Deploying Deep Learning on Embedded Devices
– When FPGAs Make Sense

Jack Erickson
HDL Technical Marketing
Deep Learning Inference on Embedded Devices

Airborne Image Analysis

Autonomous Driving

Industrial Inspection

Medical Image Analysis

Wireless Modulation Classification

Radar Signature Classification
System Requirements Drive Network Design

Industrial Inspection

Deep Learning Practitioner

Systems Engineer

Hardware/Software Engineers

Camera specs
- Accuracy
- Latency
- Cost
- Power

MATLAB EXPO

MathWorks
Challenges of Deploying Deep Learning to FPGA Hardware: Convolution

Each stride is an $11\times11\times3$ matrix multiply-accumulate $\rightarrow 105M$ floating-point multiply operations!

$96$ filters of $11\times11\times3$ of $32$-bit parameters $\rightarrow 140k$ bytes

$11\times11$ $\rightarrow 1.16M$ bytes of activations

$105M$ floating-point multiply operations!
Challenges of Deploying Deep Learning to FPGA Hardware

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

- **Off-chip RAM**
- **Block RAM**
- **DSP Slices**
Deploying Deep Learning to FPGA Hardware Requires Collaboration

<table>
<thead>
<tr>
<th>Parameters (Bytes)</th>
<th>input</th>
<th>conv 1</th>
<th>conv 2</th>
<th>conv 3</th>
<th>conv 4</th>
<th>conv 5</th>
<th>fc6</th>
<th>fc7</th>
<th>fc8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activations (Bytes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOPs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optimize
- Network / layers
- Fixed-point quantization
- Processor micro-architecture

![Diagram showing data processing flow](image)
### Data Preparation
- Data cleansing and preparation
- Human insight
- Simulation-generated data

### AI Modeling
- Model design and tuning
- Hardware accelerated training
- Interoperability

### System Design
- Integration with complex systems
- System simulation
- System verification and validation

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

---

**Iteration and Refinement**
Design and Analyze Your Networks in MATLAB

**AI Modeling**
- Model design and tuning
- Hardware accelerated training
- Interoperability

**Classification Learner** app to try different classifiers and find the best fit for your data set.

**Deep Network Designer** app to build, visualize, and edit deep learning networks.
MATLAB Interoperates with Other AI Frameworks

**AI Modeling**
- Model design and tuning
- Hardware accelerated training
- Interoperability

**Frameworks and Tools**
- TensorFlow
- Caffe2
- PyTorch
- MxNet
- Chainer

**Discussion Points**
- Keras importer
- Caffe importer

**ONNX**
Deploy from MATLAB to a Variety of Hardware Platforms
FPGA Deployment from MATLAB

- Prototype network on FPGA
- Assess memory usage, latency, and accuracy
  - Adjust network and iterate
  - Quantize to fixed-point
- Generate customized deep learning processor HDL

…all from within MATLAB!
Deep Learning HDL Toolbox Components

Application logic
- Quantize
- Analyze Profile
- Customize Estimate

Compile & Deploy Network
- Layer control instructions
- Weights & Activations

Build Processor
- HDL Coder

IP core interface
- DL Processor HDL
- FPGA Bitstream

Deep Learning Processor
- Memory Access
  - Convolution Module
    - Activations
  - Fully Connected Module
    - Activations

MathWorks
MATLAB EXPO
Get Started Prototyping on FPGA with Deep Learning HDL Toolbox™

Hardware support package
Deep learning processor with I/O and external memory interfaces
- Int8 or single
- Supported boards:
  - Xilinx: ZCU102 or ZC706
  - Intel: Arria10 SoC
- [http://mathworks.com/hardware-support.html](http://mathworks.com/hardware-support.html)

```
%% Create Target Object
hTarget = dlhdl.Target('Xilinx', 'Interface', 'Ethernet', 'IPAddress', '10.10.10.15');

%% Create Workflow Object
hw = dlhdl.Workflow('network', snet, 'Bitstream', 'zcu102_single', 'Target', hTarget);

%% Compile the Lenet series Network
dn = hw.compile;

%% Program Bitstream onto FPGA and Download Network Weights
hw.deploy;

%% Run prediction for one frame
outputs = hw.predict(inputImg);
```
Defect Detection Example

Application logic

Pre-processing: Extract regions and resize

Inference: Predict using trained network

Post-processing: Annotate and label

FPGA
Run Deep Learning on FPGA from MATLAB

**Defect Detection**

**Prerequisites**
- Xilinx ZCU102 SoC development kit
- Deep Learning HDL Toolbox™ Support Package for Xilinx FPGA and SoC
- Deep Learning Toolbox™
- Deep Learning HDL Toolbox™

**Create Folder and Copy Relevant Files**

```matlab
unzip(‘dnfpga_defectdetection.zip’);
[novDir, origDir] = cloneSetupDir(‘dnfpga_defectdetection’);
cd(novDir);
```

**Command Window**

```
>> ...
```
Profile FPGA Prototype and Iterate in MATLAB

% Load the modified and trained network
net2 = load('trainedBlemDetNet.mat');
snet_blendnet = net2.convnet;
% Use the new network in the workflow object
hw = dlhdl.Workflow('Network',snet_blendnet,'Bitstream','zcua102_single','Target
hw.compile
hw.deploy

scores = zeros(2,4);
for i = 1:num
    [scores(:,i), speed] = hw.predict(single(imgPacked2(:,i,:)),'Profile','on');
end

Deep Learning Processor Profiler Performance Results

<table>
<thead>
<tr>
<th>Network</th>
<th>LastlayerLatency(cycles)</th>
<th>LastlayerLatency(seconds)</th>
<th>FramesNum</th>
<th>Total Latency</th>
<th>Frames/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv_module</td>
<td>12213262</td>
<td>0.05551</td>
<td>1</td>
<td>12213262</td>
<td>18.0</td>
</tr>
<tr>
<td>conv1</td>
<td>412728</td>
<td>0.01196</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>norm1</td>
<td>173252</td>
<td>0.00079</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pool1</td>
<td>56036</td>
<td>0.00027</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv2</td>
<td>650082</td>
<td>0.00058</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>norm2</td>
<td>128169</td>
<td>0.00055</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pool2</td>
<td>33269</td>
<td>0.00024</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3</td>
<td>700456</td>
<td>0.00035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv4</td>
<td>200059</td>
<td>0.00173</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv5</td>
<td>408977</td>
<td>0.00186</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pool3</td>
<td>20859</td>
<td>0.00069</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fc_module</td>
<td>921217</td>
<td>0.04055</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fc6</td>
<td>1759000</td>
<td>0.00800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fc7</td>
<td>7030644</td>
<td>0.03196</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fc8</td>
<td>130772</td>
<td>0.00859</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The clock frequency of the DL processor is: 220 MHz.
Design Exploration and Customization
Collaborate to Quantize Network

Systems Engineer

Latency
Cost
Power

Deep Learning Practitioner

Hardware/Software Engineers

Parameters (Bytes)

<table>
<thead>
<tr>
<th>input</th>
<th>conv1</th>
<th>conv2</th>
<th>conv3</th>
<th>conv4</th>
<th>conv5</th>
<th>fc6</th>
<th>fc7</th>
<th>fc8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>35K</td>
<td>0.4M</td>
<td>0.9M</td>
<td>1.3M</td>
<td>0.5M</td>
<td>37M</td>
<td>16M</td>
<td>4M</td>
<td>58 M</td>
</tr>
</tbody>
</table>

Off-chip RAM

MathWorks™

MATLAB EXPO
Int8 Quantization

Create Workflow Object for trainedBlemDetNet Network

```
webapp('trainedBlemDetNet.mat',url);
end
net2 = load('trainedBlemDetNet.mat');
netUnet = net2.convnet
analyzeNetwork(netUnet_blednet);
```

Compile trainedBlemDetNet Series Network

```
hw = dlhdl.Workflow('Network',netUnet_blednet,'Bitstream','ccu182_single','Target
```

Program Bitstream onto FPGA and Download Network Weights

```
hn.deploy
```

Run Prediction for One Image

```
filename=stdin,'/testImages/ok1.png');
img=imread(filename);
predictDefect(hn, img);
```

MATLAB EXPO

MathWorks
Converge on an FPGA-Optimized Deep Learning Network

% 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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>140 MB</td>
<td>18 fps</td>
</tr>
<tr>
<td>84 MB</td>
<td>45 fps</td>
</tr>
<tr>
<td>68 MB</td>
<td>139 fps</td>
</tr>
</tbody>
</table>
Generate Custom Deep Learning Processor HDL and IP Core

% Create a custom processor object
hPC = dlhdl.ProcessorConfig;

% Customize processor characteristics
hPC.setModuleProperty('conv', 'KernelDataType', 'int8');
hPC.setModuleProperty('conv', 'ConvThreadNumber', 64);
hPC.setModuleProperty('fc', 'KernelDataType', 'int8');
hPC.setModuleProperty('fc', 'FCThreadNumber', 16);
hPC.TargetFrequency = 300;

% Create workflow object for this config, estimate performance
hw = dlhdl.Workflow('Network',quantizer,'ProcessorConfig',hPC);
hw.estimate('Performance');

% Generate HDL and IP core using HDL Coder
dlhdl.buildProcessor(hPC);

- Configure processor settings
  - Parallel threads, frequency, memory sizes
  - Quantized or single precision floating point
  - Target frequency
- Target any hardware
  - Synthesizable RTL with AXI mappings
  - Automatic Xilinx or Intel implementation
Collaborate to Converge on Deep Learning FPGA Implementation

Application logic

AI Modeling
System Design
Deployment

CPU
GPU
FPGA

Deep Learning HDL Toolbox
- Prototype from MATLAB
- Tune for system requirements
- Configure and generate RTL
Learn More

- Deep Learning Solutions in MATLAB

- Deep Learning HDL Toolbox
  https://www.mathworks.com/products/deep-learning-hdl.html

- Onramp: Deep Learning in MATLAB
  https://www.mathworks.com/learn/tutorials/deep-learning-onramp.html

- MathWorks FPGA Solutions Page