Deep Learning HDL Toolbox™ Release Notes

MATLAB®

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
Phone:	508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Deep Learning HDL Toolbox™ Release Notes

Contents

R2021b

Trimmed deep learning processor configuration **1-2**

Generic deep learning processor generation **1-2**

Custom reference design functionality for custom boards for deep learning processor IP core integration **1-2**

Deep learning processor streaming handshake modes **1-2**

Updates to estimatePerformance **1-3**

Updates to estimateResource **1-3**

Enhancements for quantization of directed acyclic graph (DAG) networks **1-3**

Network prototyping and validation without hardware **1-4**

Updated supported layers **1-4**

Functionality being removed or changed **1-4**
KernelDataType property ofdlhdl.ProcessorConfig object has been removed **1-4**
LUT property ofestimateResources function has been removed **1-4**

R2021a

Custom directed acyclic graph (DAG) network support **2-2**

Performance estimation and profiling **2-2**

Resource estimation **2-2**

Updated supported layers **2-2**

MATLAB Emulation for validate method of dlquantizer object **2-3**

dlhdl.Workflow name-value argument pair update **2-3**

Updates to supported software **2-3**

Functionality being removed or changed **2-3**
estimate function for dlhdl.Workflow object has been removed **2-3**
'ProcessorConfig' option in dlhdl.Workflow has been removed **2-3**

R2020b

Introducing Deep Learning HDL Toolbox: Prototype and implement deep learning networks on FPGAs and SoCs **3-2**

Prototype on FPGAs **3-2**

Custom series network support **3-2**

Portable Verilog and VHDL code **3-2**

Tune user-configurable parameters **3-2**

Custom board support **3-2**

Performance estimation and profiling **3-2**

Hardware Support **3-2**

Support Package for Intel FPGA and SoCs **3-3**

Support Package for Xilinx FPGA and SoCs **3-3**

R2021b

Version: 1.2

New Features

Bug Fixes

Compatibility Considerations

Trimmed deep learning processor configuration

Generate a resource-optimized deep learning processor IP core and bitstream that suit your custom convolution module layers only or fully connected module layers only networks. Generate the optimized deep learning processor IP core and bitstream by removing the convolution, fully connected, or adder modules from the deep learning processor configuration. To remove the modules:

- Set the `ModuleGeneration` property to `off`.
- Use the `optimizeConfigurationForNetwork` function.

To further optimize your processor configuration:

- Remove the Local Response Normalization (LRN) block from the processor configuration by setting the `LRNBlockGeneration` property to `off`.
- Remove the Softmax block from the processor configuration by setting the `SoftmaxBlockGeneration` property to `off`. When you set this property to `off`, the Softmax layer is still implemented in software.

See “Generate Custom Bitstream to Meet Custom Deep Learning Network Requirements”.

Generic deep learning processor generation

Generate a custom generic deep learning processor IP core by specifying `Generic Deep Learning Processor` for the `TargetPlatform` property of the `dlhdl.ProcessorConfig` object. Integrate the generated IP core with your larger FPGA design. You can:

- Specify a name for your project folder by using the `ProjectFolder` name-value argument.
- Name your deep learning processor IP core by using the `ProcessorName` name-value argument.
- Specify HDL code generation options, such as target language for the generated HDL code, by using the `HDLCoderConfig` name-value argument.

Custom reference design functionality for custom boards for deep learning processor IP core integration

Use custom reference design functionality for custom boards and designs for deep learning processor IP core integration. You can:

- Register a custom board to target for deep learning.
- Register a custom reference design to integrate the deep learning processor IP core.
- Specify your board and reference design by using the `TargetPlatform` and `ReferenceDesign` properties of the `dlhdl.ProcessorConfig` object.

See `registerDeepLearningMemoryAddressSpace`, `registerDeepLearningTargetInterface`, and `validateReferenceDesignForDeepLearning`.

Deep learning processor streaming handshake modes

In R2021b, the generated deep learning processor IP core supports streaming handshaking modes. You can send multiple data frames to and receive multiple data frames from the deep learning processor IP core by using buffer mode or streaming mode. See “Interface with the Deep Learning Processor IP Core”.

Updates to estimatePerformance

Prior to R2021b, you could estimate performance of a network for only these bitstreams:

- `zcu102_single`
- `zcu102_int8`
- `zc706_single`
- `zc706_int8`
- `arria10soc_single`
- `arria10soc_int8`

In R2021b, you can estimate performance for your custom bitstream by using the `estimatePerformance` function. Create a processor configuration by using `dlhdl.ProcessorConfig`. Estimate performance by using the created processor configuration.

Estimate the performance of your network for multiple frames and for a bitstream by using the `FrameCount` name-value argument of the `estimatePerformance` function.

For more information, see `estimatePerformance`.

Updates to estimateResource

In R2021b, you can use the `estimateResource` function to:

- Estimate the resource usage for any Xilinx® and Intel® devices that have been registered by using a device registration function.
- Display the resource estimates as a percentage of the total resources for Xilinx devices.
- Retrieve the lookup table (LUT) utilization estimates for these devices:
  - Xilinx Zynq®-7000 ZC706
  - Intel Arria® 10 SoC
  - Xilinx Zynq UltraScale+™ MPSoC ZCU102

Enhancements for quantization of directed acyclic graph (DAG) networks

In R2021b, Deep Learning HDL Toolbox supports quantization of:

- Addition layers that have more than two inputs.
- Addition layer followed by ReLU, Leaky ReLU, and Clipped ReLU layers.

Deep Learning HDL Toolbox supports channel-wise quantization for depth-wise separable convolution layers for improved accuracy of quantized network predictions.

Deep Learning HDL Toolbox supports quantization of these DAG networks:

- GoogLeNet
- MobileNet
- SqueezeNet

Network prototyping and validation without hardware

In R2021b, you can prototype, verify prediction accuracy, and retrieve intermediate layer-level results for your custom deep learning networks without the need for hardware. Create a simulation object by using the `dlhdl.Simulation` class. Verify network prediction accuracy by using the `prediction` function of the simulation object. Retrieve intermediate layer-level performance by using the `activations` function of the simulation object. The `dlhdl.Simulation` object accepts `single` data type networks, `int8` data type quantized networks, and `dlhdl.ProcessorConfig` objects as inputs. See `dlhdl.Simulator`.

Updated supported layers

Deep Learning HDL Toolbox now provides support for these layers:

- Softmax layer hardware implementation

For `int8` data type quantization, Deep Learning HDL Toolbox now provides support for these layers:

- Depth concatenation layer
- Softmax layer
- Addition layer followed by ReLU, leaky ReLU,or clipped ReLU layers

Functionality being removed or changed

KernelDataType property ofdlhdl.ProcessorConfig object has been removed
Errors

This property has been removed. Use the `ProcessorDataType` property of the `dlhdl.ProcessorConfig` object instead.

LUT property ofestimateResources function has been removed
Errors

This property has been removed as the `estimateResources` method reports LUT utilization by default.

R2021a

Version: 1.1

New Features

Compatibility Considerations

Custom directed acyclic graph (DAG) network support

Compile and deploy your custom DAG networks. Retrieve predictions from the deployed network by using MATLAB®. For a list of supported networks, see Supported Networks, Layers, Boards, and Tools. The deep learning compiler analyzes the DAG network graph and generates the instructions, address mapping, and schedule to run the DAG network on the new deep learning processor. Deploy larger DAG networks onto FPGA boards with smaller resources by quantizing your DAG networks to use `int8` data types. See Quantization of Deep Neural Networks.

Performance estimation and profiling

Estimate performance by using the `estimatePerformance` function on the `dlhdl.ProcessorConfig` object before building your custom deep learning processor. For more information, see estimatePerformance. Retrieve the processor configuration of the shipping (reference) bitstream, by using the `dlhdl.ProcessorConfig` object. See dlhdl.ProcessorConfig. Perform design space exploration to find the deep learning processor configuration that fits your performance requirements by comparing the performance of your custom deep learning processor configuration to the performance of the shipping (reference) bitstream processor configuration.

You cannot estimate performance by using the `estimate` method for the `dlhdl.Workflow` object. For more information, see “Functionality being removed or changed” on page 2-3.

Resource estimation

Estimate resource utilization by using the `estimateResources` function on the `dlhdl.ProcessorConfig` object before building your custom deep learning processor. For more information, see estimateResources. Retrieve resource utilization of shipping (reference) bitstreams by using the `getBuildInfo` function on the `dlhdl.Workflow` object. See getBuildInfo. Perform design space exploration to find the deep learning processor configuration that fits your FPGA resource budget by comparing the resource utilization of your custom deep learning processor configuration to the resource utilization of the shipping (reference) bitstream.

Updated supported layers

Deep Learning HDL Toolbox now provides support for these layers:

- Addition layer
- Depth-wise separable convolution layer
- Depth concatenation layer

For `int8` data type quantization, Deep Learning HDL Toolbox now provides support for these layers:

- Average pooling layer
- Global average pooling layer
- Addition layer
- Clipped ReLU layer
- Leaky ReLU layer
- Depth-wise separable convolution layer

See Supported Networks, Layers, Boards, and Tools.

MATLAB Emulation for validate method of dlquantizer object

Validate the performance of your quantized network by comparing the prediction accuracy of your quantized network to that of your nonquantized network, without the need for hardware by using MATLAB emulation. See validate.

dlhdl.Workflow name-value argument pair update

For the list of name-value pair arguments that have been removed from `dlhdl.Workflow`, see “Functionality being removed or changed” on page 2-3.

Updates to supported software

Deep Learning HDL Toolbox has been tested with:

- Xilinx Vivado® Design Suite 2020.1
- Intel Quartus® Pro 18.1

Functionality being removed or changed

estimate function for dlhdl.Workflow object has been removed
Errors

This function has been removed.

'ProcessorConfig' option in dlhdl.Workflow has been removed
Errors

The 'ProcessorConfig' name-value pair for `dlhdl.Workflow` has been removed.

R2020b

Version: 1.0

New Features

Introducing Deep Learning HDL Toolbox: Prototype and implement deep learning networks on FPGAs and SoCs

With Deep Learning HDL Toolbox, you can prototype and implement deep learning networks on FPGAs and SoCs. Deploy and run deep learning networks on supported Xilinx and Intel FPGA and SoC devices. Improve deep learning network design, performance, and resource utilization by using profiling and estimating tools to explore tradeoffs and customize the network. Using HDL Coder™, you can generate HDL and an IP core to target FPGAs or SoCs.

Prototype on FPGAs

Use MATLAB and fixed bitstreams to compile, deploy, and run inference for pretrained series networks on target Intel and Xilinx FPGA and SoC boards. For more information, see Prototype Deep Learning Networks on FPGA.

Custom series network support

Compile and deploy your custom series networks using the same fixed-bitstreams as the pre-trained networks. For more information, see Prototype Deep Learning Networks on FPGA and SoCs Workflow.

Portable Verilog and VHDL code

Generate portable Verilog® and VHDL® code from your series deep learning network.

Tune user-configurable parameters

Customize your deep learning network implementation by tuning user-configurable parameters such as Thread Number, Input, and Output Memory Size. For more information, see Custom Processor Configuration Workflow.

Custom board support

Integrate the code generated from your customized design into your reference design for deploying to your custom board. For more information, see Generate Custom Processor IP.

Performance estimation and profiling

Gather layer-level latency and throughput estimates for your series networks. For more information, see estimate.

Hardware Support

Prototype and deploy deep learning networks to Intel and Xilinx FPGA boards. Use Ethernet based LIBIIO to rapidly deploy your series deep learning networks to your target Intel and Xilinx FPGA and SoC boards. For more information, see LIBIIO/Ethernet Connection Based Deployment.

Support Package for Intel FPGA and SoCs

You can use the Deep Learning HDL Toolbox Support Package for Intel FPGA and SoC Devices to communicate with, deploy series networks, and retrieve inference results from target Intel FPGA and SoC platforms. To download the support package, use the Add-on Explorer. For more information, see Deep Learning HDL Toolbox Support Package for Intel FPGA and SoC Devices.

Support Package for Xilinx FPGA and SoCs

You can use the Deep Learning HDL Toolbox Support Package for Xilinx FPGA and SoC Devices to communicate with, deploy series networks, and retrieve inference results from target Xilinx FPGA and SoC platforms. To download the support package, use the Add-on Explorer. For more information, see Deep Learning HDL Toolbox Support Package for Xilinx FPGA and SoC Devices.