Vitis AI User Documentation

vai_q_caffe and vai_q_tensorflow are the names of our Vitis AI quantizer, where ‘q’ stands for quantizer and caffe/tensorflow are the framework names. This section helps you to quantize a Resnet-50 model quickly. See the Model Quantization for a full introduction of the VAI quantizer.

TensorFlow Version

Use the following the steps to run vai_q_tensorflow.

1. Prepare floating-point frozen model and dataset.

Table 1. Input Files for vai_q_tensorflow

No.	Name	Description
1	frozen_graph	Frozen Resnet-50 model.
2	calib_images	Before launching quantization for ResNet-50, prepare the calibration dataset. You can download 100 to 1000 images of ImageNet dataset from http://academictorrents.com/collection/imagenet-2012 or http://www.image-net.org/download.php and then change the calibration dataset path in the input_fn.
3	input_fn	A Python function to read images in the calibration dataset and perform pre-processing (e.g. resize, normalization).

Input files for vai_q_tensorflow are shown in the above table. The frozen model can be downloaded from the Xilinx model zoo (https://github.com/Xilinx/Vitis-AI/tree/master/AI-Model-Zoo). Scripts to evaluate the models can also be found in the model zoo.

`input_fn` is a python function to read images in the calibration dataset and preform pre-processing. An `calib_image_list` is an image list file for calibration and `calib_image_dir` is the directory containing the calibration image files. Function `calib_input` is the required input function for quantizer.

```
import tensorflow as tf
import os

_R_MEAN = 123.68
_G_MEAN = 116.78
_B_MEAN = 103.94

class Data_loader(object):
  def __init__(self, out_height, out_width, smallest_side=256):
    self._sess = tf.Session()
    self._out_height = out_height
    self._out_width = out_width
    self._smallest_side = smallest_side

    self._decode_jpeg_data = tf.placeholder(dtype=tf.string)
    self._decode_jpeg = tf.image.decode_jpeg(self._decode_jpeg_data, channels=3)

    self._image_pl = tf.placeholder(tf.float32, shape=(None, None, 3))
    self._resized_image = self._aspect_preserving_resize(self._image_pl,
self._smallest_side)

  def _center_crop(self, image):
    image_height, image_width = image.shape[:2]
    offset_height = (image_height - self._out_height) // 2
    offset_width = (image_width - self._out_width) // 2
    image = image[offset_height:offset_height + self._out_height,
                  offset_width:offset_width + self._out_width, :]
    return image

  def _smallest_size_at_least(self, height, width, smallest_side):
    """Computes new shape with the smallest side equal to `smallest_side`.

    Computes new shape with the smallest side equal to `smallest_side` while
    preserving the original aspect ratio.
```

The calibration image list file `calib_image_list` looks like this:

```
ILSVRC2012_val_00000001.JPEG
ILSVRC2012_val_00000002.JPEG
ILSVRC2012_val_00000003.JPEG
ILSVRC2012_val_00000004.JPEG
...
```

2. Activate Tensorflow running environment.

```
conda activate vitis-ai-tensorflow
```

3. Run vai_q_tensorflow to quantize the TensorFlow frozen models.

```
vai_q_tensorflow quantize \
  --input_frozen_graph resnet_v1_50_inference.pb \
  --input_nodes input \
  --input_shapes ?,224,224,3 \
  --output_nodes resnet_v1_50/predictions/Reshape_1 \
  --input_fn input_fn.calib_input \
  --method 1 \
  --gpu 0 \
  --calib_iter 20 \
  --output_dir ./quantize_results \
```

Here `--input_fn` is set to be `"input_fn.calib_input"`. `input_fn` is the name of python script and `calib_input` is the function name in `input_fn.py`. The script may take several minutes to finish. Running the script displays messages as shown below:

```
INFO: Checking Float Graph...
INFO: Float Graph Check Done.
2020-03-07 06:46:35.567522: W tensorflow/contrib/decent_q/utils/quantize_utils.cc:538]
Convert mean node resnet_v1_50/pool5 to AvgPool
2020-03-07 06:46:35.572301: W tensorflow/contrib/decent_q/utils/quantize_utils.cc:628] Scale
output of avg_pool node resnet_v1_50/pool5 to simulate DPU.
INFO: Calibrating for 20 iterations...
100% (20 of 20)
|###########################################################################################
#########| Elapsed Time: 0:21:11 Time:  0:21:11
INFO: Calibration Done.
INFO: Generating Deploy Model...
[DEPLOY WARNING] Node resnet_v1_50/predictions/Reshape_1(Type: Reshape) is not quantized and
cannot be deployed to DPU,because it has unquantized input node:
resnet_v1_50/predictions/Softmax. Please deploy it on CPU.
INFO: Deploy Model Generated.
******************** Quantization Summary ********************
INFO: Output:
  quantize_eval_model: ./quantize_results/quantize_eval_model.pb
  deploy_model: ./quantize_results/deploy_model.pb
```

Two files will be generated in `quantize_results` directory. The `deploy_model.pb` could be fed to VAI compiler for the following compilation processes targeting hardware platform DPUCZDX8G. The `quantize_eval_model.pb` can be used for model evaluation and dump on GPU or CPU. It is also the input file for compilation processes targeting hardware platform DPUCAHX8H.

Caffe Version

vai_q_caffe takes a floating-point model as an input model and uses a calibration dataset to generate a quantized model. Use the following steps to create and quantize Resnet50 floating-point model.

1. Prepare a floating-point model for Resnet-50. You can download one from the internet or from Xilinx modelzoo (https://github.com/Xilinx/Vitis-AI/tree/master/AI-Model-Zoo).
2. Prepare the calibration dataset used by vai_q_caffe. You can download 100 to 1000 images of ImageNet dataset from http://academictorrents.com/collection/imagenet-2012 or http://www.image-

net.org/download.php and then change the `source` and `root_folder` of `image_data_param` in ResNet-50 prototxt accordingly. For example, the ImageData layer in prototxt looks like the following:

```
layer {
  name: "data"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    mirror: false
    crop_size: 224
    mean_value: 104
    mean_value: 107
    mean_value: 123
  }
  image_data_param {
    source: "/path/calibration.txt"
    root_folder: "/path/calibration_images/"
    batch_size: 20
    shuffle: false
  }
}
```

For quantize calibration, calibration data without label is enough. But due to the implementation, a image list file with two columns is required. Just set the second column to a random value or zero. This is an example of "`calibration.txt`".

```
n01440764_985.JPEG 0
n01443537_9347.JPEG 0
n01484850_8799.JPEG 0
```

3. Activate the caffe running environment:

```
conda activate vitis-ai-caffe
```

4. Start quantization:

```
vai_q_caffe quantize -model float.prototxt -weights float.caffemodel
```

If your targeting hardware platform is DPUCAHX8H, another option "`-keep_fixed_neuron`" should be added to the command. Refer to Chapter 4 for details.

```
vai_q_caffe quantize -model float.prototxt -weights float.caffemodel -keep_fixed_neuron
```

This invokes the vai_q_caffe tool to perform quantization with the appropriate parameters. The running time of this command varies from a few seconds to several minutes, depending on hardware and the size of the neural network. Four files are generated in the output directory, including `deploy.prototxt` and `deploy.caffemodel`, which could be fed to VAI compiler for the following compilation process.

Pytorch Version

vai_q_pytorch is designed to work as a Pytorch plugin. We provide simplest APIs to introduce our FPAG-friendly quantization feature. vai_q_pytorch package is already installed in docker image "vitis-ai-pytorch" environment. A Resnet18 example is in our open-source repo. Use the following steps to quantize a Resnet18 model.

1. Prepare a floating-point model for Resnet-18. You can download one from Pytorch official site.

   ```
   wget https://download.pytorch.org/models/resnet18-5c106cde.pth -O resnet18.pth
   ```

2. Prepare the calibration dataset used by vai_q_pytorch. You can download 100 to 1000 images of ImageNet dataset from http://academictorrents.com/collection/imagenet-2012 or http://www.image-net.org/download.php
3. Copy float model `resnet18.pth`, example code `resnet18_quant.py`, and calibration images to docker image and modify default `data_dir` and `model_dir` in `resnet18_quant.py` accordingly.
4. Activate the Pytorch running environment:

   ```
   conda activate vitis-ai-pytorch
   ```

5. Evaluate float model

   ```
   python resnet18_quant.py --quant_mode 0
   ```

6. Quantize, using a subset (200 images) of validation data for calibration. Because we are in quantize calibration process, the displayed loss and accuracy are meaningless.

   ```
   python resnet18_quant.py --quant_mode 1 --subset_len 200
   ```

7. Evaluate quantized model and generate xmodel file for compiler.

   ```
   python resnet18_quant.py --quant_mode 2
   ```

   A `ResNet_int.xmodel` file will be generated under folder `quantize_result`. It could be fed to VAI compiler for following compilation process.