1. Compress the neural network model. – DECENT
2. Compile the neural network model. -- DNNC
3. Program with DNNDK APIs.

## Network Compression (TensorFlow Version)

**DECENT takes a floating-point network model, pre-trained weights, and a calibration dataset in Caffe format as inputs to generate a lightweight quantized model with INT8 weights.**

**Table 5: Input files for DECENT_Q**

| No. | Name | Description |
|---|---|---|
| 1 | frozen_resnet_v1_50.pb | Floating-point frozen graph for ResNet-50 |
| 2 | calibration dataset | A subset of the ImageNet Dataset (without labels) containing 100 images |
| 3 | input_fn | A python function to read images in calibration dataset and do preprocess |

A script file named `resnet_v1_50_input_fn.py` can be found in `$dnndk_pkg/host_x86/models/resnet50`, shown in the following figure.

- A script file named decent_q.sh can be found in $dnndk_pkg/host_x86/models/resnet50, shown in the following figure. This invokes the DECENT_Q tool to perform quantization with the appropriate parameters.

The script might take several minutes to finish. When quantization is done, the following two files are generated under the `quantize_results` directory. Then you can use `deploy_model.pb` to compile the model using DNNC.

**Table 6: DECENT_Q Output Files**

| No. | Name | Description |
|---|---|---|
| 1 | deploy_model.pb | Quantized model for DNNC (extended TensorFlow format) |
| 2 | quantize_eval_model.pb | Quantized model for evaluation |

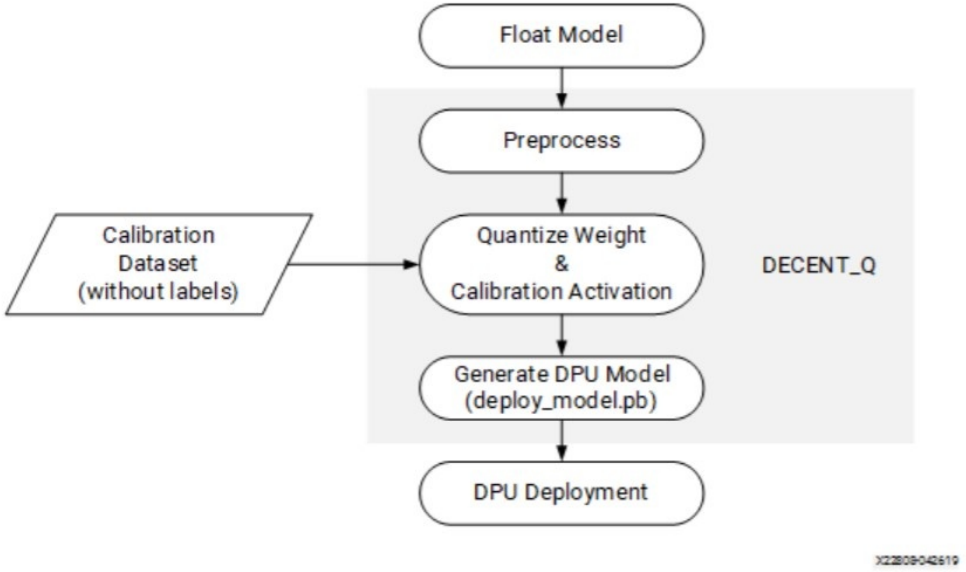
**- The script file for compiling TensorFlow ResNet-50 model, dnnc.sh, can be found in $dnndk_pkg/host_x86/models/tensorflow/resnet50, as shown in the following figure. For TensorFlow model compilation, you must specify the parser type using tensorflow through the -- parser option**

- The kernels resnet50_0 runs on the DPU. DNNC generates an ELF object file for this kernel in the output_dir directory, with the name dpu_resnet50_0.elf. The other kernel, resnet50_1, is for “Softmax” operations, which are not supported by DPU and must be deployed and run on the CPU.

- After calibration, the quantized model is transformed to DPU deployable model named deploy.prototxt/deploy.caffemodel by Caffe version DECENT or **deploy_model.pb for TensorFlow version DECENT, which follows the data format of DPU**. Then it can be compiled by DNNC compiler and deployed to DPU. Note that the quantized model cannot be taken in by standard vision Caffe or TensorFlow framework.

## DECENT Working Flow

The overall workflow of model quantization are as follows:



### *Preparing the Neural Network Model*

Before running DECENT_Q, prepare the frozen TensorFlow model in floating-point format and calibration set, including:

**Table 10: Input Files for DECENT_Q**

| No. | Name | Description |
|---|---|---|
| 1 | `frozen_graph.pb` | Floating-point frozen graph for ResNet-50. |
| 2 | `calibration dataset` | A subset of the training set containing 100 to 1000 images. |
| 3 | `Input_fn` | An input function to convert the calibration dataset to the frozen_graph 's input data during quantize calibration. Usually will do some data preprocessing and augmentation. |

#### How to Get the Frozen Graph

In most situations, training a model with TensorFlow gives you a folder containing a GraphDef file (usually ending with the `.pb` or `.pbtxt` extension) and a set of checkpoint files. What you need for mobile or embedded deployment is a single GraphDef file that has been 'frozen', or had its variables converted into inline constants so everything is in one file. To handle the conversion, TensorFlow provided `freeze_graph.py`, which is automatically installed with DECENT_Q.

An example of command-line usage is:

```
$ freeze_graph \
    --input_graph=/tmp/inception_v1_inf_graph.pb \
    --input_checkpoint=/tmp/checkpoints/model.ckpt-1000 \
    --input_binary=true \
    --output_graph=/tmp/frozen_graph.pb \
    --output_node_names=InceptionV1/Predictions/Reshape_1
```

**Note:** Because the operations of data preprocessing and loss functions are not needed for inference and deployment, the `frozen_graph.pb` should only include the main part of the model. In particular,

- Run DECENT_Q

Run the following command line to quantize the model:

```
$decent_q quantize \
 --input_graph_def frozen_graph.pb \
 --input_nodes ${input_nodes} \
--input_shapes ${input_shapes} \
--output_nodes ${output_nodes} \
--input_fn Pre-defined|Custom input_fn \
```