

# MATLAB EXPO

## Automated Optical Inspection and Defect Detection with Deep Learning

*Harshita Bhurat*

*Product Manager – Image Processing and Computer Vision*



# 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

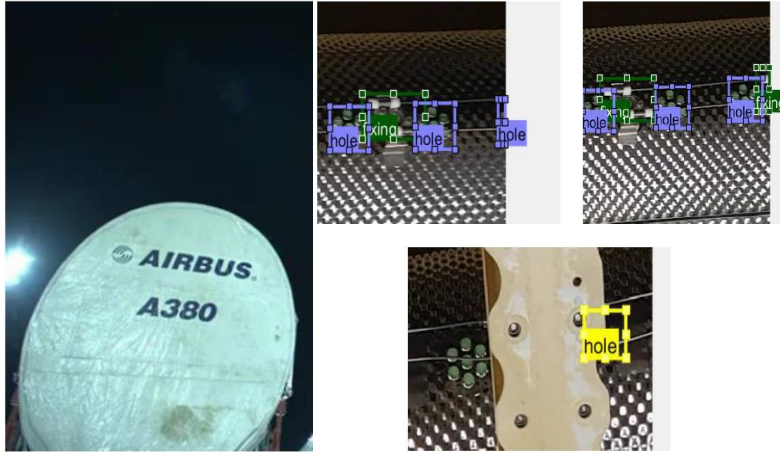
Machine Vision

Visual Inspection

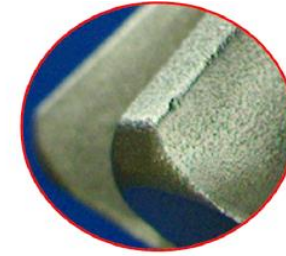
Automated Inspection

# Customer References

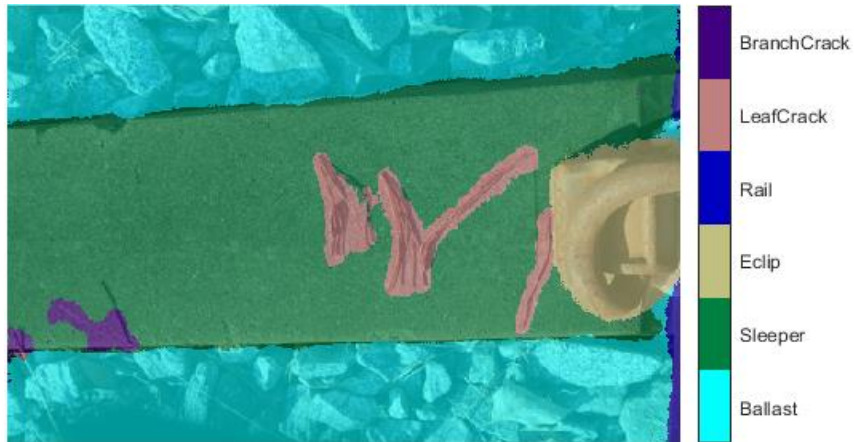
**AIRBUS**



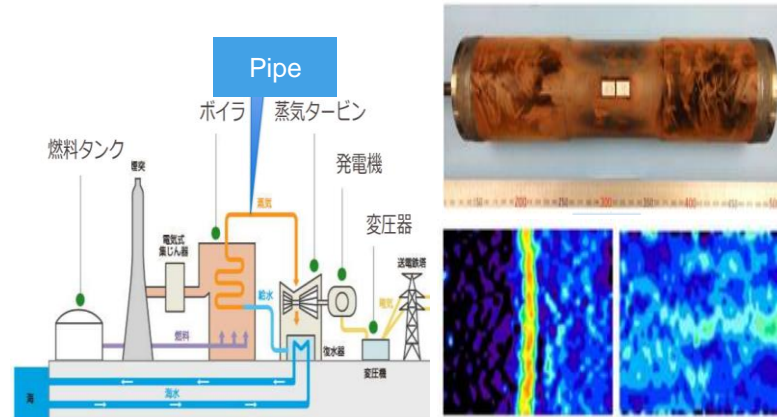
Automatic Defect Detection



Visual Inspection of Automotive Parts

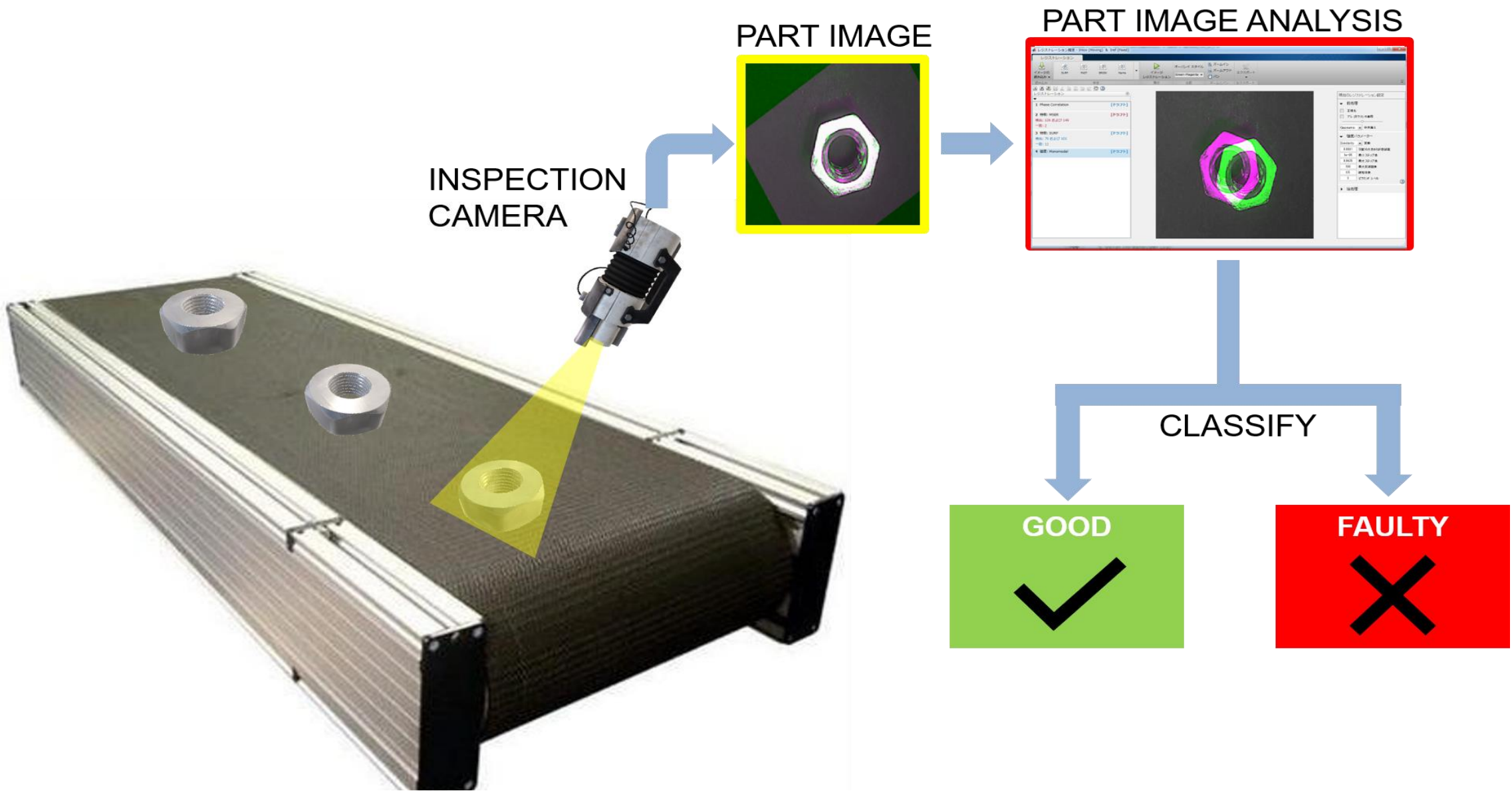


Defect Detection in Railway Components



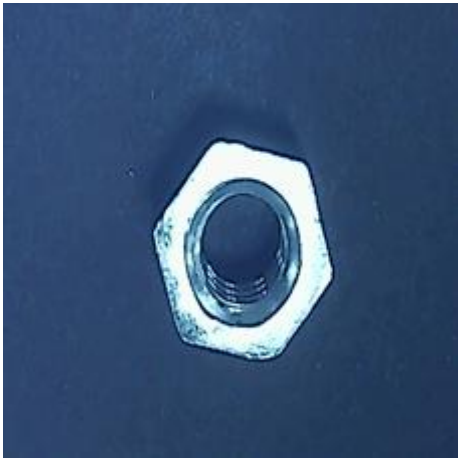
Assess Pipe Weld Damage at Power Plants

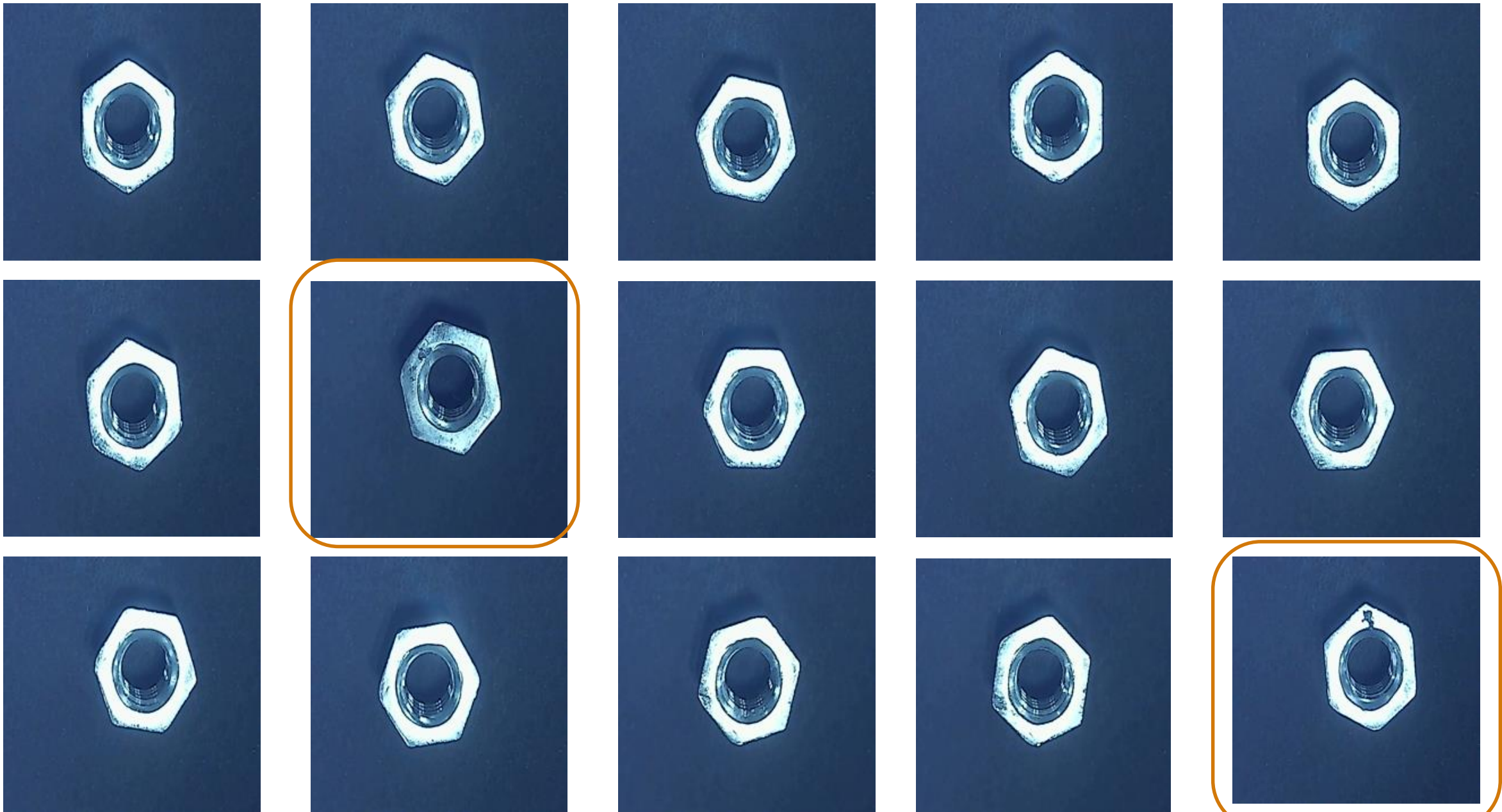


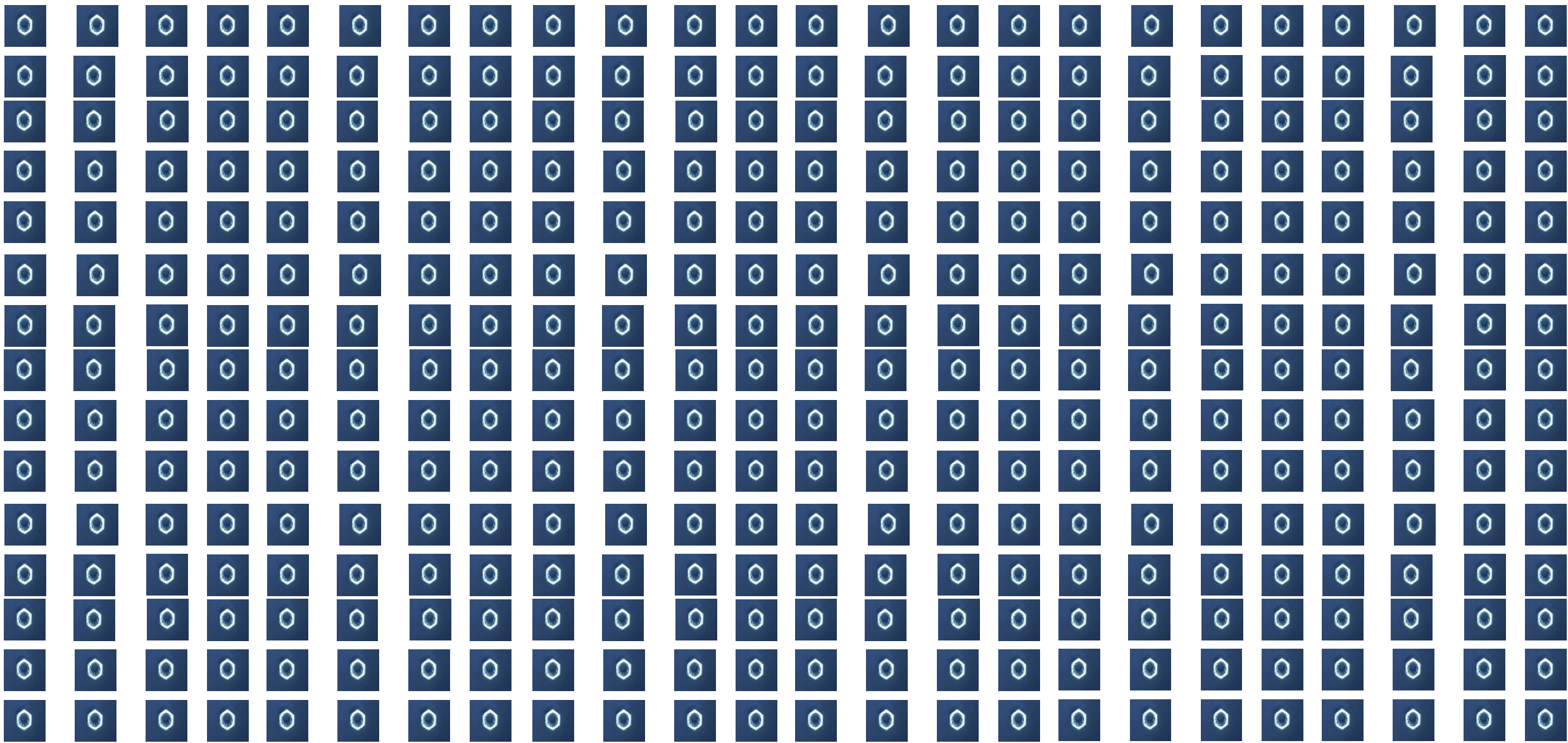


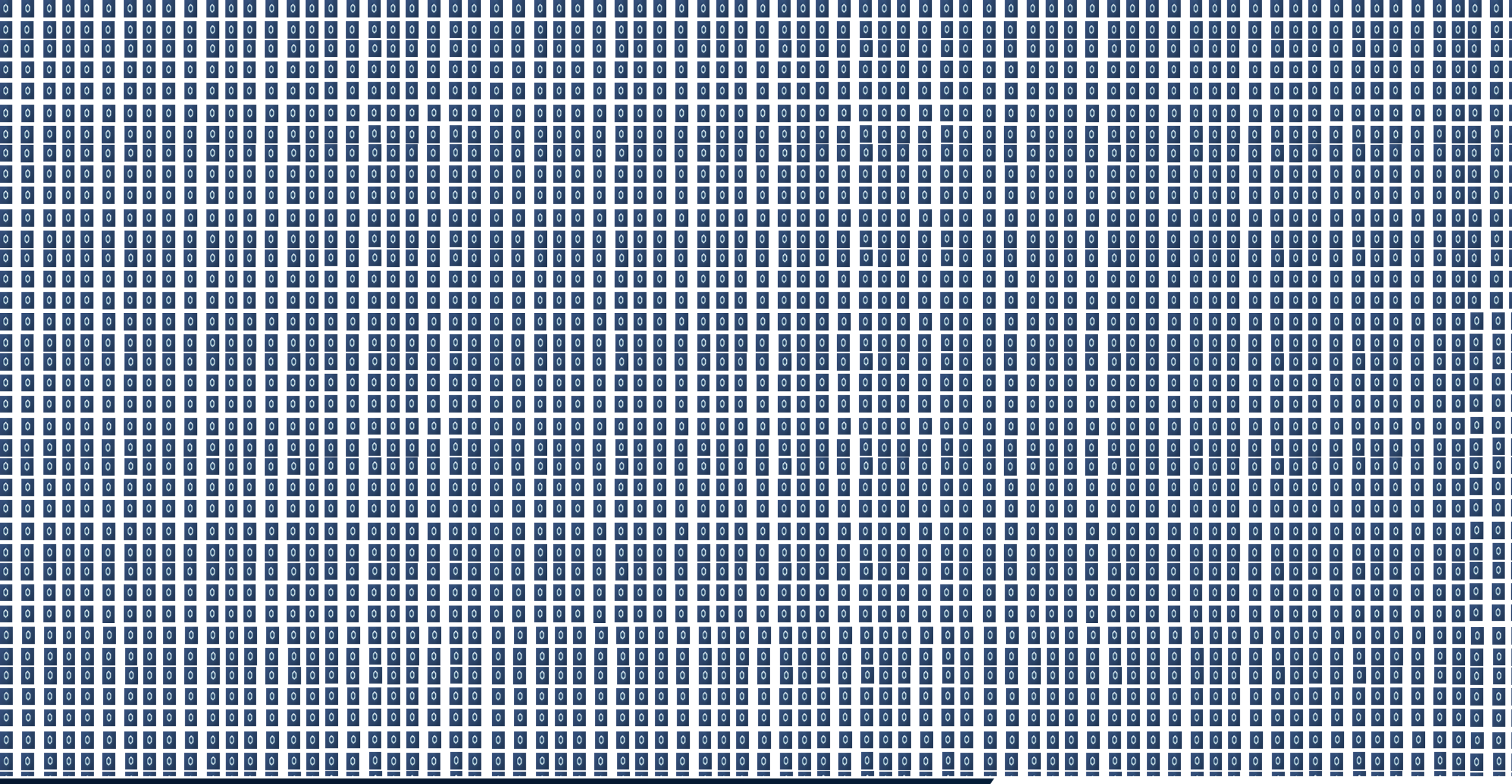


# Can you find the defective hex nut?





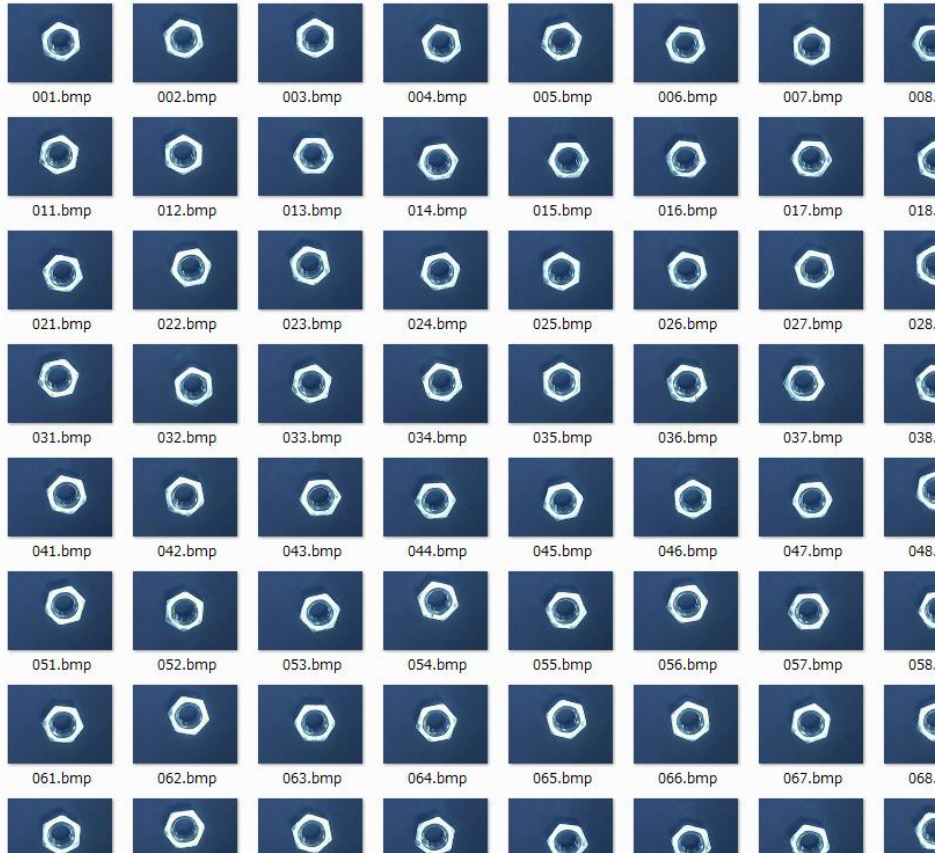




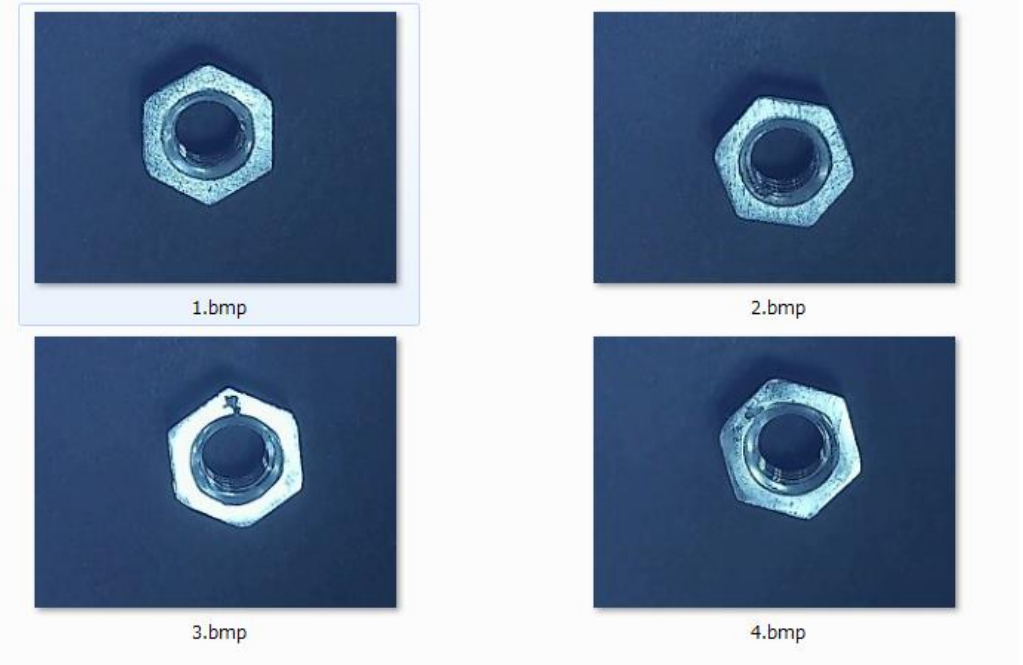


# Finding Defective Hex Nuts

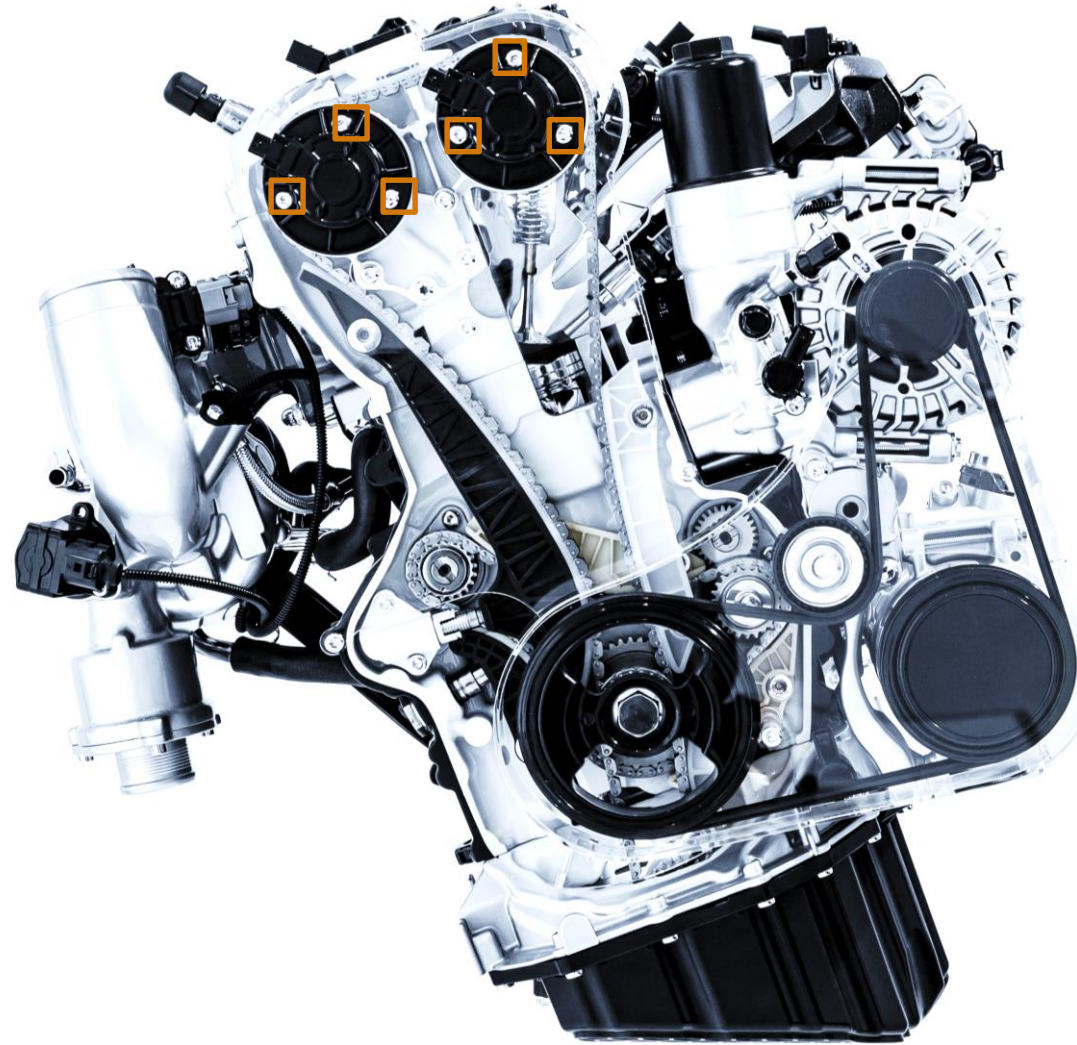
## Good



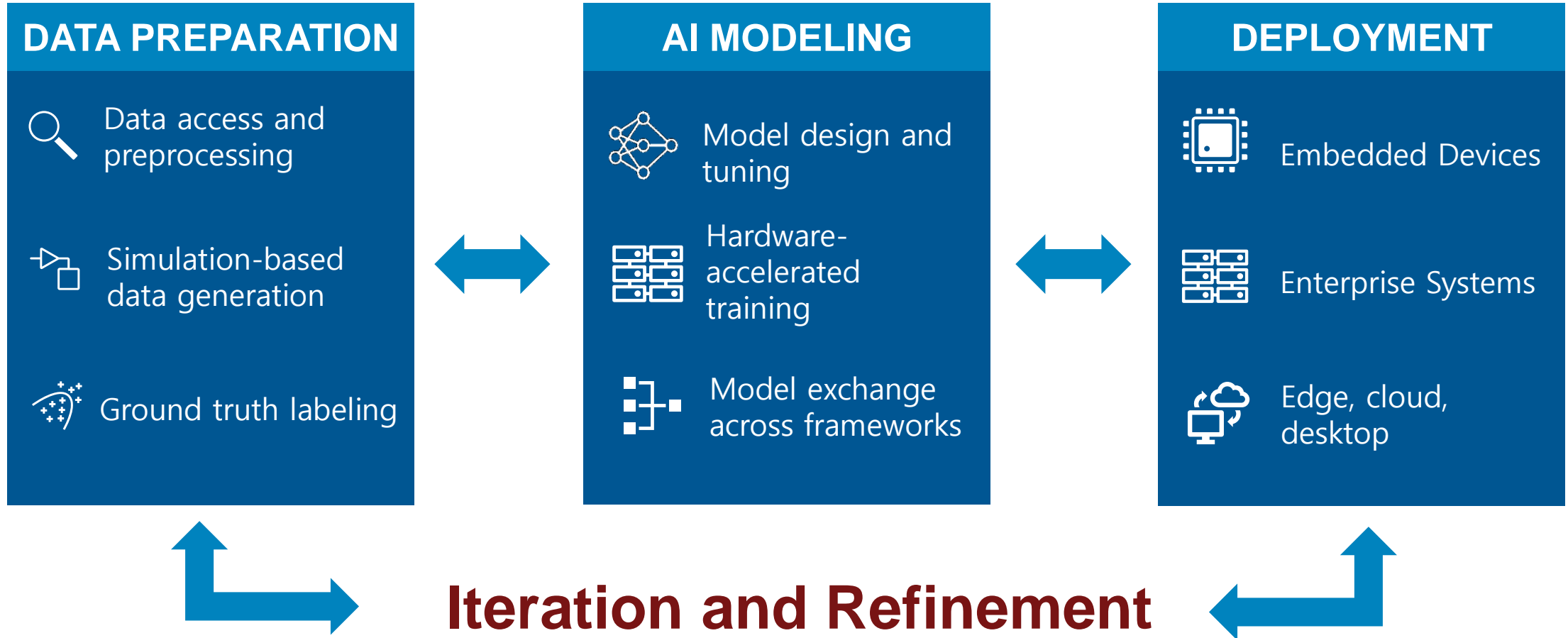
## Defective



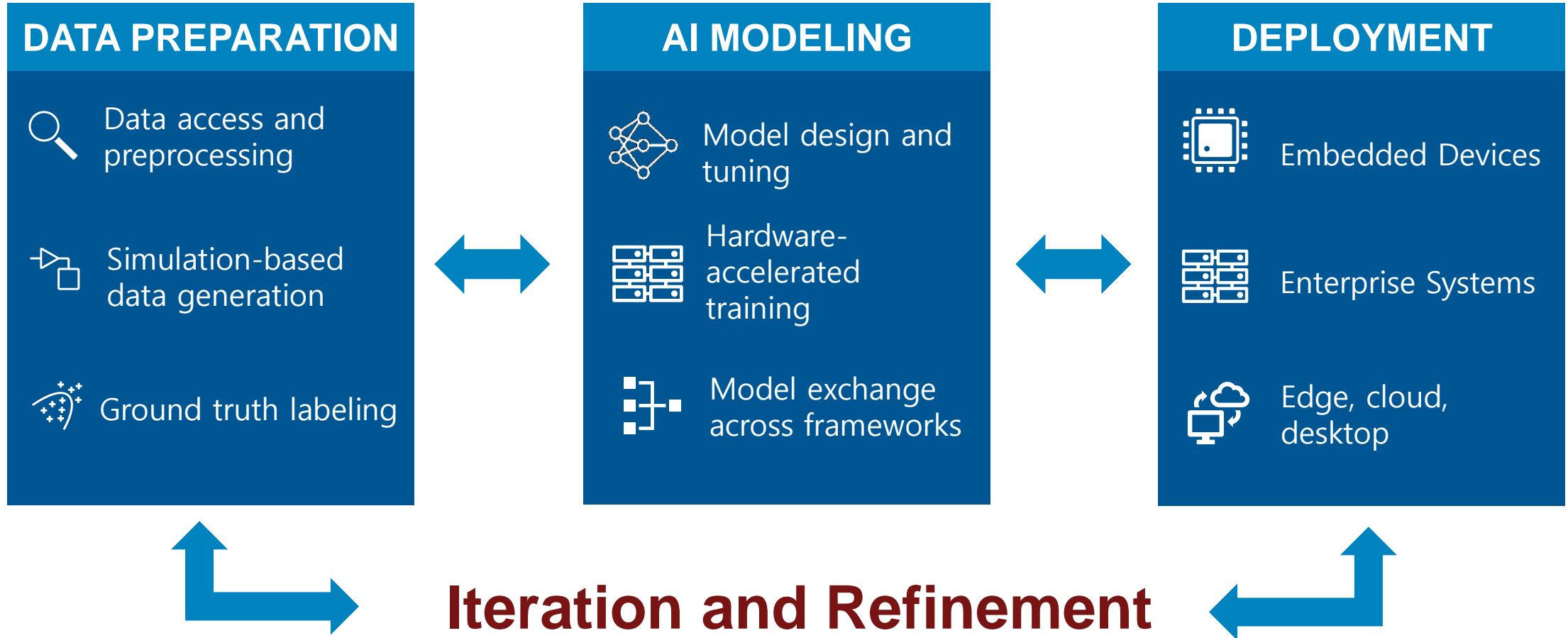
# Detecting Parts



# 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?

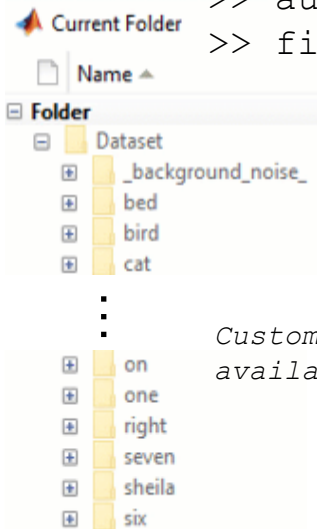
How do I access large data that might not fit in memory?

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

## Datstores

Loads image/signal data into memory as and when needed

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

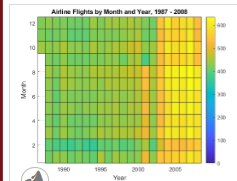


*Custom Datstores also available*

## Tall Arrays

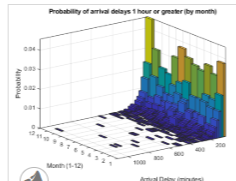
Work with out-of-memory numeric data

- Train deep neural networks for numeric arrays



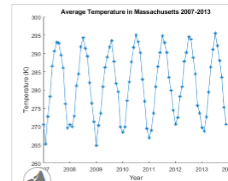
Analyze Big Data in MATLAB Using Tall Arrays

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.

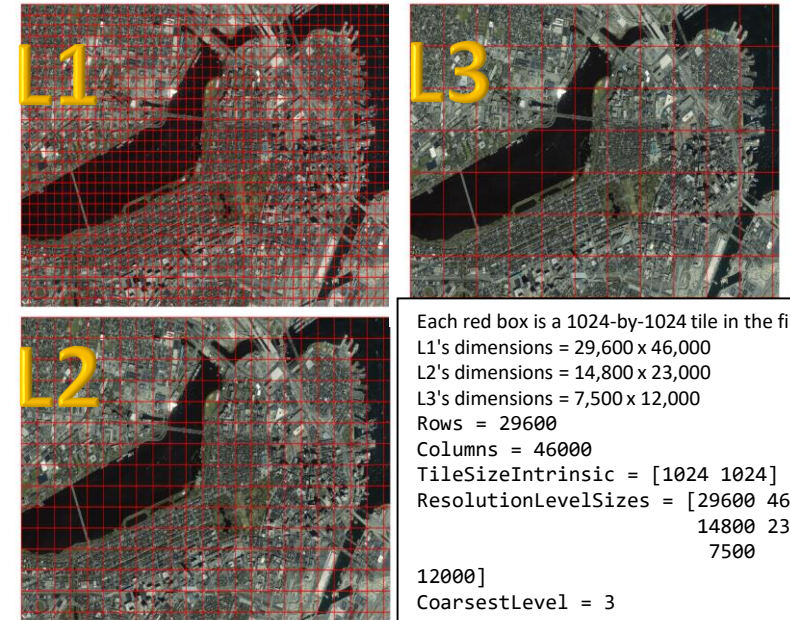


Process Big Data in the Cloud

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?

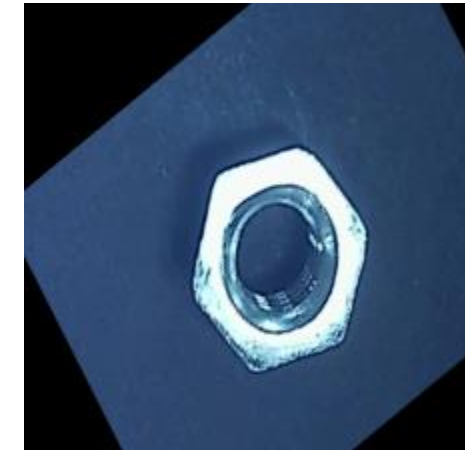
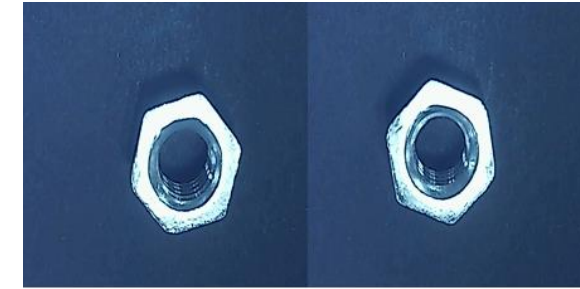


How do I preprocess data  
and get the right features?

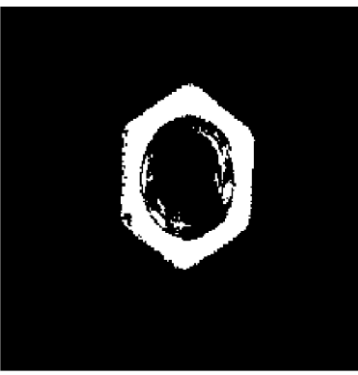
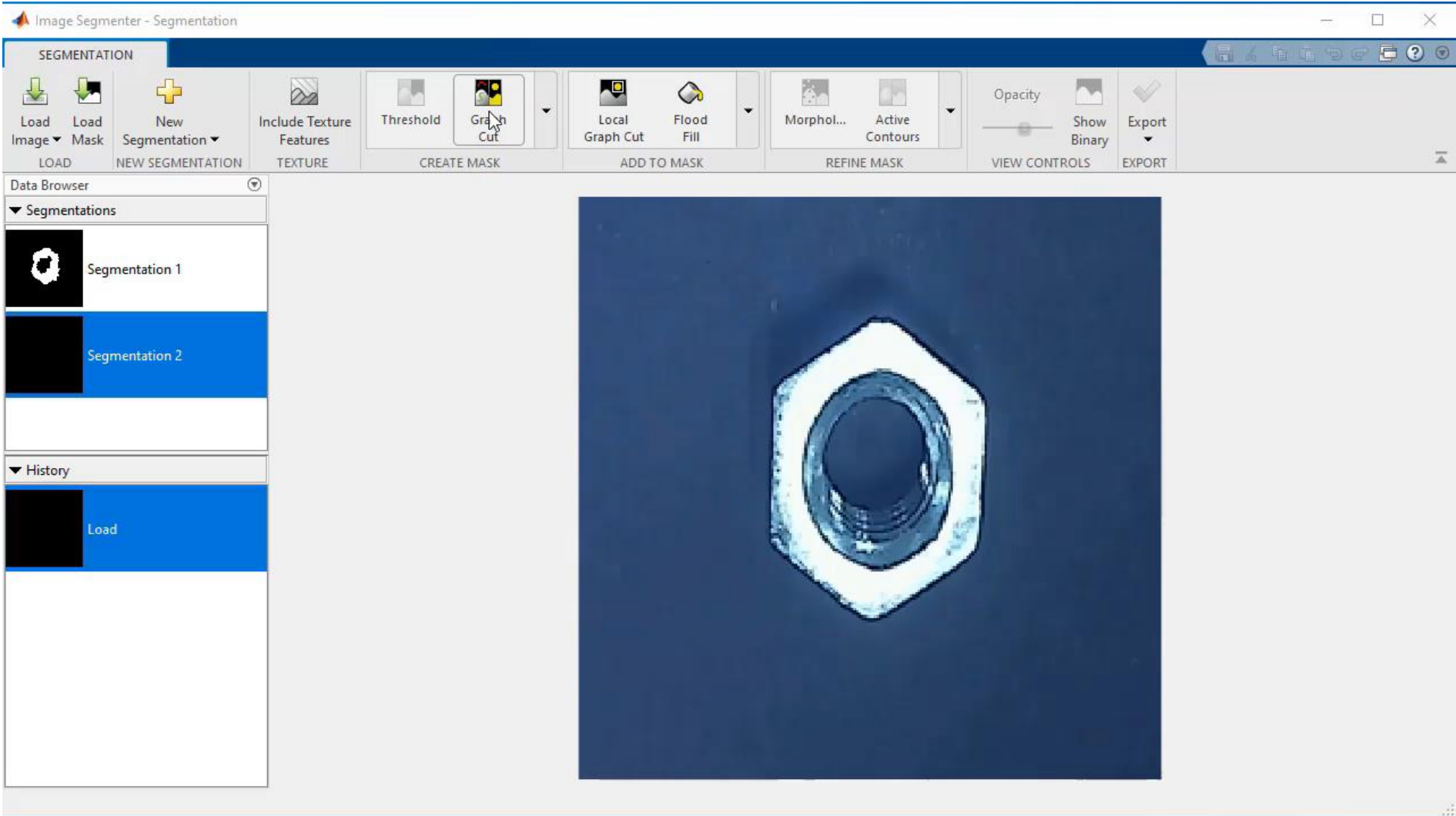
# Pre-processing Data – Registration Estimator App

The screenshot displays the Registration Estimator app interface. The title bar reads "Registration Estimator - Imov (Moving Image) & Iref (Fixed Image)". The main window is divided into several sections:

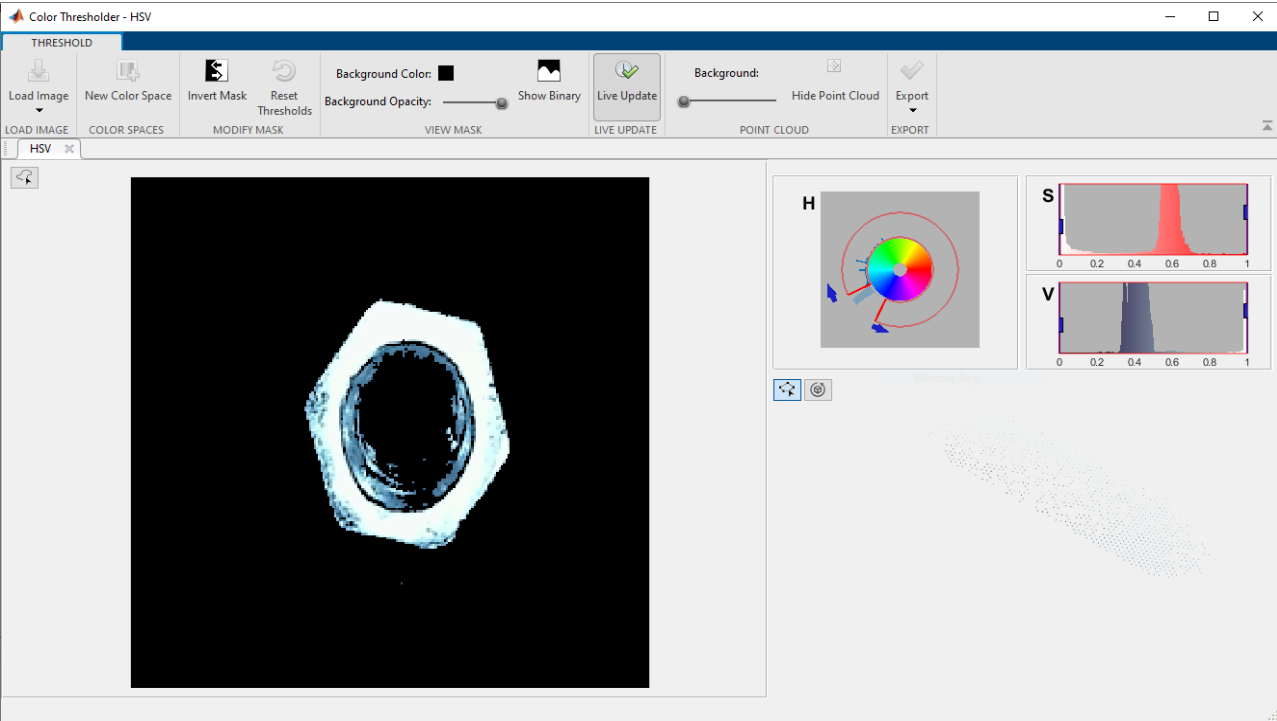
- REGISTRATION** toolbar: Includes "Load Images" (LOAD), "SURF", "FAST", "BRISK", "Harris" (TECHNIQUE), "Register Images" (RUN), "Overlay Style" (Green-Magenta), and "Export" (EXPORT).
- Registrations** list (left):
  - 1 Phase Correlation [DRAFT]
  - 2 Feature: MSER [DRAFT]  
Detected: 104 and 101  
Matched: 5
  - 3 Feature: SURF [DRAFT]  
Detected: 79 and 101  
Matched: 12
- Central Image:** A grayscale image of a nut with two overlapping images. One is colored green and the other magenta. Yellow lines connect corresponding feature points between the two images.
- Current Registration Settings** (right):
  - Feature Parameters
    - Projective Transformation
    - Number of Detected Features (slider)
    - Quality of Matched Features (slider)
    - Has Rotation
  - Post-processing



# Pre-processing Data – Image Segmenter App



# Preprocessing Data - Apps



Color Thresholder

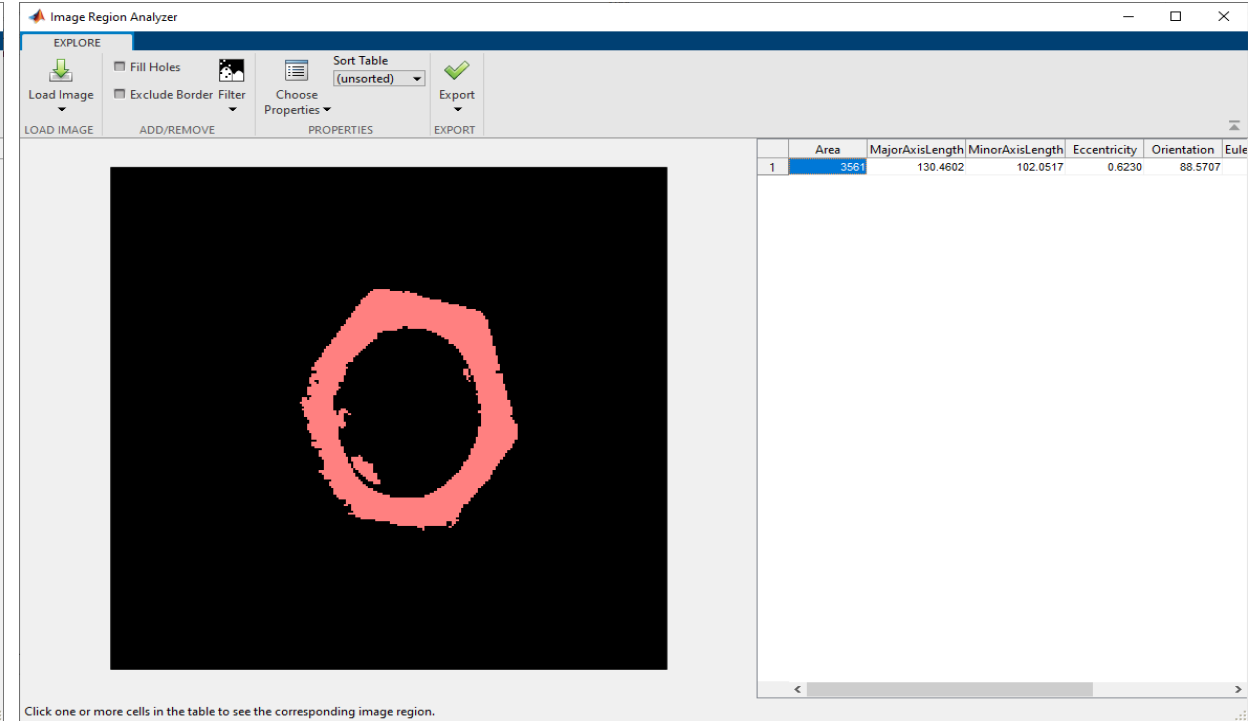
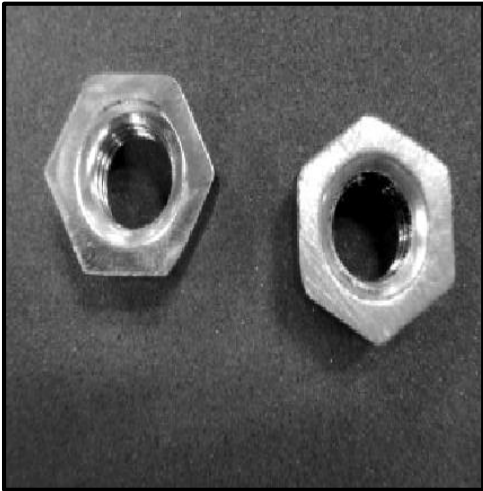


Image Region Analyzer

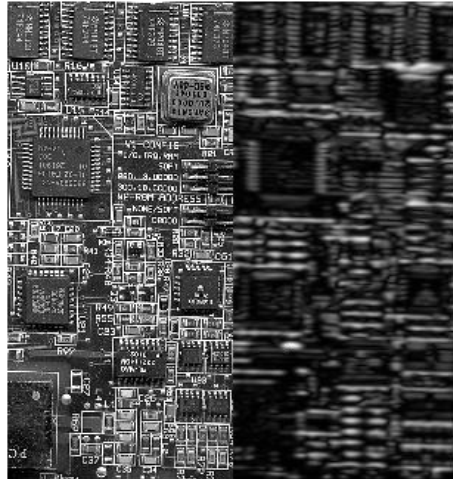


# Pre-processing Data – Built-in Algorithms

imadjust



imgaborfilt



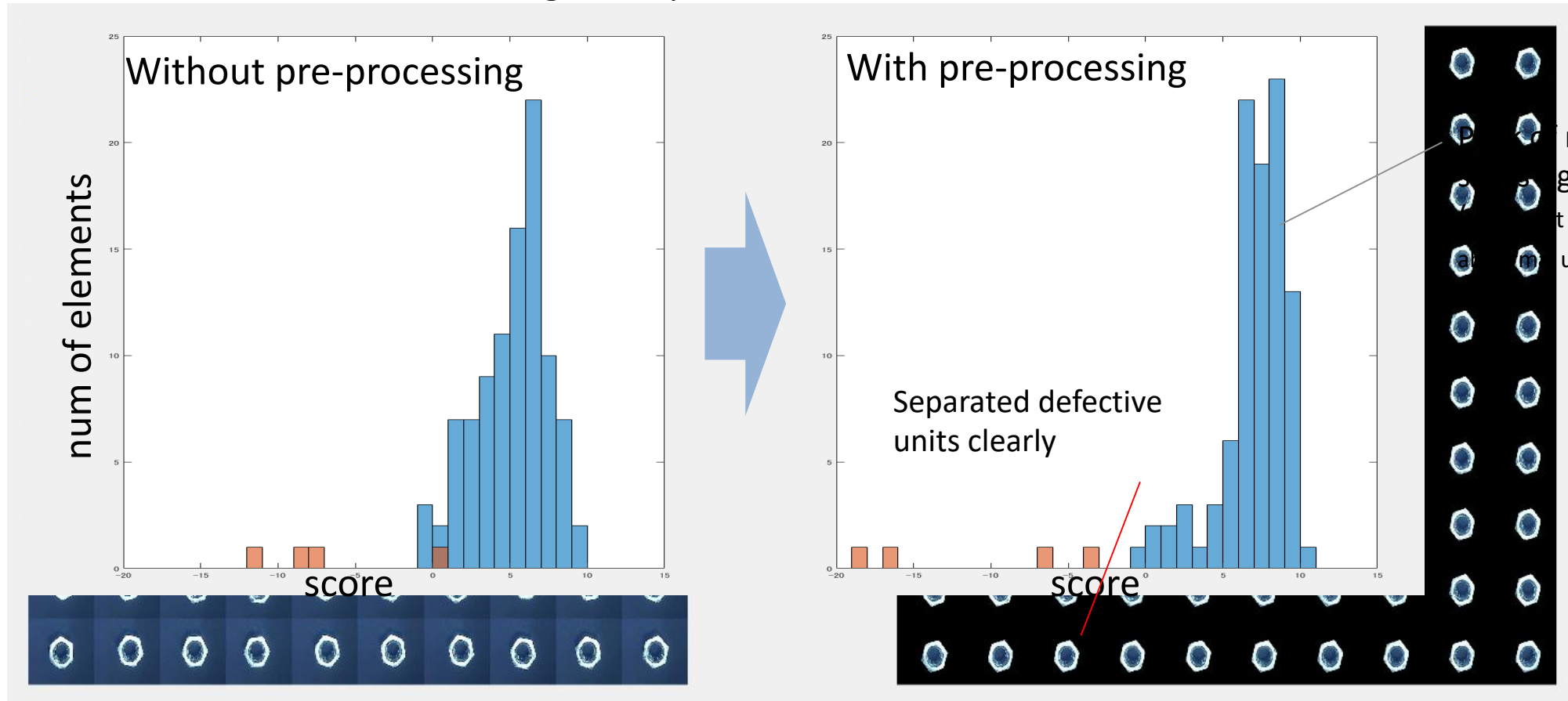
fibermetric



# Defect detection using AlexNet: Results with preprocessing

Without pre-processing

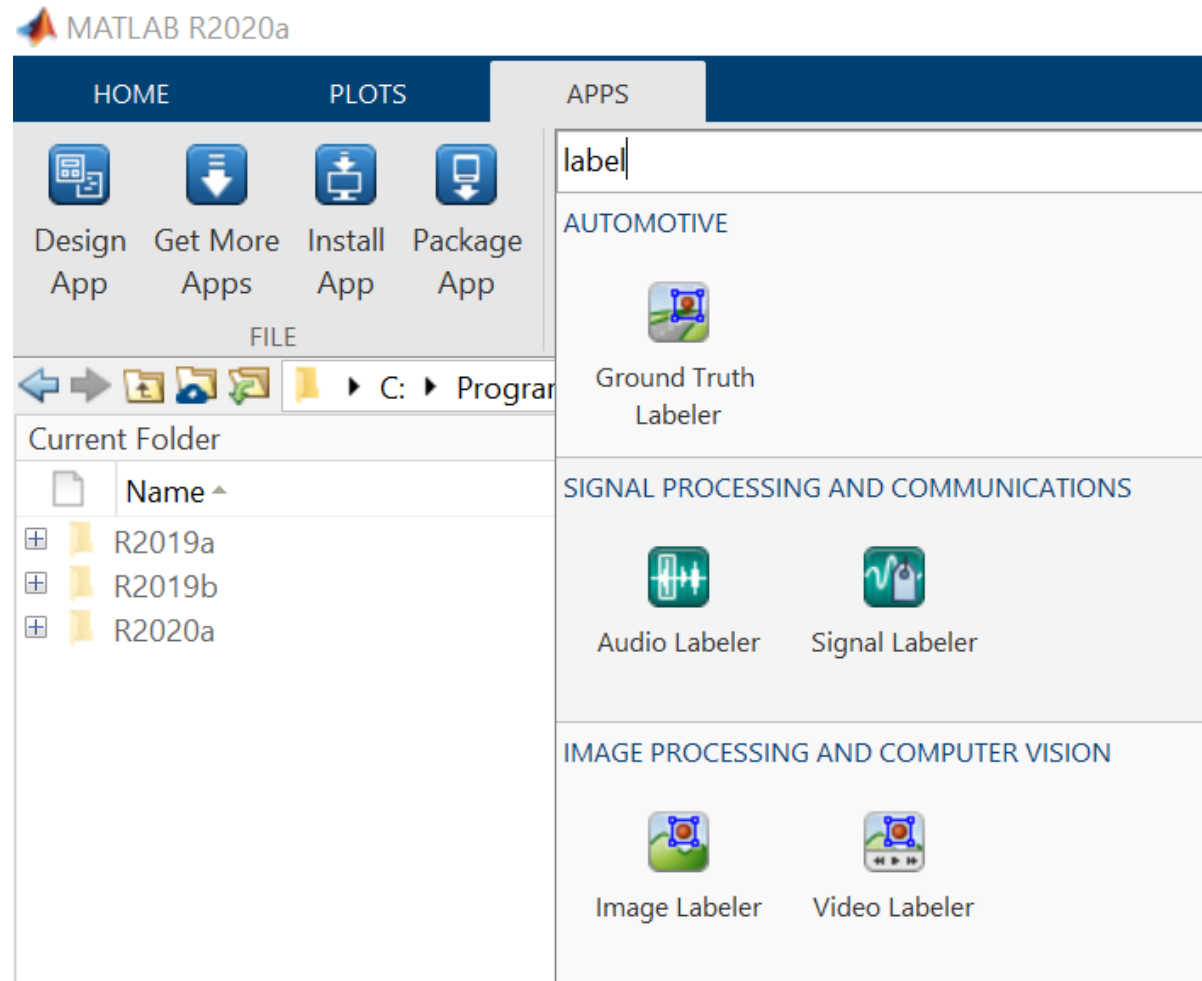
With pre-processing



# Data Access and Preprocessing – Common Challenges

How do I label my data faster?

# Data Preprocessing - Labeling

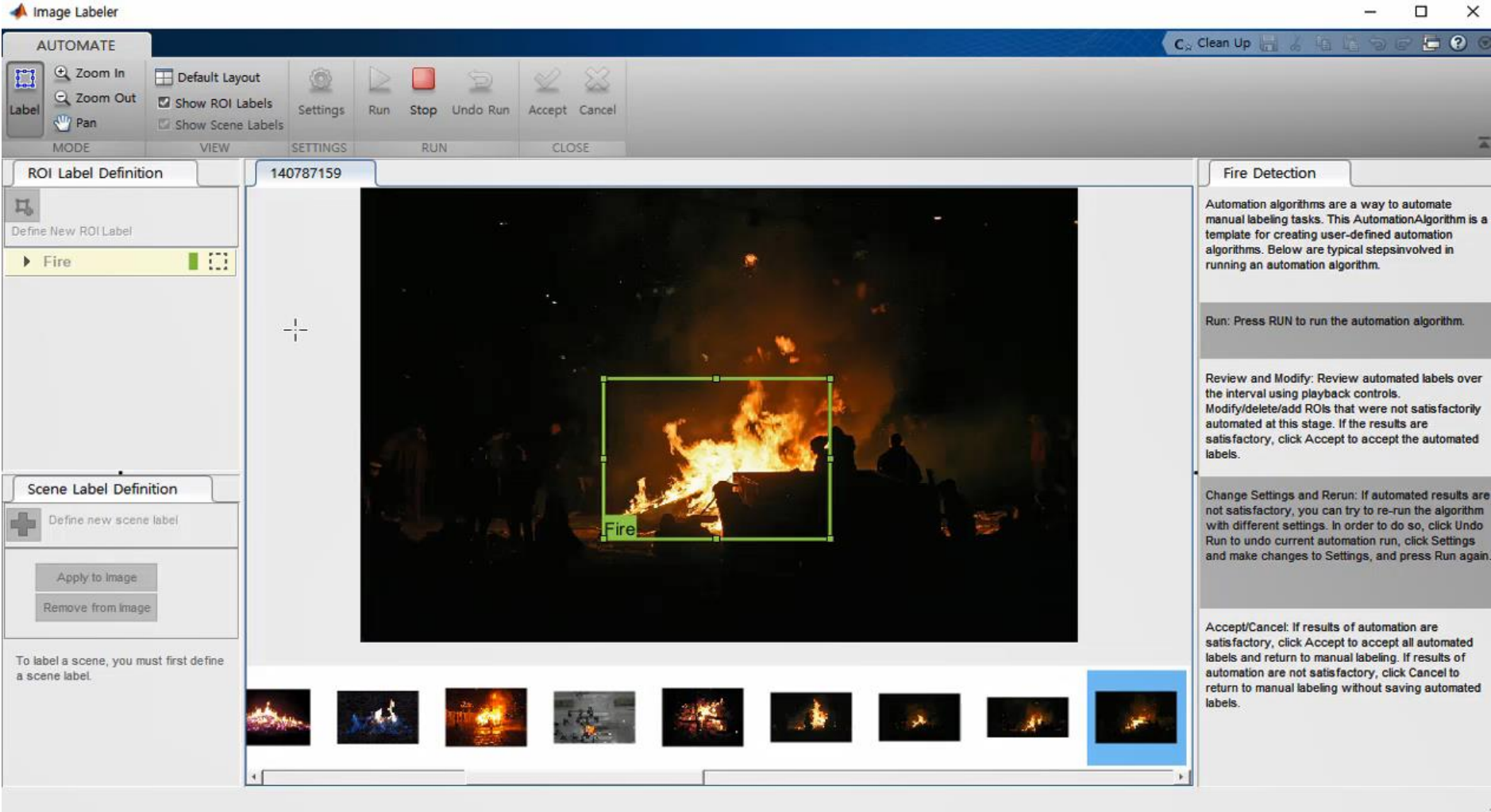




# Image & Video Labeler

Image Labeler + Video labeler

Big-Image Labeler



# Big Image Labeler

Image Labeler  
+ Video 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.

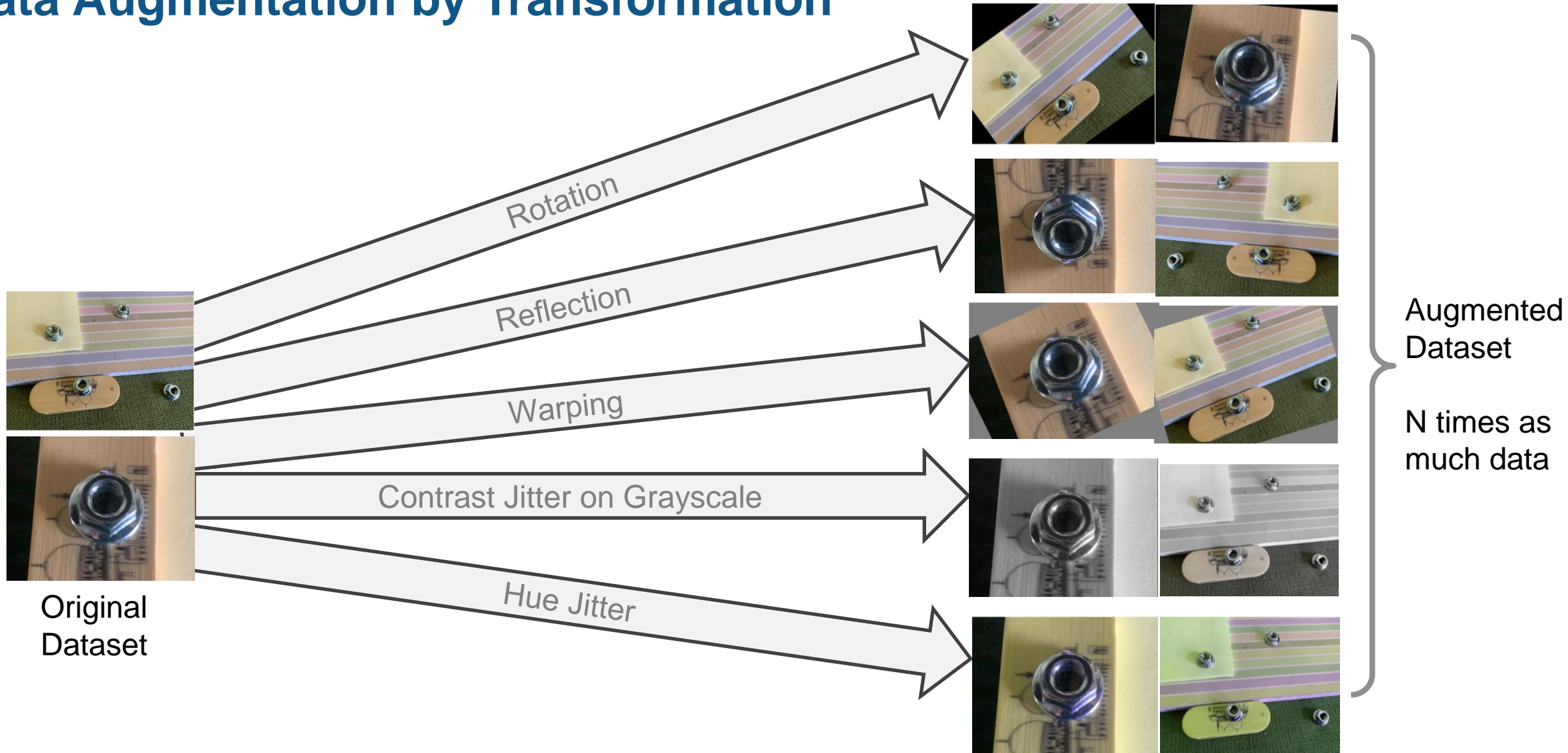
The main workspace is divided into two panels. The left panel, titled "Overview Region: [42497 48641 1536 1024]", shows a circular overview of a large image with a small purple square indicating the current subimage's location. Below this panel is a "Tooling/Options" section with several sub-sections: "DISPLAY/EXPORT Annotations", "Define/Extract Training Chips", "Recall Session", "LABEL VISIBILITY" (with checkboxes for Overview and Subimage), "Labeling Mode" (with radio buttons for Manually Label, Predict/Verify, and Rectangle), and "ROI Type" (with radio buttons for Freehand and Rectangle). The "Segmentation/Filter Settings" section includes "Sensitivity" (0 to 1), "Border Simplification" (0 to 1), "Minimum Size" (0 to 1e6), "Allow holes?", "Segmentation Option" (Threshold, Imextended, Gray Connectivity, Custom), and "Border Simplification" (Subimage, Overview Image) with "Max Area Pct" (0.85) and "Max Overlap" (0.7). A "LABEL" table is also present, listing "InitialAutoSegmentation" regions with IDs 1 through 4.

The right panel, titled "Subimage from: Scan004\_cellspot\_pyramid", shows a magnified view of a cell spot image. The axes are labeled with coordinates: X-axis from 42500 to 44000, and Y-axis from 48700 to 49600. The image shows several cell spots, with three of them outlined in magenta and labeled "InitialAutoSegmentation" in green boxes. Below the image is a "LABEL" table with columns "LABEL" and "ID", listing "InitialAutoSegmentation" regions with IDs 1 through 4. The table also includes buttons for "Select By Label", "GoTo (first)", "Delete Selected", "Rename Selected", and "Validate".

# Data Access and Preprocessing – Common Challenges

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

# Data Augmentation by Transformation



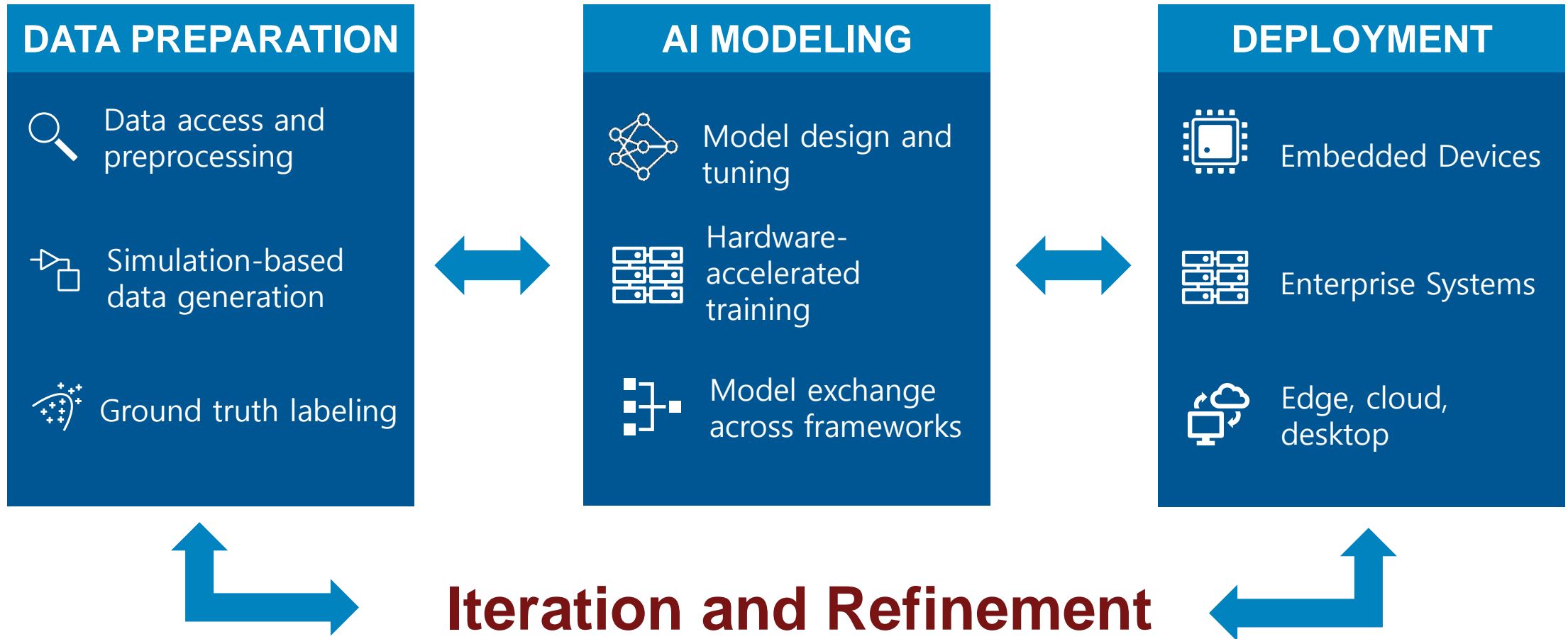


# Data Augmentation : Generative Adversarial Networks (GANs)



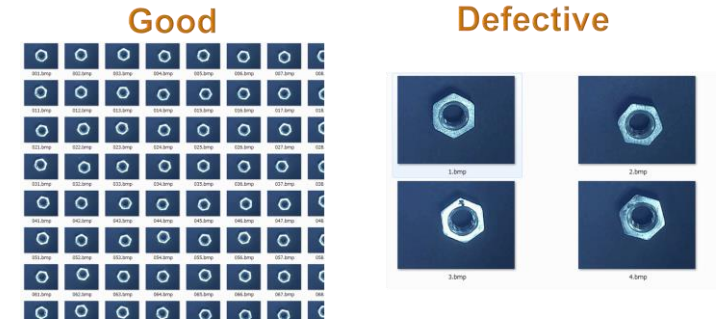


# Defect Detection Workflow

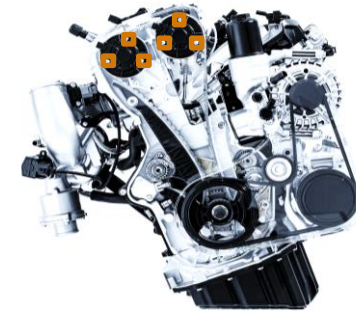


# Deep Learning for Defect Detection

Deep learning for  
Classification



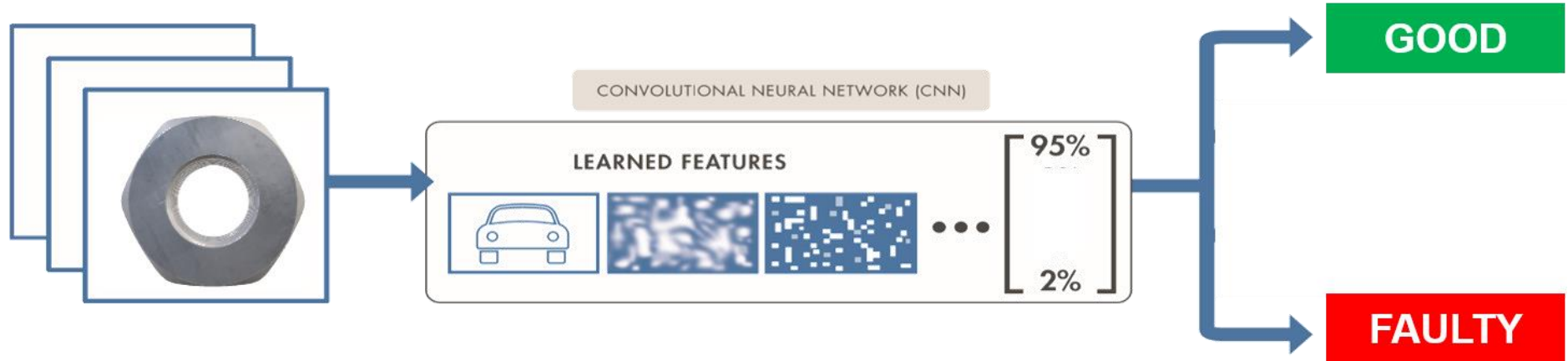
Deep Learning for Object  
Detection



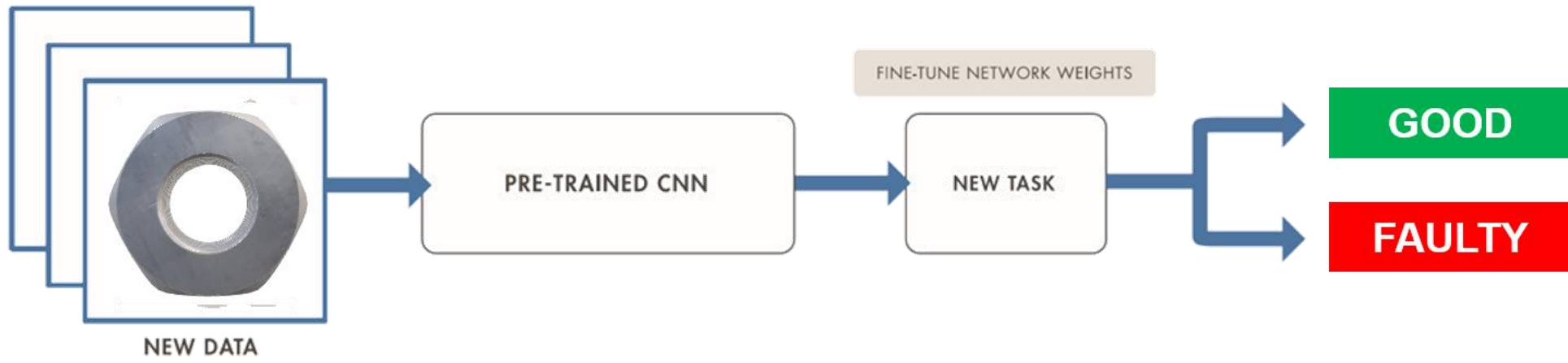
## Deep learning for Classification

# Two Approaches for Deep Learning

## 1. Train a deep neural network from scratch



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

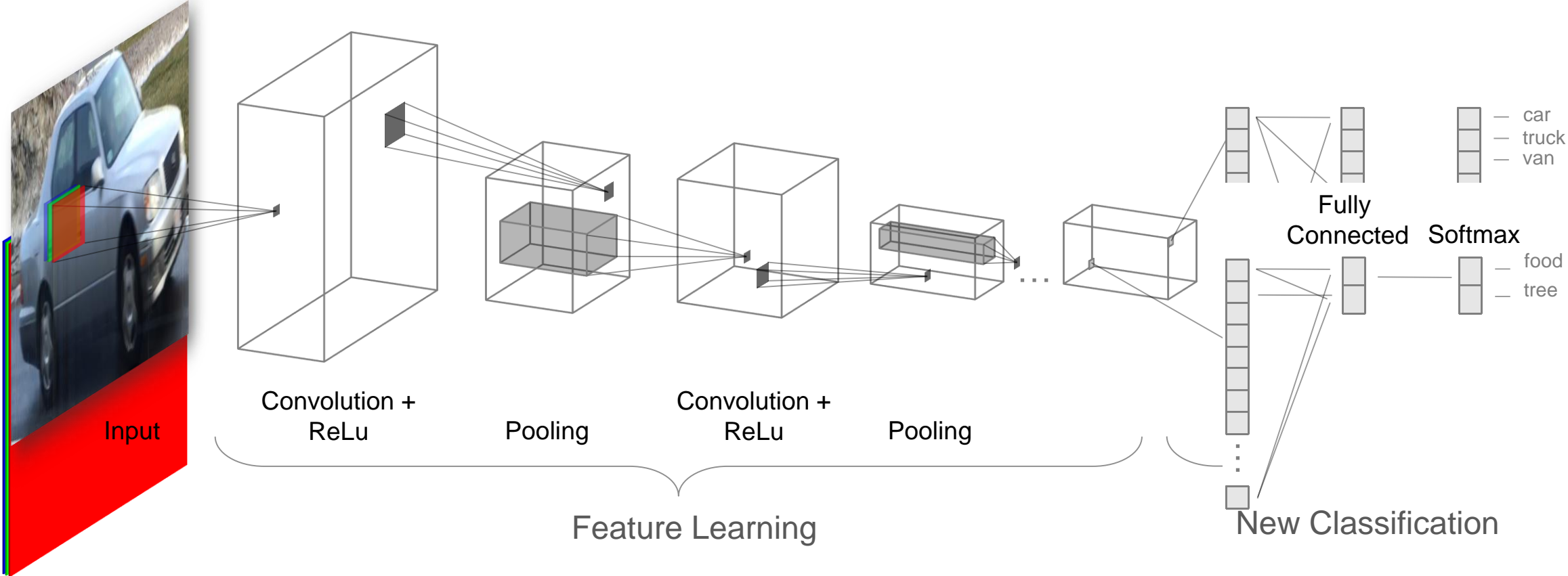


# Train a Deep Neural Network from Scratch

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'. The 'fc' layer is selected in the library. On the right, the 'PROPERTIES' panel shows the configuration for the 'fullyConnectedLayer' (name: 'fc', input size: 'auto', output size: '10', weights: '[]', bias: '[]', weight learn rate factor: '1', weight L2 factor: '1', bias learn rate factor: '1', bias L2 factor: '0', weights initializer: 'glorot', bias initializer: 'zeros'). Below the properties is an 'OVERVIEW' section showing 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)

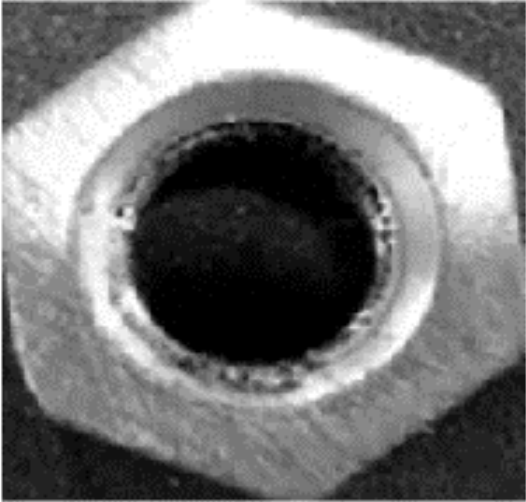
The screenshot displays the MATLAB Deep Network Designer software interface. At the top, the title bar reads "Deep Network Designer" and the main header says "MATLAB Deep Network Designer". Below the header, there are navigation links: "Getting Started", "Compare Pretrained Networks", and "Transfer Learning". The main area is a grid of 21 neural network architecture diagrams, each with a label and a warning icon (a yellow triangle with an exclamation mark). The architectures shown are:

- SqueezeNet
- GoogLeNet
- ResNet-50
- DarkNet-53
- DarkNet-19
- ShuffleNet
- NasNet-Mobile
- NasNet-Large
- Xception
- Places365-Goog...
- MobileNet-v2
- DenseNet-201
- ResNet-18
- Inception-ResNe...

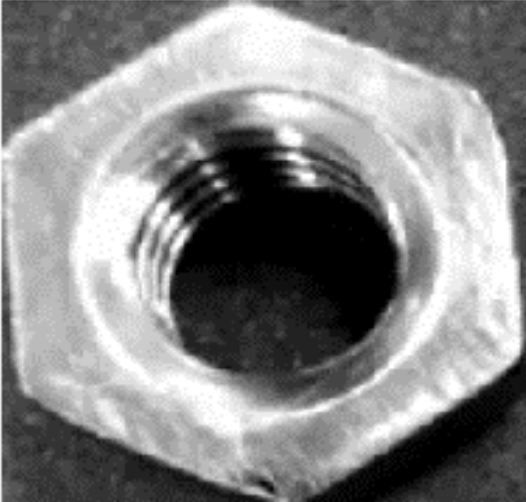
The diagrams use colored boxes (orange, green, purple) to represent different layers and their connections, illustrating the unique structures of each model.

# Classification with Trained MobileNetV2

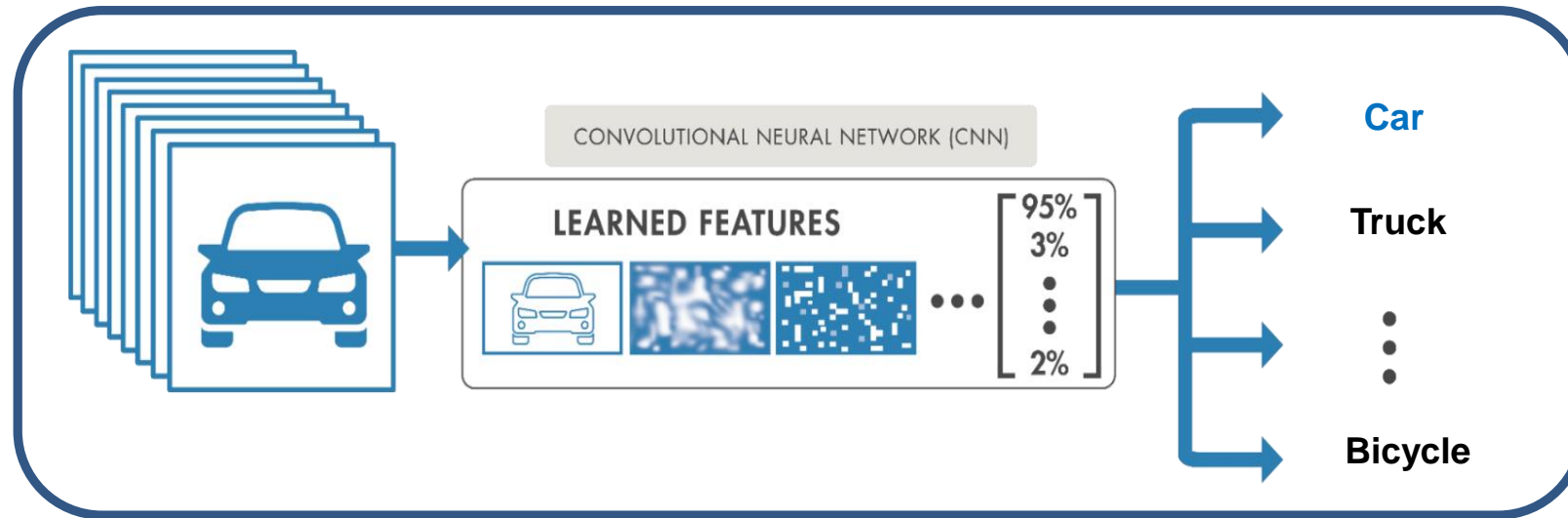
Bad



Good



# Challenges with Deep Learning Models

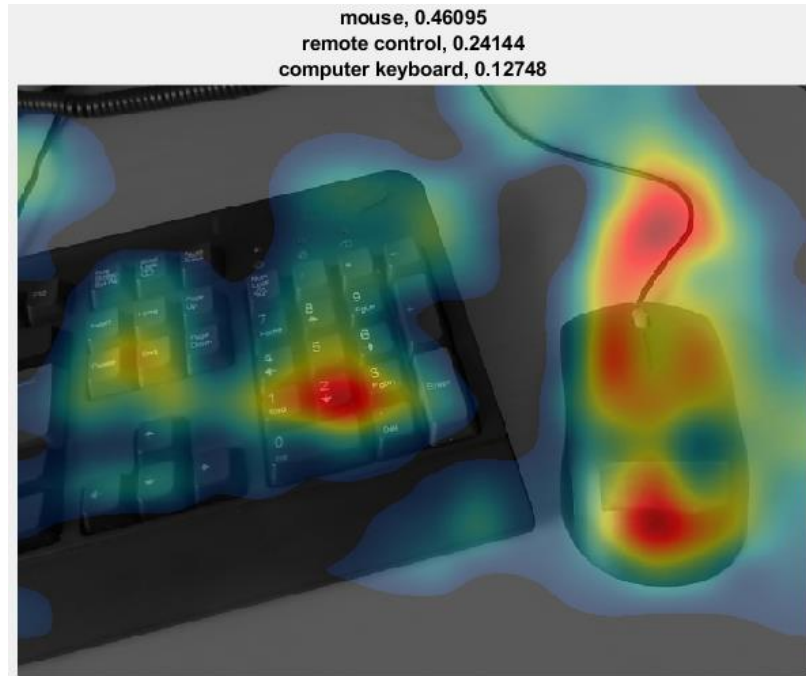


*Explainable AI  
is required*

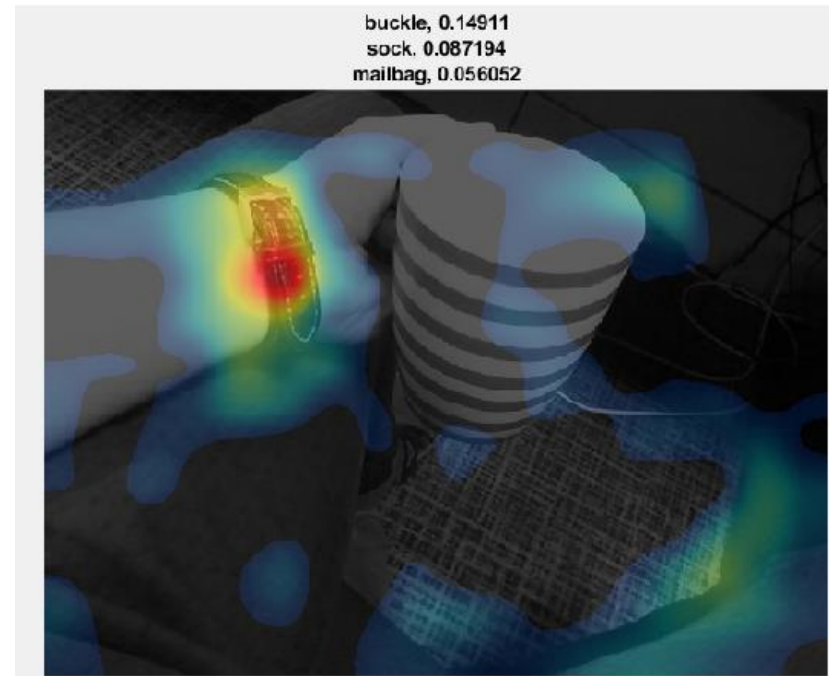


- [Class Activation Mapping \(CAM\)](#)
- [Grad-CAM](#)

# Class Activation Mapping to Investigate Network Predictions



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

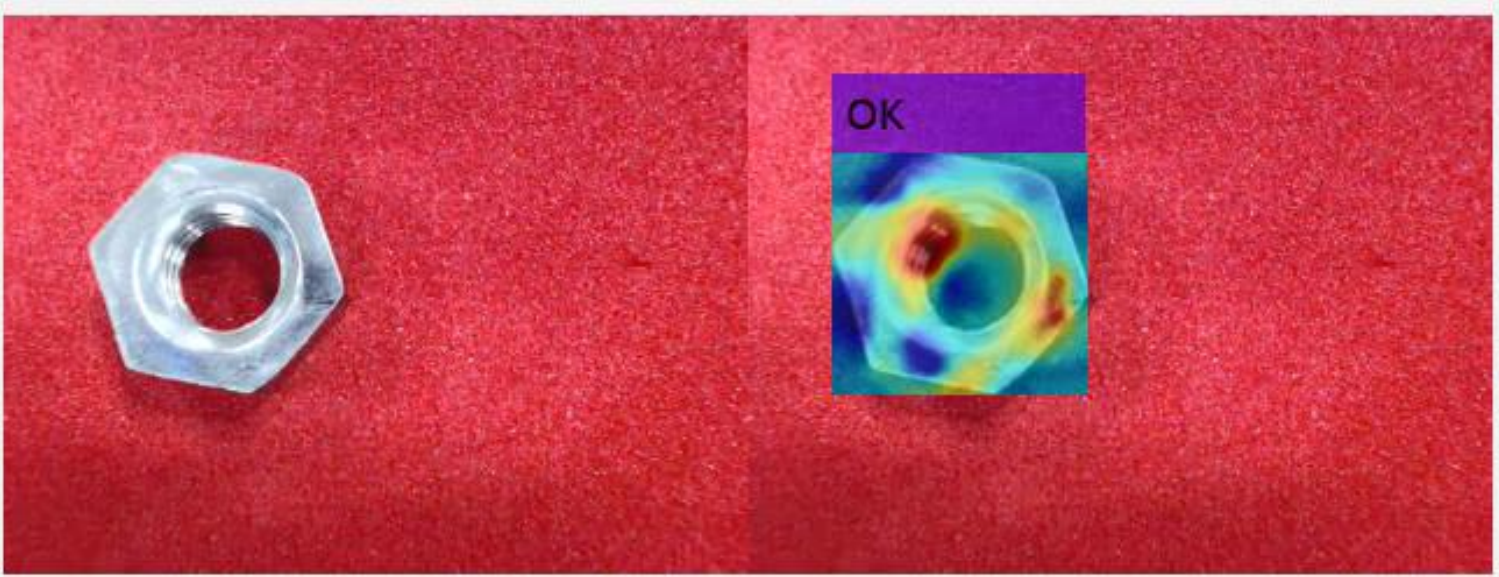




Incorrectly classified “coffee mug” as “buckle” due to the watch

# Visualization of Features with CAM

Captured Image

Classification and CAM

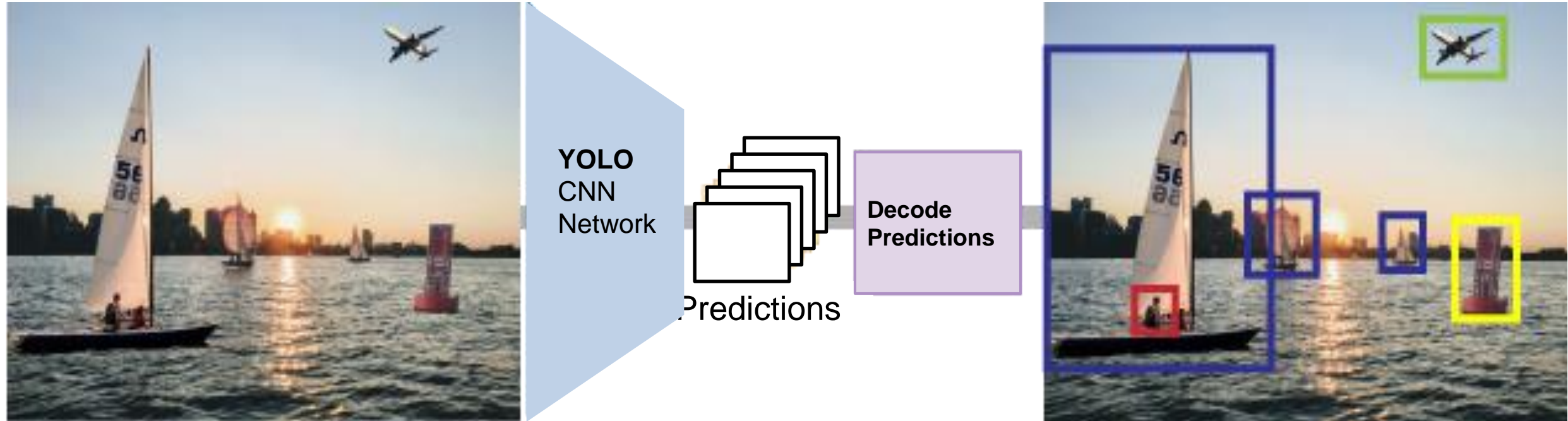


-  OK → Reacts to whole surface
-  Bad → Reacts to the scratch

## Deep Learning for Object Detection

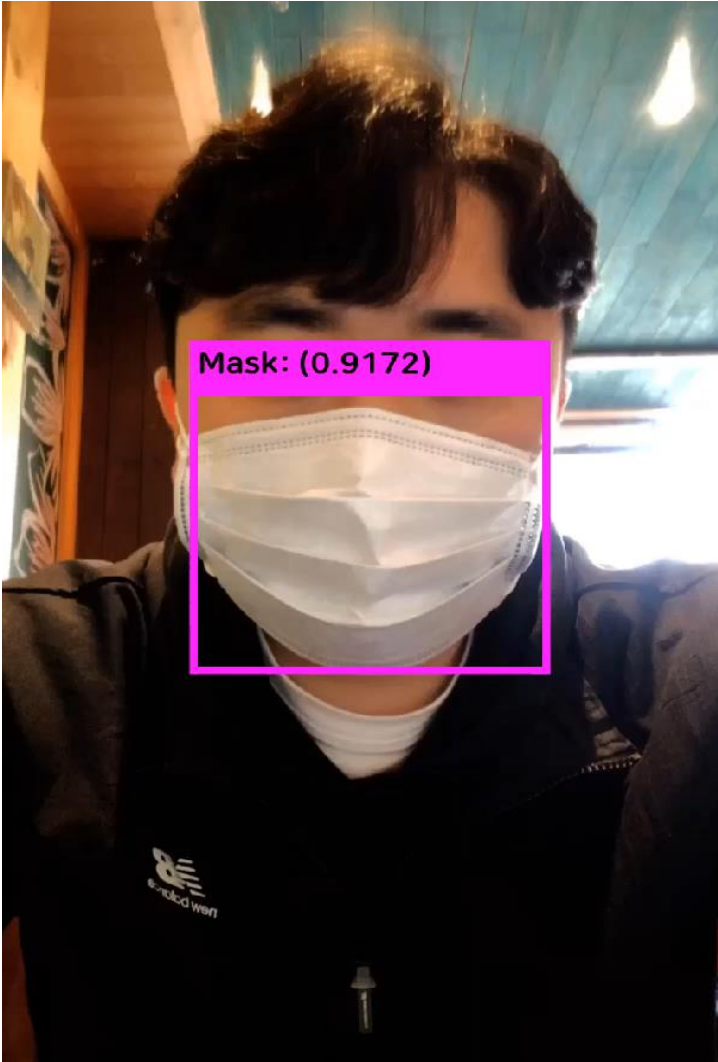


# Detecting Objects with You Only Look Once (YOLO) v2



Build, test, and deploy a deep learning solution that can detect objects in images and video

# Mask Detection with YOLO v2

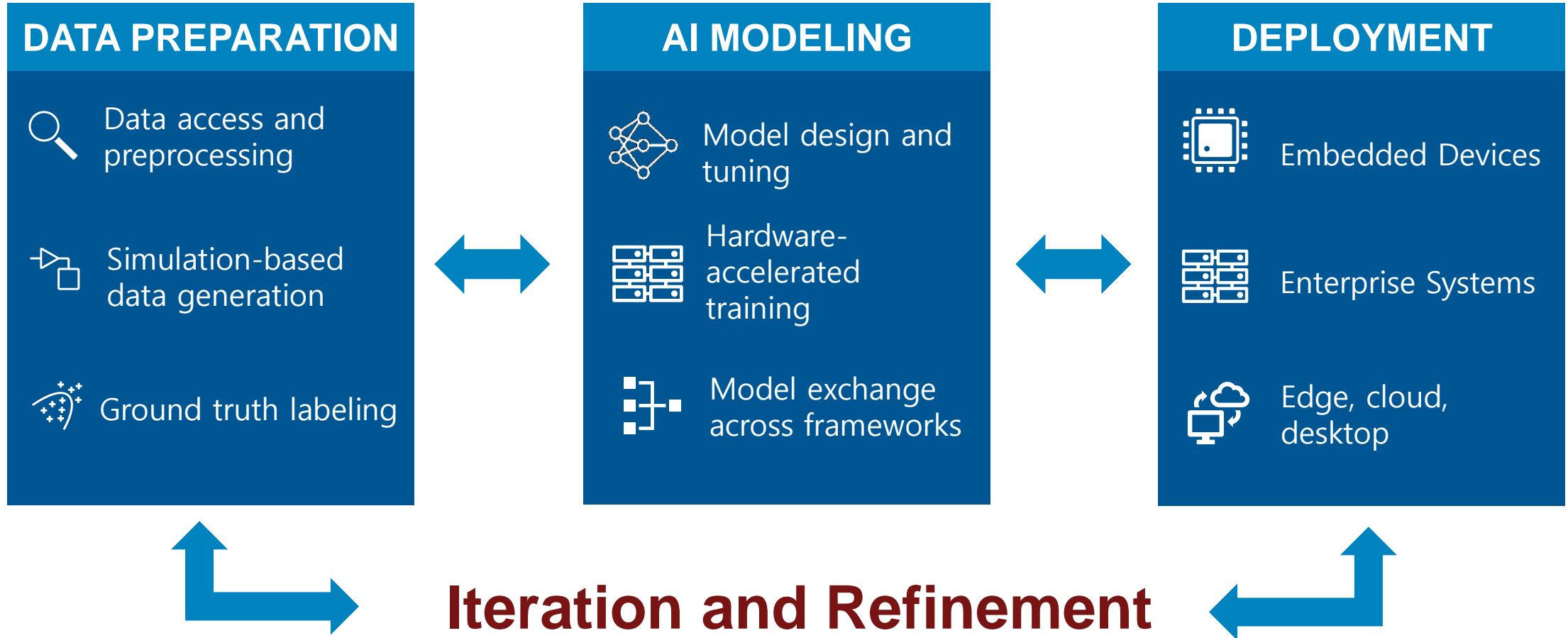


# Experiment Manager

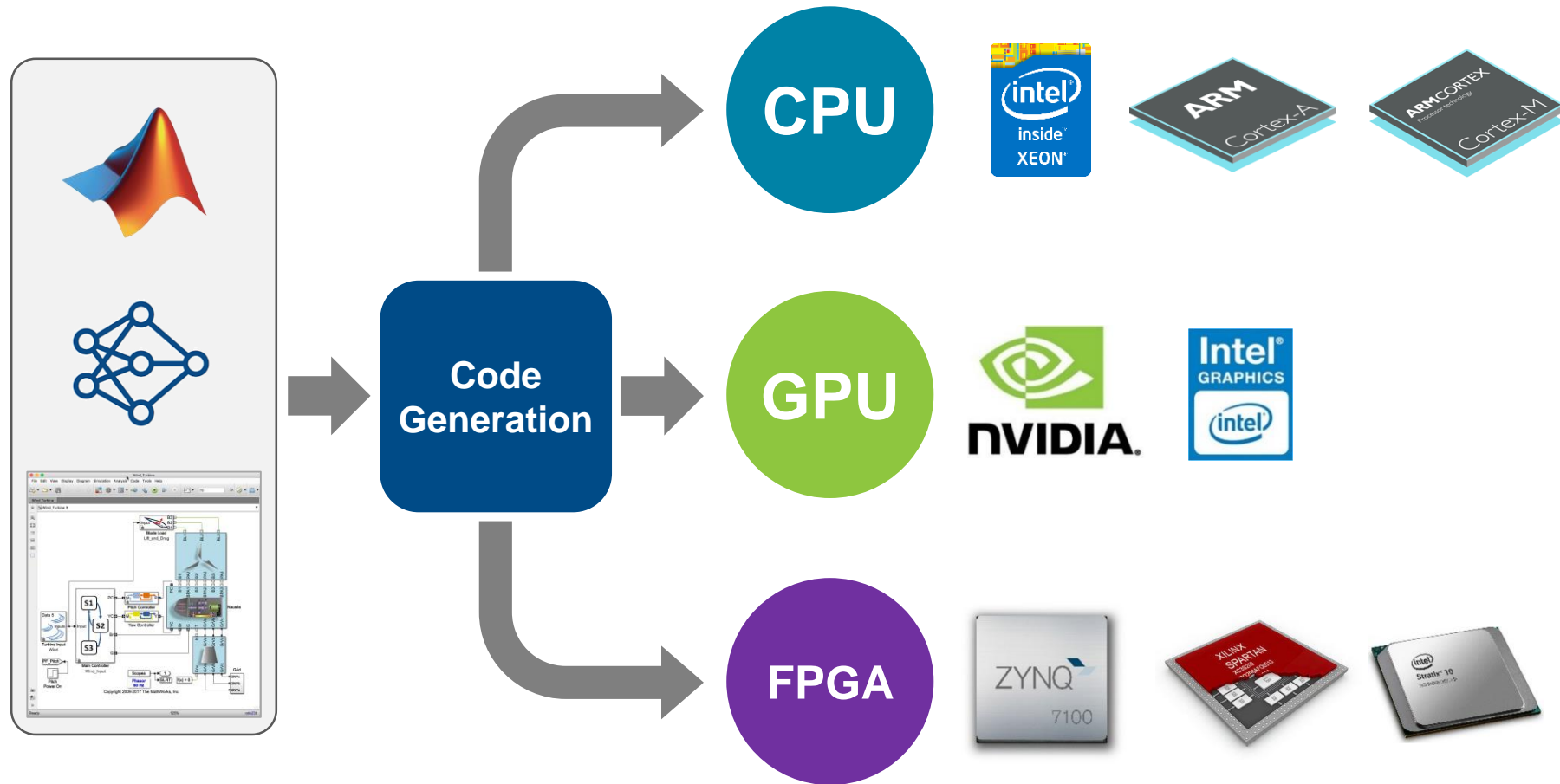
The screenshot displays the MATLAB Experiment Manager interface. At the top, there is a toolbar with icons for file operations (New, Open, Save, Duplicate), environment management (Layout), execution (Run, Stop), and result review (Training Plot, Confusion Matrix, Filter, Export). Below the toolbar is the 'EXPERIMENT BROWSER' on the left, which shows a tree view of the experiment structure: DigitsClassifier > Baseline Establishment > Sweep Initial Learning Rate, Baseline run, and Baseline Tuning > Result1 (Running), Larger Initial Learning Rate Range, Sweep Learning Rate Conv Size and Add Conv-Batch-ReLu Banks, Vary Filter Size of First Conv2D Layer, and Train Validation Split Study. The main area shows 'Result Details' for 'Baseline Tuning' on 2/7/2020 at 12:53:36 PM, with a progress bar indicating 7/16 trials completed. A summary table shows 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         |                  |               |                 |

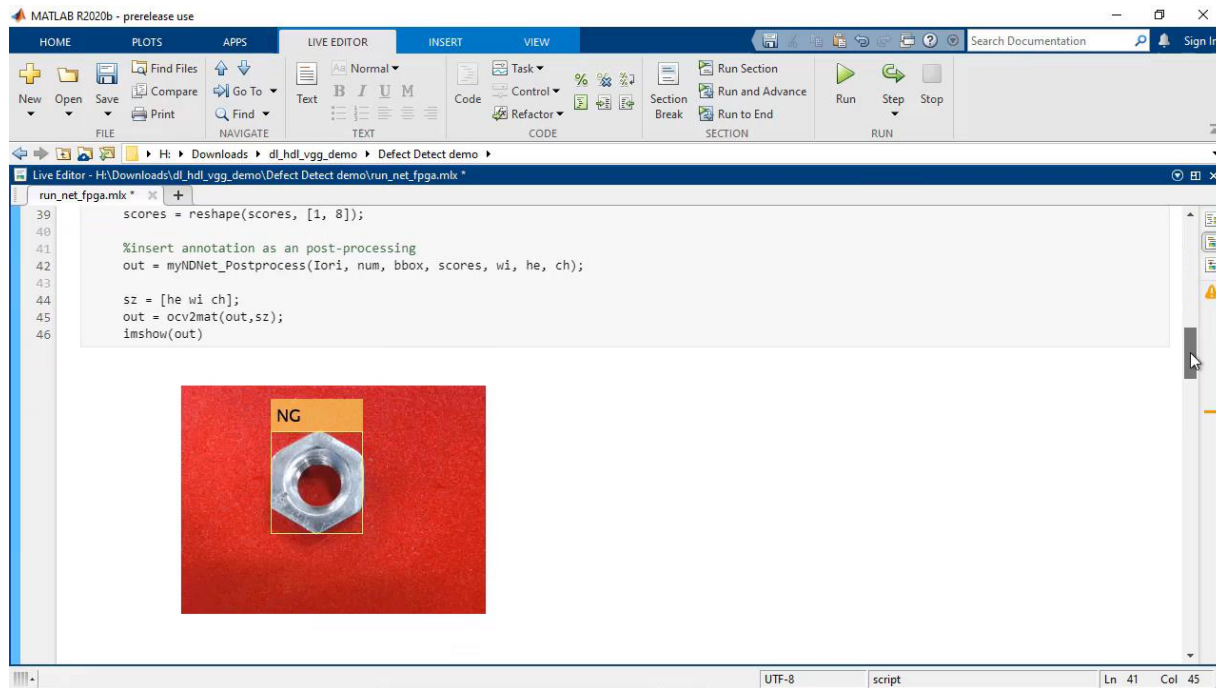
# Defect Detection Workflow



# 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

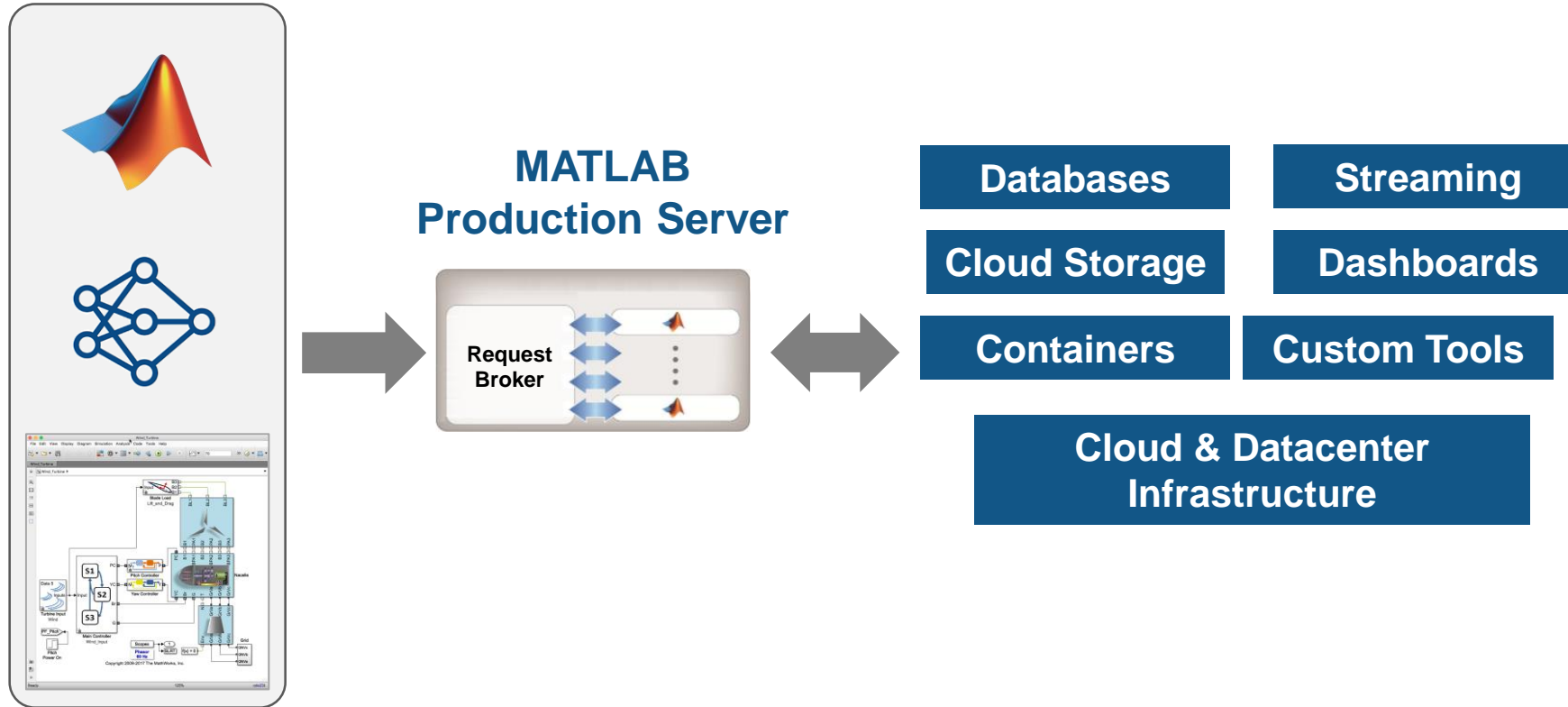
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
29294 techcon 20 0 770124 208236 81036 R 89.7 5.1 2:59.16 nutsDet_exe
29310 techcon 20 0 770124 208236 81036 S 39.0 5.1 0:50.61 nutsDet_exe
29312 techcon 20 0 770124 208236 81036 R 39.0 5.1 0:50.22 nutsDet_exe
29311 techcon 20 0 770124 208236 81036 S 38.7 5.1 0:50.49 nutsDet_exe
29013 techcon 20 0 11596 3840 2076 S 3.5 0.1 0:10.81 sshd
29325 techcon 20 0 5984 2760 2176 R 2.6 0.1 0:03.85 top
8 root 20 0 0 0 0 I 0.3 0.0 0:58.59 rcu_preempt
2130 root 20 0 0 0 0 I 0.3 0.0 0:23.68 kworker/5:2
17151 root 20 0 0 0 0 I 0.3 0.0 0:02.27 kworker/2:1
25518 root 20 0 0 0 0 I 0.3 0.0 0:22.65 kworker/3:1
28340 root 20 0 0 0 0 I 0.3 0.0 0:01.86 kworker/0:2
29296 techcon 20 0 770124 208236 81036 S 0.3 5.1 0:00.19 QXcbEventReader
1 root 20 0 154280 5224 3504 S 0.0 0.1 0:12.00 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.18 kthreadd
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
7 root 20 0 0 0 0 S 0.0 0.0 0:05.58 ksoftirqd/0
9 root 20 0 0 0 0 I 0.0 0.0 0:00.38 rcu_sched
10 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_bh
11 root rt 0 0 0 0 S 0.0 0.0 0:00.14 migration/0
12 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuphp/0
13 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuphp/1
14 root rt 0 0 0 0 S 0.0 0.0 0:00.15 migration/1
15 root 20 0 0 0 0 S 0.0 0.0 0:00.28 ksoftirqd/1
17 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/1:0H
18 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuphp/2
19 root rt 0 0 0 0 S 0.0 0.0 0:00.14 migration/2
20 root 20 0 0 0 0 S 0.0 0.0 0:00.22 ksoftirqd/2
```

Defect detection deployed on ARM Cortex-A microprocessor

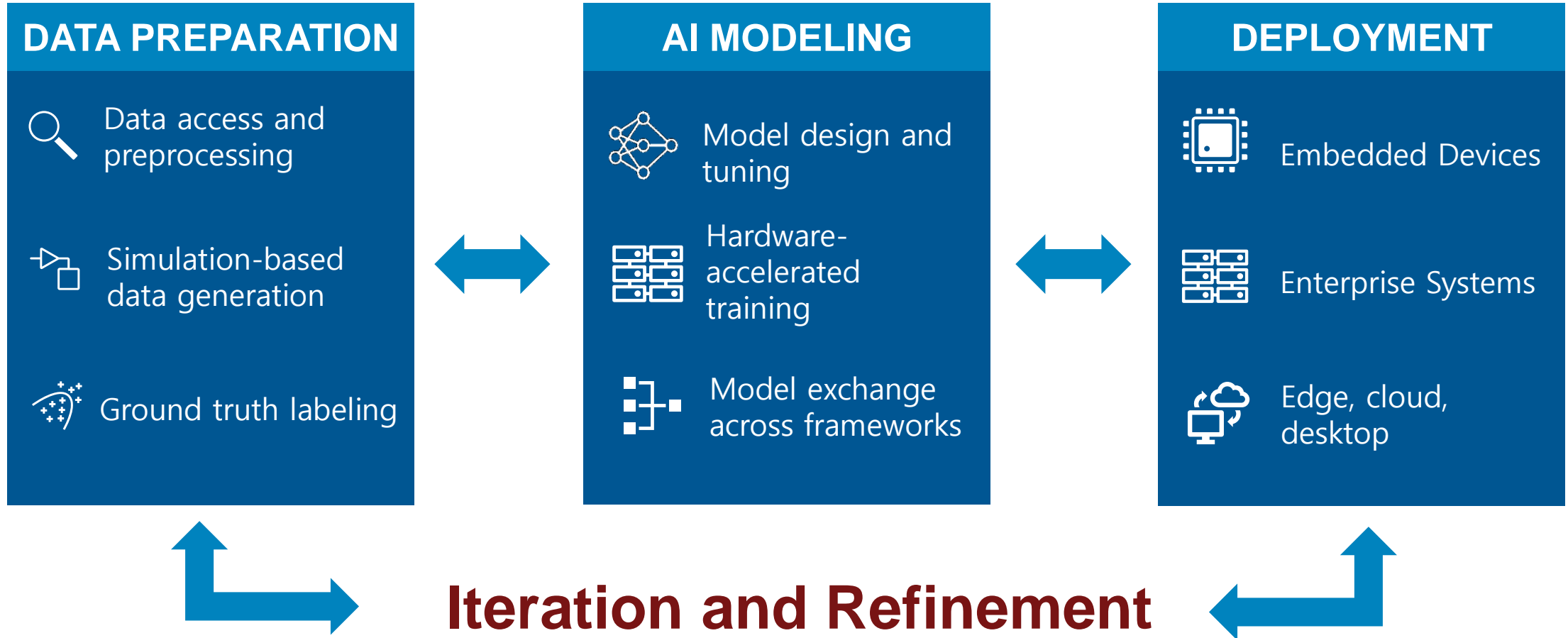
## Resources:

- [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](#)

# Deploy to Enterprise IT Infrastructure



# Defect Detection Workflow



# Key Takeaways

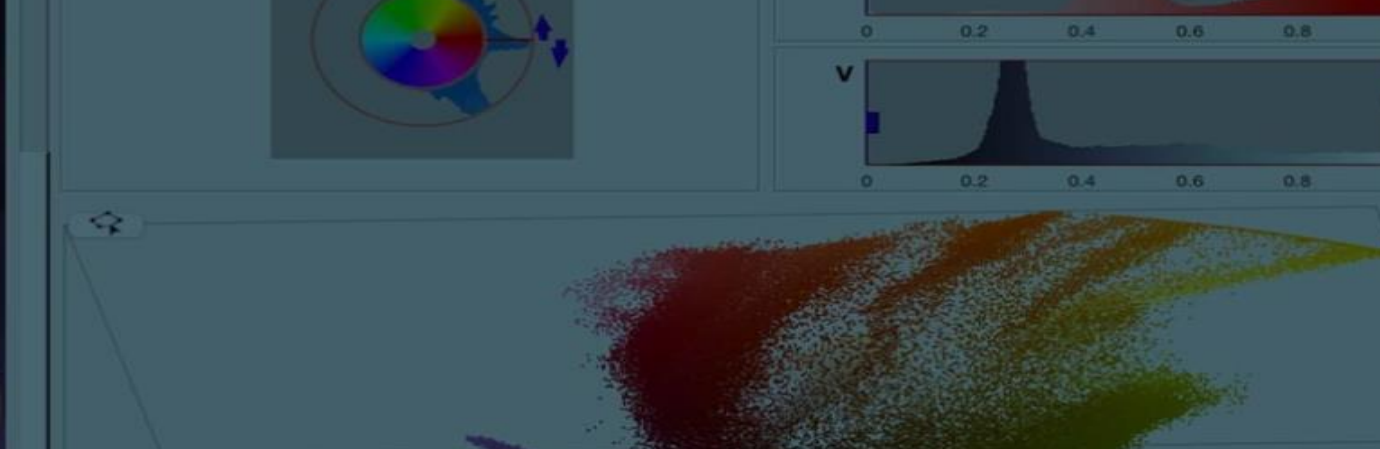
- Interactive and easy to use apps help explore, iterate and automate workflows
- Flexibility and options to choose networks and optimizations based on data and requirements
- MATLAB provides an easy and extensible framework for defect detection from data access to deployment

## Image Processing Toolbox

Perform image processing, visualization, and analysis

[Watch video](#)

[Download a free trial](#)



## Computer Vision Toolbox

Design and test computer vision, 3D vision, and video processing systems

[Watch video](#)

[Download a free trial](#)



Pedestrian



Vehicle

## Deep Learning Toolbox

Design, train, and analyze deep learning networks

[Watch video](#)

[Download a free trial](#)





**THANK YOU!**