# AI Quantizer and AI Compiler – TensorFlow2 and PyTorch

Vitis AI Development Environment 2.0

## Abstract

This lab illustrates the Vitis™ AI quantizer and Vitis AI compiler flow for the TensorFlow2 and PyTorch frameworks. You will explore the required options for these tools and then run the tools.

This lab should take approximately 45-60 minutes.

CloudShare users: You may face some delay in running this lab depending on your region. Because this lab will be supported only from the Amsterdam region, other regions are not supported.

## CloudShare Users Only

You are provided three attempts to access a lab, and the time allotted to complete each lab is 2X the time expected to complete the lab. Once the timer starts, you cannot pause the timer. Also, each lab attempt will reset the previous attempt—that is, your work from a previous attempt is not saved.

## Objectives

After completing this lab, you will be able to:

* Describe the AI quantizer flow and AI compiler flow for the TensorFlow2 and PyTorch frameworks
* Quantize a pre-trained model
* Compile the quantized model for the DPUCZDX8G architecture
* Verify the files generated after quantization and compilation for the TensorFlow 2 and PyTorch model networks

## Introduction

There are two stages for developing deep learning applications: training and inference. The training stage is used to design a neural network for a specific task (such as image classification) using a huge amount of training data. The inference stage involves the deployment of a previously designed neural network to handle new input data not seen during the training stage.

The Vitis AI toolchain provides an innovative workflow to deploy deep learning inference applications on the DPU with the following four steps:

* Quantize the neural network model
* Compile the neural network model
* Program with the Vitis AI programming interface
* Run and evaluate the deployed DPU application

In this lab, you will be focusing on quantization and compilation.

Model Quantization

The Vitis AI quantizer supports the TensorFlow (both 1.x and 2.x), PyTorch, and Caffe frameworks. vai\_q\_tensorflow, vai\_q\_pytorch, and vai\_q\_caffe are the Vitis AI quantizer, where q stands for quantizer and tensorflow, pytorch, and caffe are the framework names.

The Vitis AI quantizer for TensorFlow 1.x and TensorFlow 2.x are implemented in different ways and are released separately. For TensorFlow 1.x, the Vitis AI quantizer is based on Tensorflow 1.15. The vai\_q\_pytorch quantizer supports PyTorch from 1.2-1.9. After adding quantization features, the Vitis AI quantizer rebuilds and redistributes a standalone package. For TensorFlow 2.x, the Vitis AI quantizer is a Python package with several quantization APIs. You can import this package, and the Vitis AI quantizer works like a plugin for TensorFlow.

Post training quantization (PTQ) requires only a small set of unlabeled images to analyze the distribution of activations. The running time of quantize calibration varies from a few seconds to several minutes, depending on the size of the neural network.

Generally, there is some drop in accuracy after quantization. However, for some networks (such as Mobilenet), the accuracy loss might be large. In this situation, quantization aware training (QAT) can be used to further improve the accuracy of the quantized models.

QAT requires the original training dataset. Several epochs of finetuning are needed and the finetune time varies from several minutes to several hours. It is recommended to use small learning rates when performing QAT.

For PTQ, the cross-layer equalization algorithm is implemented. Cross-layer equalization can improve calibration performance, especially for networks including depth-wise convolution.

Note: For the Vitis AI development environment 1.4 onwards, the term "quantize calibration" is replaced with "post training quantization", and "quantize finetuning" is replaced with "quantization aware training."

| Frameworks | Versions | Post Training Quantization (PTQ) | Quantize Aware Training (QAT) | Fast Finetuning (Advanced Calibration) |
| --- | --- | --- | --- | --- |
| TensorFlow 1.x | Based on 1.15 | Yes | Yes | No |
| TensorFlow 2.x | Supports 2.3 | Yes | Yes | Yes |
| PyTorch | Supports 1.2 – 1.9 | Yes | Yes | Yes |
| Caffe | – | Yes | Yes | No |

Compiling

The Vitis AI compiler is the unified interface to a compiler family targeting the optimization of neural network computations to a family of DPUs. Each compiler maps a network model into a highly optimized DPU instruction sequence.

The XIR-based compiler takes the quantized TensorFlow or PyTorch model as the input. The TensorFlow model is transformed to XIR format by xcompiler. However for PyTorch, the quantizer will output an XIR format xmodel.

First, it transforms the input models into the XIR format as the foundation of the following processes. Most of the variations among different frameworks are eliminated and transferred to a unified representation in XIR.

The compiler then applies various optimizations on the graph and breaks up the graph into several subgraphs on the basis of whether the operation can be executed on the DPU. More architecture-aware optimizations are applied for each subgraph, as required. For the DPU subgraph, the compiler generates the instruction stream and attaches to it.

Finally, the optimized graph with the necessary information and instructions for VART is serialized into a compiled xmodel file.

| DPU Name | Hardware Platform |
| --- | --- |
| DPUCZDX8G | Zynq® UltraScale+™ MPSoC |
| DPUCAHX8H | Alveo™ U50LV, U55C Data Center accelerator cards |
| DPUCADF8H | Alveo U200, U250 Data Center accelerator cards |
| DPUCVDX8G | Versal® ACAP VCK190 (Versal AI Core Series) evaluation kit |
| DPUCVDX8H | Versal ACAP VCK5000 evaluation kit |

You can find the arch.json files for those platforms in /opt/vitis\_ai/compiler/arch.

In this lab, you will be focusing on the AI quantizer and AI compiler for TensorFlow2 and PyTorch.

Understanding the Lab Environment

Customizable environment variables enable you to tailor your environment for specific machine configurations. The only environment variable (shown below) used in the customer training environment (CustEd\_VM) points to the training directory where all the lab files are located.

This environment variable can be customized according to your specific location and can be set for Linux systems in the /etc/profile file.

The following is the environment variable used in the customer training VM:

| Environment Variable Name | Description |
| --- | --- |
| $TRAINING\_PATH | Points to the space allocated for students to work through their labs. This directory includes prebuilt images and starting points for the labs and demos. In the customer training VM, $TRAINING\_PATH sets to the /home/xilinx/training directory. |

Note: Environment variables are not supported from the Vitis IDE GUI. When using this tool, you must manually replace $TRAINING\_PATH with the value of the variable, which in the customer training virtual machine, is /home/xilinx/training.

## General Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step 1:  Reviewing the TF2 Arguments & Scripts |  | Step 2:  Running  the TF2 Quantizer & Compiler |  | Step 3:  Reviewing the PyTorch Arguments & Scripts |  | Step 4:  Running  the PyTorch  Quantizer & Compiler |

Reviewing the TensorFlow2 AI Quantizer Arguments and Training Scripts Step

The overall model quantization flow is outlined in the following figure.

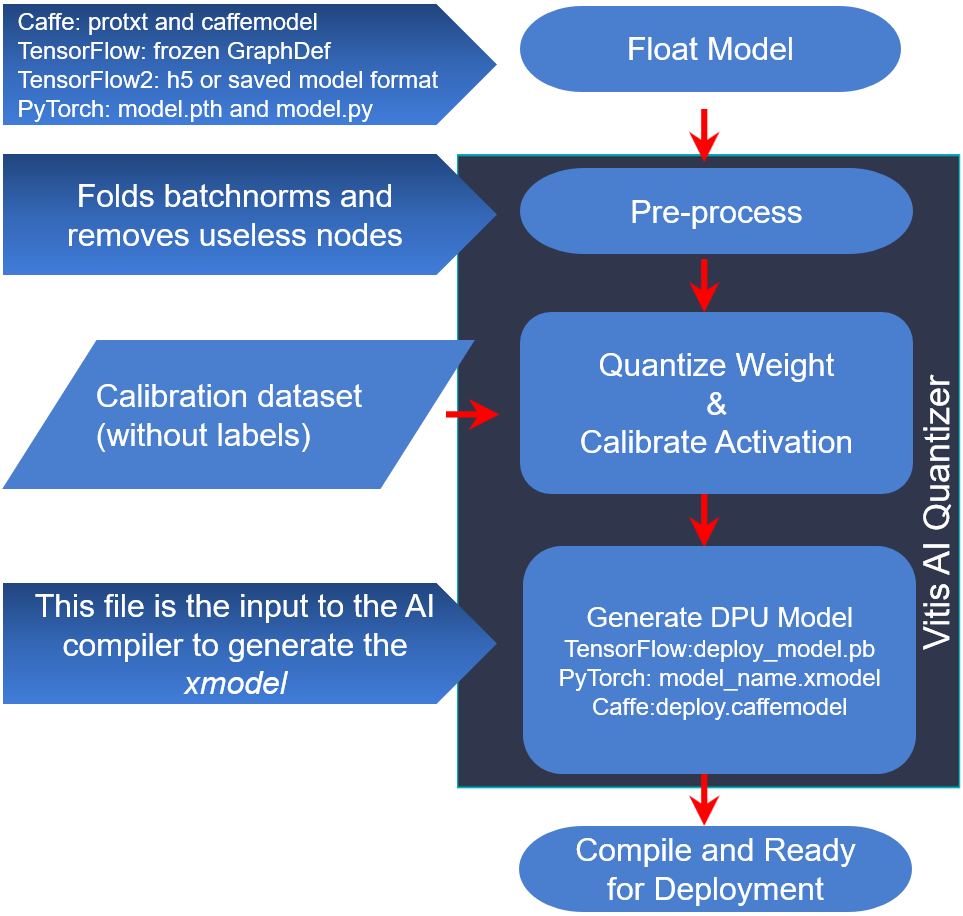


Figure 2‑: Vitis AI Quantizer Flow

The Vitis AI quantizer takes a floating-point model as input, performs pre-processing (folding batch norms and removing nodes not required for inference), and then quantizes the weights/biases and activations to the given bit width.

To capture activation statistics and improve the accuracy of quantized models, the Vitis AI quantizer must run several iterations of inference to calibrate the activations. A calibration image dataset input is therefore required. Generally, the quantizer works well with 100–1000 calibration images. This is because there is no need for back propagation; the un-labeled dataset is sufficient.

After calibration, the quantized model is transformed into a DPU-deployable model (named deploy\_model.pb for vai\_q\_tensorflow, model\_name.h5 for vai\_q\_tensorflow2, model\_name.xmodel for vai\_q\_pytorch, or deploy.prototxt/deploy.caffemodel for vai\_q\_caffe), which follows the data format of a DPU.

This model can then be compiled by the Vitis AI compiler and deployed to the DPU. The quantized model cannot be taken in by the standard version of the Caffe, TensorFlow, or PyTorch framework.

1-1. Copy the lab files to the docker location.

1-1-1. Enter the following command to copy the files to the specified directory:

[host]$ cp -rf $TRAINING\_PATH/vai\_q\_c\_tf2\_pt /home/xilinx/Vitis-AI

You will be using the following scripts and text file for running the TF2 quantizer and compiler.

* 1\_tf2\_quantize.sh
* This script calls the vitis-ai quantizer for TensorFlow2 with all the required arguments.
* 4\_tf2\_compile\_for\_mpsoc.sh
* This script calls the vitis-ai compiler for TensorFlow2 with all the required arguments.
* val.txt
* For quantize, the evaluation image list file.

1-2. Review the 1\_tf2\_quantize.sh script.

1-2-1. Press <Ctrl + Alt + T> to open a new terminal window.

1-2-2. Enter the following command to review the quantizer arguments in the quantizer script:

[host]$ gedit /home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/lab/  
1\_tf2\_quantize.sh

The detailed commands used in the training script are as shown below.

python ${TF2\_NETWORK\_PATH}/code/com/train\_eval\_h5.py --model \  
${TF2\_NETWORK\_PATH}/float/resnet\_50.h5 \

--quantize=true \

--quantize\_output\_dir=${TF2\_NETWORK\_PATH}/vai\_q\_output \

--eval\_only=true \

--eval\_images=true \

--eval\_image\_path=images/ \

--eval\_image\_list=val.txt \

--label\_offset=1 \

--gpus=0

Note: Observe that the python script train\_eval\_h5.py is used to quantize the trained model. Usually, this script comes from the trainer who has created it for training the model.

The following are the arguments to this quantizer script:

| Quantizer Argument | Description |
| --- | --- |
| model | TensorFlow2 floating-point network model h5 file. |
| quantize | Set it to true. Indicates that the quantization is to be done. |
| quantize\_output\_ dir | Output directory location to store the generated output. |
| eval\_only | Set it to true for evaluation only. |
| eval\_images | Set it to true for evaluation images. |
| eval\_images\_path | Location to the evaluation images directory. |
| eval\_images\_list | Images list. |
| label\_offset | Set to one. |
| gpus | Indicates to use GPU or CPU. |

1-2-3. After you complete the review, close the file.

In this lab, you will be using the tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0 pre-trained model from the Vitis AI Model Zoo. The pre-trained model has been already downloaded and is located in the /home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt directory.

You will find the pre-trained floating model under the directory tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0 > float.

| Application | Image Classification |
| --- | --- |
| Model | resnet50 |
| Name | tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0 |
| Framework | TensorFlow2 |
| Backbone | resnet50 |
| Input Size | 224\*224 |
| OPS per image | 7.76G |
| Training Set | ImageNet training |
| Val Set | ImageNet validation |
| Vitis-AI Version | 2.0 |

With regards to the naming of tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0:

* tf2 specifies the TensorFlow2 framework.
* imagenet specifies the dataset.
* 224\_224 specifies the height of the input data x the width of the input data.
* 7.76G specifies the computation of the model; how many giga-operations per second (GOPs) per image.
* 2.0 specifies the Vitis AI development environment version.

In order for calibration and evaluation in the quantization phase to be conducted, images and labels from ImageNet have been downloaded and placed under the vai\_q\_c\_tf2\_pt/images directory.

1-3. Review the train\_eval\_h5.py training script.

Usually, this script comes from the trainer who has created it for training the model. This script is specific to the model—in this case, it is for ResNet50. The user will need to import the Vitis AI quantizer.

1-3-1. Enter the following command to review the training script that has been updated with the Vitis AI quantizer:

[host]$ gedit /home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/lab/  
tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/code/com/train\_eval\_h5.py

* Observe the declarations of variables from line no. 34 to 88:
* flags.DEFINE\_string(<NAME>, <VALUE>, <DESCRIPTION>)
* flags.DEFINE\_bool(<NAME>, <True/False>, <DESCRIPTION>)
* flags.DEFINE\_integer(<NAME>, <INTEGER>, <DESCRIPTION>)
* flags.DEFINE\_float(<NAME>, <FLOAT>, <DESCRIPTION>)
* The function get\_input\_data() at line no. 95 reads the input data and returns the training data (train\_data) and evaluation data (eval\_data):
* def get\_input\_data(num\_epochs=1):
* The main() program starts at line no. 112:
* At near line no. 114, if save\_whole\_model is True, then the file just includes weights and the whole model is saved to an h5 file.
* At near line no. 119, if eval\_images is set to False, it will read the input data. eval\_images is set to True in the 1\_tf2\_quantize.sh script.
* At near line no. 127, the model variable is updated with the weights of the trained model, which is passed (resnet50.h5) in the 1\_tf2\_quantize.sh script.
* At near line no. 129, read the images and labels and store in the img\_paths and labels list.
* At near line no. 133, verify the quantize flag, which is set to True in the 1\_tf2\_quantize.sh script.
* From line no. 135 to 141, the user will need to include in their training script:
* Importing the Vitis AI Quantizer
* Calling the Vitis AI quantizer by passing the trained model and calibration dataset
* Saving the quantized model

1-3-2. After you complete the review, close the file.

1-3-3. Close the terminal.

Running the AI Quantizer and AI Compiler for TensorFlow2 Step

You will now run the Vitis AI quantizer for TensorFlow2 using the training script, verify the generated output, and then run the Vitis AI compiler (vai\_c\_tensorflow2) for TensorFlow2.

2-1. Load the CPU image from the Docker hub.

2-1-1. Press <Ctrl + Alt + T> to open a new terminal window.

2-1-2. Enter the following commands to load the CPU image from the Docker hub:

[host]$ cd Vitis-AI

[host]$ ./docker\_run.sh xilinx/vitis-ai-cpu:latest

If there is a newer version of the Vitis AI tools that has been released, entering the above command will download the latest version (that is, not VAI 2.0). In order to use the Vitis AI 2.0 environment, enter the command with the docker image tag. You can find the tag from https://hub.docker.com/r/xilinx/vitis-ai-cpu/tags?page=1&ordering=last\_updated.

For example: [host]$ ./docker\_run.sh xilinx/vitis-ai-cpu:<Docker\_tag>

To use 2.0: [host]$ ./docker\_run.sh xilinx/vitis-ai-cpu:2.0

Note: Keep clicking to accept the terms and agreements and then enter 'y'.

The terminal output should be similar to the following:

==========================================

\_\_ \_\_\_ \_ \_ \_\_\_\_\_

\ \ / (\_) | (\_) /\ |\_ \_|

\ \ / / \_| |\_ \_ \_\_\_ \_\_\_\_\_\_ / \ | |

\ \/ / | | \_\_| / \_\_|\_\_\_\_\_\_/ /\ \ | |

\ / | | |\_| \\_\_ \ / \_\_\_\_ \ \_| |\_

\/ |\_|\\_\_|\_|\_\_\_/ /\_/ \\_\\_\_\_\_\_|

==========================================

Docker Image Version: 2.0.0.1103 (CPU)

Vitis AI Git Hash: 06d7cbb

Build Date: 2022-01-12

For TensorFlow 1.15 Workflows do:

conda activate vitis-ai-tensorflow

For Caffe Workflows do:

conda activate vitis-ai-caffe

For PyTorch Workflows do:

conda activate vitis-ai-pytorch

For TensorFlow 2.6 Workflows do:

conda activate vitis-ai-tensorflow2

Vitis-AI /workspace >

2-2. Activate the TensorFlow2 conda environment.

2-2-1. Enter the following command to activate the TensorFlow2 conda environment:

Vitis-AI /workspace > conda activate vitis-ai-tensorflow2

The terminal output should be similar to the following:

(vitis-ai-tensorflow2) Vitis-AI /workspace >

2-2-2. Enter the following commands to run the Vitis AI quantizer for Tensorflow2:

(vitis-ai-tensorflow2) Vitis-AI /workspace > cd vai\_q\_c\_tf2\_pt/lab

(vitis-ai-tensorflow2) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > sh 1\_tf2\_quantize.sh

The quantizer tool now starts running. The execution time usually depends on the calibration dataset.

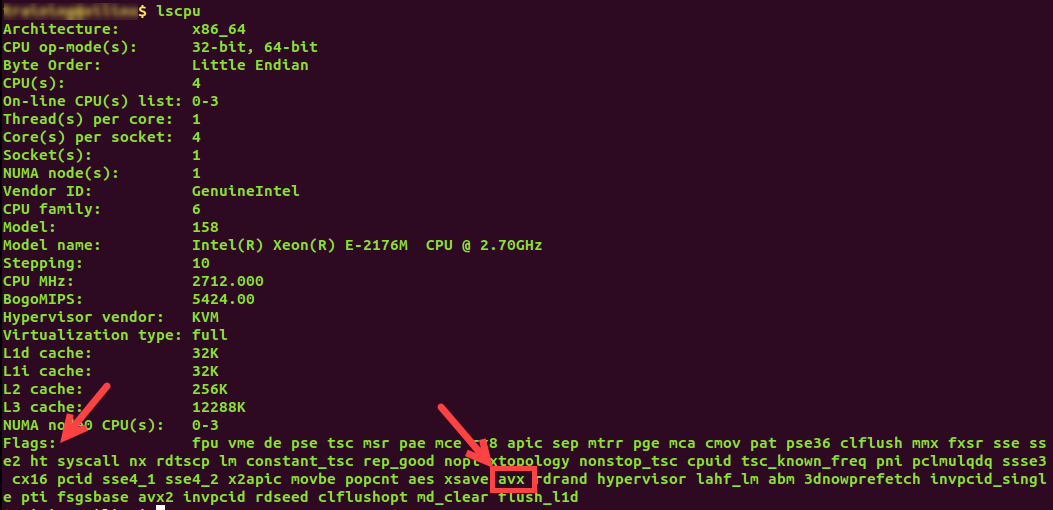
Known issue: You may see the message "Illegal instruction (core dumped)" on some PCs:

(vitis-ai-tensorflow2) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > sh 1\_tf2\_quantize.sh

Illegal instruction (core dumped)

This issue is due to the non-support of the avx instruction by the CPU of the PC. You can verify this by entering the command lscpu in the terminal and verifying the flags as shown below. If you can see the avx flag, then your CPU supports running TensorFlow2.

If your PC is not supported, then you will not see the avx flag. If so, you can just review the TensorFlow2 instructions that follow, and you can perform the PyTorch instructions in Steps 3 and 4, which should have no CPU restrictions.



**Figure 2‑2: Verifying avx Support**

It should take approximately 7-10 minutes if you are using the VM. If you are using CloudShare, it should take approximately 5-8 minutes.

After the quantization is over, you should see messages as shown below.

...

2022-03-10 23:42:36.415267: I tensorflow/core/platform/cpu\_feature\_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

/opt/vitis\_ai/conda/envs/vitis-ai-tensorflow2/lib/python3.7/site-packages/keras/utils/generic\_utils.py:497: CustomMaskWarning: Custom mask layers require a config and must override get\_config. When loading, the custom mask layer must be passed to the custom\_objects argument.

category=CustomMaskWarning)

[VAI INFO] Update custom\_layer\_type: []

[VAI INFO] Start CrossLayerEqualization...

10/10 [==============================] - 25s 3s/step

[VAI INFO] CrossLayerEqualization Done.

2022-03-10 23:43:17.371169: I tensorflow/compiler/mlir/mlir\_graph\_optimization\_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)

[VAI INFO] Start Quantize Calibration...

10/10 [==============================] - 248s 20s/step

[VAI INFO] Quantize Calibration Done.

[VAI INFO] Start Post-Quantize Adjustment...

[VAI INFO] Post-Quantize Adjustment Done.

[VAI INFO] Quantization Finished.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

Quantize finished, results in: tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/vai\_q\_output

(vitis-ai-tensorflow2) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab >

After successful execution of the quantizer script, the quantized file quantized.h5 is generated in the output directory (/home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/vai\_q\_output).

For TensorFlow2.x, the quantizer generates the quantized model in the h5 format.

2-3. Review the 4\_tf2\_compile\_for\_mpsoc.sh script.

2-3-1. Press <Ctrl + Alt + T> to open a new terminal window.

2-3-2. Enter the following command to review the compiler arguments in the compiler script:

[host]$ gedit /home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/lab/  
4\_tf2\_compile\_for\_mpsoc.sh

The detailed commands used in the script are as shown below.

vai\_c\_tensorflow2 -m ${TF2\_NETWORK\_PATH}/vai\_q\_output/quantized.h5 \

-a /opt/vitis\_ai/compiler/arch/DPUCZDX8G/ZCU102/arch.json \

-o ${TF2\_NETWORK\_PATH}/vai\_c\_output \

-n resnet50\_tf2 \

The Vitis AI compiler for Tensorflow2 arguments are listed here:

| Quantizer Argument | Description |
| --- | --- |
| -m MODEL | h5 model file |
| -a ARCH | Architecture JSON file |
| -o OUTPUT\_DIR | Output directory location to store the generated output |
| -n NET\_NAME | Prefix-name for the outputs |
| -e OPTIONS | Extra options |

2-3-3. After you complete the review, close the file.

2-4. Run the AI compiler tool by executing the 4\_tf2\_compile\_for\_mpsoc.sh script.

2-4-1. Go to the terminal where the quantizer had been executed.

2-4-2. Enter the following command to run the Vitis AI compiler for TensorFlow2:

(vitis-ai-tensorflow2) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > sh 4\_tf2\_compile\_for\_mpsoc.sh

When a network model is compiled, the required options should be specified to the Vitis AI compiler. Once compilation is successful, the Vitis AI compiler will generate the xmodel file. These files are located under the folder specified by output\_dir.

You should see the following output after compilation has completed.

(vitis-ai-tensorflow2) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > sh 4\_tf2\_compile\_for\_mpsoc.sh

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* VITIS\_AI Compilation - Xilinx Inc.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[INFO] Namespace(batchsize=1, inputs\_shape=None, layout='NHWC', model\_files=['tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/vai\_q\_output/quantized.h5'], model\_type='tensorflow2', named\_inputs\_shape=None, out\_filename='/tmp/resnet50\_tf2\_org.xmodel', proto=None)

[INFO] tensorflow2 model: /workspace/vai\_q\_c\_tf2\_pt/lab/tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/vai\_q\_output/quantized.h5

[INFO] keras version: 2.6.0

[INFO] Tensorflow Keras model type: functional

[INFO] parse raw model :100%|████████████████████████████████████████████████████████████████| 126/126 [00:00<00:00, 11566.19it/s]

[INFO] infer shape (NHWC) :100%|████████████████████████████████████████████████████████████████| 200/200 [00:00<00:00, 1037.68it/s]

[INFO] perform level-0 opt :100%|████████████████████████████████████████████████████████████████| 2/2 [00:00<00:00, 17.37it/s]

[INFO] perform level-1 opt :100%|████████████████████████████████████████████████████████████████| 2/2 [00:00<00:00, 109.23it/s]

[INFO] infer shape (NHWC) :100%|████████████████████████████████████████████████████████████████| 202/202 [00:00<00:00, 2370.90it/s]

[INFO] generate xmodel :100%|████████████████████████████████████████████████████████████████| 202/202 [00:00<00:00, 242.95it/s]

[INFO] dump xmodel: /tmp/resnet50\_tf2\_org.xmodel

[UNILOG][INFO] Compile mode: dpu

[UNILOG][INFO] Debug mode: function

[UNILOG][INFO] Target architecture: DPUCZDX8G\_ISA0\_B4096\_MAX\_BG2

[UNILOG][INFO] Graph name: resnet50, with op num: 416

[UNILOG][INFO] Begin to compile...

[UNILOG][INFO] Total device subgraph number 3, DPU subgraph number 1

[UNILOG][INFO] Compile done.

[UNILOG][INFO] The meta json is saved to "/workspace/vai\_q\_c\_tf2\_pt/lab/tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/vai\_c\_output/meta.json"

[UNILOG][INFO] The compiled xmodel is saved to "/workspace/vai\_q\_c\_tf2\_pt/lab/tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/vai\_c\_output/resnet50\_tf2.xmodel"

[UNILOG][INFO] The compiled xmodel's md5sum is 1a4f604f2d92526b93c5b9a722dfb07d, and has been saved to "/workspace/vai\_q\_c\_tf2\_pt/lab/tf2\_resnet50\_imagenet\_224\_224\_7.76G\_2.0/vai\_c\_output/md5sum.txt"

The compiler creates three files in the OUTPUT\_DIR directory:

* Compiled xmodel: resnet50\_tf2.xmodel
* For run time: meta.json
* Checksum: md5sum.txt

The compiled xmodel is used to program the DPU.

2-4-3. Enter the following command to exit the Vitis AI tool:

(vitis-ai-tensorflow2) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > exit

Reviewing the PyTorch AI Quantizer Arguments and Training   
Scripts Step

3-1. Review the 5\_pytorch\_quantize.sh script.

3-1-1. Press <Ctrl + Alt + T> to open a new terminal window.

3-1-2. Enter the following command to review the quantizer arguments in the quantizer script:

[host]$ gedit /home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/lab/  
5\_pytorch\_quantize.sh

The detailed commands used in the script are as shown below.

python resnet50\_quantize.py --quant\_mode calib

python resnet50\_quantize.py --quant\_mode test

Note: Observe that the python script resnet50\_quantize.py is used to quantize the trained model. Usually, this script comes from the trainer who has created it for training the model.

The argument --quant\_mode is set to calib for quantization, and after calibration, --quant\_mode is set to test to evaluate the quantized model.

3-1-3. After you complete the review, close the file.

3-2. Review the resnet50\_quantize.py training script.

Usually, this script comes from the trainer who has created it for training the model. This script is specific to the model—in this case, it is for ResNet50. The user will need to import the Vitis AI quantizer.

3-2-1. Enter the following command to review the training script that has been updated with the Vitis AI quantizer:

[host]$ gedit /home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/lab/  
resnet50\_quantize.py

* Observe that the PyTorch quantize is imported at near line no. 56.
* Verify that the GPU is available; else use the CPU at near line no. 64 to 68.
* Arguments are declared from near line no. 72 to 95:
* --data\_dir: Dataset directory, when quant\_mode=calib, it is for calibration, and when quant\_mode=test, it is for evaluation.
* --model\_dir: Trained model file path.
* --subset\_len: subset\_len to evaluate the model, using the whole validation dataset if it is not set.
* --quant\_mode: Quantization mode {float, calib, test}:
* float: No quantization, evaluate float model.
* calib: Quantize.
* test: Evaluate the quantized model.
* --finetune: Finetune the model before calibration.
* The main() program starts at line no. 293:
* At near line no. 295, the model name is set to resnet50.
* At near line no. 299, verify the quant\_mode argument that is passed as calib in the 5\_pytorch\_quantize.sh script.
* At near line no. 311, the function quantization has been called with the following arguments:
* title: resnet50 with optimization
* model\_name: resnet50
* file\_path: pt\_resnet50.pth
* quant\_mode: calib
* finetune: Set to default

3-2-2. After you complete the review, close the file.

3-2-3. Close the terminal.

Running the AI Quantizer and AI Compiler for PyTorch Step

You will now run the Vitis AI quantizer for PyTorch using the training script, verify the generated output, and then run the Vitis AI compiler (vai\_c\_xir) for PyTorch.

4-1. Load the CPU image from the Docker hub.

4-1-1. Press <Ctrl + Alt + T> to open a new terminal window.

4-1-2. Enter the following commands to load the CPU image from the Docker hub:

[host]$ cd Vitis-AI

[host]$ ./docker\_run.sh xilinx/vitis-ai-cpu:latest

If there is a newer version of the Vitis AI tools that has been released, entering the above command will download the latest version (that is, not VAI 2.0). In order to use the Vitis AI 2.0 environment, enter the command with the docker image tag. You can find the tag from https://hub.docker.com/r/xilinx/vitis-ai-cpu/tags?page=1&ordering=last\_updated.

For example: [host]$ ./docker\_run.sh xilinx/vitis-ai-cpu:<Docker\_tag>

To use 2.0: [host]$ ./docker\_run.sh xilinx/vitis-ai-cpu:2.0

Note: Keep clicking to accept the terms and agreements and then enter 'y'.

The terminal output should be similar to the following:

==========================================

\_\_ \_\_\_ \_ \_ \_\_\_\_\_

\ \ / (\_) | (\_) /\ |\_ \_|

\ \ / / \_| |\_ \_ \_\_\_ \_\_\_\_\_\_ / \ | |

\ \/ / | | \_\_| / \_\_|\_\_\_\_\_\_/ /\ \ | |

\ / | | |\_| \\_\_ \ / \_\_\_\_ \ \_| |\_

\/ |\_|\\_\_|\_|\_\_\_/ /\_/ \\_\\_\_\_\_\_|

==========================================

Docker Image Version: 2.0.0.1103 (CPU)

Vitis AI Git Hash: 06d7cbb

Build Date: 2022-01-12

For TensorFlow 1.15 Workflows do:

conda activate vitis-ai-tensorflow

For Caffe Workflows do:

conda activate vitis-ai-caffe

For PyTorch Workflows do:

conda activate vitis-ai-pytorch

For TensorFlow 2.6 Workflows do:

conda activate vitis-ai-tensorflow2

Vitis-AI /workspace >

4-2. Activate the PyTorch conda environment.

4-2-1. Enter the following command to activate the PyTorch conda environment:

Vitis-AI /workspace > conda activate vitis-ai-pytorch

The terminal output should be similar to the following:

(vitis-ai-pytorch) Vitis-AI /workspace >

4-2-2. Enter the following commands to run the Vitis AI quantizer for PyTorch:

(vitis-ai-pytorch) Vitis-AI /workspace > cd vai\_q\_c\_tf2\_pt/lab

(vitis-ai-pytorch) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > sh 5\_pytorch\_quantize.sh

The quantizer tool now starts running. The execution time usually depends on the calibration dataset.

In this case, it should take approximately 8-10 minutes if you are using the VM. If you are using CloudShare, it should take approximately 5-6 minutes.

After the quantization is over, you should see messages as shown below.

No CUDA runtime is found, using CUDA\_HOME='/usr/local/cuda'

[VAIQ\_NOTE]: Loading NNDCT kernels...

Using CPU

-------- Start resnet50 test

[VAIQ\_WARN]: CUDA is not available, change device to CPU

[VAIQ\_NOTE]: Quantization test process start up...

[VAIQ\_NOTE]: =>Quant Module is in 'cpu'.

[VAIQ\_NOTE]: =>Parsing ResNet...

[VAIQ\_NOTE]: Start to trace model...

[VAIQ\_NOTE]: Finish tracing.

[VAIQ\_NOTE]: Processing ops...

██████████████████████████████████████████████████| 176/176 [00:00<00:00, 1088.70it/s, OpInfo: name = ResNet/Linear[fc]/8299, type = addmm]

[VAIQ\_NOTE]: =>Doing weights equalization...

[VAIQ\_NOTE]: =>Quantizable module is generated.(pt\_resnet50/vai\_q\_output/ResNet.py)

[VAIQ\_NOTE]: =>Get module with quantization.

100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████████| 48/48 [00:54<00:00, 1.14s/it]

loss: 12.7056

top-1 / top-5 accuracy: 8.33333 / 8.33333

[VAIQ\_NOTE]: =>Converting to xmodel ...

[VAIQ\_NOTE]: =>Successfully convert 'ResNet' to xmodel.(pt\_resnet50/vai\_q\_output/ResNet\_int.xmodel)

-------- End of resnet50 test

(vitis-ai-pytorch) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab >

After successful execution of the quantizer script, the quantized file ResNet\_int.xmodel is generated in the output directory (/home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/pt\_resnet50/vai\_q\_output).

For PyTorch, the quantizer NNDCT outputs the quantized model in the XIR format directly. Use vai\_c\_xir to compile it.

4-3. Review the 6\_pytorch\_compile\_for\_mpsoc.sh script.

4-3-1. Press <Ctrl + Alt + T> to open a new terminal window.

4-3-2. Enter the following command to review the compiler arguments in the compiler script:

[host]$ gedit /home/xilinx/Vitis-AI/vai\_q\_c\_tf2\_pt/lab/  
6\_pytorch\_compile\_for\_mpsoc.sh

The detailed commands used in the script are as shown below.

#!/bin/sh

TARGET=MPSOC

NET\_NAME=resnet50

DEPLOY\_MODEL\_PATH=vai\_q\_output

ARCH=/opt/vitis\_ai/compiler/arch/DPUCZDX8G/ZCU102/arch.json

vai\_c\_xir -x pt\_resnet50/vai\_q\_output/ResNet\_int.xmodel \

-o pt\_resnet50/vai\_c\_output/${TARGET} \

-n pt\_resnet50 \

-a ${ARCH}

The Vitis AI compiler for PyTorch arguments are listed here:

| Quantizer Argument | Description |
| --- | --- |
| -x MODEL | xmodel file |
| -o OUTPUT\_DIR | Output directory location to store the generated output |
| -n NET\_NAME | Prefix name for the outputs |
| -a ARCH | Architecture JSON file |
| -e OPTIONS | Extra options |

4-3-3. After you complete the review, close the file.

4-4. Run the AI compiler tool by executing the 6\_pytorch\_compile\_for\_mpsoc.sh script.

4-4-1. Go to the terminal where the quantizer had been executed.

4-4-2. Enter the following command to run the Vitis AI compiler for PyTorch:

(vitis-ai-pytorch) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > sh 6\_pytorch\_compile\_for\_mpsoc.sh

When a network model is compiled, the required options should be specified to the Vitis AI compiler. Once compilation is successful, the Vitis AI compiler will generate the xmodel file. These files are located under the folder specified by output\_dir.

You should see the following output after compilation has completed.

(vitis-ai-pytorch) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > sh 6\_pytorch\_compile\_for\_mpsoc.sh

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* VITIS\_AI Compilation - Xilinx Inc.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[UNILOG][INFO] Compile mode: dpu

[UNILOG][INFO] Debug mode: function

[UNILOG][INFO] Target architecture: DPUCZDX8G\_ISA0\_B4096\_MAX\_BG2

[UNILOG][INFO] Graph name: ResNet, with op num: 415

[UNILOG][INFO] Begin to compile...

[UNILOG][INFO] Total device subgraph number 3, DPU subgraph number 1

[UNILOG][INFO] Compile done.

[UNILOG][INFO] The meta json is saved to "/workspace/vai\_q\_c\_tf2\_pt/lab/pt\_resnet50/vai\_c\_output/MPSOC/meta.json"

[UNILOG][INFO] The compiled xmodel is saved to "/workspace/vai\_q\_c\_tf2\_pt/lab/pt\_resnet50/vai\_c\_output/MPSOC/pt\_resnet50.xmodel"

[UNILOG][INFO] The compiled xmodel's md5sum is a9474e9927141281f1dbba1173a2f218, and has been saved to "/workspace/vai\_q\_c\_tf2\_pt/lab/pt\_resnet50/vai\_c\_output/MPSOC/md5sum.txt"

(vitis-ai-pytorch) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab >

The compiler creates three files in the OUTPUTPATH directory:

* Compiled xmodel: pt\_resnet50.xmodel
* For run time: meta.json
* Checksum: md5sum.txt

The compiled xmodel is used to program the DPU.

4-4-3. Enter the following command to exit the Vitis AI tool:

(vitis-ai-pytorch) Vitis-AI /workspace/vai\_q\_c\_tf2\_pt/lab > exit

## Summary

In this lab, you reviewed the inputs for the Vitis AI quantizer (for both the TensorFlow2 and PyTorch frameworks) and then ran the Vitis AI quantizer and Vitis AI compiler (for both TensorFlow2 and PyTorch) by using the output from the quantizer.

# 