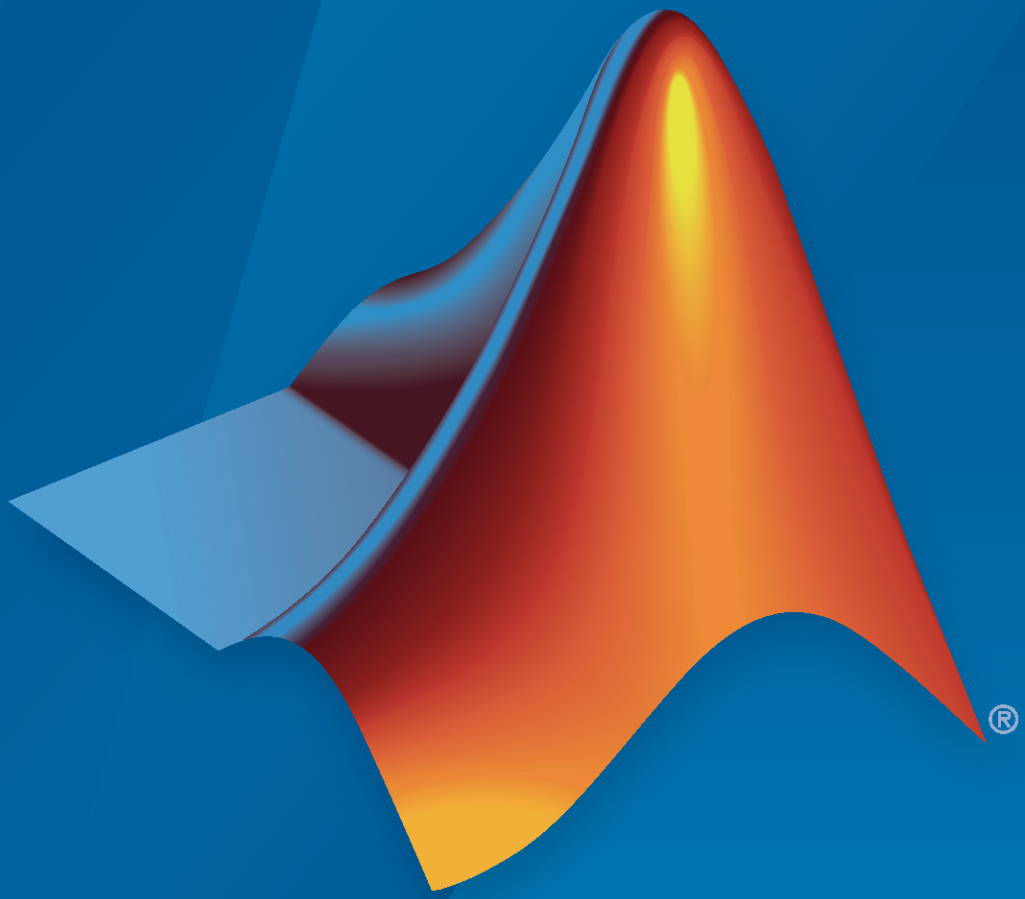
Deep Learning HDL Toolbox™

Getting Started Guide



MATLAB®

R2020b

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™ Getting Started Guide

© COPYRIGHT 2020 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Revision History

September 2020	Online only	New for Version 1.0 (Release 2020b)

Contents

1 Install and Set up Prerequisite Products and Third-Party Tools

2 Tutorials

Install and Set up Prerequisite Products and Third-Party Tools

Deep Learning HDL Toolbox Product Description

Prototype and deploy deep learning networks on FPGAs and SoCs

Deep Learning HDL Toolbox™ provides functions and tools to prototype and implement deep learning networks on FPGAs and SoCs. It provides pre-built bitstreams for running a variety of deep learning networks on supported Xilinx® and Intel® FPGA and SoC devices. Profiling and estimation tools let you customize a deep learning network by exploring design, performance, and resource utilization tradeoffs.

Deep Learning HDL Toolbox enables you to customize the hardware implementation of your deep learning network and generate portable, synthesizable Verilog® and VHDL® code for deployment on any FPGA (with HDL Coder™ and Simulink®).

Use Deep Learning on FPGA Bitstreams

The Deep Learning HDL Toolbox hardware support packages provide bitstreams that you can use to deploy various deep learning networks on the target platform.

This table illustrates the mapping between the target boards, data types, and bitstream names.

Target Board	Data Type	Bitstream Name
Xilinx Zynq® -7000 ZC706	`single`	`'zc706_single'`
Xilinx Zynq-7000 ZC706	`int8`	`'zc706_int8'`
Xilinx Zynq UltraScale™ ZCU102	`single`	`'zcu102_single'`
Xilinx Zynq UltraScale ZCU102	`int8`	`'zcu102_int8'`
Intel Arria® 10 SoC development kit	`single`	`'arria10soc_single'`
Intel Arria 10 SoC development kit	`int8`	`'arria10soc_int8'`

For an example that illustrates how you can use these bitstreams names and deploy your network when running the workflow, see “Prototype Deep Learning Networks on FPGA and SoCs Workflow”.

Check Host Computer Connection to FPGA Boards

After you install MATLAB® and the support packages, to connect the host computer to one of the supported FPGA boards:

1. Connect the host computer to the FPGA board by using a USB JTAG cable.
   a. To learn how to set up the JTAG connection, see “JTAG Connection” on page 1-9.
   b. To learn how to set up each board, see “Configure Board-Specific Setup Information” on page 1-6.

   Note To download the weights faster, use USB 3.0 for the USB JTAG cable. Weight downloading can become slower if you use USB 2.0 for the cable.
2. Check whether the JTAG driver is installed on the host computer. Install the driver if it is not already installed.
3. Connect the host computer to the FPGA board by using an Ethernet cable. For more information, see “Guided SD Card Setup” (Deep Learning HDL Toolbox Support Package for Intel FPGA and SoC Devices) or “Guided SD Card Setup” (Deep Learning HDL Toolbox Support Package for Xilinx FPGA and SoC Devices).
4. To troubleshoot Ethernet connection issues for Intel boards, see “Command Line Session with Intel SoC Device” (Deep Learning HDL Toolbox Support Package for Intel FPGA and SoC Devices). To troubleshoot Ethernet connection issues for Xilinx boards, see “Troubleshoot Xilinx Zynq Platform and Development Computer Connection” (Deep Learning HDL Toolbox Support Package for Xilinx FPGA and SoC Devices).
5. Install and set up the path to the relevant third-party synthesis tool. To learn more, see the "Install Synthesis Tools and Set up Tool Path" section below.

Install Synthesis Tools and Set up Tool Path

Before you can deploy your neural network to one of the supported FPGA boards, you must install the required third-party synthesis tools. The workflow supports these synthesis tools:

- Intel Quartus® Prime Standard Edition 18.1 and later.
- Xilinx Vivado® Design Suite 2019.1 and later.

To run the workflow and deploy the network to the FPGA board, set up the tool path to your installed Xilinx Vivado or Intel Quartus Prime standard edition executable file by running the `hdlsetuptoolpath` function.

If your synthesis tool is Intel Quartus Prime, enter this command:

```
hdlsetuptoolpath('ToolName','Altera Quartus II','ToolPath',...
 'C:\intel\18.1\quartus\bin\quartus.exe');
```

If your synthesis tool is Xilinx Vivado, enter this command:

```
hdlsetuptoolpath('ToolName','Xilinx Vivado','ToolPath',...
 'C:\Xilinx\Vivado\2019.1\bin\vivado.bat');
```

See also `hdlsetuptoolpath` (HDL Coder).

Verify FPGA Board Connection

To verify that the connection to the JTAG connection to the FPGA board has been setup and is ready, run the “Get Started with Deep Learning FPGA Deployment on Intel Arria 10 SoC” example. This example runs the entire workflow and displays the prediction results.

To verify that the Ethernet connection to the FPGA board has been setup and is ready to run, run the “Get Started with Deep Learning FPGA Deployment on Xilinx ZCU102 SoC” example. This example runs the entire workflow and displays the prediction results.

Configure Board-Specific Setup Information

Note For setting up the boards, you do not need an Ethernet cable or an SD card.

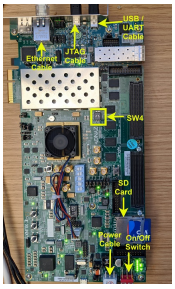
Xilinx Zynq-7000 ZC706 Evaluation Board

This figure shows how to set up the Xilinx Zynq-7000 ZC706 evaluation board. To set up the board:

1. Configure SW4 shown in the image below, and to use Digilent USB-TO-JTAG interface using the following configuration table:

Configuration Source	SW4 switch 1	SW4 switch 2
None	0	0
Cable Connector J3	1	0
Digilent USB-TO-JTAG Interface	0	1
JTAG (flying lead)Header J62	1	1

2. Plug the power cord and then connect the host computer to the FPGA board by using a JTAG cable as shown in the image below:



3. To use Ethernet, see “Create Target Object That Has an Ethernet Interface and Set IP Address”.
4. To learn more about the board configuration, see Xilinx ZC706 Evaluation Board User Guide.

After you have set up the connection to the board, to run the workflow:

1. Create a workflow object by using the bitstream name that is provided for the board as mentioned in “Use Deep Learning Bitstreams” (Deep Learning HDL Toolbox Support Package for Xilinx FPGA and SoC Devices) .
2. To learn more about the workflow, see “Prototype Deep Learning Networks on FPGA and SoCs Workflow”.

Intel Arria 10 SoC development kit

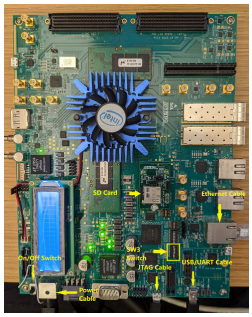
This figure shows how to set up the Intel Arria 10 SoC development kit. To set up the board:

1. Plug the power cord and then connect the host computer to the FPGA board by using a JTAG cable.
2. Specify the SW3 switch settings:

Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Off	On	On	On	On	Off	Off	Off

3. Connect two DDR4 plugin boards to the memory plugin slot.
4. To use Ethernet, see “Create Target Object That Has an Ethernet Interface and Set IP Address”.

This figure shows the configuration settings for the Intel Arria 10 SoC development kit.



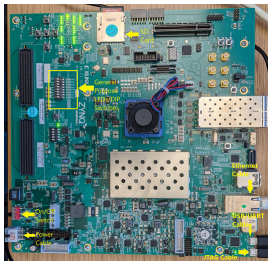
To learn more about the board configuration, see Arria 10 SoC Development Kit User Guide.

After you have set up the connection to the board, to run the workflow:

1. Create a workflow object by using the bitstream name that is provided for the board as mentioned in “Use Deep Learning Bitstreams” (Deep Learning HDL Toolbox Support Package for Intel FPGA and SoC Devices) .
2. To learn more about the workflow, see “Prototype Deep Learning Networks on FPGA and SoCs Workflow”.

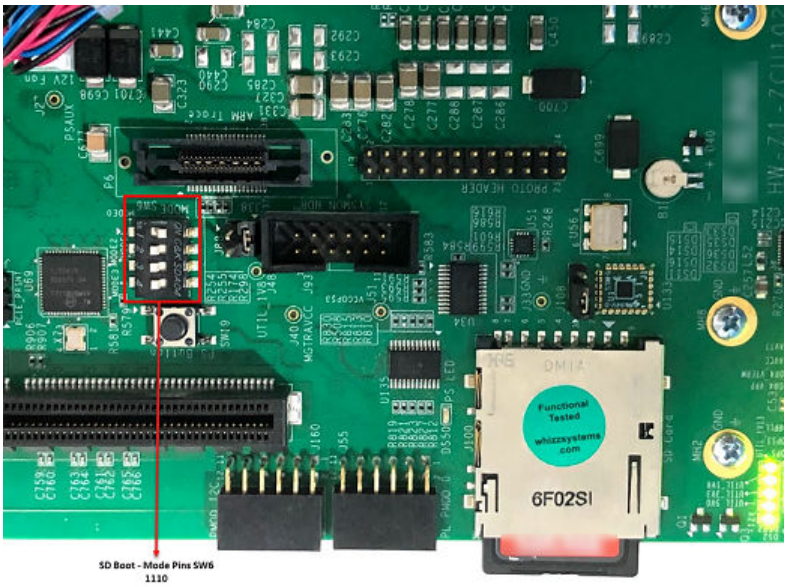
Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA Development Board

1. Set up the Xilinx Zynq UltraScale+ MPSoC ZCU102 evaluation kit as shown in the figure below:



To setup the board:

1. Plug in the power cord. If using JTAG connect the FPGA board to the host computer using a JTAG cable. If using Ethernet connect the FPGA board to the host computer using an Ethernet cable.
2. Configure `SW6` switch which is shown in the image below:



Use the configuration table below to configure the switch settings:

Boot Mode	Mode Pins [3:0]	SW6 Switch Position [3:0]
JTAG	0, 0, 0, 0	on, on, on, on
QSPI32	0, 0, 1, 0	on, on, off, on
SD	1, 1, 1, 0	off, off, off, on

The SW6 default position is QSPI32. For the SW6 DIP switch moving the switch towards the ON label is a 0.

3 To use Ethernet, see “Create Target Object That Has an Ethernet Interface and Set IP Address”.

4 To learn more about the ZCU102 hardware setup, please refer to Xilinx documentation

After you have set up the connection to the board, to run the workflow:

1 Create a workflow object by using the bitstream name that is provided for the board as mentioned in “Use Deep Learning Bitstreams” (Deep Learning HDL Toolbox Support Package for Xilinx FPGA and SoC Devices) .

2 To learn more about the workflow, see “Prototype Deep Learning Networks on FPGA and SoCs Workflow”.

JTAG Connection

Vendor	Required Hardware	Required Software
Intel	USB Blaster I or USB Blaster II download cable	• USB Blaster I or II driver • For Windows® operating systems: Quartus Prime executable directory must be on system path. • For Linux® operating systems: versions below Quartus II 13.1 are not supported. Quartus II 14.1 is not supported. Only 64-bit Quartus is supported. Quartus library directory must be on `LD_LIBRARY_PATH` *before* starting MATLAB. Prepend the Linux distribution library path before the Quartus library on `LD_LIBRARY_PATH`. For example, `/lib/x86_64-linux-gnu:$QUARTUS_PATH`.
Xilinx	Digilent® download cable. • If your board has an onboard Digilent USB-JTAG module, use a USB cable. • If your board has a standard Xilinx 14 pin JTAG connector, use with HS2 or HS3 cable from Digilent.	• For Windows operating systems: Xilinx Vivado executable directory must be on system path. • For Linux operating systems: Digilent Adept2
	FTDI USB-JTAG cable • Supported for boards with onboard FT4232H, FT232H, or FT2232H devices implementing USB-to JTAG	Supported for Windows operating systems. **Note** FTDI USB JTAG support is only available for MATLAB as AXI Master and for FPGA Data Capture.
Microsemi®	JTAG connection not supported	

Note When simulating your FPGA design through Digilent JTAG cable with Simulink or MATLAB, you cannot use any debugging software that requires access to the JTAG; for example, Vivado Logic Analyzer.

Tutorials

Try Deep Learning on FPGA with only Five Additional Lines of MATLAB Code

This example shows how to use Deep Learning HDL Toolbox to deploy a pretrained deep learning network to a target board and identify objects on a live webcam connected to the development computer by adding only five lines of MATLAB code to the “Try Deep Learning in 10 Lines of MATLAB Code” example.

1. To connect to the webcam and get a pretrained neural network, run these commands.

   ```
   camera = webcam; % Connect to the camera
   net = alexnet;   % Load the neural network
   ```

   If you need to install the webcam and Alexnet add-ons, a message appears with a link to help you download the free add-ons using Add-On Explorer. Alternatively, see Deep Learning Toolbox Model *for AlexNet Network* and MATLAB Support Package for USB Webcams for installation instructions.

   After you install Deep Learning Toolbox Model for AlexNet network you can use it to classify images. AlexNet is a pretrained convolutional neural network (CNN) that has been trained on more than a million images and can classify images into 1000 object categories (for example, keyboard, mouse, and so on).
2. Run the following three lines of code to set up the interface to the target board, create the workflow object, and deploy the network to the target board.

   ```
   hT = dlhdl.Target('Xilinx');
   hW = dlhdl.Workflow('Network',net,'Bitstream','zcu102_single','Target',hT);
   hW.deploy;
   ```
3. Run the following code to show and classify live images. Point the webcam at an object. The neural network reports what class of object it thinks the webcam is showing, classifying images until you press **Ctrl+C**. The code resizes the image for the network by using `imresize`.

   ```
   while true
       im = snapshot(camera);       % Take a picture
       image(im);                   % Show the picture
       im = imresize(im,[227 227]); % Resize the picture for alexnet
       [prediction, speed] = hW.predict(single(im),'Profile','on');
       [val, idx] = max(prediction);
       label = net.Layers(end).ClassNames{idx}; % classify the image
       title(char(label));          % Show the class label
       drawnow
      end
   ```

   In this example, the network correctly classifies a coffee mug. Experiment with objects in your surroundings to see how accurate the network is.

For next steps, see “Deep Learning on FPGA Solution and Workflows”.

See Also

`alexnet` | `dlhdl.Target` | `dlhdl.Workflow`

More About

- “Prototype Deep Learning Networks on FPGA and SoCs Workflow”
- “Supported Networks, Layers and Boards”