MathWorks®

esa cnes
5th SEFUW
SpacE FPGA Users Workshop
14, 15 and 16 March 2023
ESTEC, Noordwijk, The Netherlands

Image Recognition for Space Applications using Deep Learning on FPGAs & SoCs

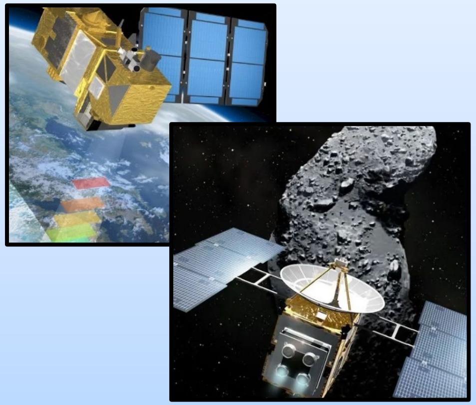
Lunar Crater Detection

Stephan van Beek
svanbeek@mathworks.com

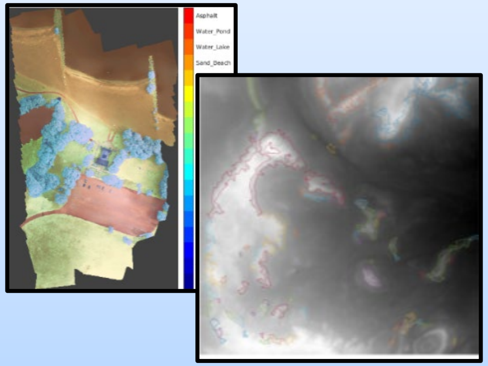
European Technical Specialist
SoC/FPGA/ASIC Design Flows

Artificial Intelligence on Embedded Devices

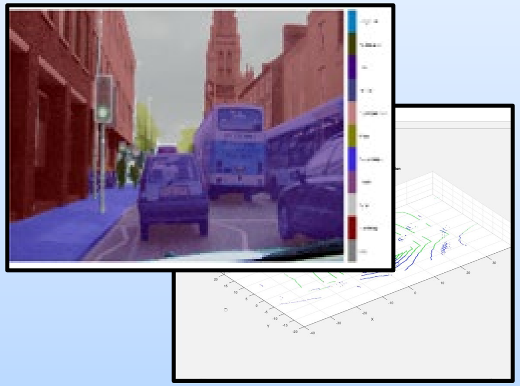
Satellite Navigation



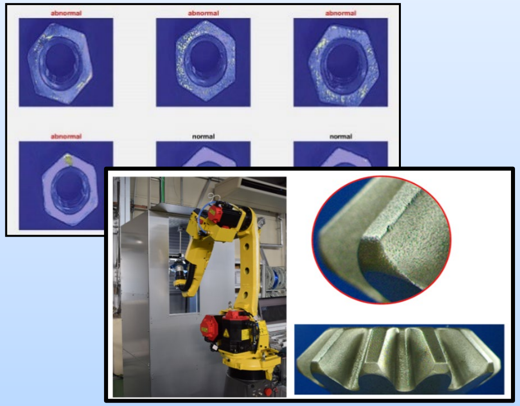
Airborne Image Analysis



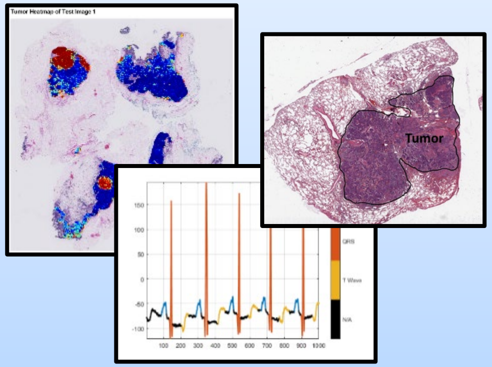
Autonomous Driving



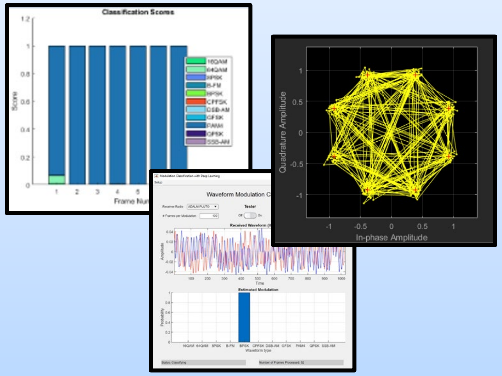
Industrial Inspection



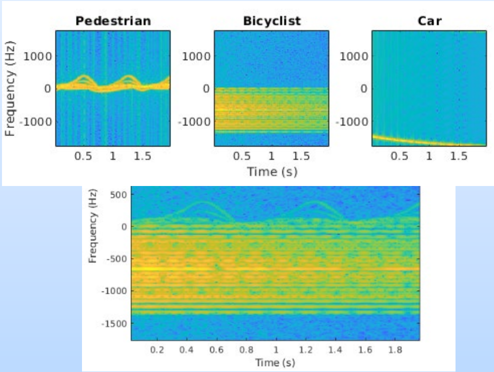
Medical Image Analysis



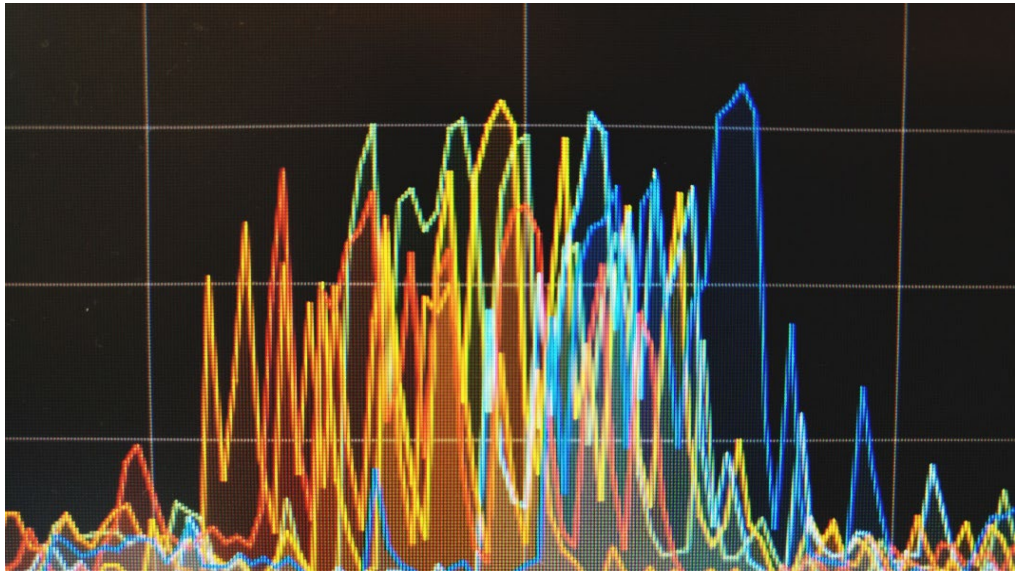
Wireless Modulation Classification



Radar Signature Classification



Machine learning has been deployed on ground segment applications for several years ➔ now moving into space

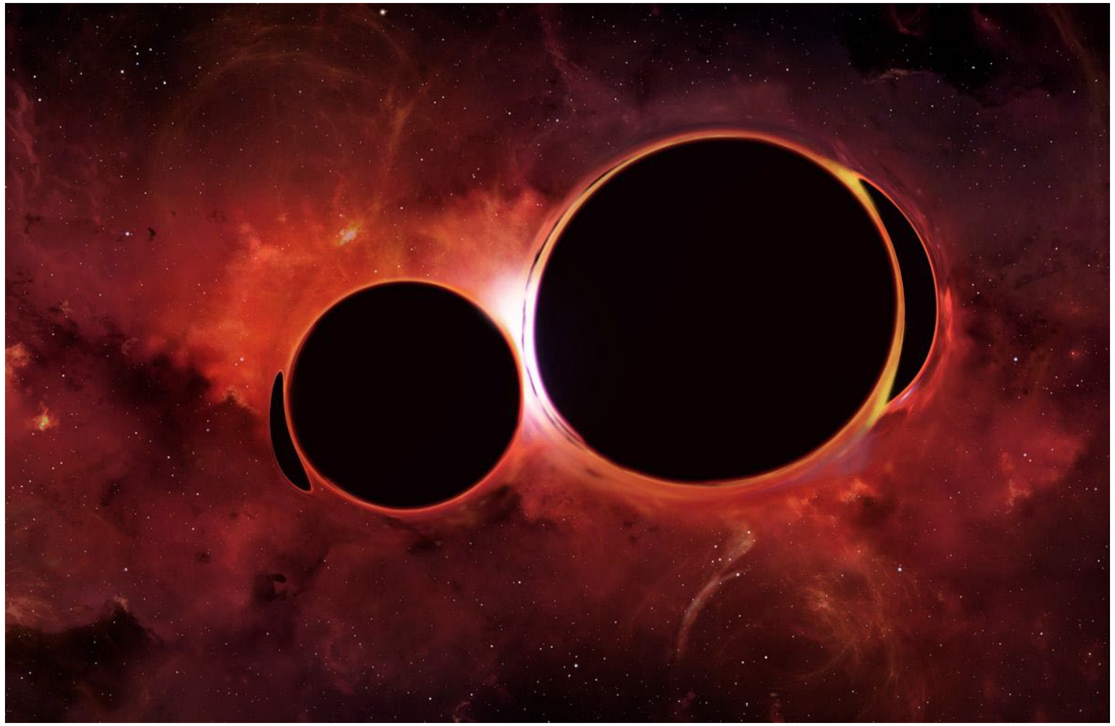


Telemetry Outlier Detection

Geospatial Analytics

Deep Learning Helps Detect Gravitational Waves

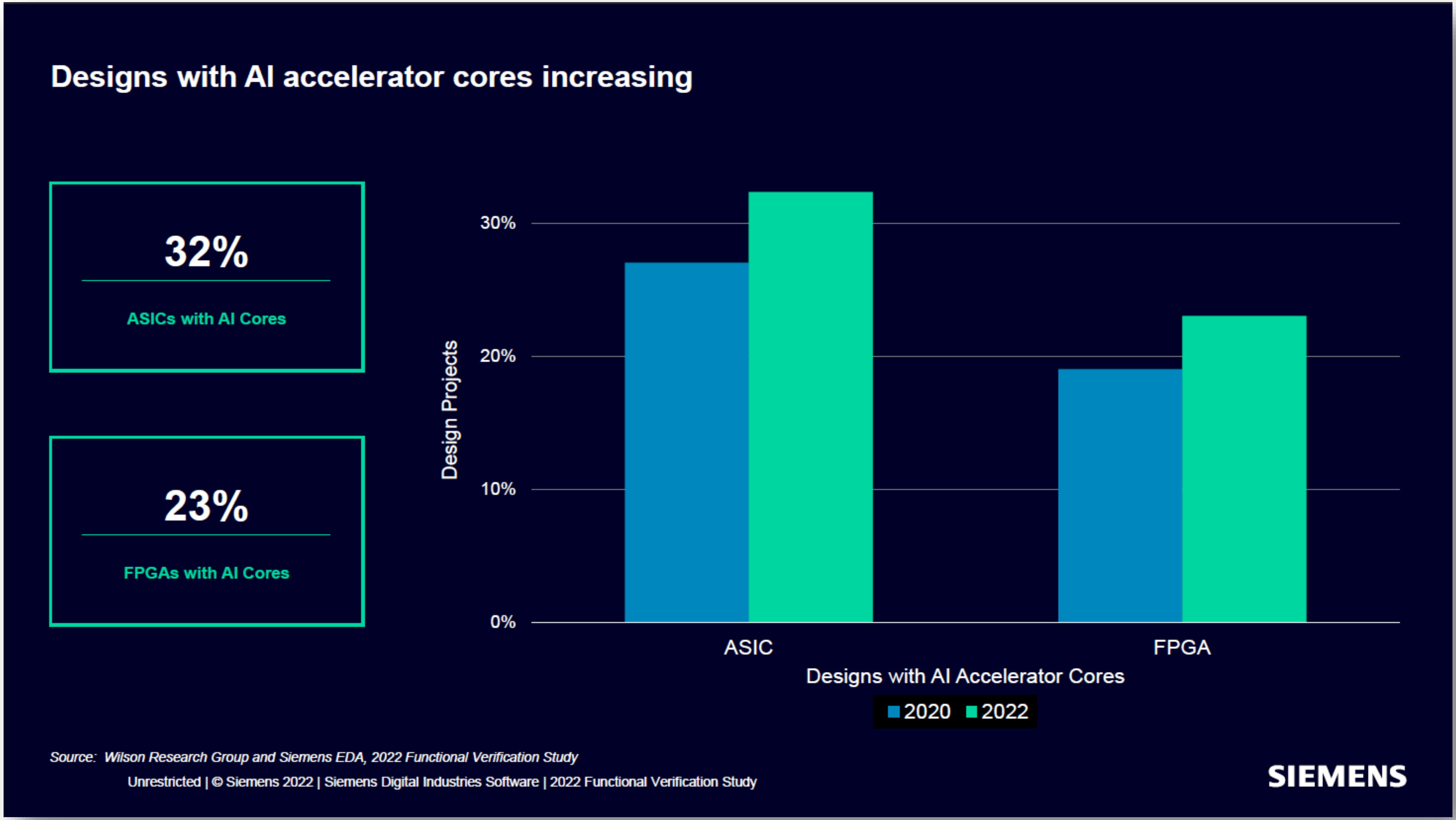
Hunting for Black Holes with Artificial Intelligence



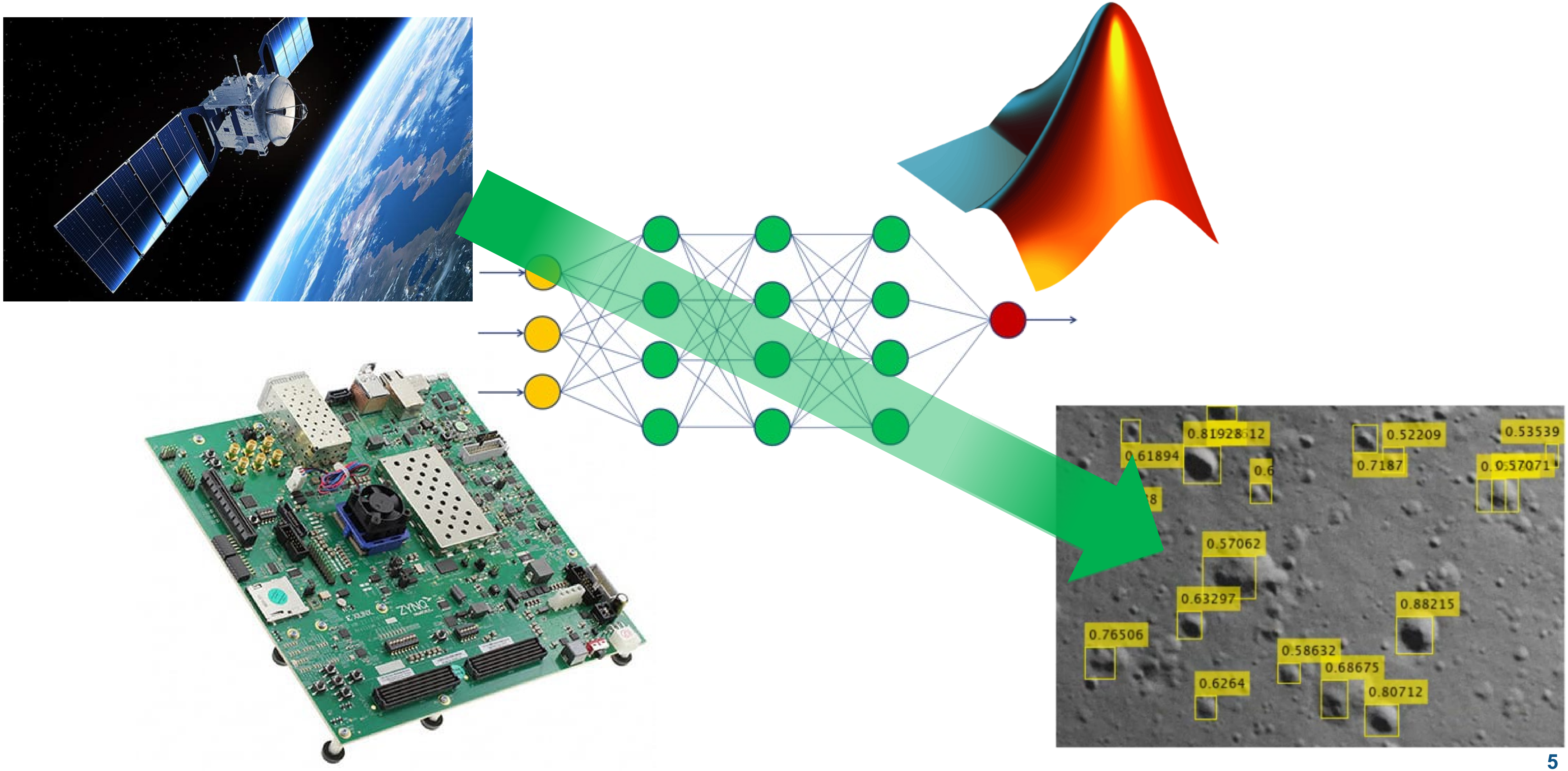
Max Planck Institute used AI and laser interferometry to detect gravitational waves caused by space-time distortions in our solar system.

https://www.mathworks.com/company/mathworks-stories/deep-learning-ai-and-laser-interferometry-detects-gravitational-waves.html

Industry Trends



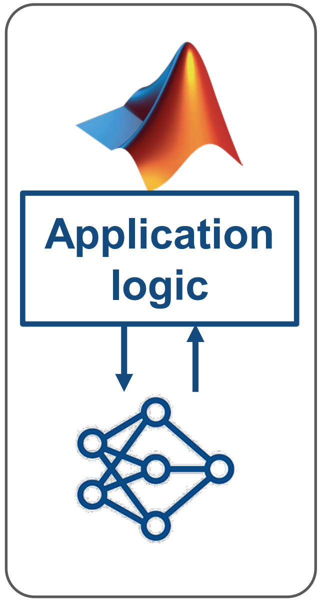
Let's have a look at an example: Lunar Crater Detection

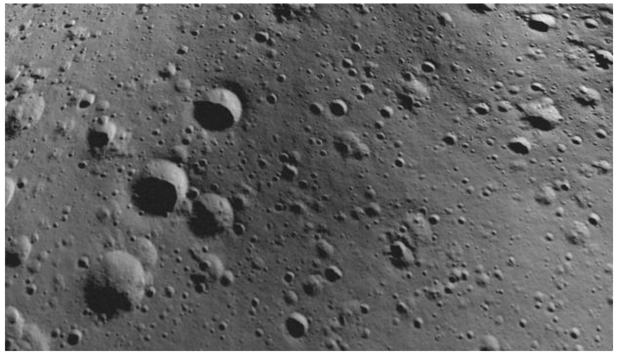


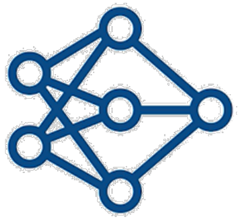
Object Detection with MATLAB

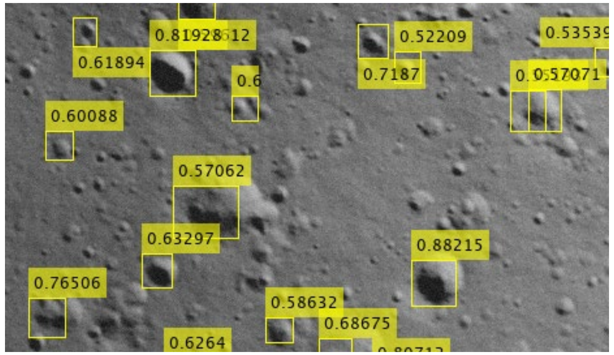
Images and video generated by PANGU and courtesy of the University of Dundee and STAR-Dundee.

Crater Detection Example









Pre-processing: Extract regions and resize

Inference: Predict using trained network

FPGA

Post-processing: Annotate and label

Unique Platform Differentiators

MATLAB is an Interoperable and Integrated Platform for AI-Driven Systems

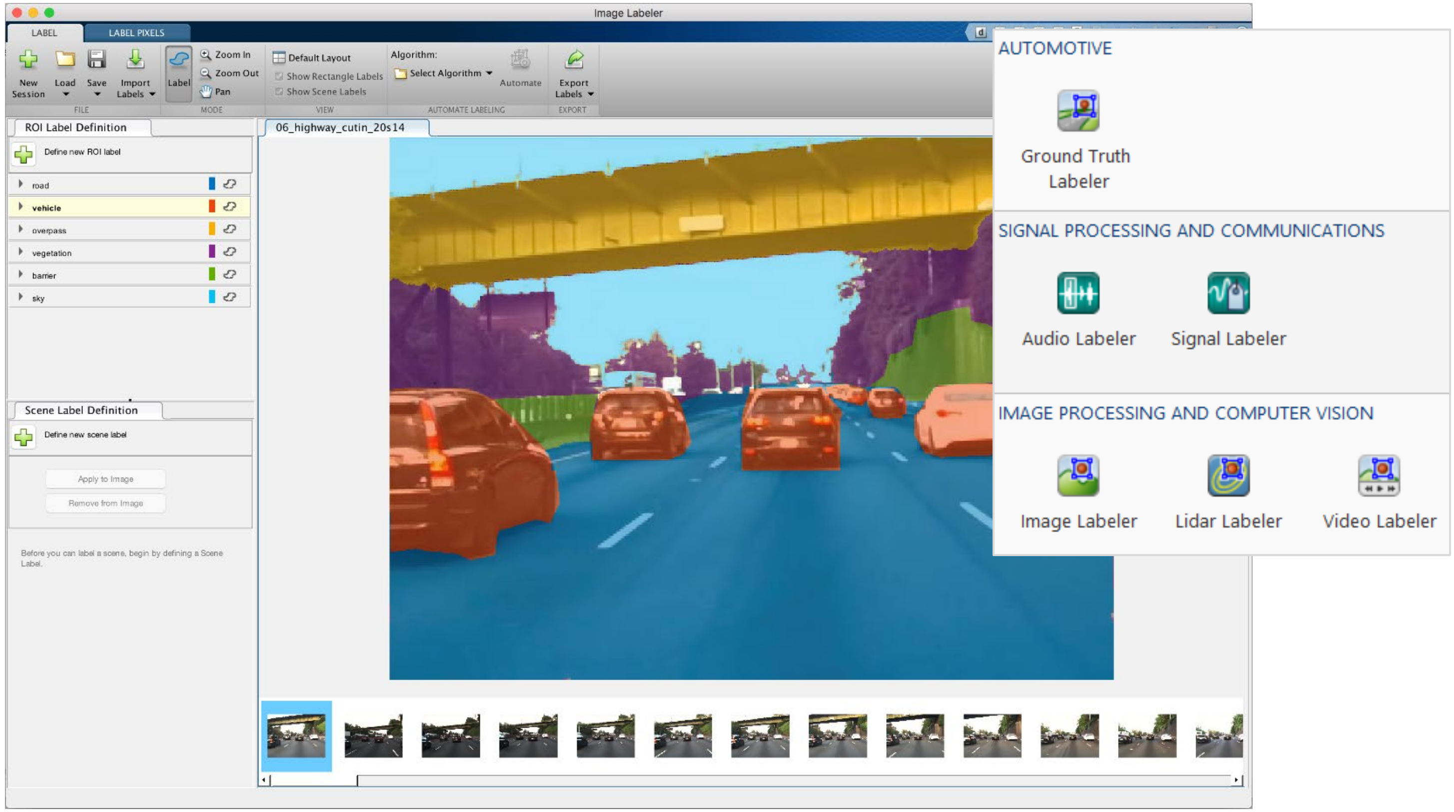
Data Preparation	AI Modeling	Simulation & Test	Deployment
Data cleansing and preparation	Model design and tuning	Integration with complex systems	Embedded devices
Human insight	Hardware accelerated training	System simulation	Enterprise systems
Simulation-generated data	Interoperability	System verification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

Spend less time preprocessing and labeling data

Synchronize disparate time series, filter noisy signals, automate labeling of video, and more.

Data Preparation

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



Use labeling apps for deep learning workflows like semantic segmentation

Start with a complete set of algorithms and pre-built models

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

Algorithms

Machine learning
Trees, Naïve Bayes, SVM…

Deep learning
CNNs, GANs, LSTM, MIMO…

Reinforcement learning
DQN, A2C, DDPG…

Regression
Linear, nonlinear, trees…

Unsupervised learning
K-means, PCA, GMM…

Predictive maintenance
RUL models, condition indicators…

Bayesian optimization

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…

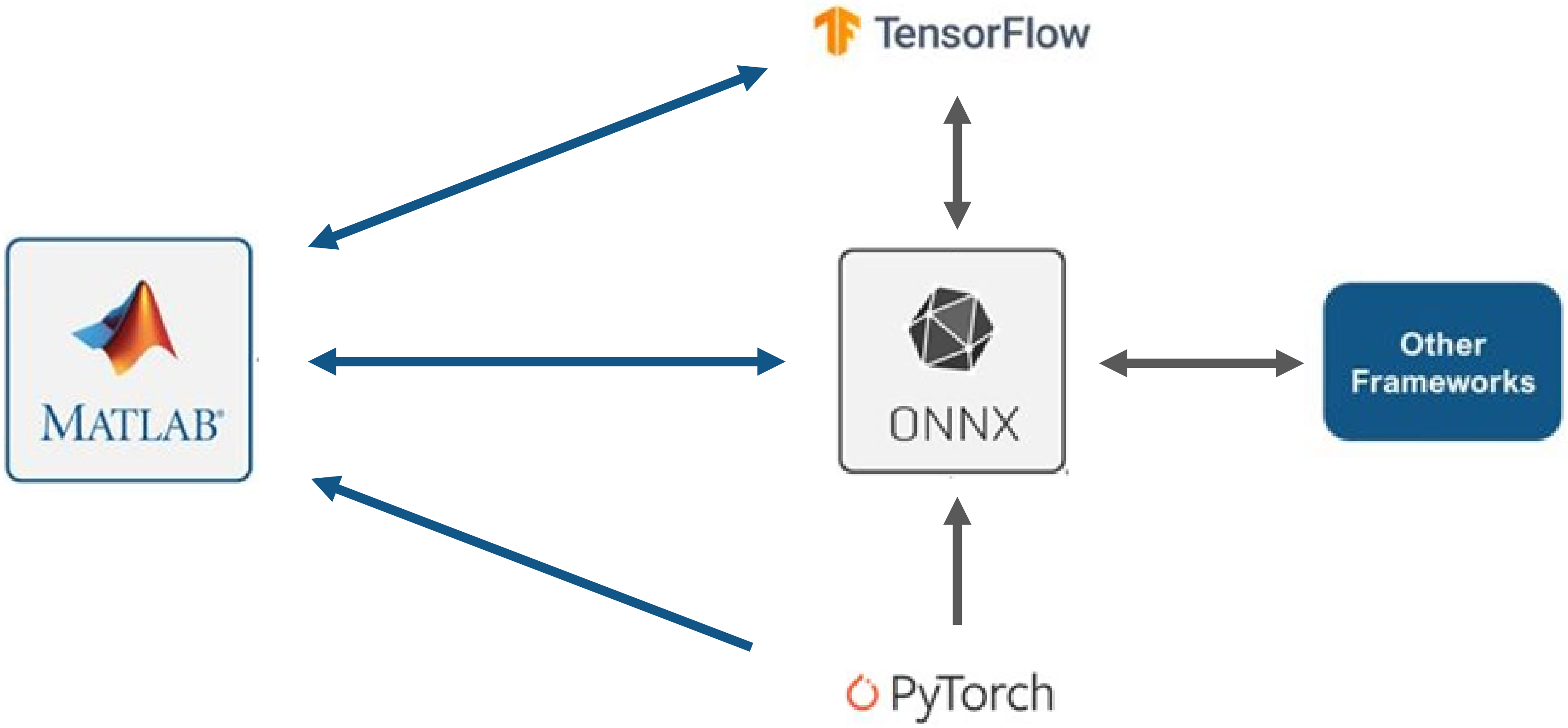
Signal and speech processing
Denoising, music genre recognition, keyword spotting, radar waveform classification…

…and more…

Interoperability with Python Based Frameworks

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

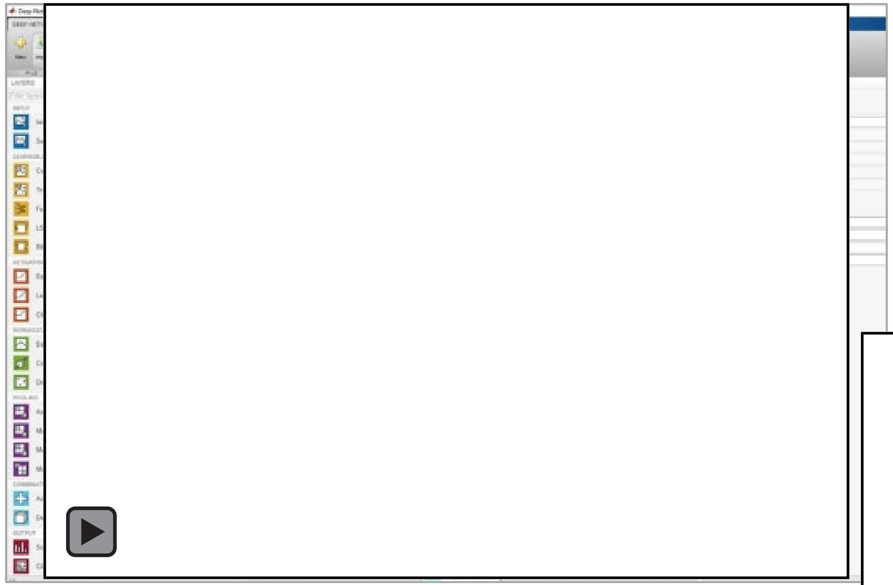


Increase productivity using Apps for design and analysis

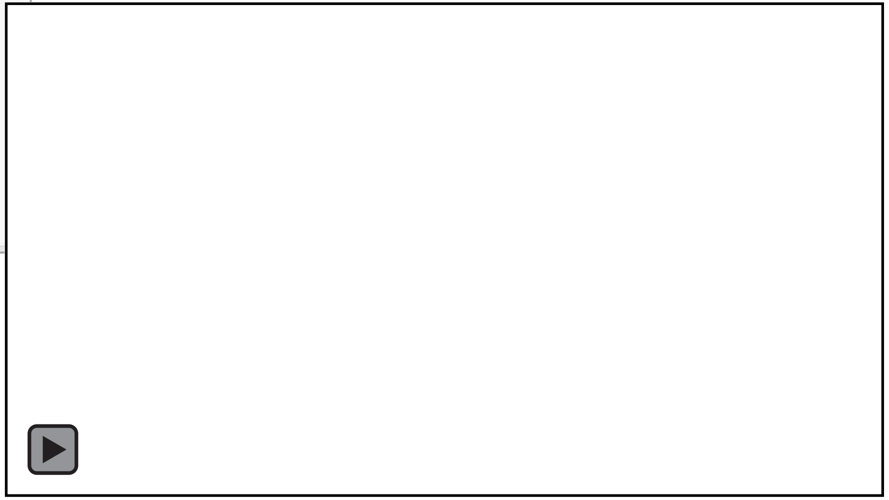
Use MATLAB Apps to design deep learning networks, explore a wide range of classifiers, train regression models, train an optical character recognition model, and more.

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability



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



Experiment Manager app to manage multiple deep learning experiments, analyze and compare results and code

Hardware acceleration and scaling are critical for training

MATLAB accelerates AI training on GPUs, cloud, and datacenter resources without specialized programming.

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

Azure

aws

intel

docker

NVIDIA

FPGA is a good choice for lower power deep learning applications

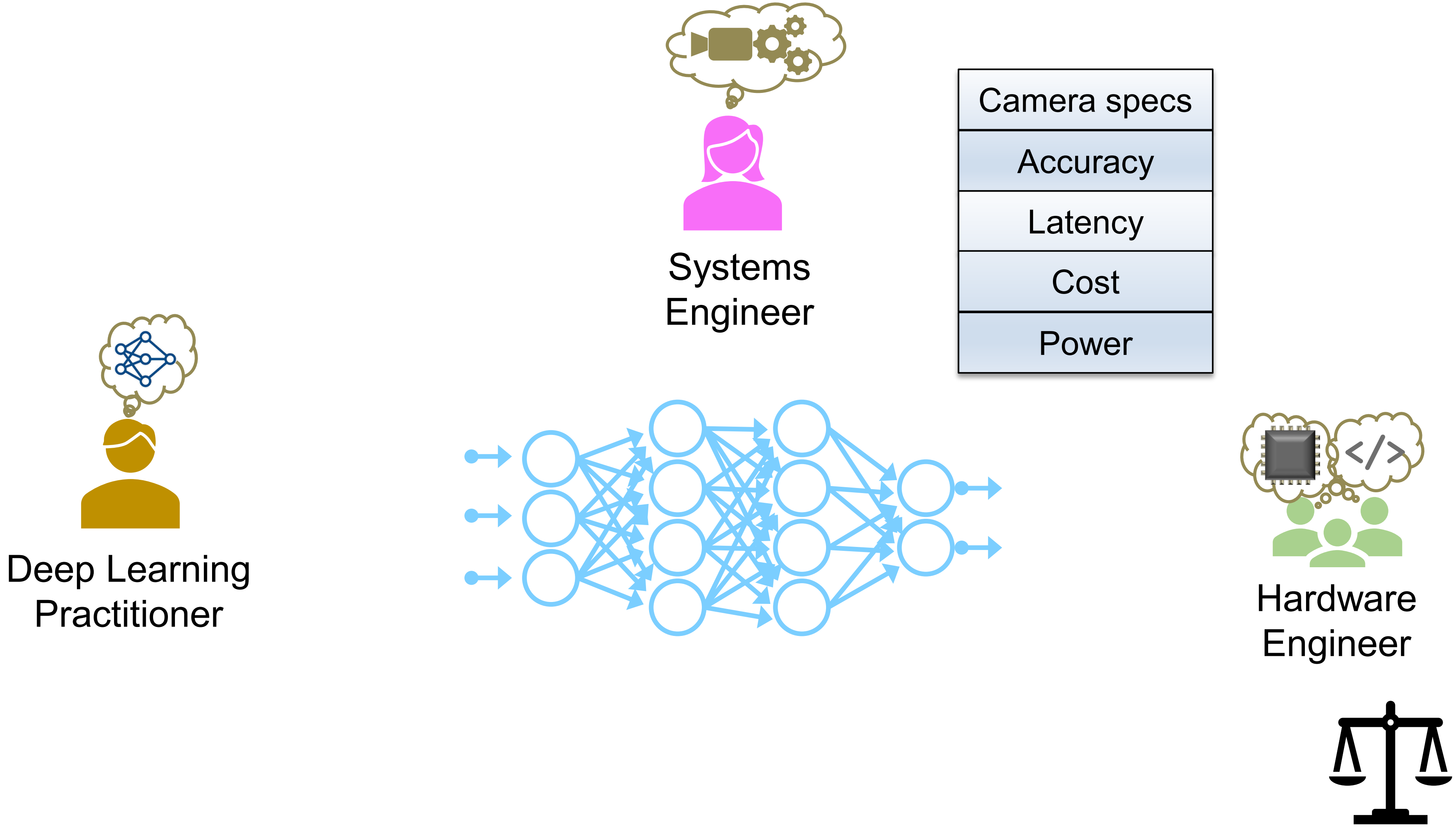
Deployment

- Embedded devices
- Enterprise systems
- Edge, cloud, desktop

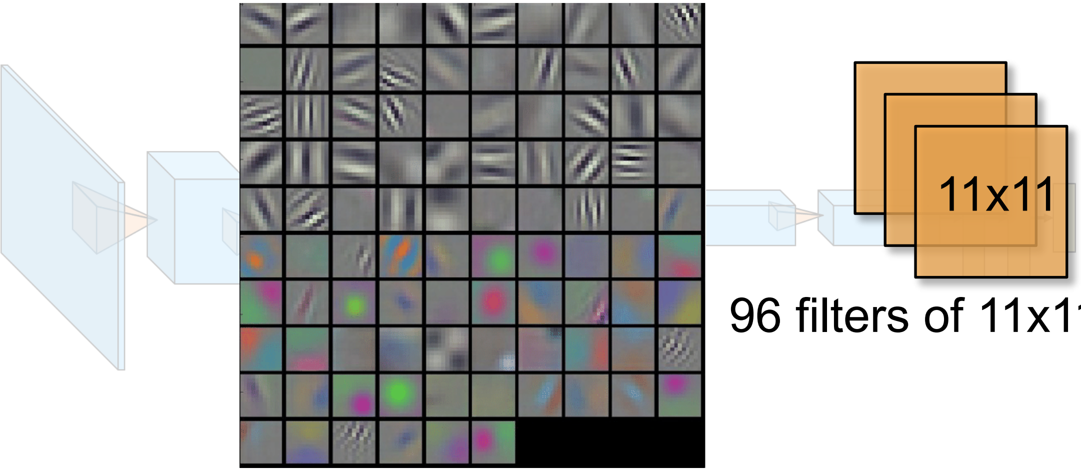
	GPU	ARM	FPGA	ASIC
Speed	High	Low	High	High
Power Consumption	High	Low	Low	Lowest
Engineering Cost	Medium	Low	Medium	High

- Low Latency
- High speed I/O connectivity

System Requirements Drive Network Design

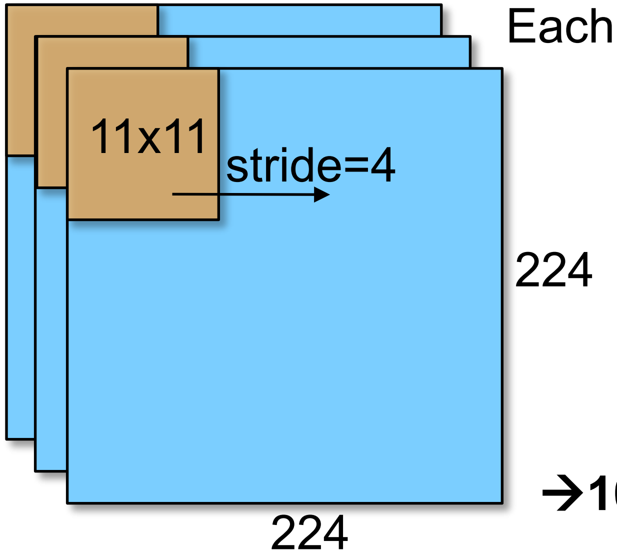


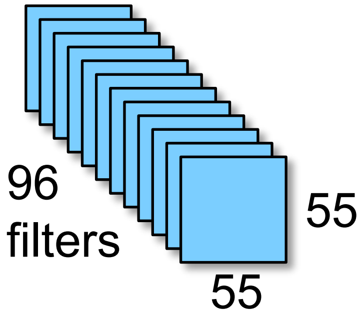
Challenges of Deploying Deep Learning to FPGA Hardware:



96 filters of 11x11x3 of 32-bit parameters → **140k bytes**

Each stride is an 11x11x3 matrix multiply-accumulate



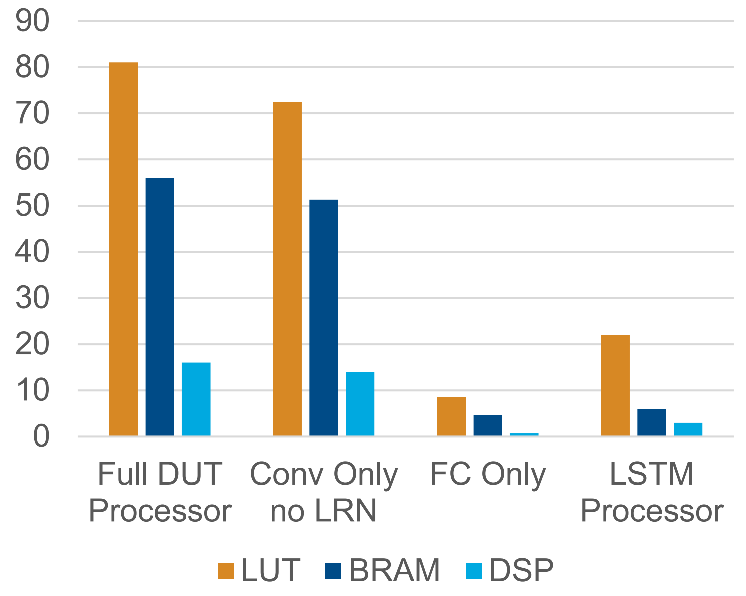


→ **1.16M bytes of activations**

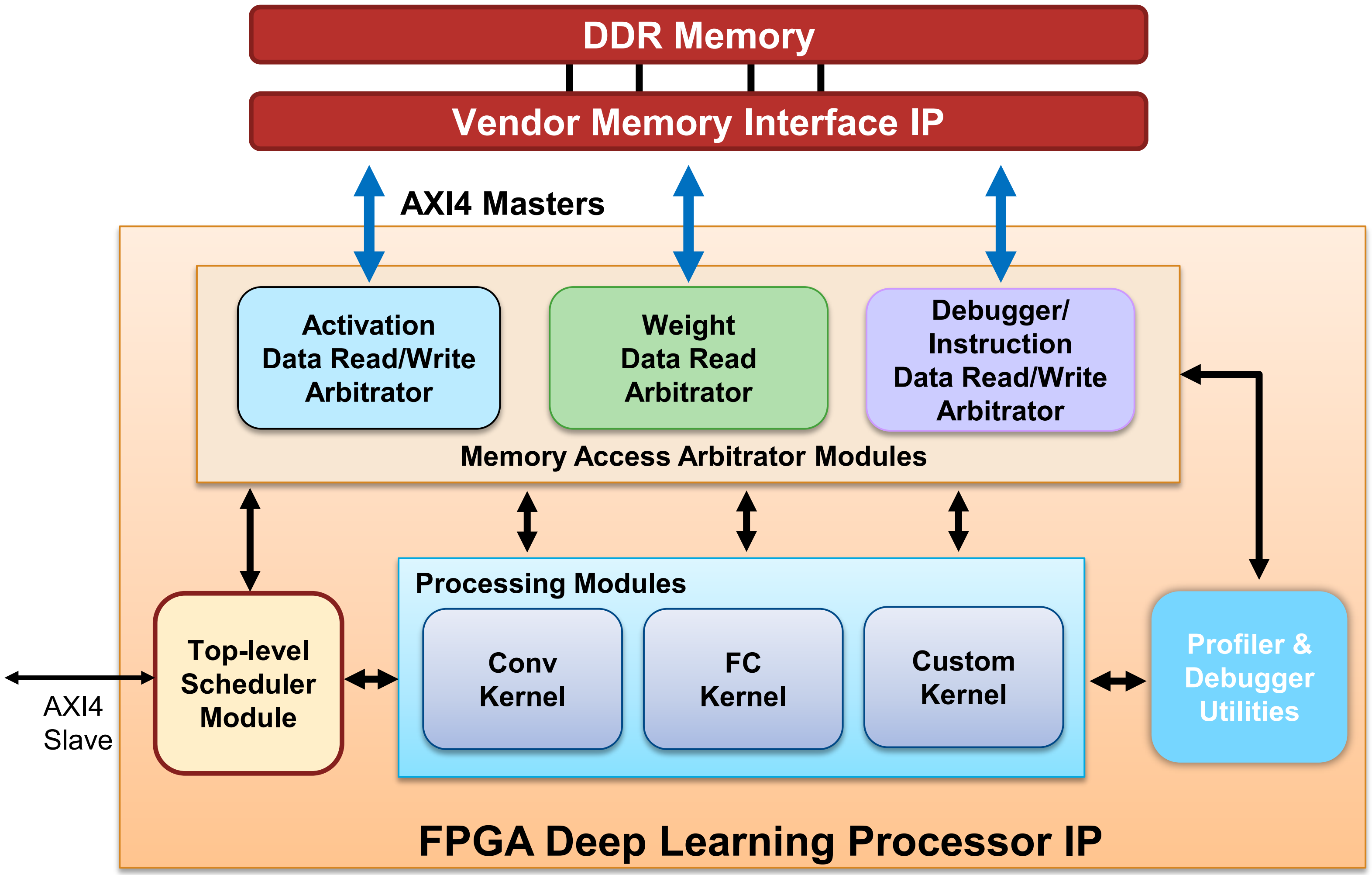
→ **105M** floating-point multiply operations!

Customizable Deep Learning Processor

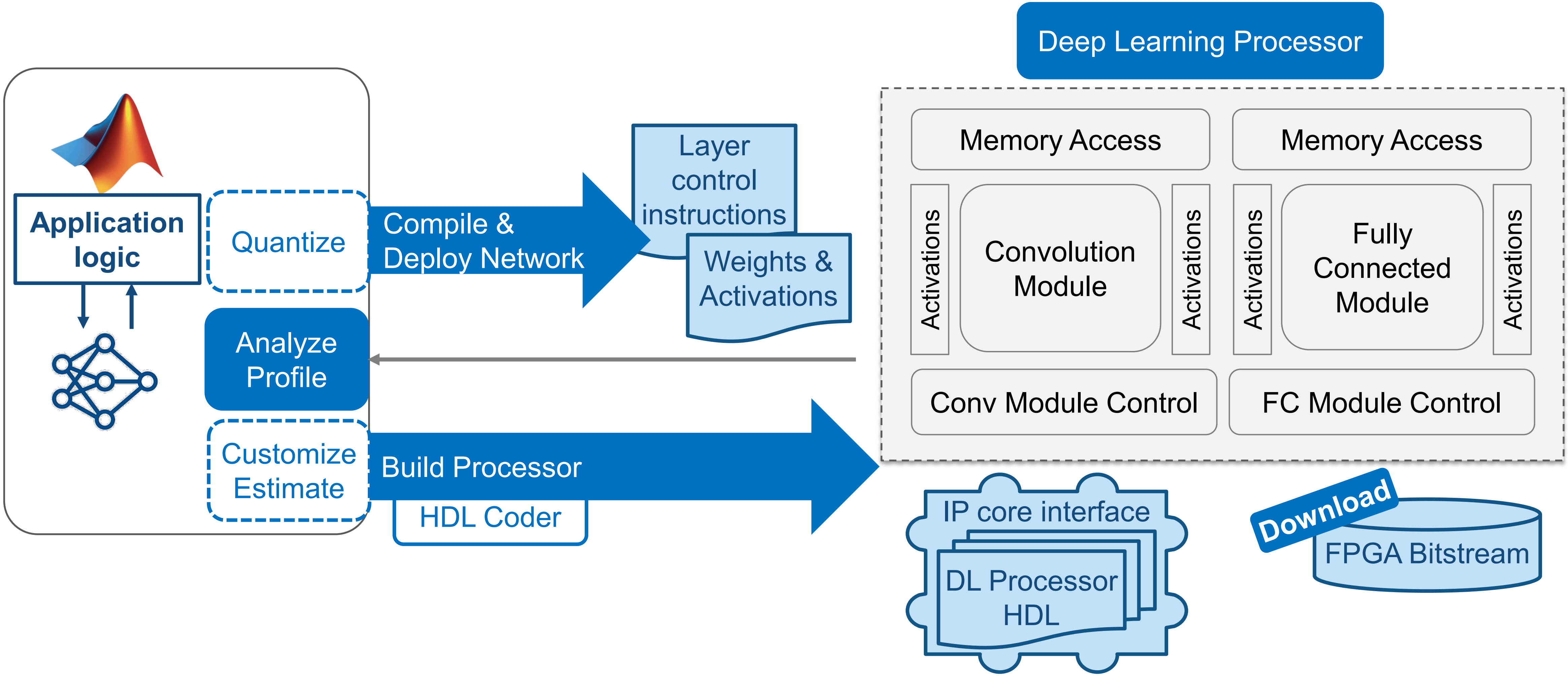
- Spend FPGA resource for only the layer kernels used in your network



Percentage resource usage on ZCU102 board



Deep Learning HDL Processor steps



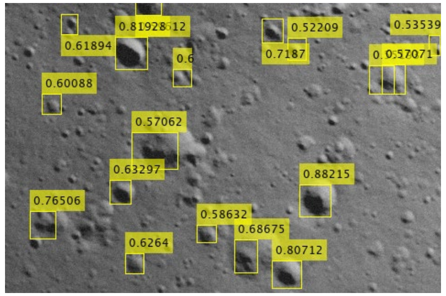
Profile FPGA Prototype and Iterate in MATLAB

Application logic

Re-train

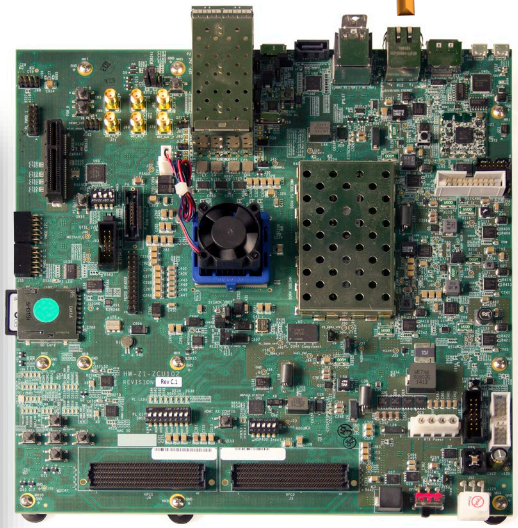
```
>> deepNetworkDesigner
```

```
>>wobj=dlhdl.Workflow('Network', detector.Network, 'Bitstream', 'zcu102_single');
>> dn = wobj.compile;
>> wobj.Target = dlhdl.Target('Xilinx', 'Interface', 'Ethernet', 'IPAddress', '192.168.4.2');
>> wobj.deploy;
>> [predict_out, speed] = wobj.predict(img_pre,'Profile','on');
```



Layer control instructions

Weights & Activations

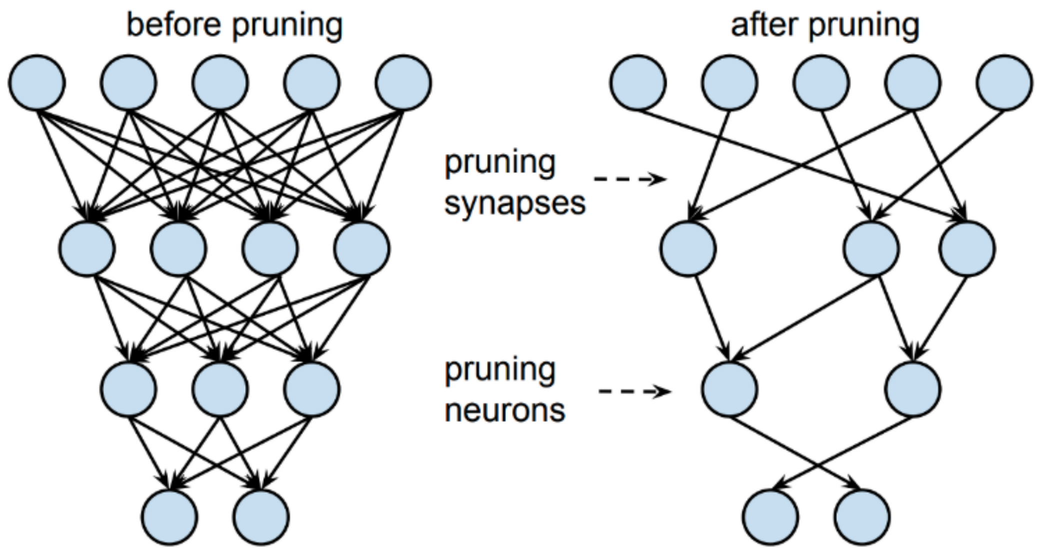


Deep Learning Processor Profiler Performance Results

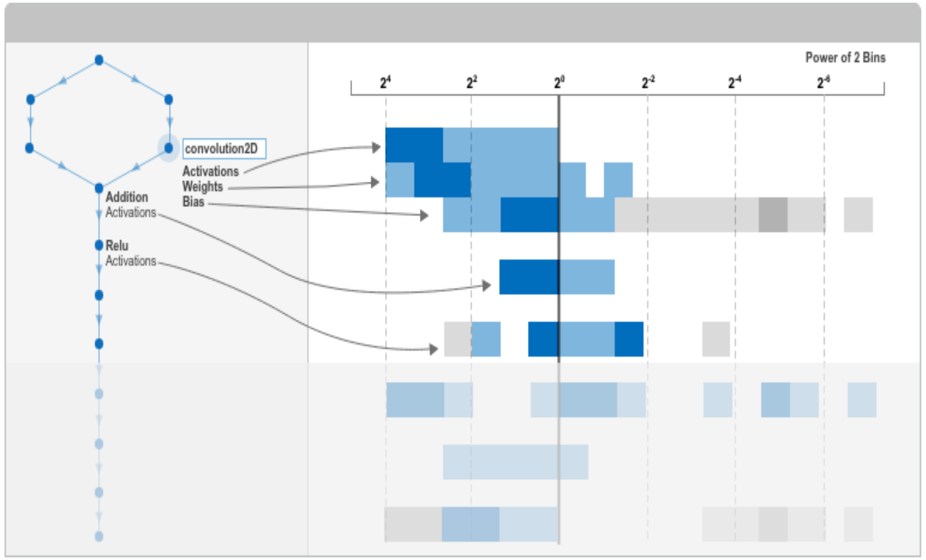
	LastFrameLatency(cycles)	LastFrameLatency(seconds)	FramesNum	Total Latency	Frames/s
Network	1729236	0.00786	1	1729753	127.2
conv_1	205293	0.00093			
maxpool1	161098	0.00073			
conv_2	212825	0.00097			
maxpool2	79776	0.00036			
conv_3	178397	0.00081			
maxpool3	44220	0.00020			
conv_4	161974	0.00074			
yolov2Conv1	305766	0.00139			
yolov2Conv2	306061	0.00139			
yolov2ClassConv	73795	0.00034			

* The clock frequency of the DL processor is: 220MHz

Two Compression Techniques

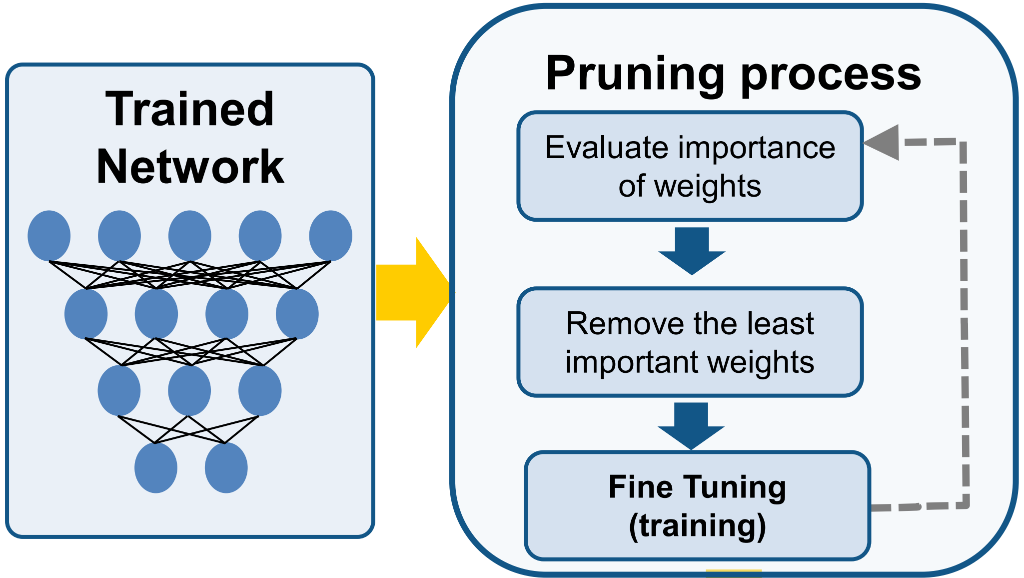


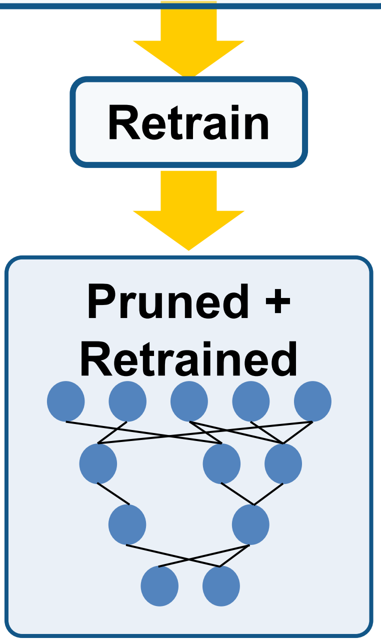
Pruning
deep neural networks



Quantization of
deep neural networks

Taylor Approximation Pruning

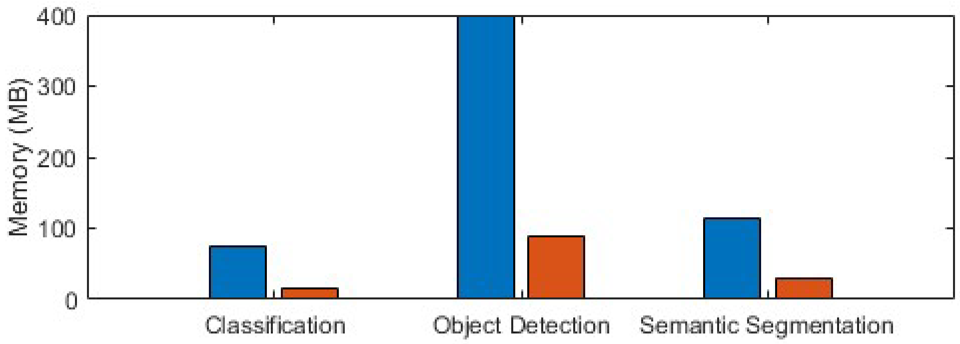


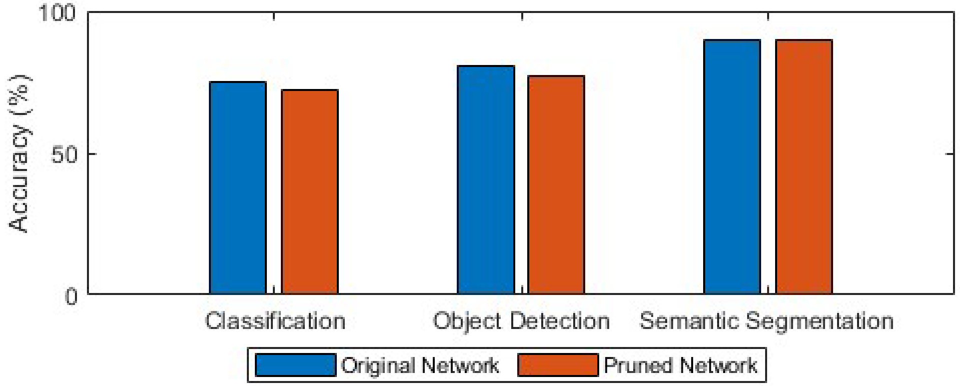


Remove **unimportant** parts of the network

```
prunableNetwork = taylorPrunableNetwork(dlnet)
```

```
prunableNetwork =
  TaylorNetworkPruner with properties ...
```





Recurrent Neural Networks (RNN) can be simplified by compressing LSTM Layers

- **Data**: time series of temps/currents/voltages
  → predict **s**tate **o**f **c**harge at each time step

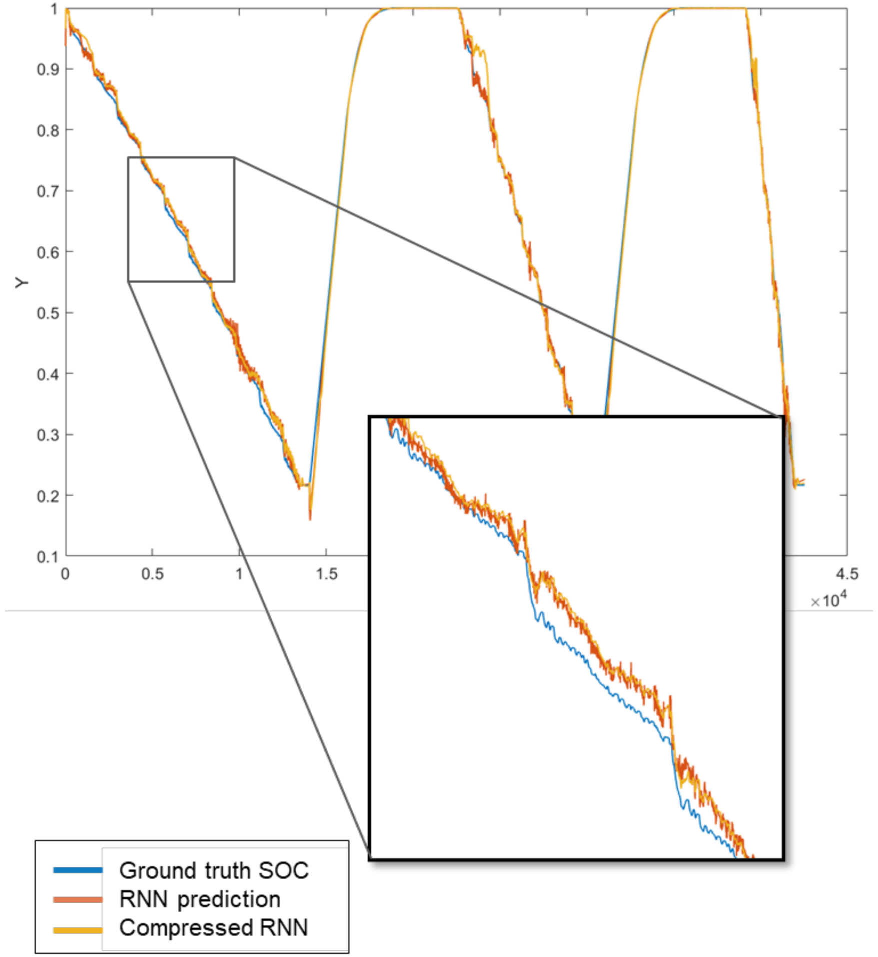
- **Model**: Recurrent regression NN with two LSTMs

463.4k	8	0 ⚠	0 ❗
total learnables	layers	warnings	errors

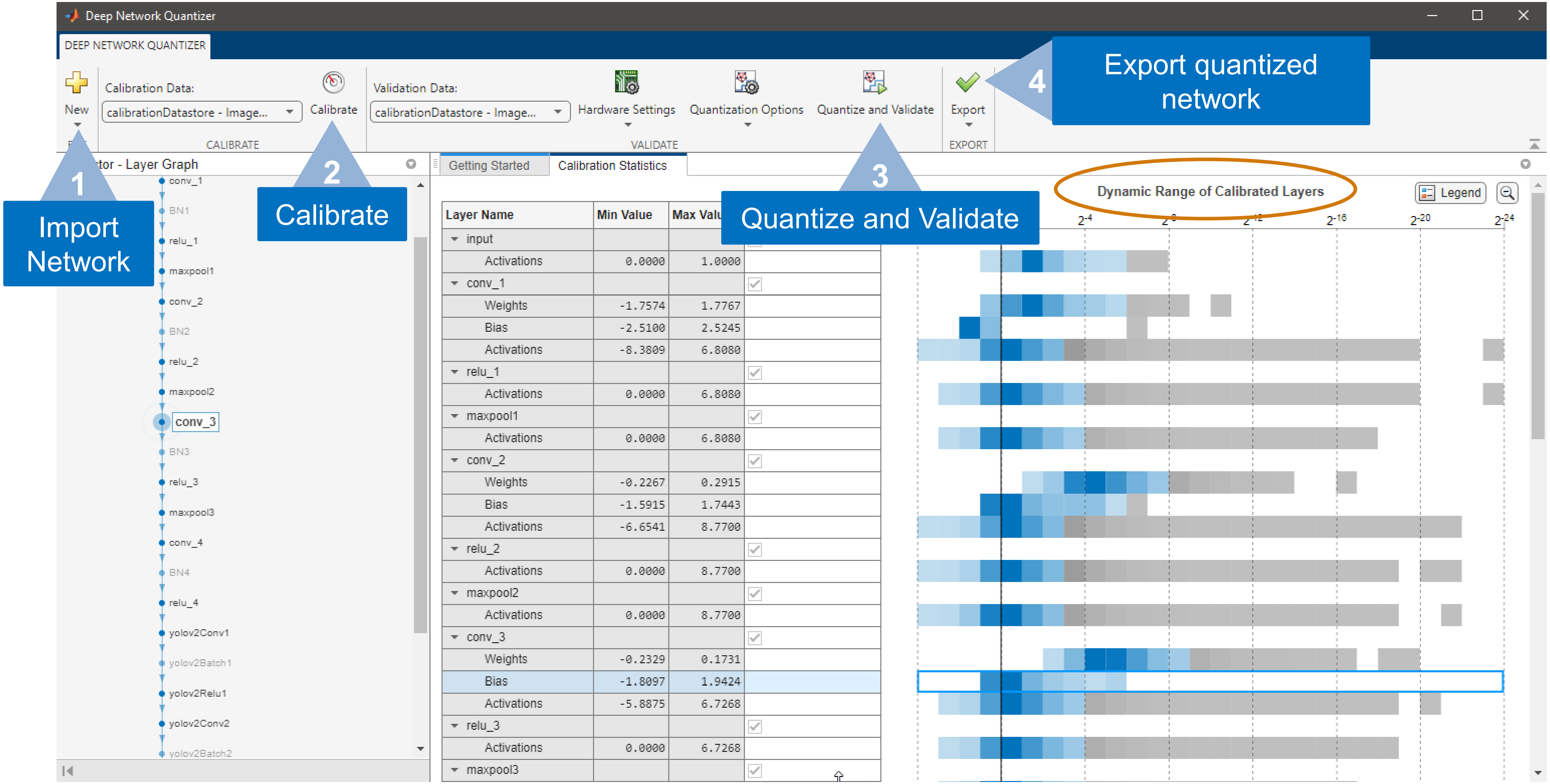
- **New in R2022b - Projected Layer Pruning**

41.8k	8	0 ⚠	0 ❗
total learnables	layers	warnings	errors

  – Memory Footprint: **1.85 MB → 167 KB** (**91%** reduction)
  – Inference Latency: **3.3 sec → 1.7 sec** (**48%** reduction)



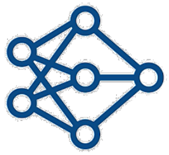
Deep Network Quantizer - Int8 Quantization

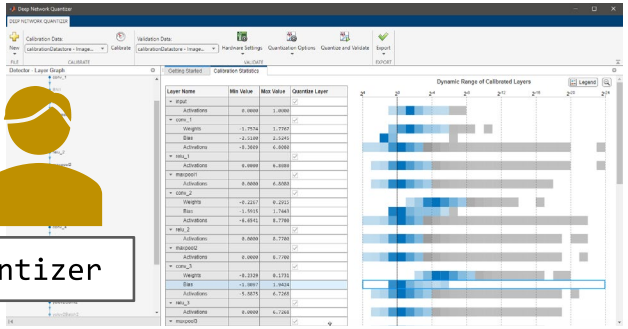


Quantize Deep Learning Network and Processor in MATLAB

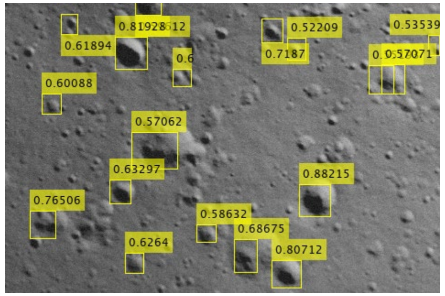
```
>>wobj=dlhdl.Workflow('Network', detector.Network, 'Bitstream','zcu102_int8');
>> dn = wobj.compile;
>> wobj.Target = dlhdl.Target('Xilinx', 'Interface', 'Ethernet', 'IPAddress', '192.168.4.2');
>> wobj.deploy;
>> [predict_out, speed] = wobj.predict(img_pre,'Profile','on');
```

Application logic



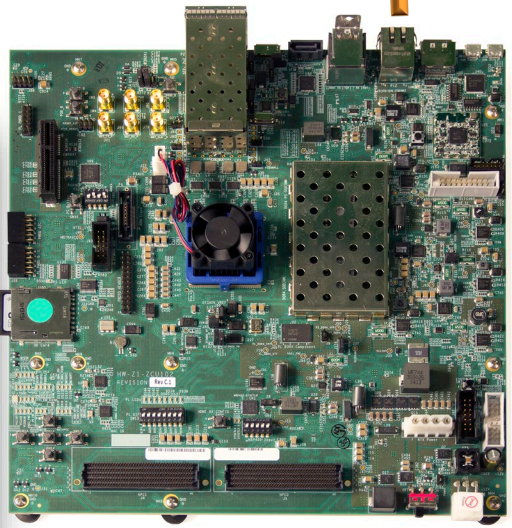


```
>> deepNetworkQuantizer
```



Layer control instructions

Weights & Activations



Deep Learning Processor Profiler Performance Results

	LastFrameLatency(cycles)	LastFrameLatency(seconds)	FramesNum	Total Latency	Frames/s
Network	575217	0.00230	1	575799	434.2
conv_1	99836	0.00040			
maxpool1	65019	0.00026			
conv_2	66616	0.00027			
maxpool2	31568	0.00013			
conv_3	53503	0.00021			
maxpool3	18154	0.00007			
conv_4	46168	0.00018			
yolov2Conv1	84962	0.00034			
yolov2Conv2	85134	0.00034			
yolov2ClassConv	24226	0.00010			

* The clock frequency of the DL processor is: 250MHz

Converge on an FPGA-Optimized Deep Learning Network

Application logic

```
% Create target object
hTarget = dlhdl.Target(…)

% Create workflow object, using the target
hW = dlhdl.Workflow(…);

% Compile the network
hW.compile;

% Program the bitstream and deploy the compiled network and weights
hW.deploy;

% Run prediction
[score, speed] = hW.predict(img, ‘Profile’, ‘on’);
```

```
>> deepNetworkQuantizer
```

Quantize

Parameters	Speed
48 MB	127.2 fps
44 MB	433.8 fps

Layer control instructions

Weights & Activations

int8 Bitstream

Bitstream Name	ConvThreadNumber	FCThreadNumber	Lookup Table(LUT) Utilization(%)	Block RAM (BRAM) Utilization(%)	DSP Utilization (%)
zcu102_single	16	4	90	63.7	15
zcu102_int8	64	16	62	49	31

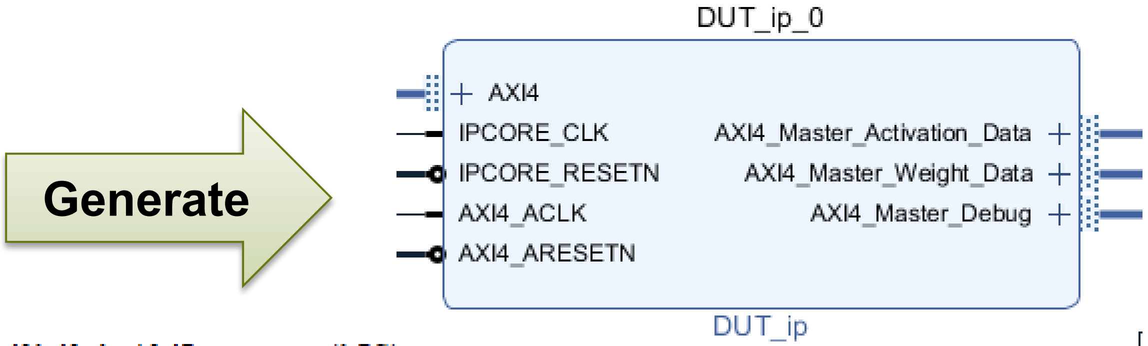
Integrate the Deep Learning Processor into your bigger system

- Generate Generic Deep Learning Processor IP core
- Define clean input/output frame hand-shaking protocol
- Drop the generated Deep Learning IP core into your bigger system

Processor Config

```
        Top Module Properties
DeepLearningIPInputInterface: 'DDR Interface'
              KernelDataType: 'single'

      System Level Properties
              TargetPlatform: 'Generic Deep Learning Processor'
             TargetFrequency: 200
               SynthesisTool: 'Xilinx Vivado'
             ReferenceDesign: ''
     SynthesisToolChipFamily: 'Zynq UltraScale+'
     SynthesisToolDeviceName: 'xczu9eg-ffvb1156-2-e'
    SynthesisToolPackageName: ''
     SynthesisToolSpeedValue: ''
```




```
>> dlhdl.buildProcessor(hPC)
```

Network Examples

Network Examples	Application Area	Type	Release
VGG16/VGG19	Classification	CNN	R2021b
Darknet19	Classification	CNN	
ResNet18/ResNet50	Classification/Detection	CNN	
YOLO v2	Object detection	CNN	
MobileNet v2	Classification/Detection	CNN	
1-Dimentional CNN networks	Classification/Detection	CNN	R2022a
Segmentation networks	Segmentation	CNN	
LSTM networks	Signal processing	RNN	R2022b
YOLO v3	Object detection	CNN, MIMO	
GRU network	Signal processing	RNN	R2023a
YAMnet (Audio toolbox)	Classification/Detection	CNN	

Why MATLAB & MathWorks for AI?

Domain-specialized workflows for **engineering and science**

Multi-platform **deployment of full applications and systems**

SIMULINK®

Platform productivity

Interoperability with Python and DL Python-based frameworks

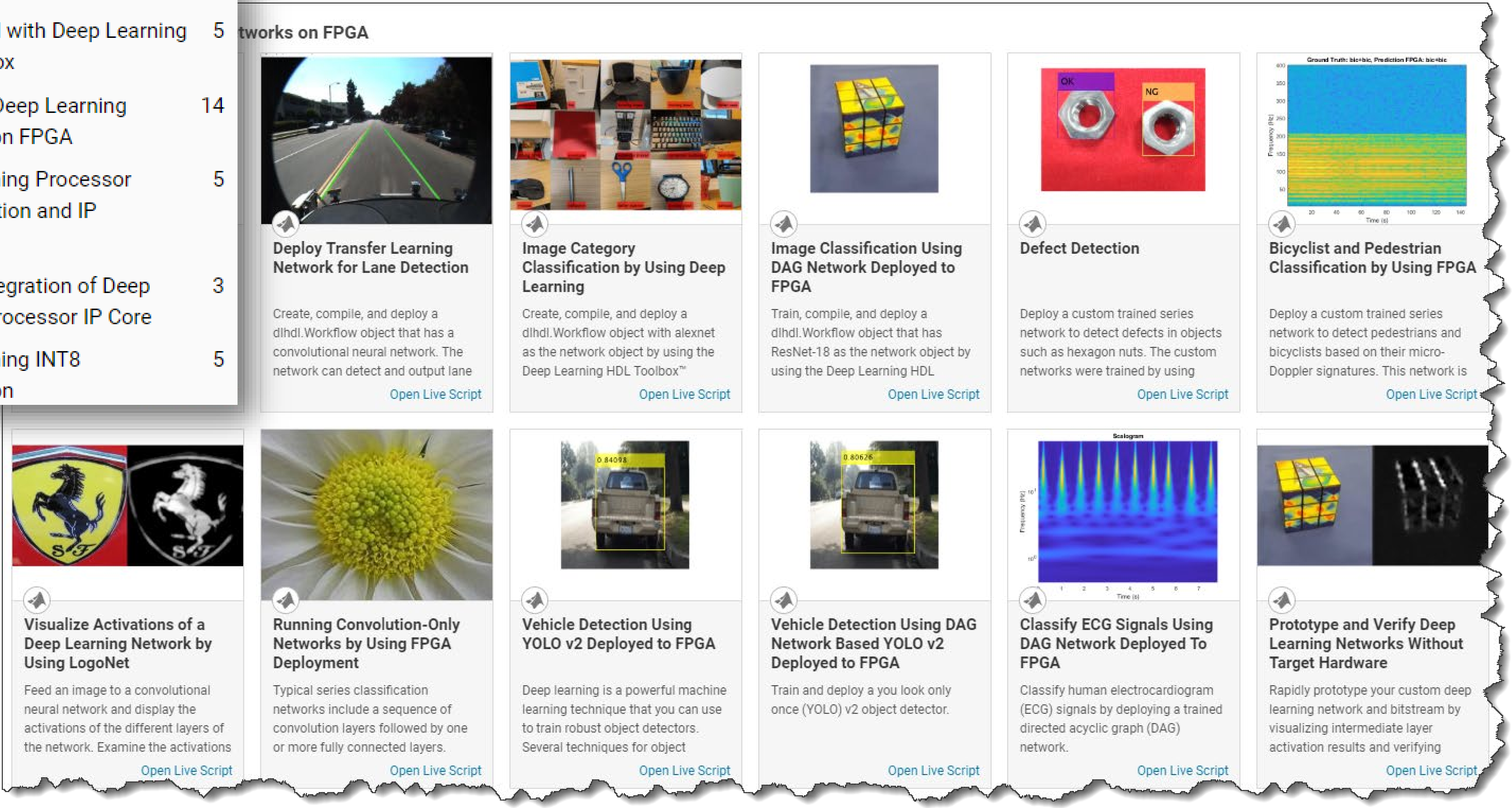
TensorFlow™
PyTorch
ONNX

People


ANY
QUESTIONS
?

Examples

Deep Learning HDL Toolbox



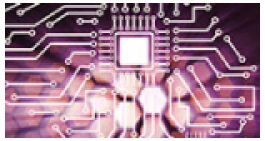
Training Resources

Machine Learning Onramp

Free

6 modules | 2 hours | Languages

Learn the basics of practical machine learning methods for classification problems.



Machine Learning with MATLAB

7 modules | 12 hours | Languages

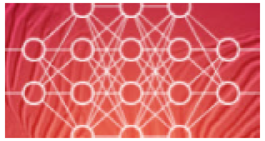
Explore data and build predictive models.

Deep Learning Onramp

Free

5 modules | 2 hours | Languages

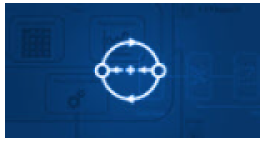
Get started quickly using deep learning methods to perform image recognition.



Deep Learning with MATLAB

13 modules | 8 hours | Languages

Learn the theory and practice of building deep neural networks with real-life image and sequence data.



Reinforcement Learning Onramp

Free

5 modules | 3 hours | Languages

Master the basics of creating intelligent controllers that learn from experience.

https://matlabacademy.mathworks.com/

Deep Learning Onramp

0%

Start course

Share | Certificate | Settings

Course Description

Get started quickly using deep learning methods to perform image recognition.

Course Author

Renee Bach

Format	Self-paced online
Duration	2 hours
Language	English (set language)

Modules

- > Introduction 5 min
- > Using Pretrained Networks 20 min
- > Managing Collections of Image Data 30 min
- > Performing Transfer Learning 60 min
- > Conclusion 10 min

MathWorks training options for AI topics

MATLAB Skills Assessment

Foundational/ Core MATLAB
- MATLAB Fundamentals (3 days)
- MATLAB for Data Processing and Visualization (1 day)
- Processing Big Data with MATLAB (1 day)
- Building Interactive Applications in MATLAB (1 day)

Image & Video Processing
- Image Processing with MATLAB (2 days)
- Computer Vision with MATLAB (1 day)

Machine and Deep Learning
- Signal Processing and Feature Extraction for Data Analytics with MATLAB (1 day)
- Machine Learning with MATLAB (2 days)
- Deep Learning with MATLAB (2 days)

Implementation
- Generating HDL Code from Simulink (2 days)
- DSP for FPGAs (3 days)
- Embedded Coder for Production Code Generation (2 days)
- Polyspace for C/C++ Code Verification (2 days)

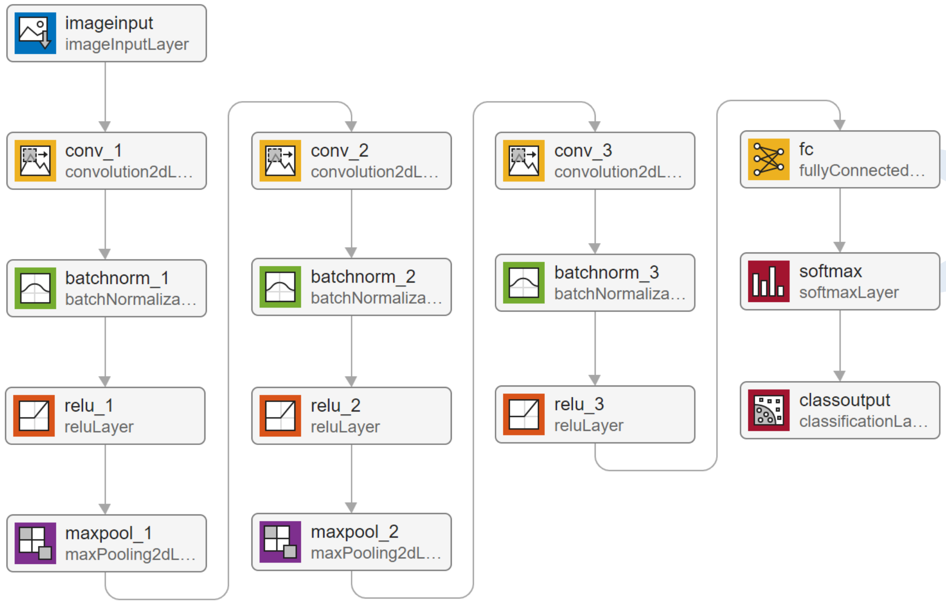
Resources for Further Learning

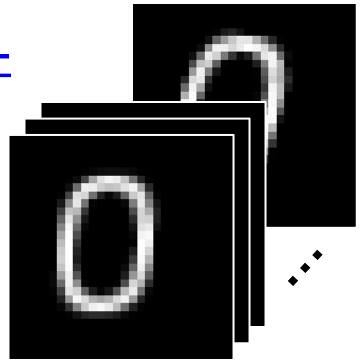
- Crater Detection - Deep Learning
  - Deep Learning Solutions in MATLAB
  - Deep Learning Verification Library
  - Deep Learning Models
  - MATLAB with TensorFlow and PyTorch
  - Importing Models from TensorFlow, PyTorch, and ONNX
  - TensorFlow-Keras Layers Supported for Conversion into Built-In MATLAB Layers
  - What’s New in Interoperability with TensorFlow and PyTorch
- Crater Detection - Deep Learning ➔ FPGA
  - Deep Learning HDL Toolbox
  - Deep Learning HDL Toolbox Supported Networks, Layers, Boards and Tools

Neuron Coverage for Deep Learning

R2022b

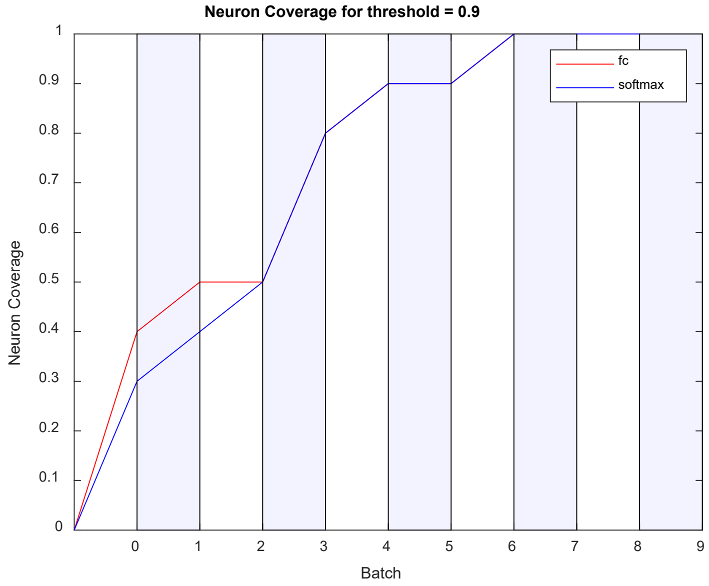
https://github.com/matlab-deep-learning/neuron-coverage-for-deep-learning





		LayerCoverage
1	fc	0.2000
2	softmax	0.1000

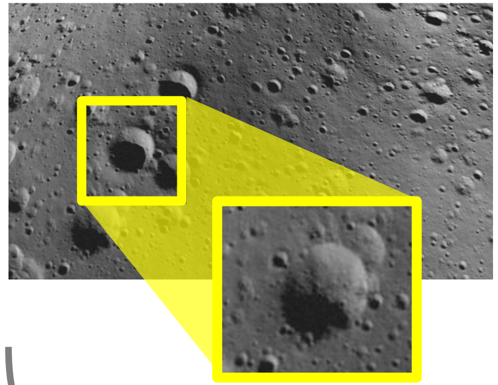
		LayerCoverage
1	fc	0.3000
2	softmax	0.2000



Deep Learning Toolbox Verification Library

R2022b

Verify deep learning network robustness against adversarial examples and to compute the output bounds for a set of input bounds.



+ δ → **Neural Network** →

Crater

No crater

Formal Verification

verified **unproven** **violated**

https://www.mathworks.com/help/deeplearning/deep-learning-verification.html
https://www.mathworks.com/matlabcentral/fileexchange/118735-deep-learning-toolbox-verification-library