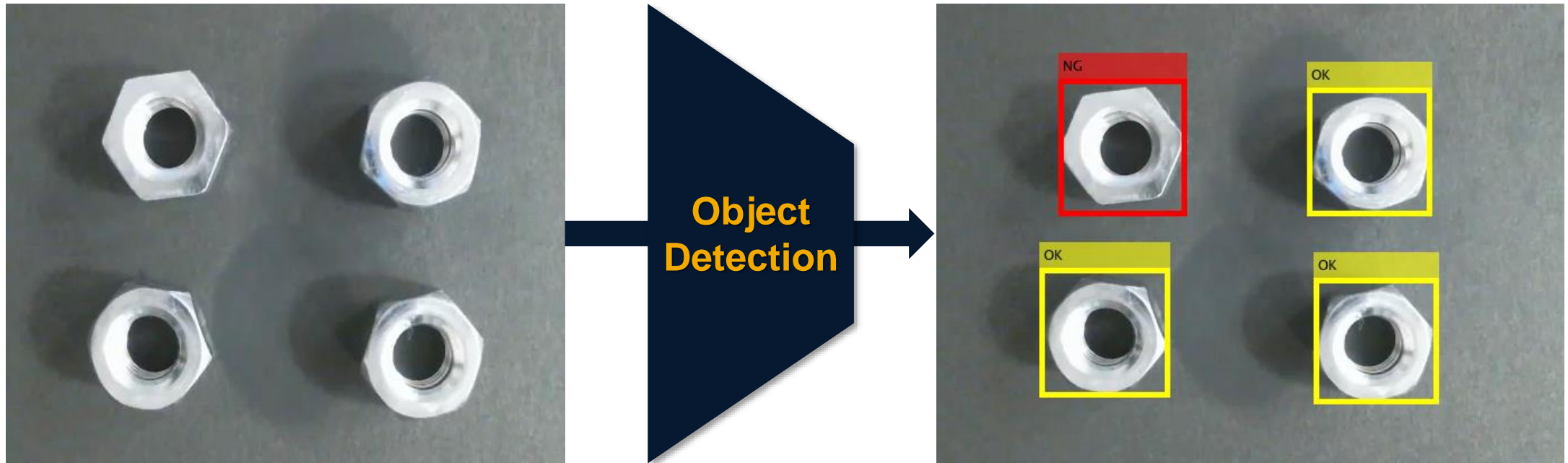


-  OK → Reacts to whole surface
-  Bad → Reacts to the scratch

The network judges the unit as Bad by seeing the scratched area

## Deep Learning for Object Detection

# Object Detection in Image/Vision System



- **Object detection**

- Computer technology related to computer vision and image processing that deals with **detecting instances of semantic objects** of a certain class (such as humans, buildings, or cars) in digital images and videos

# Increase productivity using built-in functions for design and analysis




Use MATLAB high-level API for changing network type, backbone network for better performance easily.

Choose Network to Train

```
nettoTrain = YOLOv2 ;
```

- YOLOv2
- YOLOv2
- SSD
- FasterRCNN

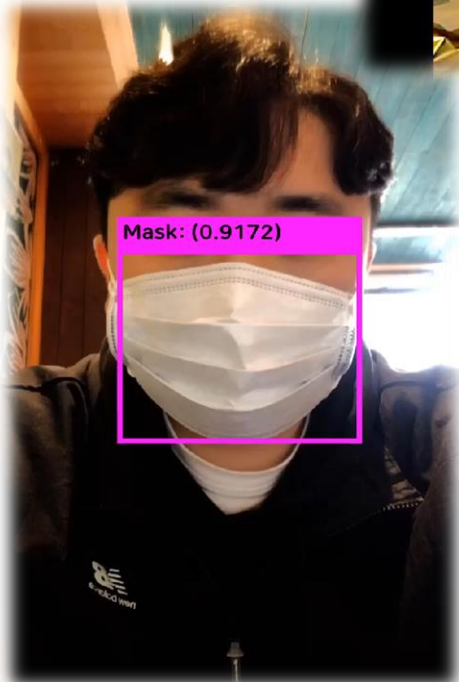
### AI Modeling

-  Model design and tuning
-  Hardware accelerated training
-  Interoperability

# Practical Object Detection Examples using Deep Learning



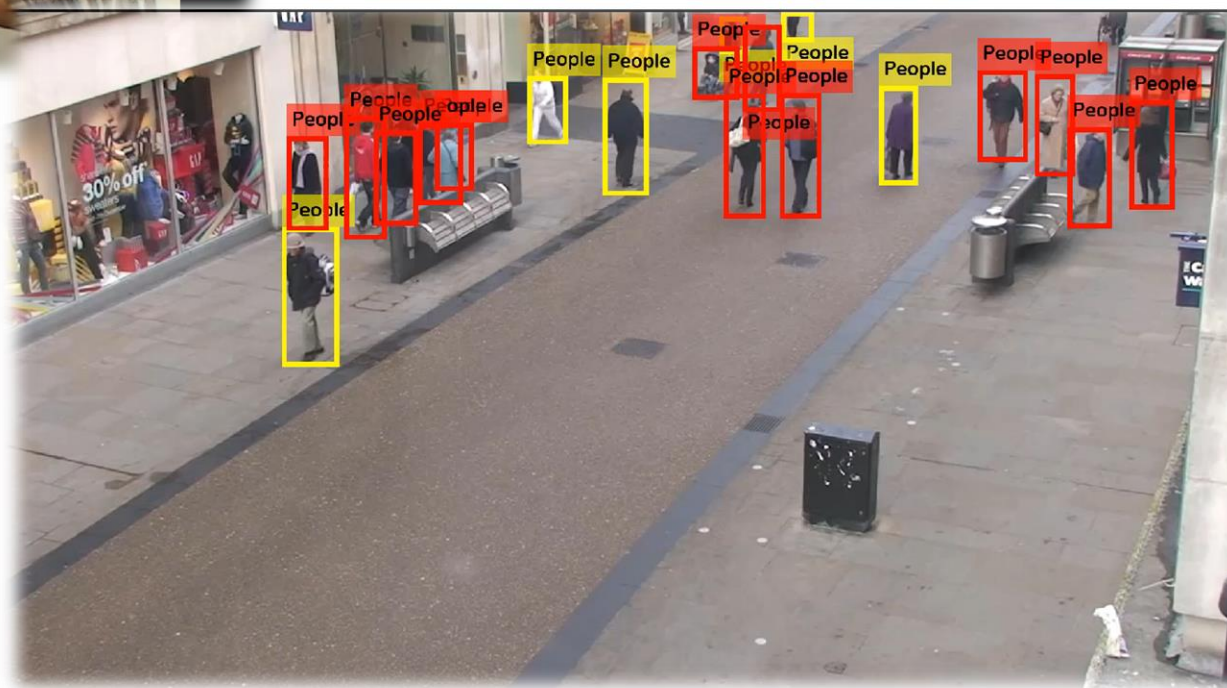
[Repository Link](#)



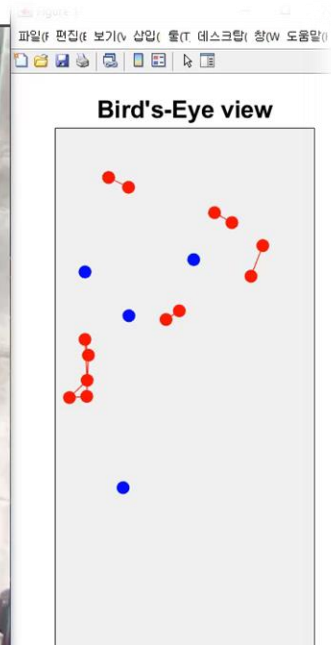
[Repository Link](#)



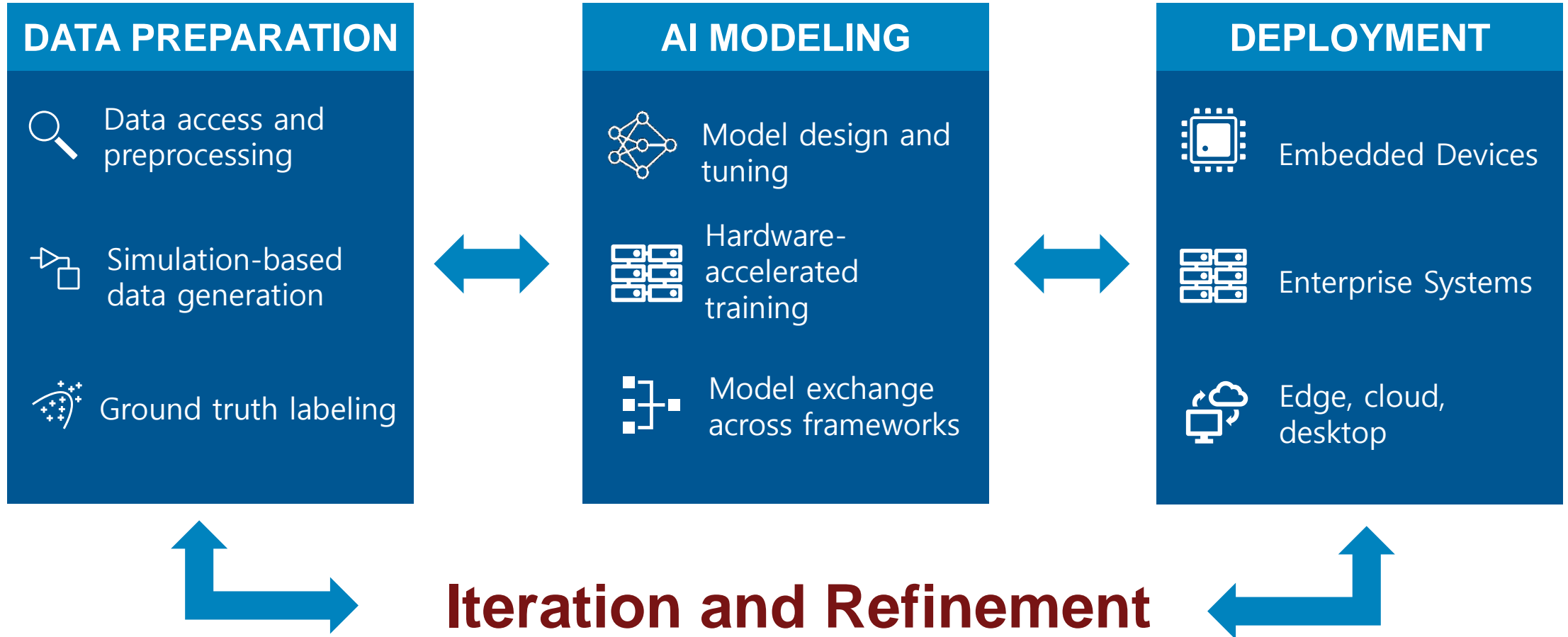
MATLAB Deep Learning  
mathworks.github.io  
<https://www.mathworks.com/solutions/deep-le...>



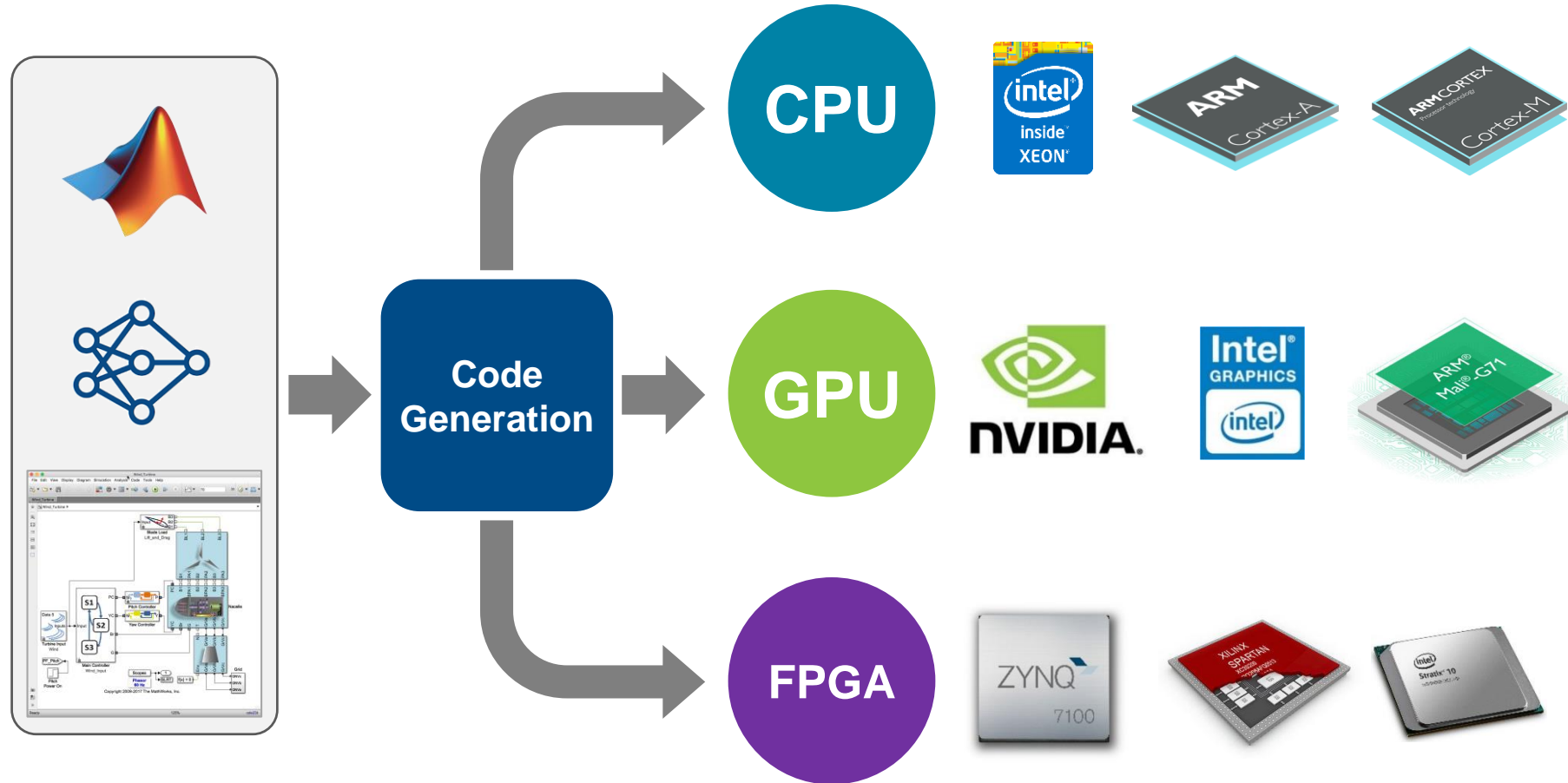
[Repository Link](#)



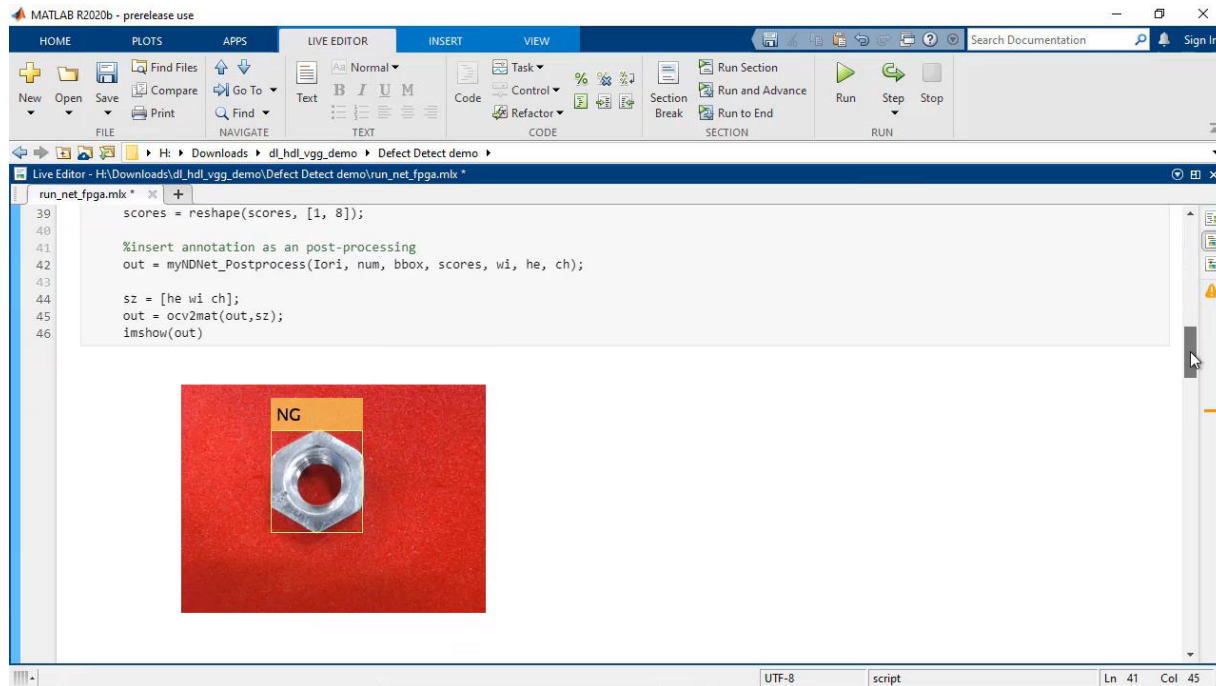
# Defect Detection Workflow



# Deploy to Any Processor with Best-in-class Performance



# Deploy to Hardware



Deploy defect detection algorithms from MATLAB to ZCU102 board from Xilinx



Deploy defect detection algorithms from MATLAB to Jetson AGX Xavier



# Deploy to Hardware

```
top - 22:06:20 up 1 day, 23:20, 3 users, load average: 1.55, 0.87, 0.37
Threads: 167 total, 3 running, 102 sleeping, 0 stopped, 0 zombie
%Cpu(s): 25.3 us, 0.9 sy, 0.0 ni, 73.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4080036 total, 2438672 free, 203504 used, 1437860 buff/cache
KiB Swap: 4194300 total, 4194300 free, 0 used, 3652220 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
29294	techcon	20	0	770124	208236	81036	R	89.7	5.1	2:59.16	nutsDet_exe
29310	techcon	20	0	770124	208236	81036	S	39.0	5.1	0:50.61	nutsDet_exe
29312	techcon	20	0	770124	208236	81036	R	39.0	5.1	0:50.22	nutsDet_exe
29311	techcon	20	0	770124	208236	81036	S	38.7	5.1	0:50.49	nutsDet_exe

Nuts Defect Detection Demo

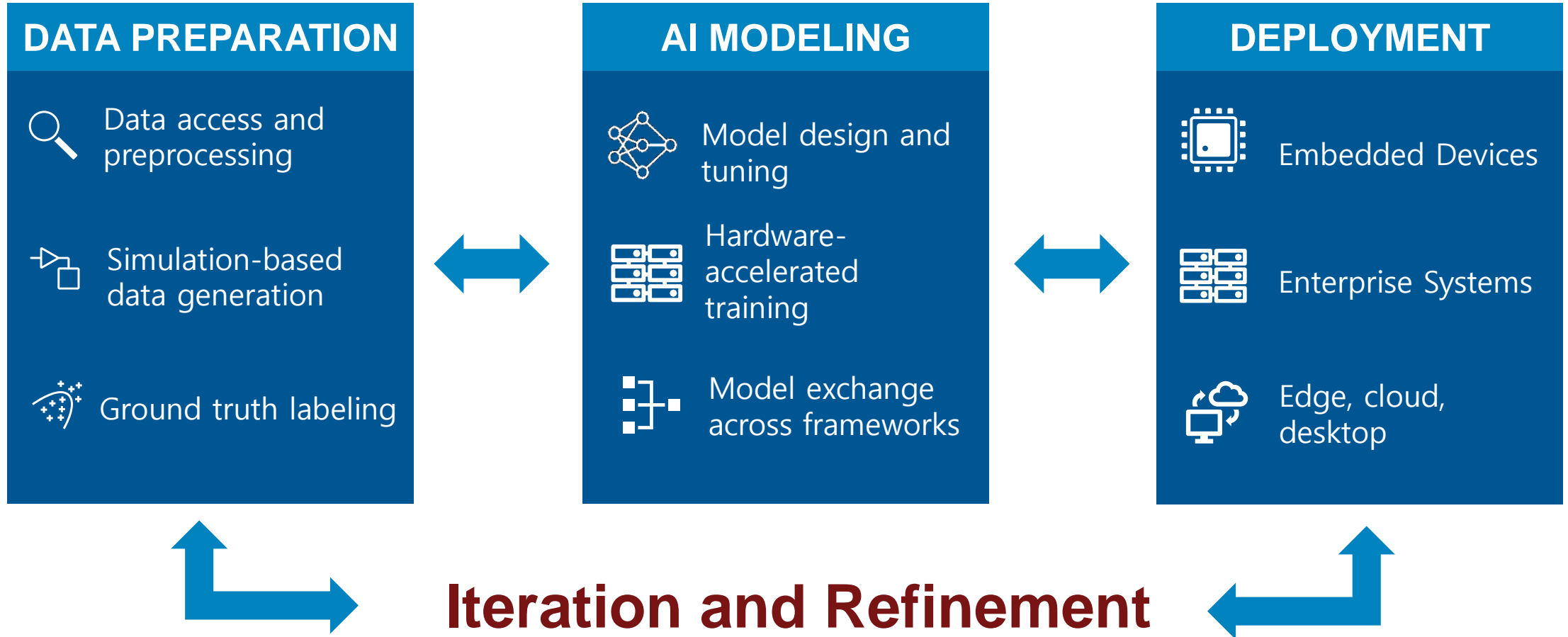
OK 1.87 FPS NG

Defect detection deployed on ARM Cortex-A microprocessor

## Additional Resources

- [Deploying Deep Neural Networks to GPUs and CPUs Using MATLAB Coder and GPU Coder](#)
- [Using GPU Coder to Prototype and Deploy on NVIDIA Drive, Jetson](#)
- [Real-Time Object Detection with YOLO v2 Using GPU Coder](#)
- [Image Classification on ARM CPU: SqueezeNet on Raspberry Pi](#)
- [Deep Learning on an Intel Processor with MKL-DNN](#)

# Defect Detection Workflow



# Key Takeaways

- Interactive and easy to use apps help explore, iterate and automate workflows
- Flexibility and options to choose networks and optimizations based on data and requirements
- MATLAB provides an easy and extensible framework for defect detection from data access to deployment

# MATLAB EXPO

*Thank You*

