Deploying Deep Learning on Embedded Devices – When FPGAs Make Sense

Dr Rishu Gupta

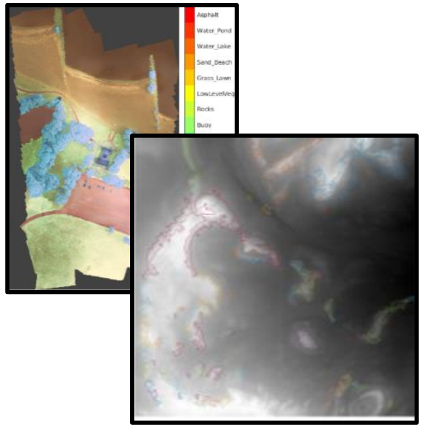
Senior Application Engineer

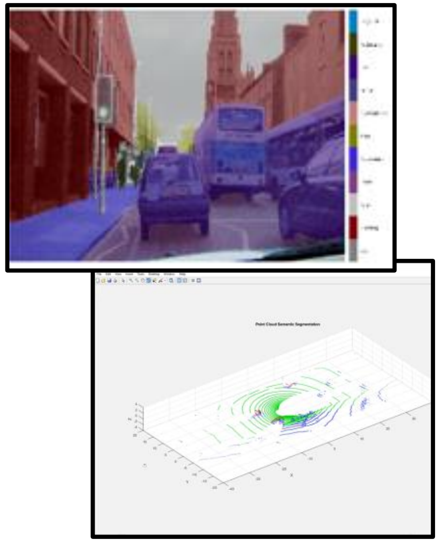
MathWorks

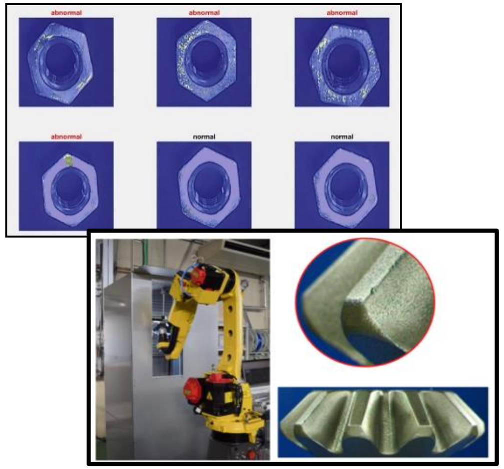
Deep learning applications can be found across many industries

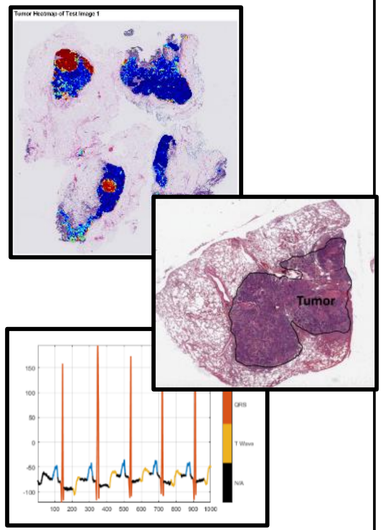
Industries

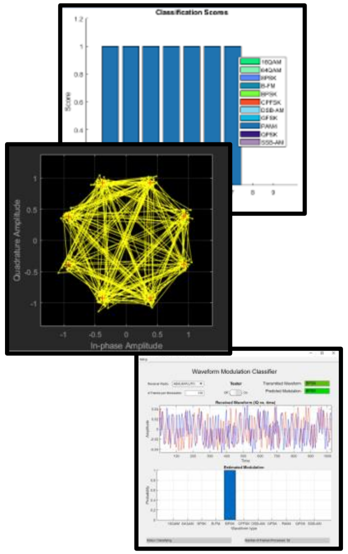
	Aerospace & Defense	Automotive	Industrial Automation	Medical Devices	Communications
Applications	Airborne Image Analysis	Autonomous Driving	Defect Detection	Medical Image and Signal Segmentation	Modulation Classification







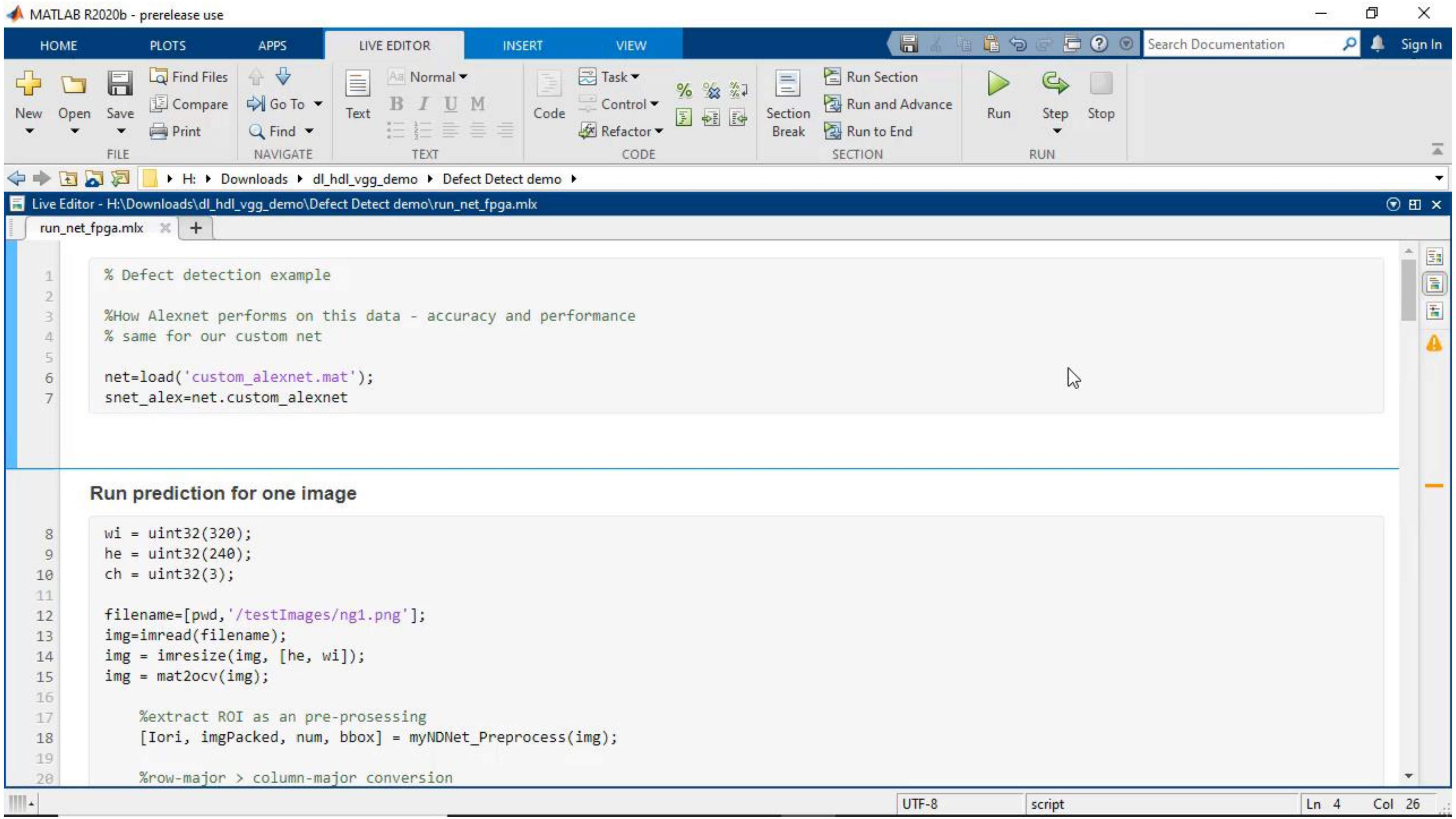




Key Takeaways

- MATLAB provides an easy workflow to prototype and deploy deep learning algorithms on different embedded platforms
  - Ease of deploying to GPUs like Nvidia Jetson, Intel and ARM based CPUs/microprocessors
  - Ease of deploying to Xilinx/Intel FPGAs and SoCs without hardware expertise
  - Optimizing the deep learning networks through INT8 quantization
- We will use defect detection as an example to illustrate.

Demo Overview – Defect Detection Application



Why FPGAs /ASICs?

Same applies for deep learning problems

System Throughput

"Real-time image processing for an aircraft head's up display"

"Evaluate the algorithm in field testing to analyze system performance"

"Optimal performance @ Piezo resonance frequency"

Power

"11 year device with a 1 A*hr battery"

Latency

"Be able to stop the robot with millimeter accuracy in less than 0.5 seconds without causing damage to the robot"

"Audio transducer prototypes must run in real time with low latencies"

"Motor control latency < 1us"

Deep Learning Deployment: Inference on the Edge

Industries

	Aerospace & Defense	Automotive	Industrial Automation	Medical Devices	Communications

Requirements

- High Compute
- Batch processing - Inference speed
- Real time - Latency
- Environmental Conditions (Space)
- Low power, Custom application
- High Speed I/O

Domains:

- Image processing and Computer Vision
- Radar Signal Processing …

Tasks:

- Image Classification
- Object Detection
- Semantic Segmentation ..

- **Red – GPUs are ideal**
- **Blue – FPGAs are ideal**

Reference articles:
https://www.arrow.com/en/research-and-events/articles/fpga-vs-cpu-vs-gpu-vs-microcontroller
http://mil-embedded.com/articles/fpga-gpu-evolution-continues/

Deployment is hard: Challenges

- Deployment to the edge is challenging because of resource constraints

Embedded constraints
Limited memory, Power
Real time performance - Latency

- Manual workflows are tedious and require a significant front end cost
- How to decide the right target platform for your application/ How to have a consistent process to deploy to multiple embedded platforms?

Especially on FPGAs

- Large scale matrix computations
  - TFLOPS: 230M weights and 724M MACs

- Complex architecture
  - Scale of data movement across the DDR

Workflow:

- Exploring multiple networks
- Exploring the resource and performance tradeoffs

```
    int t_size = interpreter->tensors_size();
    for (int i = 0; i < t_size; i++) {
      if (interpreter->tensor(i)->name)
        LOG(INFO) << i << ": " << interpreter->tensor(i)->name << ", "
                  << interpreter->tensor(i)->bytes << ", "
                  << interpreter->tensor(i)->type << ", "
                  << interpreter->tensor(i)->params.scale << ", "
                  << interpreter->tensor(i)->params.zero_point << "\n";
    }
  }

  if (s->number_of_threads != -1) {
    interpreter->SetNumThreads(s->number_of_threads);
  }

  int image_width = 224;
  int image_height = 224;
  int image_channels = 3;
  std::vector<uint8_t> in = read_bmp(s->input_bmp_name, &image_width,
                                     &image_height, &image_channels, s);

  int input = interpreter->inputs()[0];
  if (s->verbose) LOG(INFO) << "input: " << input << "\n";

  const std::vector<int> inputs = interpreter->inputs();
  const std::vector<int> outputs = interpreter->outputs();
```

MATLAB supports the entire deep learning workflow – from Data to Deployment

PREPARE DATA

- Data access and preprocessing
- Simulation-based data generation
- Ground truth labeling

TRAIN MODEL

- Model design and tuning
- Hardware-accelerated training
- Model exchange across frameworks

DEPLOY SYSTEM

- Embedded Devices
- Enterprise Systems
- Edge, cloud, desktop

Iteration and Refinement

Deep Learning Workflow – Deployment

PREPARE DATA

- Data access and preprocessing
- Simulation-based data generation
- Ground truth labeling

TRAIN MODEL

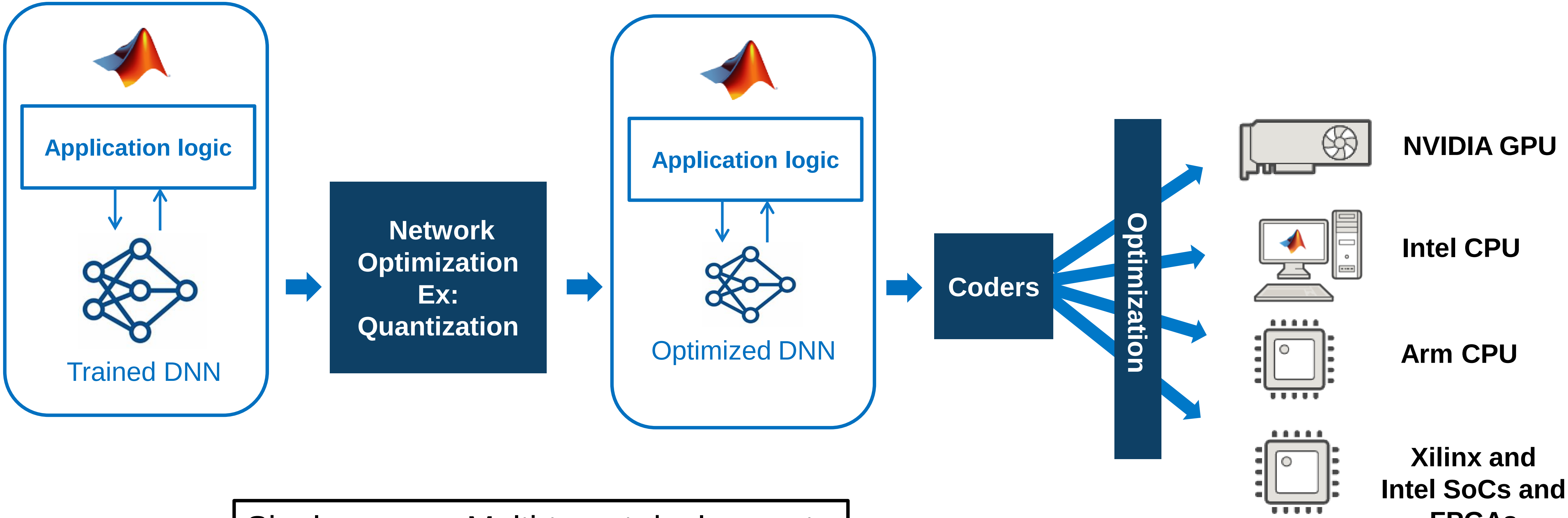
- Model design and tuning
- Hardware-accelerated training
- Model exchange across frameworks

DEPLOY SYSTEM

- Embedded Devices
- Enterprise Systems
- Edge, cloud, desktop

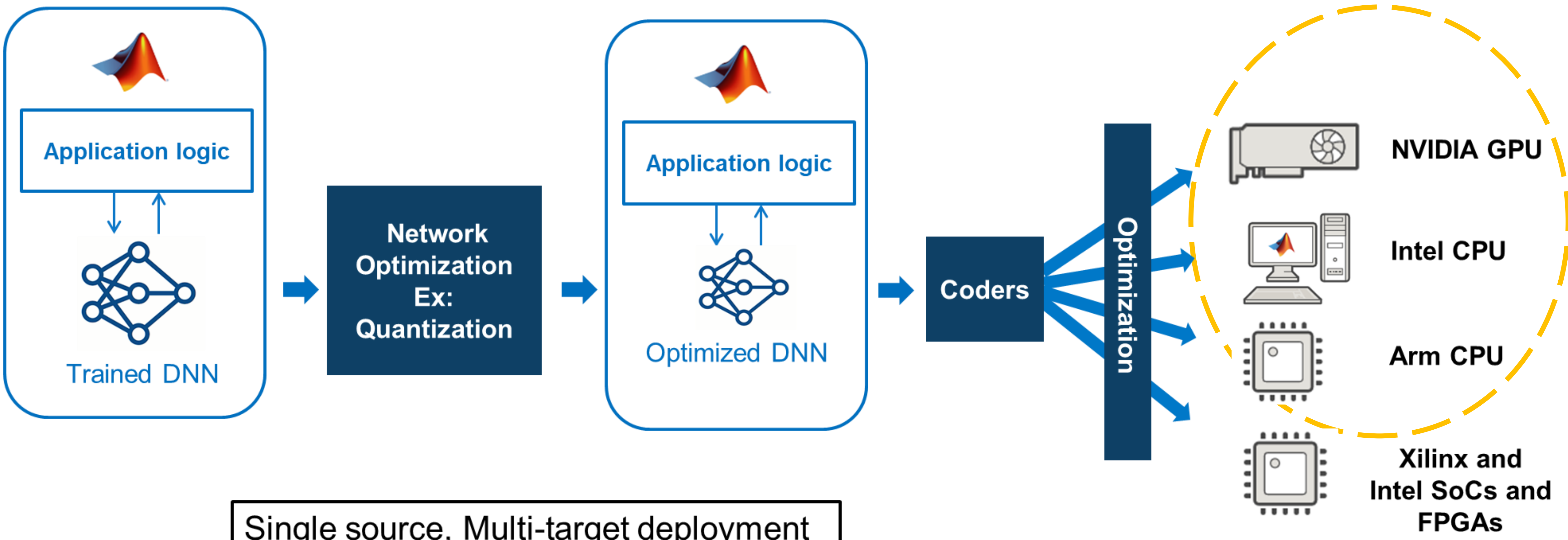
Iteration and Refinement

MATLAB enables multi-target deployment



Single source, Multi-target deployment

Multi-target deployment

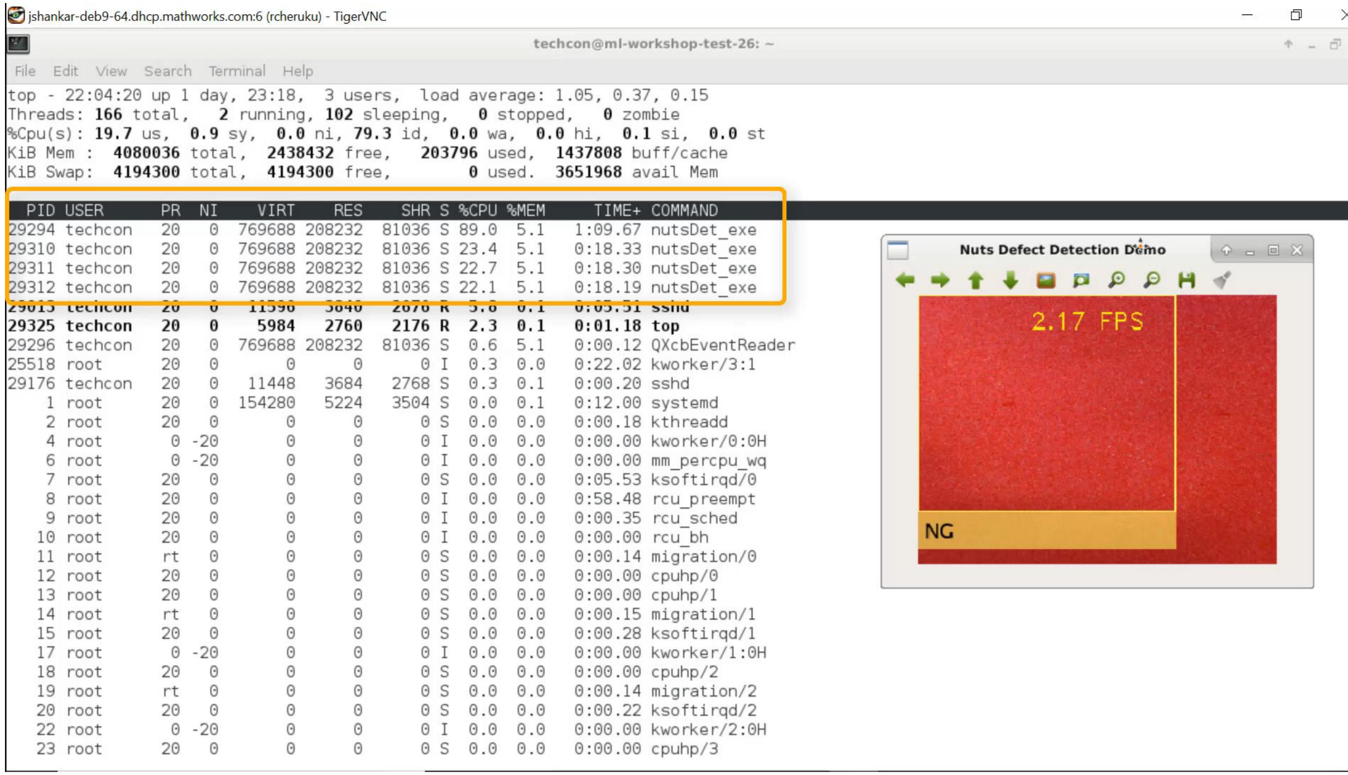


Single source, Multi-target deployment

Prototyping and Deployment workflow: GPUs and CPUs

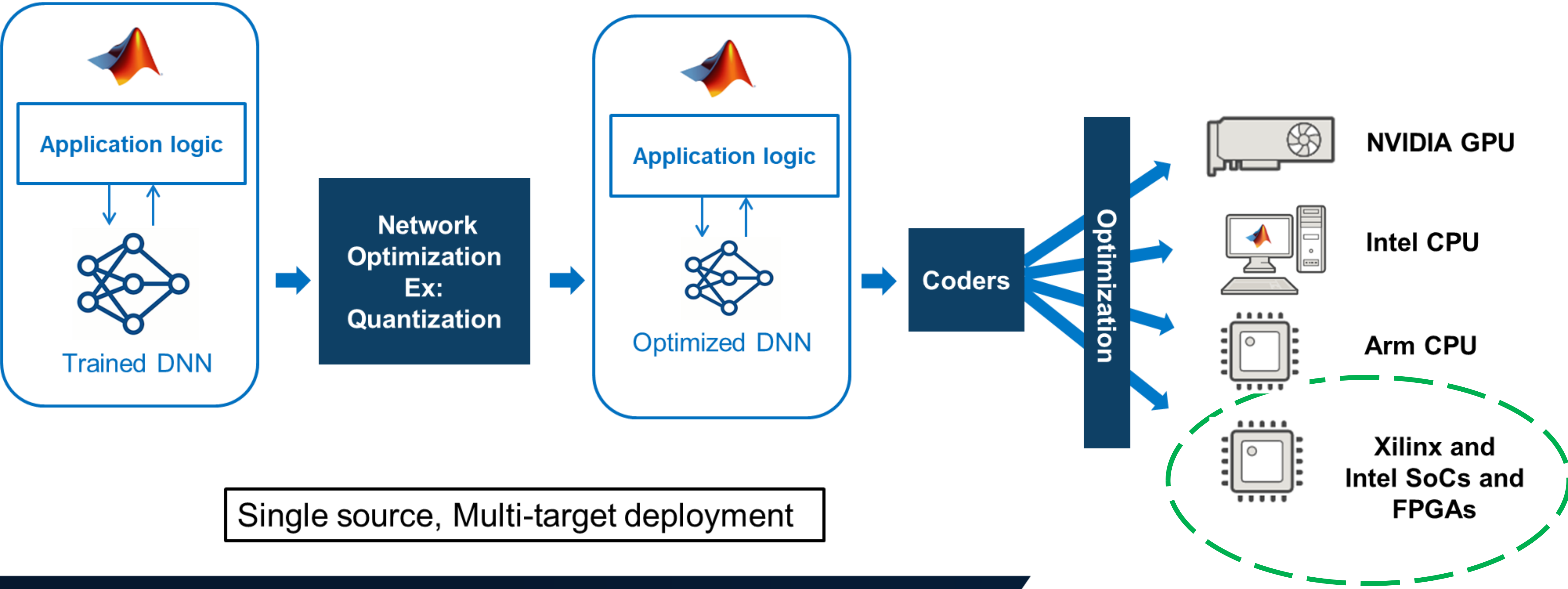
Resources:

- [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 deployed on ARM Cortex-A microprocessor

Multi-target deployment



Single source, Multi-target deployment

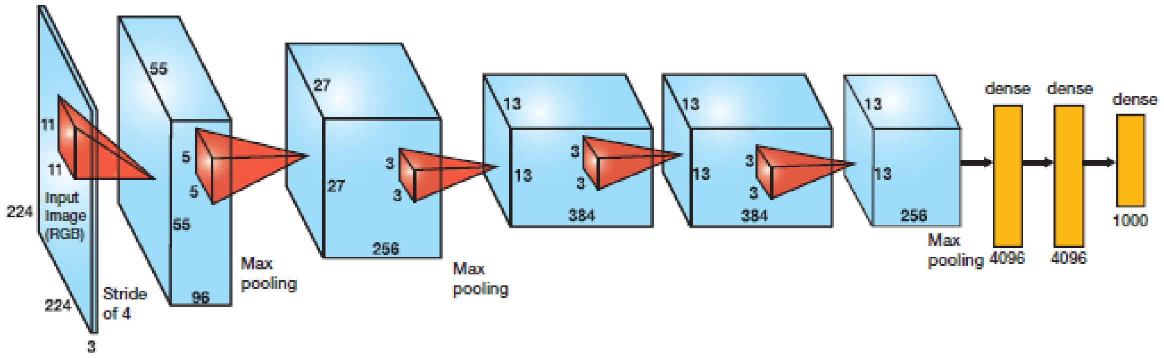
Challenges of deploying Deep learning models on FPGAs

- Large scale matrix computations
  - TFLOPS: 230M weights and 724M MACs
- Complex architecture
  - Scale of data movement across the DDR

Workflow:

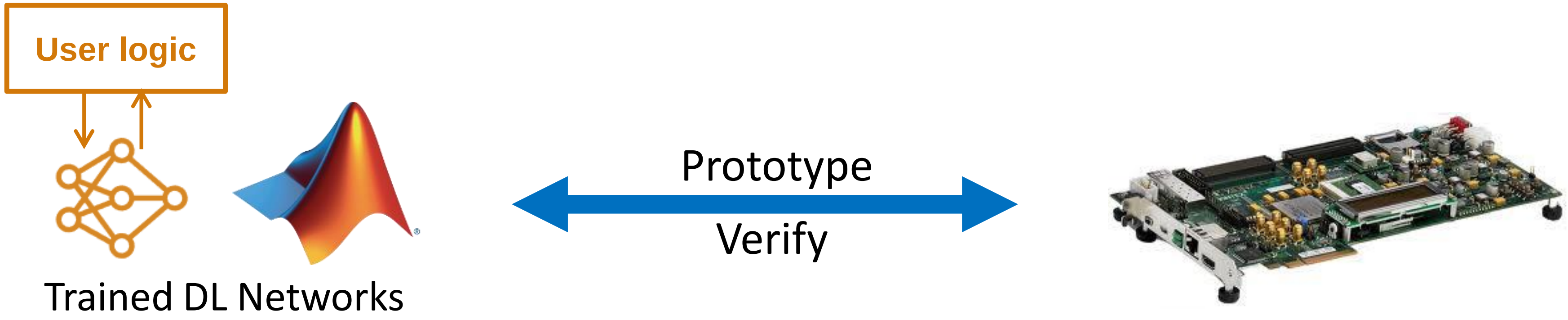
- Exploring multiple networks
- Exploring the resource and performance tradeoffs

	input	conv 1	conv 2	conv 3	conv 4	conv 5	fc6	fc7	fc8	Total	
Parameters (Bytes)	n/a	140K	1.2M	3.5M	5.2M	1.8M	148 M	64M	16M	230 M	DDR
Activations (Bytes)	588K	1.1M	728K	252K	252K	168K	16K	16K	4K	3.1 M	BRAMs
FLOPs	n/a	105 M	223 M	149 M	112M	74M	37M	16M	4M	720 M	DSPs



Deep learning networks are too big for FPGAs

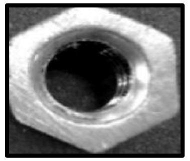
Prototyping and Deploying Deep Learning Networks from MATLAB to FPGA



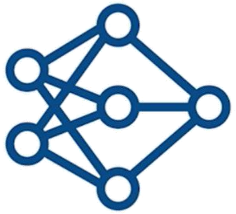
1. No HDL Knowledge Required
2. Ease of prototyping on FPGA from MATLAB
3. Ease of exploring various DL networks and customizing them to your application

```
function out = targetFunction(img)
%#codegen
coder.inline('never');

%extract ROI as an pre-prosessing
[imgPacked, num, bbox] =
myNDNet_Preprocess(img);

%classify detected nuts by using CNN
scores = zeros(2,4);
for i = 1:num
    scores(:,i) =
predict(imgPacked(:,:,i));
end

%insert annotation as an post-processing
out = myNDNet_Postprocess(img, num, bbox,
scores);

end
```

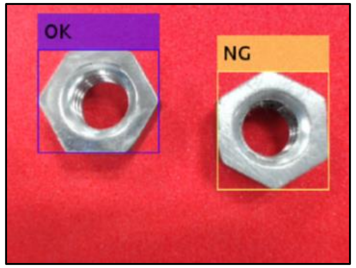
Extract regions



Resize

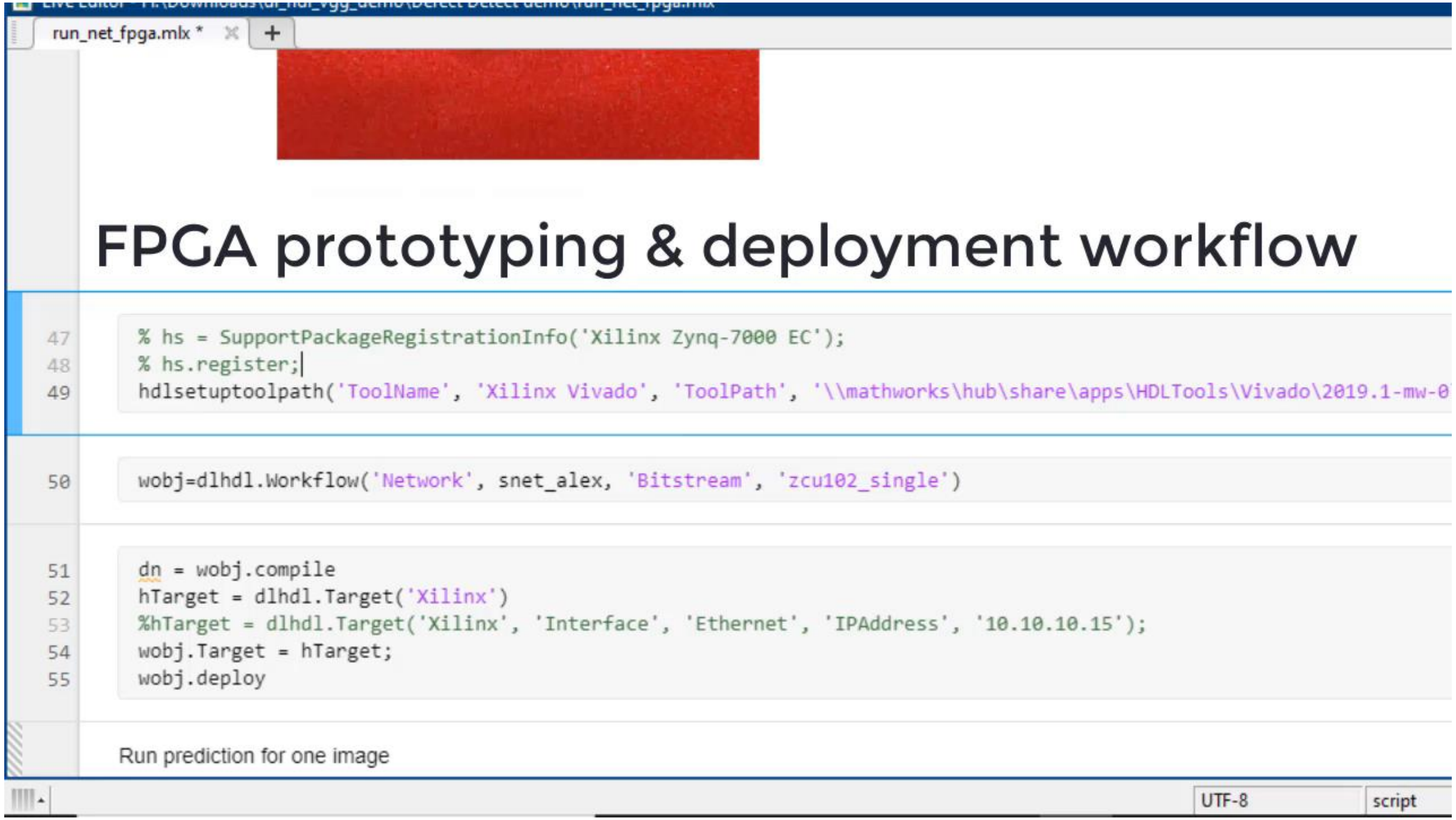


Prediction from the trained network aka Inference



Annotate and label

Prototyping and Deploying Deep Learning Networks from MATLAB to FPGA



run_net_fpga.mlx *

FPGA prototyping & deployment workflow

```
% hs = SupportPackageRegistrationInfo('Xilinx Zynq-7000 EC');
% hs.register;
hdlsetuptoolpath('ToolName', 'Xilinx Vivado', 'ToolPath', '\\mathworks\hub\share\apps\HDLTools\Vivado\2019.1-mw-0
```

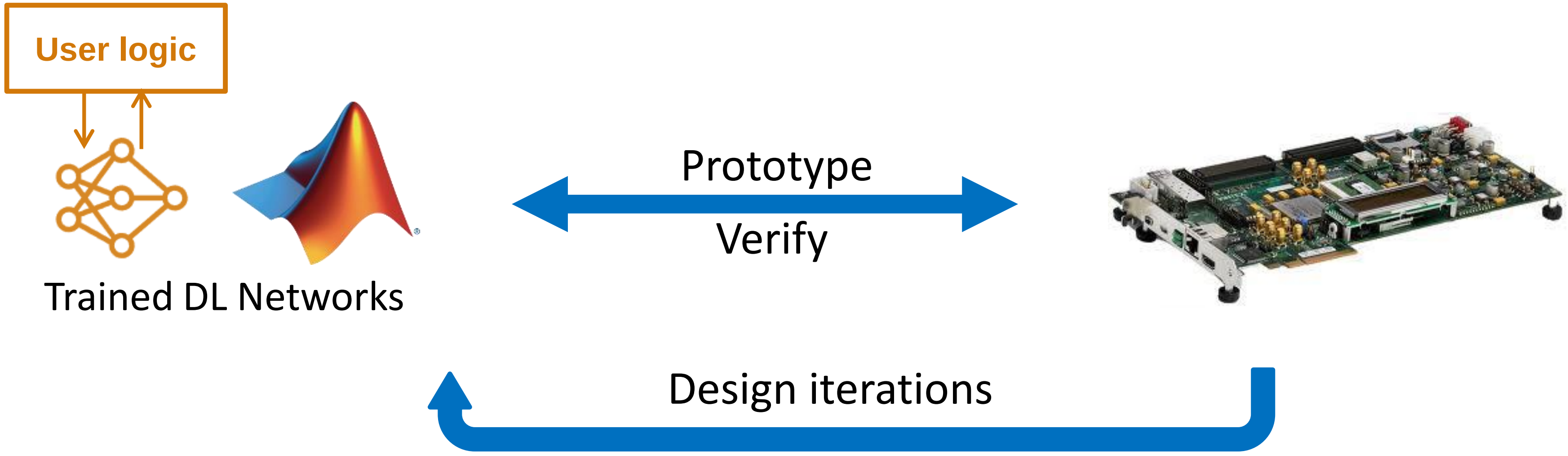
```
wobj=dlhdl.Workflow('Network', snet_alex, 'Bitstream', 'zcu102_single')
```

```
dn = wobj.compile
hTarget = dlhdl.Target('Xilinx')
%hTarget = dlhdl.Target('Xilinx', 'Interface', 'Ethernet', 'IPAddress', '10.10.10.15');
wobj.Target = hTarget;
wobj.deploy
```

Run prediction for one image

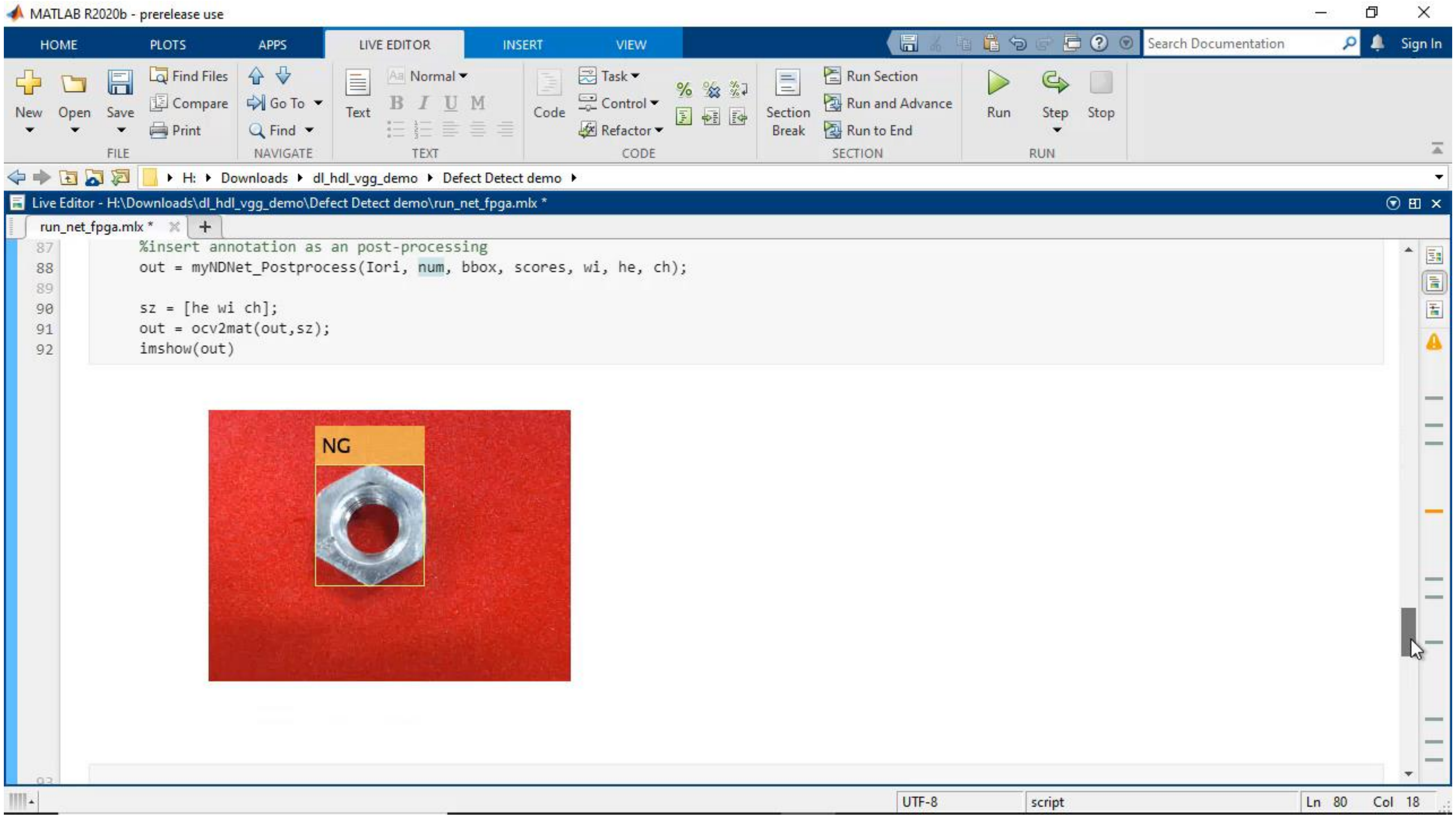
UTF-8 script

Prototyping: Design Exploration and Customization

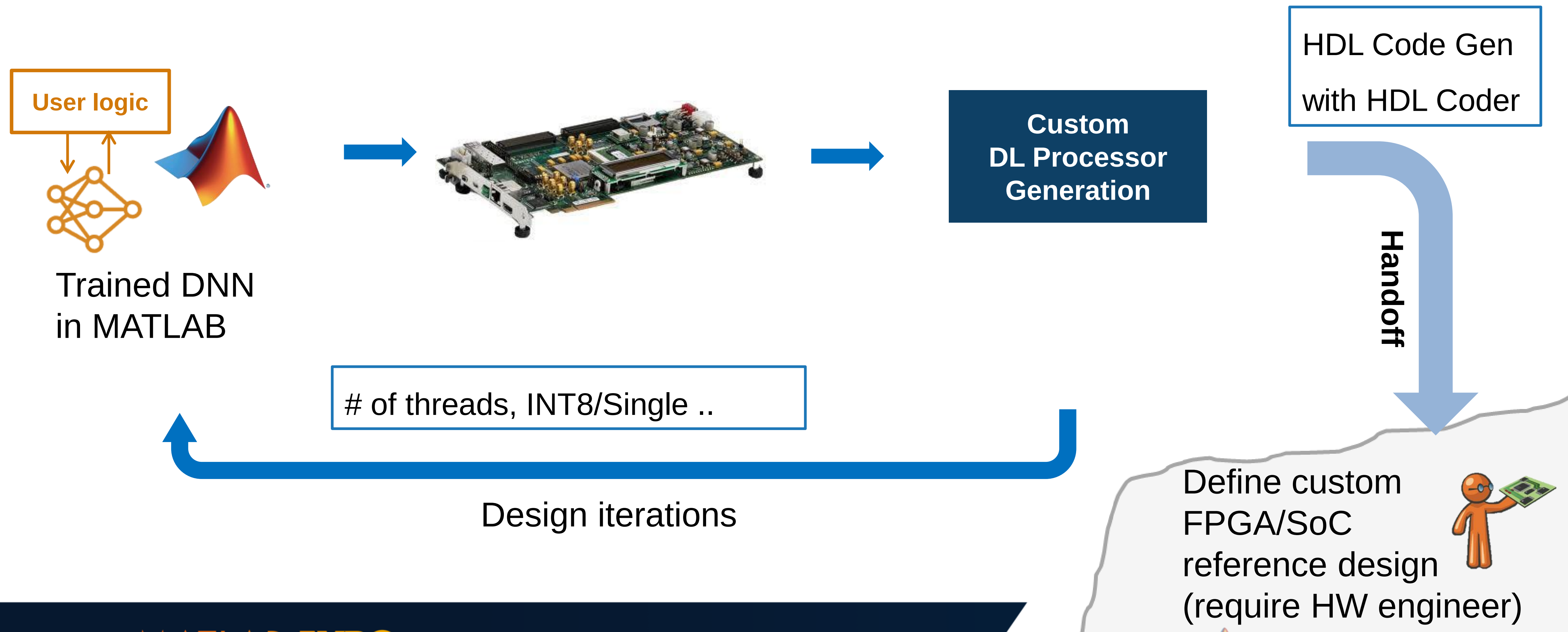


- Most cases, you want to customize the network for your application: Deep Network Designer workflow
- Iteratively deploy and run on the FPGA

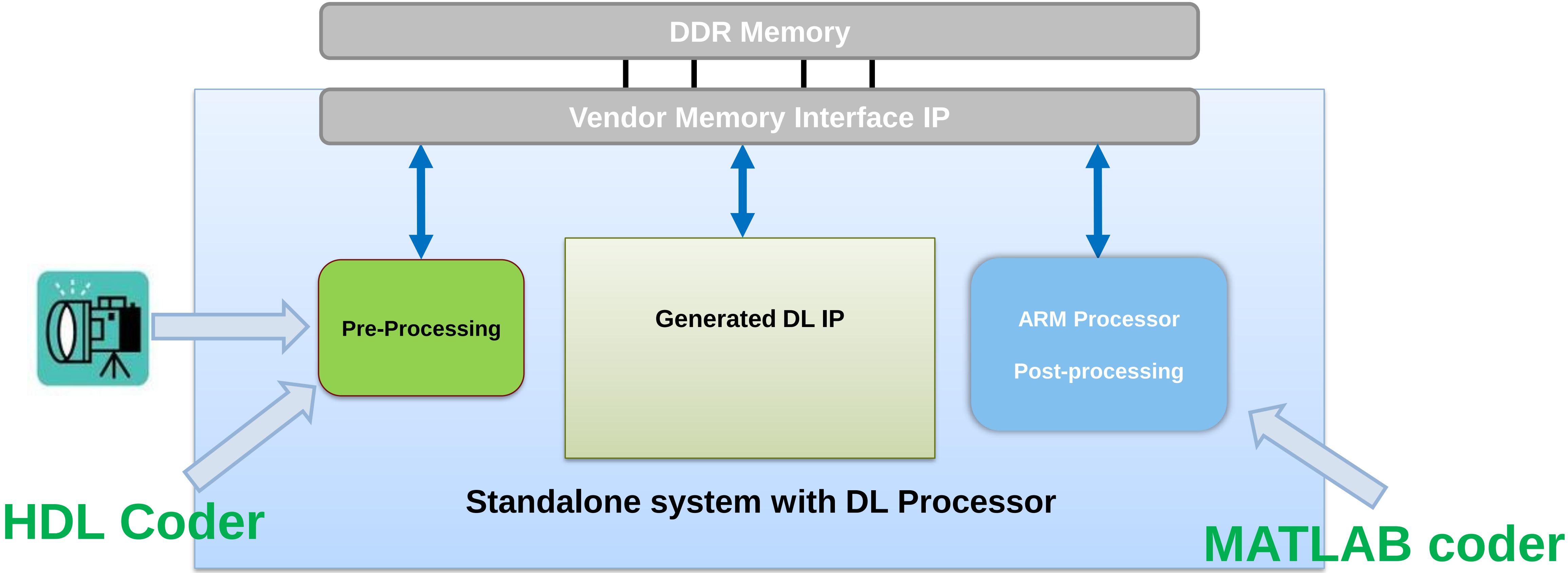
Design Exploration and Customization



Generate Custom DL Processor & Integrate Deep Learning network into your application



Integrate Deep Learning network into your System

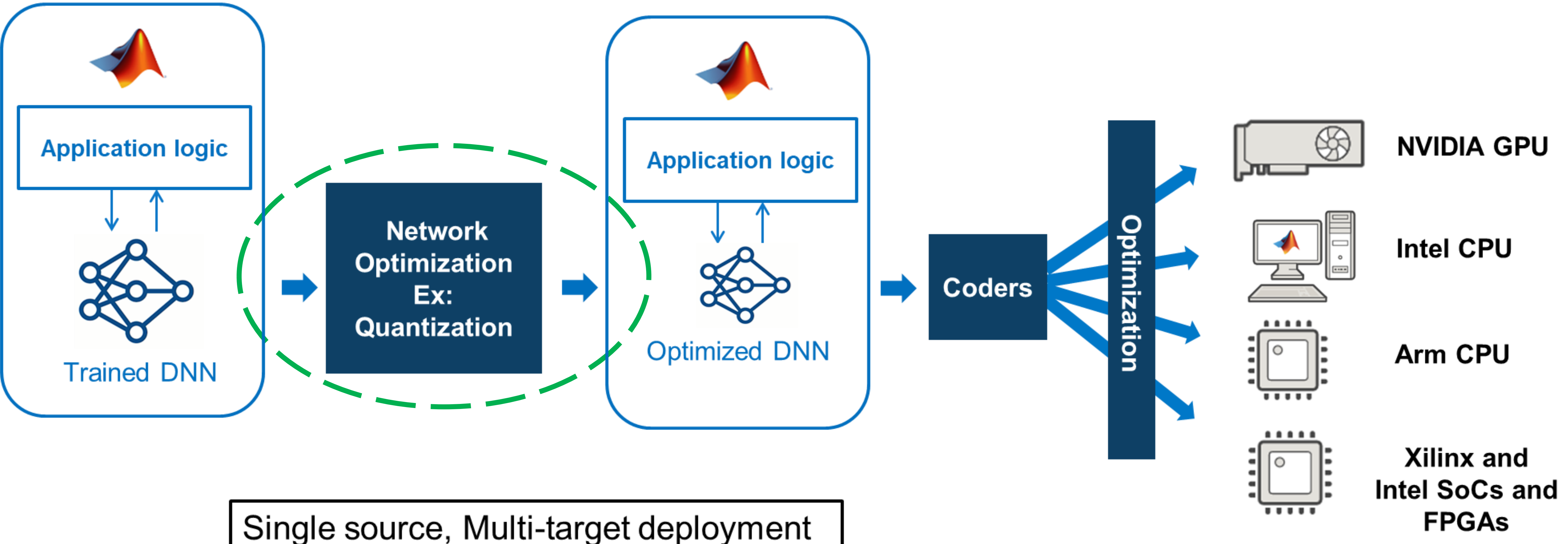


Prototyping and Deploying Deep Learning Networks from MATLAB to FPGA

- Supported boards:
  - Xilinx boards - MPSoC - ZCU102, ZC706
  - Intel Arria 10 SoC
  - Custom boards via code generation

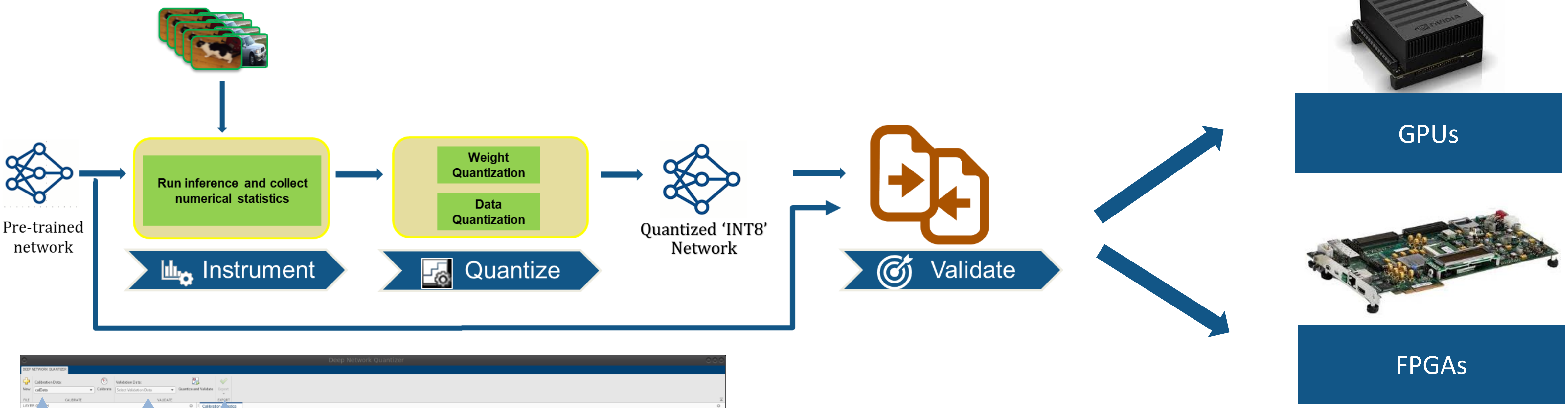
- Supported Networks
  - CNNs: series networks – VGG, Alexnet etc.
  - object detectors - YoloV2

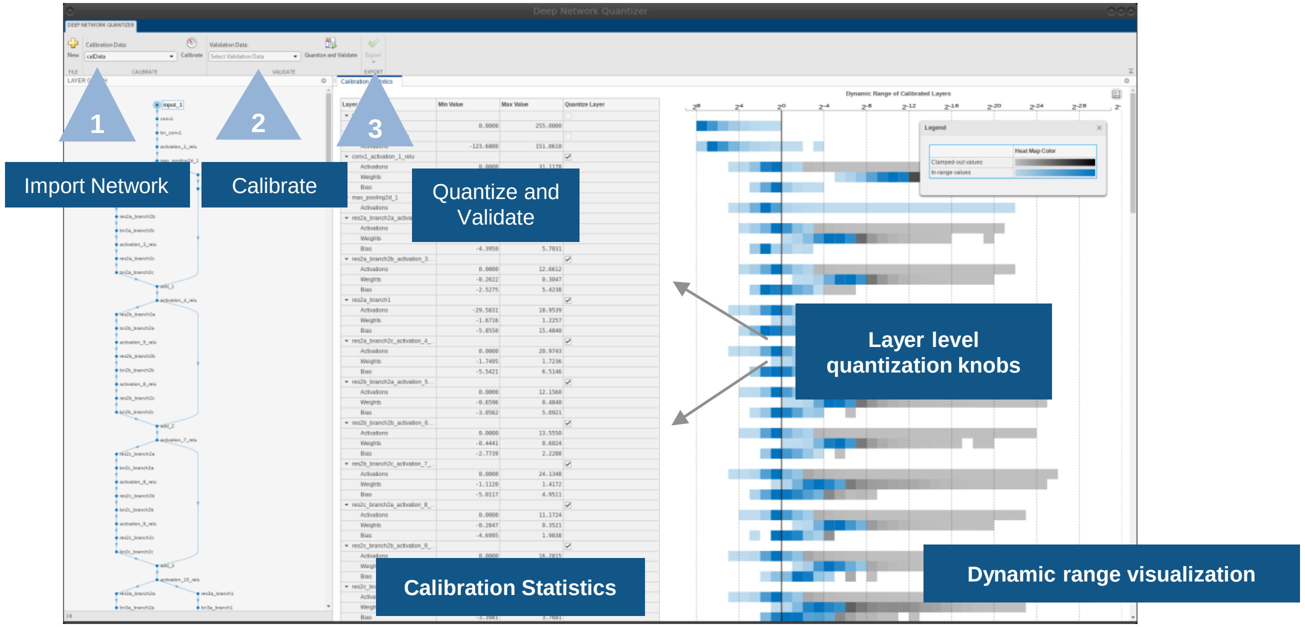
Multi-target deployment



Single source, Multi-target deployment

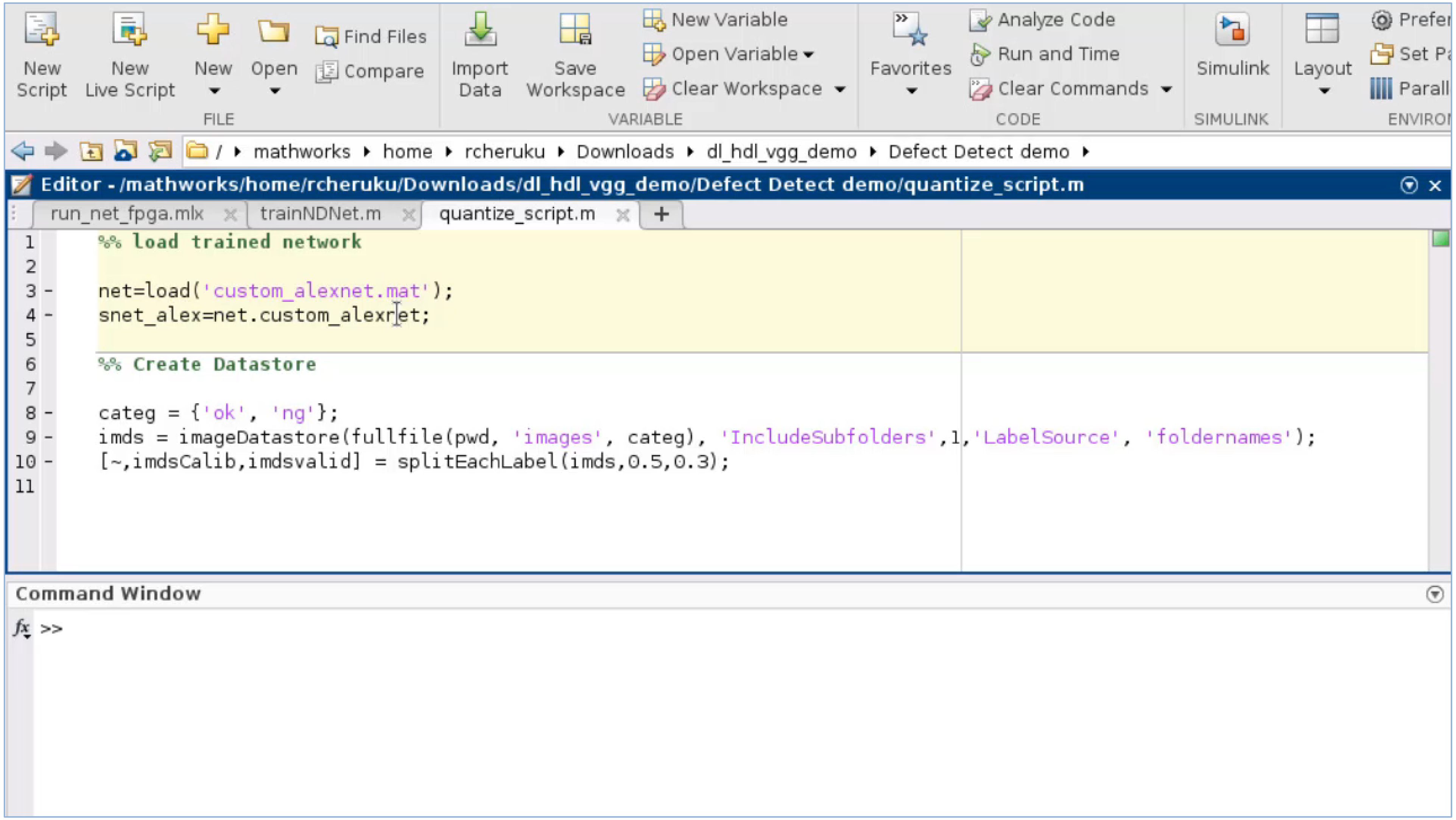
Model Quantization Library





- Workflow to quantize & validate a network to INT8

INT8 Quantization

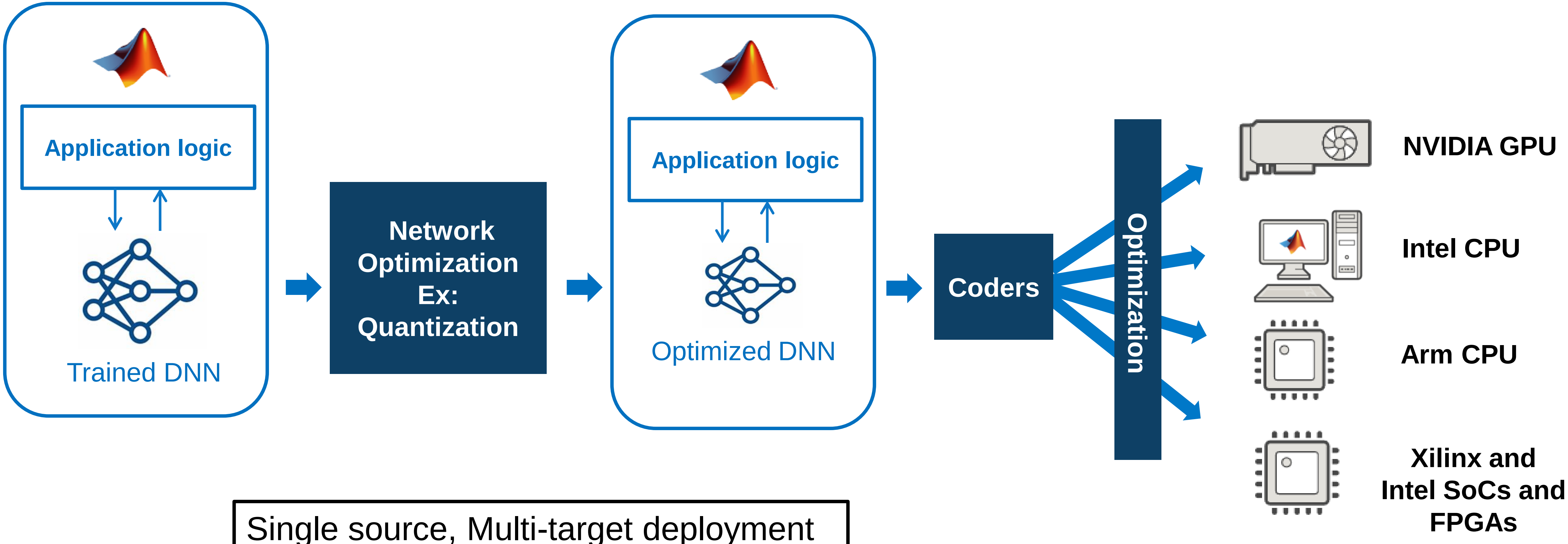


INT8 Quantization

```
wfObj.compile

         offset_name            offset_address     allocated_space
    ______________________     ______________     _________________

    "InputDataOffset"           "0x00000000"       "16.0 MB"
    "OutputResultOffset"        "0x01000000"       "4.0 MB"
    "SystemBufferOffset"        "0x01400000"       "28.0 MB"
    "InstructionDataOffset"     "0x03000000"       "4.0 MB"
    "ConvWeightDataOffset"      "0x03400000"       "4.0 MB"
    "FCWeightDataOffset"        "0x03800000"       "12.0 MB"
    "EndOffset"                 "0x04400000"       "Total: 68.0 MB"
```

```
              Deep Learning Processor Profiler Performance Results

                   LastLayerLatency(cycles)   LastLayerLatency(seconds)       FramesNum      Total Latency     Frames/s
                         -------------             -------------              ---------        ---------       ---------
Network                    2138658                  0.00713                       1            2138697          140.3
    conv_module             650904                  0.00217
        conv_1              260213                  0.00087
        maxpool_1            93888                  0.00031
        crossnorm           126372                  0.00042
        conv_2              150355                  0.00050
        maxpool_2            20144                  0.00007
    fc_module              1487754                  0.00496
        fc_1               1479124                  0.00493
        fc_2                  8629                  0.00003
 * The clock frequency of the DL processor is: 300MHz
```

MATLAB enables multi-target deployment



Single source, Multi-target deployment

Customer References

Airbus: Artificial Intelligence & Deep Learning for Automatic Defect Detection

- *An integrated tool to design, train and deploy deep learning models*
- *Interactive prototyping and testing in a very short amount of time*
- *Direct translation from MATLAB language to CUDA code*

> "Having the possibility to **test**, **modify**, **train** and **test again** the code in a **short timeframe** was key to success."

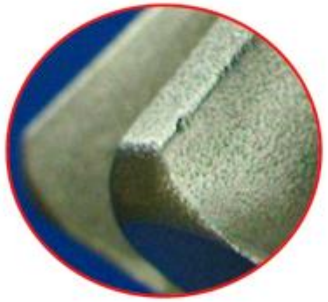
Running on NVIDIA Jetson

Customer References

Musashi Seimitsu Industry: Detect Abnormalities in Automotive Parts

> "Using camera connection, preprocessing, and various pretrained models in MATLAB enabled us to work on the entire workflow. Through discussions with consultants, our team gained many tips for solving problems, growing the skills of our engineers."

- *Enable a seamless development workflow from image capture to implementation on embedded GPU*
- *Image annotation for training and Preprocessing of captured images*
- *Deployment to NVIDIA Jetson using GPU Coder*



Deep Learning Deployment Solution Summary

- **MATLAB provides an end to end workflow for the complete application**
  - **offers an easy automated workflow for optimal deployment on different embedded platforms**
  - **simplifies the workflow for FPGAs both for design exploration & prototyping as well as HDL code generation**
- **Call to action:**
  - Deep Learning onramp
  - **Services**
    - **Training- Deep Learning using MATLAB**
    - **Consulting**
  - Contact your rep to try GPU Coder or HDL Coder

Contact Details:

Email: rishu.g@mathworks.com

LinkedIn: https://www.linkedin.com/in/rishu-gupta-72148914/